Developpez.com
Club des développeurs

Introduction au langage Python par l'exemple

par Serge Tahé (home)

Date de publication : 30/01/2012

Dernière mise à jour : 04/04/2012

Les thèmes suivants sont abordés :
- les bases du langage ;
- classes et héritage ;
- les exceptions ;
- architectures en couches et programmation par interfaces ;
- utilisation du SGBD MySQL ;
- architectures client / serveur TCP/IP ;
- services web.

MCours.com

I - Avant-propos.... 4
II - Remerciements.... 5
III - Installation d'un interpréteur Python.... 6
III-A - ActivePython.... 6
III-B - Python Tools for Visual Studio.... 8
IV - Les bases.... 10
IV-A - Un exemple de programme Python.... 10
IV-B - Changements de type.... 12
IV-C - La portée des variables.... 14
IV-D - Les listes, tuples et dictionnaires.... 15
IV-D-1 - Listes à une dimension.... 15
IV-D-2 - Le dictionnaire.... 17
IV-D-3 - Les tuples.... 18
IV-D-4 - Les listes à plusieurs dimensions.... 19
IV-D-5 - Liens entre chaînes et listes.... 19
IV-E - Les expressions régulières.... 20
IV-F - Mode de passage des paramètres des fonctions.... 22
IV-G - Les fichiers texte.... 23
V - Exercice d'application - [IMPÔTS].... 25
V-A - Le problème.... 25
V-B - Version avec listes.... 26
V-C - Version avec fichiers texte.... 27
VI - Les classes et objets.... 31
VI-A - Une classe Objet.... 31
VI-B - Une classe Personne.... 32
VI-C - La classe Personne avec un constructeur.... 32
VI-D - La classe Personne avec contrôles de validité dans le constructeur.... 33
VI-E - Ajout d'une méthode faisant office de second constructeur.... 34
VI-F - Une liste d'objets Personne.... 35
VI-G - Création d'une classe dérivée de la classe Personne.... 36
VI-H - Création d'une seconde classe dérivée de la classe Personne.... 36
VII - Les exceptions.... 38
VIII - Architecture en couches et programmation par interfaces.... 44
VIII-A - Introduction.... 44
VIII-B - Les entités de l'application.... 44
VIII-C - La couche [dao].... 46
VIII-D - La couche [métier].... 48
VIII-E - La couche [console].... 51
IX - Exercice d'application - [IMPOTS] avec objets.... 53
IX-A - La couche [DAO].... 53
IX-B - La couche [metier].... 55
IX-C - La couche [console].... 56
IX-D - Résultats.... 57
X - Utilisation du SGBD MySQL.... 58
X-A - Installation du module MySQLdb.... 58
X-B - Installation de MySQL.... 59
X-C - Connexion à une base MySQL - 1.... 62
X-D - Connexion à une base MySQL - 2.... 64
X-E - Création d'une table MySQL.... 65
X-F - Remplissage de la table [personnes].... 67
X-G - Exécution de requêtes SQL quelconques.... 69
XI - Exercice [IMPOTS] avec MySQL.... 73
XI-A - Transfert d'un fichier texte dans une table MySQL.... 73
XI-B - Le programme de calcul de l'impôt.... 76
XI-C - La classe [ImpotsMySQL].... 77
XI-D - Le script console.... 77
XII - Les fonctions réseau de Python.... 79
XII-A - Obtenir le nom ou l'adresse IP d'une machine de l'Internet.... 79

Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

XII-B - Un client Web 80
XII-C - Un client SMTP 82
XII-D - Un second client SMTP 86
XII-E - Client / serveur d'écho 89
XII-F - Serveur Tcp générique 92
XIII - Des services Web en Python 95
XIII-A - Application client/ serveur de date/heure 95
XIII-A-1 - Le serveur 95
XIII-A-2 - Deux tests 96
XIII-A-3 - Un client programmé 97
XIII-B - Récupération par le serveur des paramètres envoyés par le client 98
XIII-B-1 - Le service Web 98
XIII-B-2 - Le client GET 99
XIII-B-3 - Le client POST 100
XIII-C - Récupération des variables d'environnement d'un service Web 101
XIII-C-1 - Le service Web 101
XIII-C-2 - Le client programmé 103
XIV - Exercice [IMPOTS] avec un service Web 104
XIV-A - Le serveur 104
XIV-B - Un client programmé 106
XV - Traitement de documents XML 108
XVI - Exercice [IMPOTS] avec XML 112
XVI-A - Le service Web 112
XVI-B - Le client programmé 114

Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

I - Avant-propos

Ce document propose une liste de scripts Python dans différents domaines :

- les fondamentaux du langage ;
- la gestion de bases de données MySQL ;
- la programmation réseau TCP/ IP ;
- la programmation Web ;
- les architectures trois couches et la programmation par interfaces.

Ce n'est pas un cours Python mais un recueil d'exemples destinés à des développeurs ayant déjà utilisé un langage de script tel que Perl, PHP, Vbscript ou des développeurs habitués aux langages typés tels Java ou C# et qui seraient intéressés par la découverte d'un langage de script orienté objet. Ce document est inapproprié pour des lecteurs n'ayant jamais ou peu programmé.

Ce document n'est pas non plus un recueil de "bonnes pratiques". Le développeur expérimenté pourra ainsi trouver que certains codes pourraient être mieux écrits. Ce document a pour seul objectif de donner des exemples à une personne désireuse de s'initier rapidement au langage Python. Elle approfondira ensuite son apprentissage avec d'autres documents.

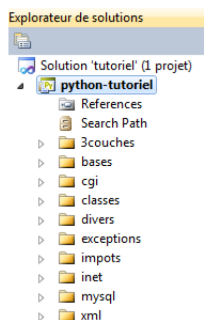
Ceux pour qui ce document « va trop vite » ou n'est pas assez explicite préféreront le tutoriel approfondi de Gérard Swinnen [**http://python.developpez.com/cours/TutoSwinnen/**] ou plus généralement les tutoriels de *developpez.com* sur Python [**http://python.developpez.com/cours**]. L'information ultime peut être trouvée à l'URL [**http://docs.python.org/**].

Les scripts sont commentés et leur exécution console reproduite. Parfois des explications supplémentaires sont fournies. Le document nécessite une lecture active : pour comprendre un script, il faut à la fois lire son code, ses commentaires et ses résultats console.

Les exemples du document sont disponibles à l'adresse :

[**http://tahe.ftp-developpez.com/divers/python/exemples/python_v2_exemples.zip**].

Ils sont livrés sous la forme d'une solution Visual Studio 2010 mais les scripts peuvent être utilisés en dehors de cet environnement.



Ce document peut comporter des erreurs ou des insuffisances. Le lecteur peut m'envoyer ses remarques à l'adresse [serge.tahe at univ-angers.fr].

Note importante : Ce document contient de nombreux exemples de code avec des lignes numérotées. Certains navigateurs rendent incorrectement la version HTML du document. La numérotation des lignes est erronée, ce qui rend incompréhensibles les commentaires qui la référencent. Parmi les navigateurs qui rendent correctement la numérotation on trouve **Firefox 11**, **Chrome 18** (6 avril 2012).

Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

II - Remerciements

Je remercie chaleureusement les personnes de **Developpez.com** qui ont contribué à l'édition de ce document :

- **http://www.developpez.net/forums/u35657/djibril/** pour la coordination de l'équipe d'édition ;
- **http://www.developpez.net/forums/u124512/claudeleloup/** et **http://www.developpez.net/forums/u386096/kdmbella/** pour la relecture orthographique ;
- **Pascal** pour ses propositions d'amélioration du document.

Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

III - Installation d'un interpréteur Python

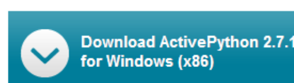
III-A - ActivePython

Les exemples de ce document ont été testés avec l'interpréteur Python d'*ActiveState* :

http://www.activestate.com/activepython/downloads.

Download Python: ActivePython Community Edition

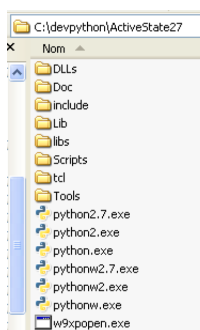
ActivePython is the leading commercial-grade distribution of the open source Python scripting language. Download ActivePython Community Edition free binaries for your development projects and internal deployments.



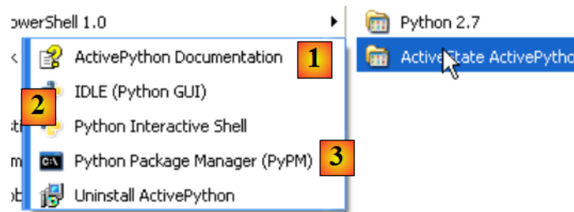
Notes :

- ne pas télécharger la version 64 bits d'ActivePython. Elle ne permet pas l'installation du module MySQLdb dont on a besoin dans la suite du document.

L'installation d'ActivePython donne naissance à l'arborescence suivante :



et au menu suivant dans l'arborescence des applications :



- [1] : la documentation d'ActivePython. Celle-ci est très riche et un bon réflexe est de la consulter dès qu'on rencontre un problème ;
- [2] : un interpréteur (shell) Python interactif ;
- [3] : le gestionnaire de packages d'ActivePython. Nous nous en servirons pour installer le package Python permettant de travailler avec des bases de données MySQL.

Nous n'utiliserons pas l'interpréteur Python interactif. Il faut simplement savoir que les scripts de ce document pourraient être exécutés avec cet interpréteur. Pratique pour tester le fonctionnement d'une fonctionnalité Python, il l'est peu pour des scripts devant être réutilisés. Voici un exemple :

Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

```
Python Shell
File Edit Shell Debug Options Windows Help
ActivePython 2.7.1.4 (ActiveState Software Inc.) based on
Python 2.7.1 (r271:86832, Feb  7 2011, 11:30:38) [MSC v.1500 32 bit
win32
Type "copyright", "credits" or "license()" for more information.
>>> nom='python'
>>> print 'nom=%s' % (nom)
nom=python
>>> print "type=%s" % (type(nom))
type=<type 'str'>
>>>
```

Le prompt >>> permet d'émettre une instruction Python qui est immédiatement exécutée. Le code tapé ci-dessus a la signification suivante :

```
1. >>> nom='python'
2. >>> print 'nom=%s' % (nom)
3. nom=python
4. >>> print "type=%s" % (type(nom))
5. type=<type 'str'>
6. >>>
```

Lignes :

- 1 : initialisation d'une variable. En Python, on ne déclare pas le type des variables. Celles-ci ont automatiquement le type de la valeur qu'on leur affecte. Ce type peut changer au cours du temps ;
- 2 : affichage du nom. '*nom=%s*' est un format d'affichage où *%s* est un paramètre formel désignant une chaîne de caractères. (*nom*) est le paramètre effectif qui sera affiché à la place de *%s* ;
- 3 : le résultat de l'affichage ;
- 4 : l'affichage du type de la variable *nom ;*
- 5 : la variable nom est de type *str* (string).

Dans la suite, nous proposons des scripts qui ont été testés de la façon suivante :

- le script peut être écrit avec un éditeur de texte quelconque. Ici nous avons utilisé *Notepad++* qui offre une coloration syntaxique des scripts Python ;
- le script est exécuté dans une fenêtre DOS.

```
Invite de commandes
               64 fichier(s)           127 544 octets
                3 Rép(s)  130 827 698 176 octets libres

C:\data\travail\2010-2011\python\tutoriel>%python% exemple_01.py
nom=dupont
tableau[0]=un
tableau[1]=deux
tableau[2]=3
tableau[3]=4
[chaine1,chaine2,chaine1chaine2]
chaine=chaine1
type[4]=<type 'int'>
type[chaine1]=<type 'str'>
type[['un', 'deux', 3, 4]]=<type 'list'>
type[a change]=<type 'str'>
res1=14
(res1,res2,res3)=[un,0,100]
t[0]=un
t[1]=0
t[2]=100
t[0]=8
t[1]=5
somme=13
```

- on se placera dans le dossier du script à tester ;
- la variable **%python%** est une variable système :

```
1. C:\data\travail\2010-2011\python\tutoriel>echo %python%
2. C:\devpython\ActiveState27\python.exe
```

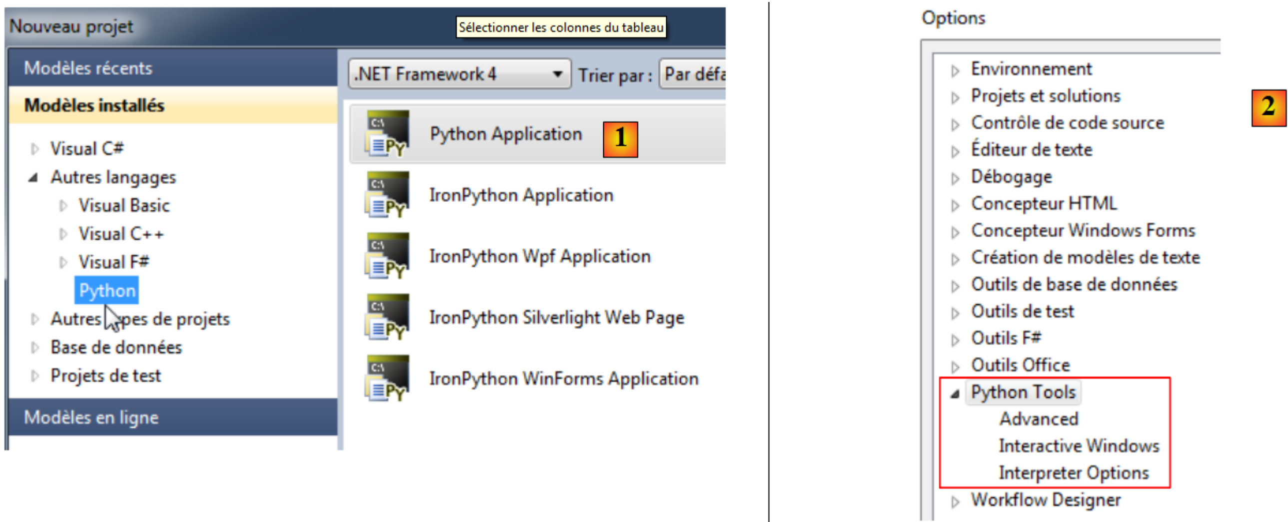
- ligne 1 : affichage de la valeur de la variable d'environnement *%python% ;*
- ligne 2 : sa valeur : *<installdir>\python.exe* où *installdir* est le dossier d'installation d'ActivePython.

Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

III-B - Python Tools for Visual Studio

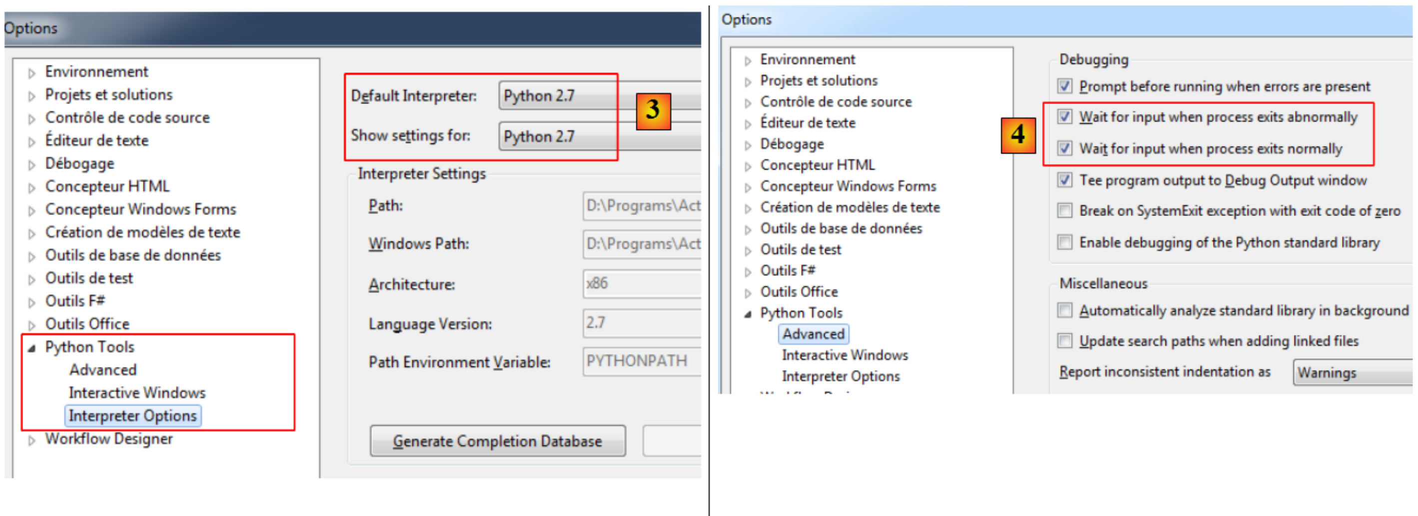
[Python Tools for Visual Studio] est disponible (février 2012) à l'URL suivante : [**http://pytools.codeplex.com/**]. Son exécution ajoute des fonctionnalités Python à Visual Studio 2010. La version Express de Visual Studio 2010 est gratuite et disponible à l'URL [**http://www.microsoft.com/visualstudio/en-us/products/2010-editions/express**].

Une fois les extensions [Python Tools for Visual Studio] installées, on peut alors créer des projets Python avec VS 2010 :



- pour tous les exemples de ce document, le choix [1] convient ;
- [2] : le menu (Outils / Options) permet de configurer l'environnement Python.

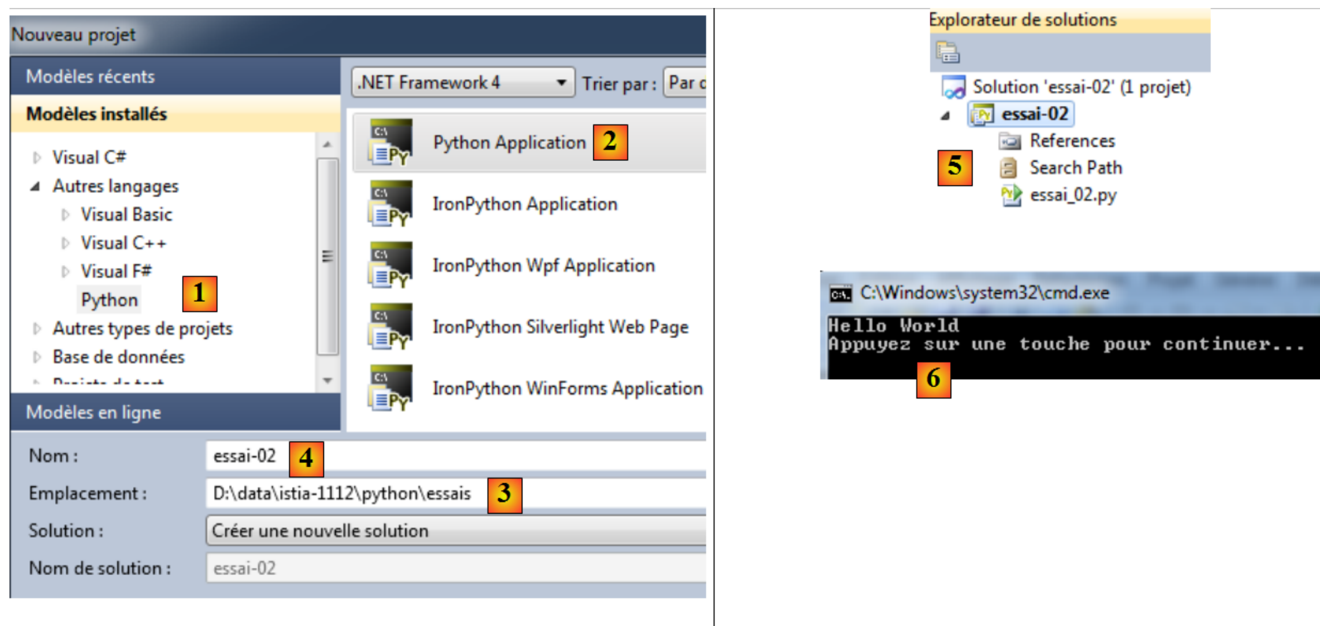
L'interpréteur d'ActivePython a normalement dû être détecté par Visual Studio [3] :



- en [4], on demande à ce que l'exécution du programme se termine par une pause que l'utilisateur interrompt en tapant une touche au clavier.

Créons un projet Python (Fichier/Nouveau/Projet) :

Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

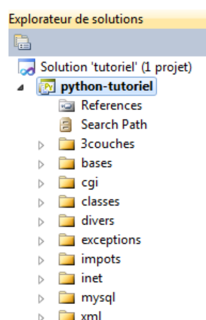


- en [1,2] : choisir un projet Python de type [Python Application] ;
- en [3] : choisir un dossier pour le projet ;
- en [4] : donner un nom au projet ;
- en [5] : le projet généré.

Le fichier [essai_02.py] est un programme Python :

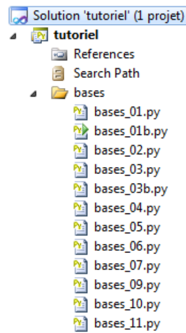
```
print ( 'Hello World' )
```

On l'exécute par [Ctrl-F5] : le résultat apparaît dans une fenêtre DOS [6]. Tous les codes du document ont également été testés de cette façon. Les exemples sont livrés sous la forme d'une solution VS 2010 :



Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

IV - Les bases



IV-A - Un exemple de programme Python

Ci-dessous, on trouvera un programme présentant les premières caractéristiques de Python.

Programme (bases_01)

```
1. # -*- coding=utf-8 -*-
2.
3. # ----------------------------------
4. def affiche(chaine):
5.     # affiche chaine
6.     print "chaine=%s" % (chaine)
7.
8.
9. # ----------------------------------
10. def afficheType(variable):
11.     # affiche le type de variable
12.     print "type[%s]=%s" % (variable,type(variable))
13.
14. # ----------------------------------
15. def f1(param):
16.     # ajoute 10 à param
17.     return param+10
18.
19. # ----------------------------------
20. def f2():
21.     # rend 3 valeurs
22.     return ("un",0,100);
23.
24.
25. # -------------------------------- programme principal ------------------------------------
26. # ceci est un commentaire
27. # variable utilisée sans avoir été déclarée
28. nom="dupont"
29.
30. # un affichage écran
31. print "nom=%s" % (nom)
32.
33. # une liste avec des éléments de types différents
34. liste=["un","deux",3,4]
35.
36. # son nombre d'éléments
37. n=len(liste)
38.
39. # une boucle
40. for i in range(n):
41.   print "liste[%d]=%s" % (i,liste[i])
42.
43. # initialisation de 2 variables avec un tuple
44. (chaine1,chaine2)=("chaine1","chaine2")
```


Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

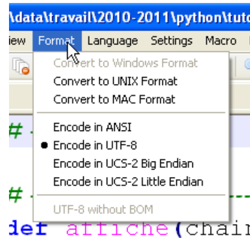
Programme (bases_01)

```
45.
46. # concaténation des 2 chaînes
47. chaine3=chaine1+chaine2
48.
49. # affichage résultat
50. print "[%s,%s,%s]" % (chaine1,chaine2,chaine3)
51.
52. # utilisation fonction
53. affiche(chaine1)
54.
55. # le type d'une variable peut être connu
56. afficheType(n)
57. afficheType(chaine1)
58. afficheType(liste)
59.
60. # le type d'une variable peut changer en cours d'exécution
61. n="a change"
62. afficheType(n)
63.
64. # une fonction peut rendre un résultat
65. res1=f1(4)
66. print "res1=%s" % (res1)
67.
68. # une fonction peut rendre une liste de valeurs
69. (res1,res2,res3)=f2()
70. print "(res1,res2,res3)=[%s,%s,%s]" % (res1,res2,res3)
71.
72. # on aurait pu récupérer ces valeurs dans une variable
73. liste=f2()
74. for i in range(len(liste)):
75.     print "liste[%s]=%s" % (i, liste[i])
76.
77. # des tests
78. for i in range(len(liste)):
79.     # n'affiche que les chaînes
80.     if (type(liste[i])=="str"):
81.         print "liste[%s]=%s" % (i,liste[i])
82.
83. # d'autres tests
84. for i in range(len(liste)):
85.     # n'affiche que les entiers >10
86.     if (type(liste[i])=="int" and liste[i]>10):
87.         print "liste[%s]=%s" % (i,liste[i])
88.
89. # une boucle while
90. liste=(8,5,0,-2,3,4)
91. i=0
92. somme=0
93. while(i<len(liste) and liste[i]>0):
94.     print "liste[%s]=%s" % (i,liste[i])
95.     somme+=liste[i] #somme=somme+t[i]
96.     i+=1 #i=i+1
97. print "somme=%s" % (somme)
98. # fin programme
```

À noter :

- ligne 1 : un commentaire particulier servant à déclarer le type d'encodage du script, ici UTF-8. Cela est dépendant de l'éditeur de texte utilisé. Ici avec Notepad++ :

Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.



- ligne 4 : le mot-clé **def** définit une fonction ;
- lignes 5-6 : le contenu de la fonction. Il est décalé à droite d'une tabulation. C'est cette indentation associée au caractère **:** de l'instruction *def* qui définit le contenu de la fonction. Cela est vrai pour toutes les instructions ayant du contenu : *if, else, while, for, try, except ;*
- ligne 12 : Python gère en interne le type des variables. On peut connaître le type d'une variable avec la fonction *type(variable)* qui rend une variable de type '*type*'. L'expression *"%s' % (type(variable))* est une chaîne de caractères représentant le type de la variable ;
- ligne 25 : le programme principal. Celui-ci vient après la définition de toutes les fonctions du script. Son contenu est non indenté ;
- ligne 28 : en Python, on ne déclare pas les variables. Python est sensible à la casse. La variable *Nom* est différente de la variable *nom*. Une chaîne de caractères peut être entourée de guillemets " ou d'apostrophes '. On peut donc écrire '*dupont*' ou "*dupont*" ;
- ligne 31 : l'expression *"xxxx**%s**yyyy**%s**zzzz" % (**100**, **200**)* est la chaîne de caractères *"xxxx**100**yyy**200**szzzz"* où chaque **%s** a été remplacé par un élément du tuple. **%s** est le format d'affichage des chaînes de caractères. Il existe d'autres formats ;
- ligne 34 : il y a une différence entre un **tuple** (1,2,3) (notez les parenthèses) et une **liste** [1,2,3] (notez les crochets). Le tuple est non modifiable alors que la liste l'est. Dans les deux cas, l'élément n° i est noté [i] ;
- ligne 40 : *range(n)* est le tuple (0,1,2,...,n-1) ;
- ligne 74 : *len(var)* est le nombre d'éléments de la collection *var* (tuple, liste, dictionnaire...) ;
- ligne 86 : les autres opérateurs booléens sont *or* et *not*.

Les résultats écran sont les suivants :

Résultats
```
1. nom=dupont
2. liste[0]=un
3. liste[1]=deux
4. liste[2]=3
5. liste[3]=4
6. [chaine1,chaine2,chaine1chaine2]
7. chaine=chaine1
8. type[4]=<type 'int'>
9. type[chaine1]=<type 'str'>
10. type[['un', 'deux', 3, 4]]=<type 'list'>
11. type[a change]=<type 'str'>
12. res1=14
13. (res1,res2,res3)=[un,0,100]
14. liste[0]=un
15. liste[1]=0
16. liste[2]=100
17. liste[0]=8
18. liste[1]=5
19. somme=13
```

IV-B - Changements de type

On s'intéresse ici aux changements de type avec des données de type **str** (chaîne de caractères), **int** (entier), **float** (réel), **bool** (booléen).

Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Programme (bases_01b)

```
1. # -*- coding=utf-8 -*-
2.
3. # changements de type
4. # int --> str, float, bool
5. x=4
6. print x, type(x)
7. x=str(4)
8. print x, type(x)
9. x=float(4)
10. print x, type(x)
11. x=bool(4)
12. print x, type(x)
13.
14. # bool --> int, float, str
15. x=True
16. print x, type(x)
17. x=int(True)
18. print x, type(x)
19. x=float(True)
20. print x, type(x)
21. x=str(True)
22. print x, type(x)
23.
24. # str --> int, float, bool
25. x="4"
26. print x, type(x)
27. x=int("4")
28. print x, type(x)
29. x=float("4")
30. print x, type(x)
31. x=bool("4")
32. print x, type(x)
33.
34. # float --> str, int, bool
35. x=4.32
36. print x, type(x)
37. x=str(4.32)
38. print x, type(x)
39. x=int(4.32)
40. print x, type(x)
41. x=bool(4.32)
42. print x, type(x)
43.
44. # gestion des erreurs de changement de type
45. try:
46.     x=int("abc")
47.     print x, type(x)
48. except ValueError, erreur:
49.     print erreur
50.
51. # cas divers du booléen
52. x=bool("abc")
53. print x, type(x)
54. x=bool("")
55. print x, type(x)
56. x=bool(0)
57. print x, type(x)
58. x=None
59. print x, type(x)
60. x=bool(None)
61. print x, type(x)
```

De nombreux changements de type sont possibles. Certains peuvent échouer, comme celui des lignes 45-49 qui essaient de transformer la chaîne 'abc' en nombre entier. On a géré l'erreur avec une structure *try / except*. Une forme générale de cette structure est la suivante :

```
1. try:
2.     actions
3. except Exception, Message:
```


Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

```
4.     actions
5. finally:
6.     actions
```

Si l'une des actions du **try** provoque une exception (signale une erreur), il y a branchement immédiat sur la clause **except**. Si les actions du **try** ne lancent pas d'exception, la clause **except** est ignorée. Les attributs *Exception* et *Message* de l'instruction *except* sont facultatifs. Lorsqu'ils sont présents, *Exception* précise le type d'exception interceptée par l'instruction *except* et *Message* contient le message d'erreur lié à l'exception. Il peut y avoir plusieurs instructions *except*, si on veut gérer différents types d'exception dans le même *try*.

L'instruction *finally* est facultative. Si elle est présente, les actions du *finally* sont **toujours exécutées qu'il y ait eu exception ou non**.

Nous reviendrons sur les exceptions un peu plus loin.

Les lignes 52-61 montrent diverses tentatives pour transformer (ou caster) une donnée de type *str*, *int*, *float*, *NoneType* en booléen. C'est toujours possible. Les règles sont les suivantes :

- **bool(int i)** vaut *False* si i vaut 0, *True* dans tous les autres cas ;
- **bool(float f)** vaut *False* si f vaut 0.0, *True* dans tous les autres cas ;
- **bool(str chaine)** vaut *False* si chaine a 0 caractère, *True* dans tous les autres cas ;
- **bool(None)** vaut False. **None** est une valeur spéciale qui signifie que la variable existe mais n'a pas de valeur.

Les résultats écran sont les suivants :

```
1. 4 <type 'int'>
2. 4 <type 'str'>
3. 4.0 <type 'float'>
4. True <type 'bool'>
5. True <type 'bool'>
6. 1 <type 'int'>
7. 1.0 <type 'float'>
8. True <type 'str'>
9. 4 <type 'str'>
10. 4 <type 'int'>
11. 4.0 <type 'float'>
12. True <type 'bool'>
13. 4.32 <type 'float'>
14. 4.32 <type 'str'>
15. 4 <type 'int'>
16. True <type 'bool'>
17. invalid literal for int() with base 10: 'abc'
18. True <type 'bool'>
19. False <type 'bool'>
20. False <type 'bool'>
21. None <type 'NoneType'>
22. False <type 'bool'>
23. False <type 'bool'>
```

IV-C - La portée des variables

Programme (bases_02)

```
1. # -*- coding=utf-8 -*-
2.
3. # portée des variables
4. def f1():
5.     # on utilise la variable globale i
6.     global i
7.     i+=1
8.     j=10
```


Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Programme (bases_02)

```
9.      print "f1[i,j]=[%s,%s]" % (i,j)
10.
11. def f2():
12.     # on utilise la variable globale i
13.     global i
14.     i+=1
15.     j=20
16.     print "f2[i,j]=[%s,%s]" % (i,j)
17.
18. def f3():
19.     # on utilise une variable locale i
20.     i=1
21.     j=30
22.     print "f3[i,j]=[%s,%s]" % (i,j)
23.
24. # tests
25. i=0
26. j=0 # ces deux variables ne sont connues d'une fonction f
27.     # que si celle-ci déclare explicitement par l'instruction global
28.     # qu'elle veut les utiliser
29. f1()
30. f2()
31. f3()
32. print "test[i,j]=[%s,%s]" % (i,j)
```

Résultats

```
1. f1[i,j]=[1,10]
2. f2[i,j]=[2,20]
3. f3[i,j]=[1,30]
4. test[i,j]=[2,0]
```

Note :

- le script montre l'utilisation de la variable i, déclarée globale dans les fonctions f1 et f2. Dans ce cas, le programme principal et les fonctions f1 et f2 partagent la même variable i.

IV-D - Les listes, tuples et dictionnaires

IV-D-1 - Listes à une dimension

Programme (bases_03)

```
1. # -*- coding=utf-8 -*-
2.
3. # listes à 1 dimension
4. # initialisation
5. list1=[0,1,2,3,4,5]
6.
7. # parcours - 1
8. print "list1 a %s elements" % (len(list1))
9. for i in range(len(list1)):
10.     print "list1[%s]=%s" % (i, list1[i])
11.
12. list1[1]=10;
13. # parcours - 2
14. print "list1 a %s elements" % (len(list1))
15. for element in list1:
16.     print element
17.
18. # ajout de deux éléments
19. list1[len(list1):]=[10,11]
20. print ("%s") % (list1)
21.
```


Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.
http://tahe.developpez.com/divers/python/exemples/

Programme (bases_03)

```
22. # suppression des deux derniers éléments
23. list1[len(list1)-2:]=[]
24. print ("%s") % (list1)
25.
26. # ajout en début de liste d'un tuple
27. list1[:0]=[-10, -11, -12]
28. print ("%s") % (list1)
29.
30. # insertion en milieu de liste de deux éléments
31. list1[3:3]=[100,101]
32. print ("%s") % (list1)
33.
34. # suppression de deux éléments en milieu de liste
35. list1[3:4]=[]
36. print ("%s") % (list1)
```

Notes :

- la notation tableau[**i:j**] désigne les éléments **i** à **j-1** du tableau ;
- la notation [**i:**] désigne les éléments **i** et **suivants** du tableau ;
- la notation [**:i**] désigne les éléments **0** à **i-1** du tableau ;
- ligne 20 : *print (%s) % (list1)* affiche la chaîne de caractères : "*[list1[0], list1[2], ..., list1[n-1]]*".

Résultats

```
1. list1 a 6 elements
2. list1[0]=0
3. list1[1]=1
4. list1[2]=2
5. list1[3]=3
6. list1[4]=4
7. list1[5]=5
8. list1 a 6 elements
9. 0
10. 10
11. 2
12. 3
13. 4
14. 5
15. [0, 10, 2, 3, 4, 5, 10, 11]
16. [0, 10, 2, 3, 4, 5]
17. [-10, -11, -12, 0, 10, 2, 3, 4, 5]
18. [-10, -11, -12, 100, 101, 0, 10, 2, 3, 4, 5]
19. [-10, -11, -12, 101, 0, 10, 2, 3, 4, 5]
```

Le code précédent peut être écrit différemment (bases_03b) en utilisant certaines méthodes des listes :

Programme (bases_03b)

```
1. # -*- coding=utf-8 -*-
2.
3. # listes à 1 dimension
4.
5. # initialisation
6. list1=[0,1,2,3,4,5]
7.
8. # parcours - 1
9. print "list1 a %s elements" % (len(list1))
10. for i in range(len(list1)):
11.     print "list1[%s]=%s" % (i, list1[i])
12.
13. list1[1]=10
14. # parcours - 2
15. print "list1 a %s elements" % (len(list1))
16. for element in list1:
17.     print element
18.
19. # ajout de deux éléments
20. list1.extend([10,11])
```


Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Programme (bases_03b)

```
21. print ("%s") % (list1)
22.
23. # suppression des deux derniers éléments
24. del list1[len(list1)-2:]
25. print ("%s") % (list1)
26.
27. # ajout en début de liste d'un tuple
28. for i in (-12, -11, -10):
29.     list1.insert(0,i)
30. print ("%s") % (list1)
31.
32. # insertion en milieu de liste
33. for i in (101,100):
34.     list1.insert(3,i)
35. print ("%s") % (list1)
36.
37. # suppression en milieu de liste
38. del list1[3:4]
39. print ("%s") % (list1)
```

Les résultats obtenus sont les mêmes qu'avec la version précédente.

IV-D-2 - Le dictionnaire

Le programme (bases_04)

```
1. # -*- coding=utf-8 -*-
2.
3. def existe(conjoints,mari):
4.     # vérifie si la clé mari existe dans le dictionnaire conjoints
5.     if(conjoints.has_key(mari)):
6.         print "La cle [%s] existe associee a la valeur [%s]" % (mari, conjoints[mari])
7.     else:
8.         print "La cle [%s] n'existe pas" % (mari)
9.
10.
11. # ----------------------------- Main
12. # dictionnaires
13. conjoints={"Pierre":"Gisele", "Paul":"Virginie", "Jacques":"Lucette","Jean":""}
14.
15. # parcours - 1
16. print "Nombre d'elements du dictionnaire : %s " % (len(conjoints))
17. for (cle,valeur) in conjoints.items():
18.     print "conjoints[%s]=%s" % (cle,valeur)
19.
20. # liste des clés du dictionnaire
21. print "liste des cles-------------"
22. cles=conjoints.keys()
23. print ("%s") % (cles)
24.
25. # liste des valeurs du dictionnaire
26. print "liste des valeurs------------"
27. valeurs=conjoints.values()
28. print ("%s") % (valeurs)
29.
30. # recherche d'une clé
31. existe(conjoints,"Jacques")
32. existe(conjoints,"Lucette")
33. existe(conjoints,"Jean")
34.
35. # suppression d'une clé-valeur
36. del (conjoints["Jean"])
37. print "Nombre d'elements du dictionnaire : %s " % (len(conjoints))
38. print ("%s") % (conjoints)
```

Notes :

Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

- ligne 17 : *conjoints.items()* rend la liste des couples (clé,valeur) du dictionnaire *conjoints ;*
- ligne 22 : *conjoints.keys()* rend les clés du dictionnaire *conjoints ;*
- ligne 27 : *conjoints.values()* rend les valeurs du dictionnaire *conjoints ;*
- ligne 5 : *conjoints.has_key(mari)* rend *True* si la clé *mari* existe dans le dictionnaire *conjoints*, *False* sinon ;
- ligne 38 : un dictionnaire peut être affiché en une seule ligne.

Les résultats

```
1. Nombre d'elements du dictionnaire : 4
2. conjoints[Paul]=Virginie
3. conjoints[Jean]=
4. conjoints[Pierre]=Gisele
5. conjoints[Jacques]=Lucette
6. liste des cles-------------
7. ['Paul', 'Jean', 'Pierre', 'Jacques']
8. liste des valeurs------------
9. ['Virginie', '', 'Gisele', 'Lucette']
10. La cle [Jacques] existe associee a la valeur [Lucette]
11. La cle [Lucette] n'existe pas
12. La cle [Jean] existe associee a la valeur []
13. Nombre d'elements du dictionnaire : 3
14. {'Paul': 'Virginie', 'Pierre': 'Gisele', 'Jacques': 'Lucette'}
```

IV-D-3 - Les tuples

Programme (bases_05)

```
1. # -*- coding=utf-8 -*-
2.
3. # tuples
4. # initialisation
5. tab1=(0,1,2,3,4,5)
6.
7. # parcours - 1
8. print "tab1 a %s elements" % (len(tab1))
9. for i in range(len(tab1)):
10.     print "tab1[%s]=%s" % (i, tab1[i])
11.
12. # parcours - 2
13. print "tab1 a %s elements" % (len(tab1))
14. for element in tab1:
15.     print element
16.
17. # modification d'un élément
18. tab1[0]=-1
```

Les résultats

```
1. tab1 a 6 elements
2. tab1[0]=0
3. tab1[1]=1
4. tab1[2]=2
5. tab1[3]=3
6. tab1[4]=4
7. tab1[5]=5
8. tab1 a 6 elements
9. 0
10. 1
11. 2
12. 3
13. 4
14. 5
15. Traceback (most recent call last):
16.   File "exemple_05.py", line 18, in <module>
17.     tab1[0]=-1
18. TypeError: 'tuple' object does not support item assignment
```

Notes :

Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

- lignes 15-18 des résultats : montre qu'un tuple ne peut pas être modifié.

IV-D-4 - Les listes à plusieurs dimensions

Programme (bases_06)

```
1. # -*- coding=utf-8 -*-
2.
3. # listes multidimensionnelles
4.
5. # initialisation
6. multi=[[0,1,2], [10,11,12,13], [20,21,22,23,24]]
7.
8. # parcours
9. for i1 in range(len(multi)):
10.     for i2 in range(len(multi[i1])):
11.         print "multi[%s][%s]=%s" % (i1,i2,multi[i1][i2])
12.
13. # dictionnaires multidimensionnels
14. # initialisation
15. multi={"zero":[0,1], "un":[10,11,12,13], "deux":[20,21,22,23,24]}
16.
17. # parcours
18. for (cle,valeur) in multi.items():
19.     for i2 in range(len(multi[cle])):
20.         print "multi[%s][%s]=%s" % (cle,i2,multi[cle][i2])
```

Résultats

```
1. multi[0][0]=0
2. multi[0][1]=1
3. multi[0][2]=2
4. multi[1][0]=10
5. multi[1][1]=11
6. multi[1][2]=12
7. multi[1][3]=13
8. multi[2][0]=20
9. multi[2][1]=21
10. multi[2][2]=22
11. multi[2][3]=23
12. multi[2][4]=24
13. multi[zero][0]=0
14. multi[zero][1]=1
15. multi[un][0]=10
16. multi[un][1]=11
17. multi[un][2]=12
18. multi[un][3]=13
19. multi[deux][0]=20
20. multi[deux][1]=21
21. multi[deux][2]=22
22. multi[deux][3]=23
23. multi[deux][4]=24
```

IV-D-5 - Liens entre chaînes et listes

Programme (bases_07)

```
1. # -*- coding=Utf-8 -*-
2.
3. # chaîne vers liste
4. chaine='1:2:3:4'
5. tab=chaine.split(':')
6. print type(tab)
7.
8. # affichage liste
9. print "tab a %s elements" % (len(tab))
10. print ("%s") % (tab)
```


Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Programme (bases_07)

```
11.
12. # liste vers chaîne
13. chaine2=":".join(tab)
14. print "chaine2=%s" % (chaine2)
15.
16. # ajoutons un champ vide
17. chaine+=":"
18. print "chaine=%s" % (chaine)
19. tab=chaine.split(":")
20.
21. # affichage liste
22. print "tab a %s elements" % (len(tab))
23. print ("%s") % (tab)
24.
25. # ajoutons de nouveau un champ vide
26. chaine+=":"
27. print "chaine=%s" % (chaine)
28. tab=chaine.split(":")
29.
30. # affichage liste
31. print "tab a %s elements" % (len(tab))
32. print ("%s") % (tab)
```

Notes :

- ligne 5 : la méthode *chaine.split(séparateur)* découpe la chaîne de caractères *chaine* en éléments séparés par *séparateur* et les rend sous forme de liste. Ainsi l'expression *'1:2:3:4'.split(":")* a pour valeur la liste *('1','2','3','4')* ;
- ligne 13 : *'separateur'.join(liste)* a pour valeur la chaîne de caractères '*liste[0]+separateur+liste[1]+separateur +...'*.

Résultats

```
1. <type 'list'>
2. tab a 4 elements
3. ['1', '2', '3', '4']
4. chaine2=1:2:3:4
5. chaine=1:2:3:4:
6. tab a 5 elements
7. ['1', '2', '3', '4', '']
8. chaine=1:2:3:4::
9. tab a 6 elements
10. ['1', '2', '3', '4', '', '']
```

IV-E - Les expressions régulières

Programme (bases_09)

```
1. # -*- coding=utf-8 -*-
2.
3. import re
4.
5. # --------------------------------------------------------------------------
6. def compare(modele,chaine):
7.     # compare la chaine chaîne au modèle modele
8.     # affichage résultats
9.     print "\nResultats(%s,%s)" % (chaine,modele)
10.     match=re.match(modele,chaine)
11.     if match:
12.         print match.groups()
13.     else:
14.         print "La chaine [%s] ne correspond pas au modele [%s]" % (chaine,modele)
15.
16.
17. # expressions régulières en python
18. # récupérer les différents champs d'une chaîne
```


Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Programme (bases_09)

```
19. # le modèle : une suite de chiffres entourée de caractères quelconques
20. # on ne veut récupérer que la suite de chiffres
21. modele=r"^.*?(\d+).*?$"
22.
23. # on confronte la chaîne au modèle
24. compare(modele,"xyz1234abcd")
25. compare(modele,"12 34")
26. compare(modele,"abcd")
27.
28. # le modèle : une suite de chiffres entourée de caractères quelconques
29. # on veut la suite de chiffres ainsi que les champs qui suivent et précèdent
30. modele=r"^(.*?)(\d+)(.*?)$"
31.
32. # on confronte la chaîne au modèle
33. compare(modele,"xyz1234abcd")
34. compare(modele,"12 34")
35. compare(modele,"abcd")
36.
37. # le modèle - une date au format jj/mm/aa
38. modele=r"^\s*(\d\d)\/(\d\d)\/(\d\d)\s*$"
39. compare(modele,"10/05/97")
40. compare(modele," 04/04/01 ")
41. compare(modele,"5/1/01")
42.
43. # le modèle - un nombre décimal
44. modele=r"^\s*([+|-]?)\s*(\d+\.\d*|\.\d+|\d+)\s*$"
45. compare(modele,"187.8")
46. compare(modele,"-0.6")
47. compare(modele,"4")
48. compare(modele,".6")
49. compare(modele,"4.")
50. compare(modele," + 4")
51. # fin
```

Notes :

- noter le module importé en ligne 3. C'est lui qui contient les fonctions de gestion des expressions régulières ;
- ligne 10 : la comparaison d'une chaîne à une expression régulière (modèle) rend le booléen *True* si la chaîne correspond au modèle, *False* sinon ;
- ligne 12 : *match.groups()* est un tuple dont les éléments sont les parties de la chaîne qui correspondent aux éléments de l'expression régulière entourés de parenthèses. Dans le modèle :
- *^.*?(\d+).*?*, *match.groups*() sera un tuple d'un élément ;
- *^(.*?)(\d+)(.*?)$*, *match.groups*() sera un tuple de trois éléments.

Résultats

```
1. Resultats(xyz1234abcd,^.*?(\d+).*?$)
2. ('1234',)
3.
4. Resultats(12 34,^.*?(\d+).*?$)
5. ('12',)
6.
7. Resultats(abcd,^.*?(\d+).*?$)
8. La chaine [abcd] ne correspond pas au modele [^.*?(\d+).*?$]
9.
10. Resultats(xyz1234abcd,^(.*?)(\d+)(.*?)$)
11. ('xyz', '1234', 'abcd')
12.
13. Resultats(12 34,^(.*?)(\d+)(.*?)$)
14. ('', '12', ' 34')
15.
16. Resultats(abcd,^(.*?)(\d+)(.*?)$)
17. La chaine [abcd] ne correspond pas au modele [^(.*?)(\d+)(.*?)$]
18.
19. Resultats(10/05/97,^\s*(\d\d)\/(\d\d)\/(\d\d)\s*$)
20. ('10', '05', '97')
21.
22. Resultats( 04/04/01 ,^\s*(\d\d)\/(\d\d)\/(\d\d)\s*$)
```


Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Résultats

```
23. ('04', '04', '01')
24.
25. Resultats(5/1/01,^\s*(\d\d)\/(\d\d)\/(\d\d)\s*$)
26. La chaine [5/1/01] ne correspond pas au modele [^\s*(\d\d)\/(\d\d)\/(\d\d)\s*$]
27.
28. Resultats(187.8,^\s*([+|-]?)\s*(\d+\.\d*|\.\d+|\d+)\s*$)
29. ('', '187.8')
30.
31. Resultats(-0.6,^\s*([+|-]?)\s*(\d+\.\d*|\.\d+|\d+)\s*$)
32. ('-', '0.6')
33.
34. Resultats(4,^\s*([+|-]?)\s*(\d+\.\d*|\.\d+|\d+)\s*$)
35. ('', '4')
36.
37. Resultats(.6,^\s*([+|-]?)\s*(\d+\.\d*|\.\d+|\d+)\s*$)
38. ('', '.6')
39.
40. Resultats(4.,^\s*([+|-]?)\s*(\d+\.\d*|\.\d+|\d+)\s*$)
41. ('', '4.')
42.
43. Resultats( + 4,^\s*([+|-]?)\s*(\d+\.\d*|\.\d+|\d+)\s*$)
44. ('+', '4')
```

IV-F - Mode de passage des paramètres des fonctions

Programme (bases_10)

```
1. # -*- coding=utf-8 -*-
2.
3. def f1(a):
4.     a=2
5.
6. def f2(a,b):
7.     a=2
8.     b=3
9.     return (a,b)
10.
11. # ------------------------ main
12. x=1
13. f1(x)
14. print "x=%s" % (x)
15. (x,y)=(-1,-1)
16. (x,y)=f2(x,y)
17. print "x=%s, y=%s" % (x,y)
```

Résultats

```
1. x=1
2. x=2, y=3
```

Notes :

- tout est objet en Python. Certains objets sont dits "immutable" : on ne peut pas les modifier. C'est le cas des nombres, des chaînes de caractères, des tuples. Lorsque des objets Python sont passés en paramètres à des fonctions, ce sont leurs références qui sont passées sauf si ces objets sont "immutable" auquel cas, c'est la valeur de l'objet qui est passée ;
- les fonctions *f1* (ligne 3), et *f2* (ligne 6) veulent illustrer le passage d'un paramètre de sortie. On veut que le paramètre effectif d'une fonction soit modifié par la fonction ;
- lignes 3-4 : la fonction *f1* modifie son paramètre formel a. On veut savoir si le paramètre effectif va lui aussi être modifié ;
- lignes 12-13 : le paramètre effectif est *x=1*. Le résultat de la ligne 1, montre que le paramètre effectif n'est pas modifié. Ainsi le paramètre effectif **x** et le paramètre formel **a** sont **deux objets différents ;**
- lignes 6-9 : la fonction *f2* modifie ses paramètres formels **a** et **b**, et les rend comme résultats ;

Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

- lignes 15-16 : on passe à *f2* les paramètres effectifs (x,y) et le résultat de *f2* est remis dans (x,y). La ligne 2 des résultats montre que les paramètres effectifs (x,y) ont été modifiés.

On en conclut donc que lorsque des objets "immutable" sont des paramètres de sortie, il faut qu'ils fassent partie des résultats renvoyés par la fonction.

IV-G - Les fichiers texte

Programme (bases_11)

```
1. # -*- coding=utf-8 -*-
2.
3. import sys
4.
5. # exploitation séquentielle d'un fichier texte
6. # celui-ci est un ensemble de lignes de la forme login:pwd:uid:gid:infos:dir:shell
7. # chaque ligne est mise dans un dictionnaire sous la forme login => [uid,gid,infos,dir,shell]
8.
9. # --------------------------------------------------------------------------
10. def afficheInfos(dico,cle):
11.     # affiche la valeur associée à cle dans le dictionnaire dico si elle existe
12.     valeur=None
13.     if dico.has_key(cle):
14.         valeur=dico[cle]
15.     if 'list' in str(type(valeur)):
16.         print "[{0},{1}]".format(cle,":".join(valeur))
17.     else:
18.         # cle n'est pas une clé du dictionnaire dico
19.         print "la cle [{0}] n'existe pas".format(cle)
20.
21.
22. def cutNewLineChar(ligne):
23.     # on supprime la marque de fin de ligne de ligne si elle existe
24.     l=len(ligne);
25.     while(ligne[l-1]=="\n" or ligne[l-1]=="\r"):
26.         l-=1
27.     return(ligne[0:l]);
28.
29. # on fixe le nom du fichier
30. INFOS="infos.txt"
31.
32. # on l'ouvre en création
33. try:
34.     fic=open(INFOS,"w")
35. except:
36.     print "Erreur d'ouverture du fichier INFOS en écriture\n"
37.     sys.exit()
38.
39. # on génère un contenu arbitraire
40. for i in range(1,101):
41.     ligne="login%s:pwd%s:uid%s:gid%s:infos%s:dir%s:shell%s" % (i,i,i,i,i,i,i)
42.     fic.write(ligne+"\n")
43.
44. # on ferme le fichier
45. fic.close()
46.
47. # on l'ouvre en lecture
48. try:
49.     fic=open(INFOS,"r")
50. except:
51.     print "Erreur d'ouverture du fichier INFOS en écriture\n"
52.     sys.exit()
53.
54. # dictionnaire vide au départ
55. dico={}
56.
57. # lecture ligne
58. ligne=fic.readline()
```


Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Programme (bases_11)

```
59. while(ligne!=''):
60.     # on enlève le caractère de fin de ligne
61.     ligne=cutNewLineChar(ligne)
62.
63.     # on met la ligne dans un tableau
64.     infos=ligne.split(":")
65.
66.     # on récupère le login
67.     login=infos[0]
68.
69.     # on enlève les deux premiers éléments [login,pwd]
70.     infos[0:2]=[]
71.
72.     # on crée une entrée dans le dictionnaire
73.     dico[login]=infos
74.
75.     # lecture ligne
76.     ligne=fic.readline()
77.
78. # on ferme le fichier
79. fic.close()
80.
81. # exploitation du dictionnaire
82. afficheInfos(dico,"login10")
83. afficheInfos(dico,"X")
```

Notes :

- ligne 37 : pour terminer le script au milieu du code.

Le fichier infos.txt

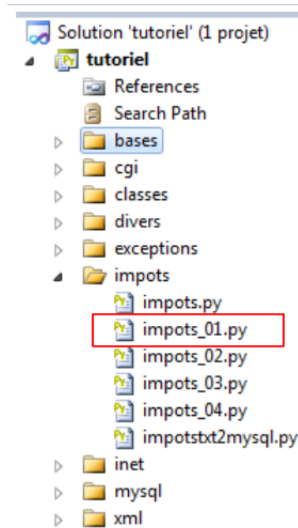
```
1. login0:pwd0:uid0:gid0:infos0:dir0:shell0
2. login1:pwd1:uid1:gid1:infos1:dir1:shell1
3. login2:pwd2:uid2:gid2:infos2:dir2:shell2
4. ...
5. login98:pwd98:uid98:gid98:infos98:dir98:shell98
6. login99:pwd99:uid99:gid99:infos99:dir99:shell99
```

Les résultats écran

```
1. [login10,uid10:gid10:infos10:dir10:shell10]
2. la cle [X] n'existe pas
```


Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

V - Exercice d'application - [IMPÔTS]



V-A - Le problème

On se propose d'écrire un programme permettant de calculer l'impôt d'un contribuable. On se place dans le cas simplifié d'un contribuable n'ayant que son seul salaire à déclarer :

- on calcule le nombre de parts du salarié **nbParts=nbEnfants/2 +1** s'il n'est pas marié, **nbEnfants/2+2** s'il est marié, où *nbEnfants* est son nombre d'enfants ;
- on calcule son revenu imposable **R=0.72*S** où S est son salaire annuel ;
- on calcule son coefficient familial **Q=R/N ;**
- on calcule son impôt I d'après les données suivantes

```
1. 12620.0 0 0
2. 13190 0.05 631
3. 15640 0.1 1290.5
4. 24740 0.15 2072.5
5. 31810 0.2 3309.5
6. 39970 0.25 4900
7. 48360 0.3 6898.5
8. 55790 0.35 9316.5
9. 92970 0.4 12106
10. 127860 0.45 16754.5
11. 151250 0.50 23147.5
12. 172040 0.55 30710
13. 195000 0.60 39312
14. 0 0.65 49062
```

Chaque ligne a trois champs. Pour calculer l'impôt I, on recherche la première ligne où *QF<=champ1*. Par exemple, si QF=30000 on trouvera la ligne :

24740 0.15 2072.5

L'impôt I est alors égal à **0.15*R - 2072.5*nbParts**. Si QF est tel que la relation *QF<=champ1* n'est jamais vérifiée, alors ce sont les coefficients de la dernière ligne qui sont utilisés. Ici :

0 0.65 49062

ce qui donne l'impôt I=0.65*R - 49062*nbParts.

Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

V-B - Version avec listes

Programme (impots_01)

```
1. # -*- coding=utf-8 -*-
2.
3. import math, sys
4.
5. def cutNewLineChar(ligne):
6.     # on supprime la marque de fin de ligne si elle existe
7.     l=len(ligne);
8.     while(ligne[l-1]=="\n" or ligne[l-1]=="\r"):
9.         l-=1
10.     return(ligne[0:l]);
11.
12.   # ------------------------------------------------------------------------
13. def calculImpots(marie,enfants,salaire,limites,coeffR,coeffN):
14.     # marié : oui, non
15.     # enfants : nombre d'enfants
16.     # salaire : salaire annuel
17.
18.     # nombre de parts
19.     marie=marie.lower()
20.     if(marie=="oui"):
21.         nbParts=float(enfants)/2+2
22.     else:
23.         nbParts=float(enfants)/2+1
24.     # une 1/2 part de plus si au moins 3 enfants
25.     if enfants>=3:
26.         nbParts+=0.5
27.     # revenu imposable
28.     revenuImposable=0.72*salaire
29.     # quotient familial
30.     quotient=revenuImposable/nbParts
31.     # est mis à la fin du tableau limites pour arrêter la boucle qui suit
32.     limites[len(limites)-1]=quotient
33.     # calcul de l'impôt
34.     i=0
35.     while(quotient>limites[i]):
36.         i=i+1
37.     # du fait qu'on a placé quotient à la fin du tableau limites, la boucle précédente
38.     # ne peut déborder du tableau limites
39.     # maintenant on peut calculer l'impôt
40.     return math.floor(revenuImposable*coeffR[i]-nbParts*coeffN[i])
41.
42.
43. # ------------------------------------------------ main
44. # définition des constantes
45. DATA="data.txt"
46. RESULTATS="resultats.txt"
47. limites=[12620,13190,15640,24740,31810,39970,48360,55790,92970,127860,151250,172040,195000,0]
48. coeffR=[0,0.05,0.1,0.15,0.2,0.25,0.3,0.35,0.4,0.45,0.5,0.55,0.6,0.65]
49. coeffN=[0,631,1290.5,2072.5,3309.5,4900,6898.5,9316.5,12106,16754.5,23147.5,30710,39312,49062]
50.
51. # lecture des données
52. try:
53.     data=open(DATA,"r")
54. except:
55.     print "Impossible d'ouvrir en lecture le fichier des donnees [DATA]"
56.     sys.exit()
57.
58. # ouverture fichier des résultats
59. try:
60.     resultats=open(RESULTATS,"w")
61. except:
62.     print "Impossible de creer le fichier des résultats [RESULTATS]"
63.     sys.exit()
64.
65. # on exploite la ligne courante du fichier des données
66. ligne=data.readline()
```


Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est
MCours.com

Programme (impots_01)

```
67. while(ligne != ''):
68.     # on enlève l'éventuelle marque de fin de ligne
69.     ligne=cutNewLineChar(ligne)
70.     # on récupère les 3 champs marié:enfants:salaire qui forment la ligne
71.     (marie,enfants,salaire)=ligne.split(",")
72.     enfants=int(enfants)
73.     salaire=int(salaire)
74.     # on calcule l'impôt
75.     impot=calculImpots(marie,enfants,salaire,limites,coeffR,coeffN)
76.     # on inscrit le résultat
77.     resultats.write("{0}:{1}:{2}:{3}\n".format(marie,enfants,salaire,impot))
78.     # on lit une nouvelle ligne
79.     ligne=data.readline()
80. # on ferme les fichiers
81. data.close()
82. resultats.close()
```

Résultats

Le fichier des données data.txt

```
1. oui,2,200000
2. non,2,200000
3. oui,3,200000
4. non,3,200000
5. oui,5,50000
6. non,0,3000000
```

Le fichier resultats.txt des résultats obtenus

```
1. oui:2:200000:22504.0
2. non:2:200000:33388.0
3. oui:3:200000:16400.0
4. non:3:200000:22504.0
5. oui:5:50000:0.0
6. non:0:3000000:1354938.0
```

V-C - Version avec fichiers texte

Dans l'exemple précédent, les données nécessaires au calcul de l'impôt avaient été trouvées dans trois listes. Elles seront désormais cherchées dans un fichier texte :

```
1. 12620:13190:15640:24740:31810:39970:48360:55790:92970:127860:151250:172040:195000:0
2. 0:0.05:0.1:0.15:0.2:0.25:0.3:0.35:0.4:0.45:0.5:0.55:0.6:0.65
3. 0:631:1290.5:2072.5:3309.5:4900:6898.5:9316.5:12106:16754.5:23147.5:30710:39312:49062
```

Programme (impots_02)

```
1. # -*- coding=utf-8 -*-
2.
3. import math,sys
4.
5. # -------------------------------------------------------------------------
6. def getTables(IMPOTS):
7.     # IMPOTS : le nom du fichier contenant les données des tables limites, coeffR, coeffN
8.     # le fichier IMPOTS existe-t-il ?
9.     try:
10.         data=open(IMPOTS,"r")
11.     except:
12.         return ("Le fichier IMPOTS n'existe pas",0,0,0)
13.     # création des 3 listes - on suppose que les lignes sont syntaxiquement correctes
14.     # -- ligne 1
15.     ligne=data.readline()
16.     if ligne== '':
17.         return ("La  ligne du fichier {0} est absente".format(IMPOTS),0,0,0)
18.     limites=cutNewLineChar(ligne).split(":")
19.     for i in range(len(limites)):
20.         limites[i]=int(limites[i])
```


Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Programme (impots_02)

```
21.     # -- ligne 2
22.     ligne=data.readline()
23.     if ligne== '':
24.         return ("La  ligne du fichier {0} est absente".format(IMPOTS),0,0,0)
25.     coeffR=cutNewLineChar(ligne).split(":")
26.     for i in range(len(coeffR)):
27.         coeffR[i]=float(coeffR[i])
28.     # -- ligne 3
29.     ligne=data.readline()
30.     if ligne== '':
31.         return ("La  ligne du fichier {0} est absente".format(IMPOTS),0,0,0)
32.     coeffN=cutNewLineChar(ligne).split(":")
33.     for i in range(len(coeffN)):
34.         coeffN[i]=float(coeffN[i])
35.     # fin
36.     return ("",limites,coeffR,coeffN)
37.
38. # -------------------------------------------------------------------------
39. def cutNewLineChar(ligne):
40.     # on supprime la marque de fin de ligne si elle existe
41.     l=len(ligne);
42.     while(ligne[l-1]=="\n" or ligne[l-1]=="\r"):
43.         l-=1
44.     return(ligne[0:l]);
45.
46.   # -------------------------------------------------------------------------
47. def calculImpots(marie,enfants,salaire,limites,coeffR,coeffN):
48.     # marié : oui, non
49.     # enfants : nombre d'enfants
50.     # salaire : salaire annuel
51.
52.     # nombre de parts
53.     marie=marie.lower()
54.     if(marie=="oui"):
55.         nbParts=float(enfants)/2+2
56.     else:
57.         nbParts=float(enfants)/2+1
58.     # une 1/2 part de plus si au moins 3 enfants
59.     if enfants>=3:
60.         nbParts+=0.5
61.     # revenu imposable
62.     revenuImposable=0.72*salaire
63.     # quotient familial
64.     quotient=revenuImposable/nbParts
65.     # est mis à la fin du tableau limites pour arrêter la boucle qui suit
66.     limites[len(limites)-1]=quotient
67.     # calcul de l'impôt
68.     i=0
69.     while(quotient>limites[i]):
70.         i=i+1
71.     # du fait qu'on a placé quotient à la fin du tableau limites, la boucle précédente
72.     # ne peut déborder du tableau limites
73.     # maintenant on peut calculer l'impôt
74.     return math.floor(revenuImposable*coeffR[i]-nbParts*coeffN[i])
75.
76.
77. # ------------------------------------------------ main
78. # définition des constantes
79. DATA="data.txt"
80. RESULTATS="resultats.txt"
81. IMPOTS="impots.txt"
82.
83. # les données nécessaires au calcul de l'impôt ont été placées dans le fichier IMPOTS
84. # à raison d'une ligne par tableau sous la forme
85. # val1:val2:val3...
86. (erreur,limites,coeffR,coeffN)=getTables(IMPOTS)
87.
88. # y a-t-il eu une erreur ?
89. if(erreur):
90.     print "{0}\n".format(erreur)
91.     sys.exit()
```


Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Programme (impots_02)

```
92.
93. # lecture des données
94. try:
95.     data=open(DATA,"r")
96. except:
97.     print "Impossible d'ouvrir en lecture le fichier des donnees [DATA]"
98.     sys.exit()
99.
100. # ouverture fichier des résultats
101. try:
102.     resultats=open(RESULTATS,"w")
103. except:
104.     print "Impossible de creer le fichier des résultats [RESULTATS]"
105.     sys.exit()
106.
107. # on exploite la ligne courante du fichier des données
108. ligne=data.readline()
109. while(ligne != ''):
110.     # on enlève l'éventuelle marque de fin de ligne
111.     ligne=cutNewLineChar(ligne)
112.     # on récupère les 3 champs marié:enfants:salaire qui forment la ligne
113.     (marie,enfants,salaire)=ligne.split(",")
114.     enfants=int(enfants)
115.     salaire=int(salaire)
116.     # on calcule l'impôt
117.     impot=calculImpots(marie,enfants,salaire,limites,coeffR,coeffN)
118.     # on inscrit le résultat
119.     resultats.write("{0}:{1}:{2}:{3}\n".format(marie,enfants,salaire,impot))
120.     # on lit une nouvelle ligne
121.     ligne=data.readline()
122. # on ferme les fichiers
123. data.close()
124. resultats.close()
```

Résultats

Les mêmes que précédemment.

Programme (impots_02b)

La méthode *getTables* ci-dessus (lignes 6-36) rend un tuple (*erreur*, *limites*, *coeffR*, *coeffN*) où *erreur* est un message d'erreur éventuellement vide. On peut vouloir gérer ces cas d'erreurs avec des exceptions. Dans ce cas :

- en cas d'erreur, la méthode *getTables* lance une exception ;
- sinon elle rend le tuple (*limites*, *coeffR*, *coeffN*).

Le code de la méthode *getTables* devient alors le suivant :

getTables

```
1. def getTables(IMPOTS):
2.     # IMPOTS : le nom du fichier contenant les données des tables limites, coeffR, coeffN
3.     # le fichier IMPOTS existe-t-il ? Si non alors l'exception IOError est lancée dans ce cas
4.     data=open(IMPOTS,"r")
5.
6.     # création des 3 listes - on suppose que les lignes sont syntaxiquement correctes
7.     # -- ligne 1
8.     ligne=data.readline()
9.     if ligne== '':
10.         raise RuntimeError ("La  ligne du fichier {0} est absente".format(IMPOTS))
11.     limites=cutNewLineChar(ligne).split(":")
12.     for i in range(len(limites)):
13.         limites[i]=int(limites[i])
14.     # -- ligne 2
15.     ligne=data.readline()
16.     if ligne== '':
```


Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

getTables

```
17.         raise RuntimeError ("La  ligne du fichier {0} est absente".format(IMPOTS))
18.     coeffR=cutNewLineChar(ligne).split(":")
19.     for i in range(len(coeffR)):
20.         coeffR[i]=float(coeffR[i])
21.     # -- ligne 3
22.     ligne=data.readline()
23.     if ligne== '':
24.         raise RuntimeError ("La  ligne du fichier {0} est absente".format(IMPOTS))
25.     coeffN=cutNewLineChar(ligne).split(":")
26.     for i in range(len(coeffN)):
27.         coeffN[i]=float(coeffN[i])
28.     # fin
29.     return (limites,coeffR,coeffN)
```

- ligne 4 : on ne teste pas si l'ouverture du fichier échoue. Si elle échoue, une exception *IOError* est lancée. On laisse celle-ci remonter au programme appelant ;
- lignes 9-10 : on lance une exception pour indiquer que la première ligne attendue est absente. On utilise une exception prédéfinie *RuntimeError*. On peut également créer ses propres classes d'exception. Nous le ferons un peu plus loin ;
- lignes 16-17 et 23-24 : on refait la même chose pour les lignes 2 et 3.

Le code du programme principal devient alors le suivant :

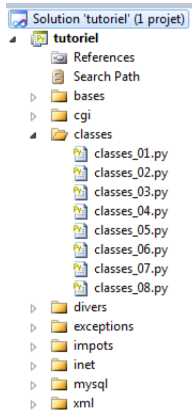
```
1. # ------------------------------------------------ main
2. # définition des constantes
3. DATA="data.txt"
4. RESULTATS="resultats.txt"
5. IMPOTS="impots.txt"
6.
7. # les données nécessaires au calcul de l'impôt ont été placées dans le fichier IMPOTS
8. # à raison d'une ligne par tableau sous la forme
9. # val1:val2:val3...
10.
11. erreur=False
12. try:
13.     (limites,coeffR,coeffN)=getTables(IMPOTS)
14. except IOError, message:
15.     erreur=True
16. except RuntimeError, message:
17.     erreur=True
18.
19. # y a-t-il eu une erreur ?
20. if(erreur):
21.     print "L'erreur suivante s'est produite : {0}\n".format(message)
22.     sys.exit()
23.
24. # lecture des données
25. ...
```

- lignes 12-17 : on appelle la méthode *getTables* et on gère les deux exceptions qu'elle peut lancer : *IOError* et *RuntimeError*. Dans cette gestion, on se contente de noter qu'il y a eu erreur ;
- lignes 20-22 : on affiche le message d'erreur de l'exception interceptée et on arrête le programme.

Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

VI - Les classes et objets

La **classe** est le moule à partir duquel sont fabriqués des **objets**. On dit de l'objet que c'est l'**instance** d'une classe.



VI-A - Une classe Objet

Programme (classes_01)

```
1. # -*- coding=utf-8 -*-
2.
3. class Objet:
4.     """une classe Objet vide"""
5.
6. # toute variable peut avoir des attributs par construction
7. obj1=Objet()
8. obj1.attr1="un"
9. obj1.attr2=100
10. # affiche l'objet
11. print "objet1=[{0},{1}]".format(obj1.attr1,obj1.attr2)
12. # modifie l'objet
13. obj1.attr2+=100
14. # affiche l'objet
15. print "objet1=[{0},{1}]".format(obj1.attr1,obj1.attr2)
16. # affecte la référence objet1 à objet2
17. obj2=obj1
18. # modifie obj2
19. obj2.attr2=0
20. # affiche les deux objets
21. print "objet1=[{0},{1}]".format(obj1.attr1,obj1.attr2)
22. print "objet2=[{0},{1}]".format(obj2.attr1,obj2.attr2)
```

Notes :

- ligne 4 : une autre forme de commentaire. Celui-ci précédé de trois " peut alors s'étaler sur plusieurs lignes ;
- lignes 3-4 : une classe objet vide ;
- ligne 7 : instanciation de la classe *Objet* ;
- ligne 17 : copie de références. Les variables *obj1* et *obj2* sont deux pointeurs (références) sur un même objet.

Résultats

```
1. objet1=[un,100]
2. objet1=[un,200]
3. objet1=[un,0]
4. objet2=[un,0]
```


Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

VI-B - Une classe Personne

Programme (classes_02)

```
1. # -*- coding=utf-8 -*-
2.
3. class Personne:
4.     # attributs de la classe
5.     # non déclarés - peuvent être créés dynamiquement
6.
7.     # méthode
8.     def identite(self):
9.         # a priori, utilise des attributs inexistants
10.         return "[{0},{1},{2}]".format(self.prenom,self.nom,self.age)
11.
12. # ---------------------------------- main
13. # les attributs sont publics et peuvent être créés dynamiquement
14. p=Personne()
15. p.prenom="Paul"
16. p.nom="Langevin"
17. p.age=48
18. # appel d'une méthode
19. print "personne={0}\n".format(p.identite())
```

Notes :

- lignes 3-10 : une classe avec une méthode ;
- ligne 8 : toute méthode d'une classe doit avoir pour premier paramètre, l'objet *self* qui désigne l'objet courant.

Résultats

```
1. personne=[Paul,Langevin,48]
```

VI-C - La classe Personne avec un constructeur

Programme (classes_03)

```
1. # -*- coding=utf-8 -*-
2.
3. class Personne:
4.
5.     # constructeur
6.     def __init__(self,prenom,nom,age):
7.         self.prenom=prenom;
8.         self.nom=nom;
9.         self.age=age;
10.
11.      # méthode
12.      def identite(self):
13.          # a priori, utilise des attributs inexistants
14.          return "[{0},{1},{2}]".format(self.prenom,self.nom,self.age)
15.
16. # ---------------------------------- main
17. # un objet Personne
18. p=Personne("Paul","Langevin",48)
19. # appel d'une méthode
20. print "personne={0}\n".format(p.identite())
```

Notes :

- ligne 6 : le constructeur d'une classe s'appelle **__init__**. Comme pour les autres méthodes, son premier paramètre est **self** ;
- ligne 18 : un objet *Personne* est construit avec le constructeur de la classe.

Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Résultats

```
1. personne=[Paul,Langevin,48]
```

VI-D - La classe Personne avec contrôles de validité dans le constructeur

Programme (classes_04)

```
1. # -*- coding=utf-8 -*-
2.
3. import re
4.
5. # -------------------
6. # une classe d'exceptions propriétaire
7. class MyError(Exception):
8.     pass
9.
10. class Personne:
11.
12.     # constructeur
13.     def __init__(self,prenom="x",nom="x",age=0):
14.          # le prénom doit être non vide
15.         match=re.match(r"^\s*(\S+)\s*$",prenom)
16.         if not match:
17.             raise MyError("Le prenom ne peut etre vide")
18.         else:
19.             self.prenom=match.groups()[0]
20.         # le nom doit être non vide
21.         match=re.match(r"^\s*(\S+)\s*$",nom)
22.         if not match:
23.             raise MyError("Le nom ne peut etre vide")
24.         else:
25.             self.nom=match.groups()[0]
26.         # l'âge doit être valide
27.         match=re.match(r"^\s*(\d+)\s*$",str(age))
28.         if not match:
29.             raise MyError("l'age doit etre un entier positif")
30.         else:
31.             self.age=match.groups()[0]
32.
33.     def __str__(self):
34.         return "[{0},{1},{2}]".format(self.prenom,self.nom,self.age)
35.
36. # ---------------------------------- main
37. # un objet Personne
38. try:
39.     p=Personne("  Paul  ","  Langevin  ",48)
40.     print "personne={0}".format(p)
41. except MyError, erreur:
42.     print erreur
43. # un autre objet Personne
44. try:
45.     p=Personne("  xx  ","  yy  ","  zz")
46.     print "personne={0}".format(p)
47. except MyError, erreur:
48.     print erreur
49. # une autre personne
50. try:
51.     p=Personne()
52.     print "personne={0}".format(p)
53. except MyError, erreur:
54.     print erreur
```

Notes :

- lignes 7-8 : une classe *MyError* dérivée de la classe *Exception*. Elle n'ajoute aucune fonctionnalité à cette dernière. Elle n'est là que pour avoir une exception propriétaire ;

Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.
http://tahe.developpez.com/divers/python/exemples/

- ligne 13 : le constructeur a des valeurs par défaut pour ses paramètres. Ainsi l'opération *p=Personne()* est équivalente à *p=Personne("x","x",0) ;*
- lignes 15-9 : on analyse le paramètre *prenom*. Il doit être non vide. Si ce n'est pas le cas, on lance l'exception *MyError* avec un message d'erreur ;
- lignes 21-25 : idem pour le nom ;
- lignes 27-31 : vérification de l'âge ;
- lignes 33-34 : la fonction *__str__* remplace la méthode qui s'appelait *identite* précédemment ;
- lignes 38-42 : instanciation d'une personne puis affichage de son identité ;
- ligne 39 : instanciation ;
- ligne 40 : affichage. L'opération demande d'afficher la personne *p* sous la forme d'une chaîne de caractères. L'interpréteur Python appelle alors automatiquement la méthode *p.__str__()* si elle existe. Cette méthode joue le même rôle que la méthode *toString()* en Java ou dans les langages .NET ;
- lignes 41-42 : gestion d'une éventuelle exception de type *MyError*. Affiche alors le message d'erreur associé à l'exception ;
- lignes 44-48 : idem pour une deuxième personne instanciée avec des paramètres erronés ;
- lignes 50-54 : idem pour une troisième personne instanciée avec les paramètres par défaut.

Résultats

```
1. personne=[Paul,Langevin,48]
2. l'age doit etre un entier positif
3. personne=[x,x,0]
```

VI-E - Ajout d'une méthode faisant office de second constructeur

Programme (classes_05)

```
1. # -*- coding=utf-8 -*-
2.
3. import re
4.
5. # -------------------
6. # une classe d'exceptions propriétaire
7. class MyError(Exception):
8.     pass
9.
10. class Personne:
11.     # attributs de la classe
12.     # non déclarés - peuvent être créés dynamiquement
13.
14.     # constructeur
15.     def __init__(self,prenom="x",nom="x",age=0):
16.          # le prénom doit être non vide
17.         match=re.match(r"^\s*(\S+)\s*$",prenom)
18.         if not match:
19.             raise MyError("Le prenom ne peut etre vide")
20.         else:
21.             self.prenom=match.groups()[0]
22.         # le nom doit être non vide
23.         match=re.match(r"^\s*(\S+)\s*$",nom)
24.         if not match:
25.             raise MyError("Le nom ne peut etre vide")
26.         else:
27.             self.nom=match.groups()[0]
28.             # l'âge doit être valide
29.         match=re.match(r"^\s*(\d+)\s*$",str(age))
30.         if not match:
31.             raise MyError("l'age doit etre un entier positif")
32.         else:
33.             self.age=match.groups()[0]
34.
35.     def initWithPersonne(self,p):
36.         # initialise l'objet courant avec une personne p
37.         self.__init__(p.prenom,p.nom,p.age)
38.
39.     def __str__(self):
```


Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Programme (classes_05)

```
40.         return "[{0},{1},{2}]".format(self.prenom,self.nom,self.age)
41.
42. # ---------------------------------- main
43. # un objet Personne
44. try:
45.     p0=Personne("  Paul  ","  Langevin  ",48)
46.     print "personne={0}".format(p0)
47. except MyError, erreur:
48.     print erreur
49. # un autre objet Personne
50. try:
51.     p1=Personne("  xx  ","  yy  ","  zz")
52.     print "personne={0}".format(p1)
53. except MyError, erreur:
54.     print erreur
55. # une autre personne
56. p2=Personne()
57. try:
58.     p2.initWithPersonne(p0)
59.     print "personne={0}".format(p2)
60. except MyError, erreur:
61.     print erreur
```

Notes :

- la différence avec le script précédent se situe au niveau des lignes 35-37. On a rajouté la méthode *initWithPersonne*. Celle-ci fait appel au constructeur *__init__*. Il n'y a pas possibilité d'avoir, comme dans les langages typés, des méthodes de même nom différenciées par la nature de leurs paramètres ou de leur résultat. Il n'y a donc pas possibilité d'avoir plusieurs constructeurs qui construiraient l'objet à partir de paramètres différents, ici un objet de type *Personne*.

Résultats

```
1. personne=[Paul,Langevin,48]
2. l'age doit etre un entier positif
3. personne=[Paul,Langevin,48]
```

VI-F - Une liste d'objets Personne

Programme (classes_06)

```
1. # -*- coding=utf-8 -*-
2.
3. class Personne:
4.     ...
5.
6. # ---------------------------------- main
7. # création d'une liste d'objets personne
8. groupe=[Personne("Paul","Langevin",48), Personne("Sylvie","Lefur",70)]
9. # identité de ces personnes
10. for i in range(len(groupe)):
11.   print "groupe[{0}]={1}".format(i,groupe[i])
```

Notes :

- lignes 3-5 : la classe *Personne* déjà décrite

Résultats

```
1. groupe[0]=[Paul,Langevin,48]
2. groupe[1]=[Sylvie,Lefur,70]
```


Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

VI-G - Création d'une classe dérivée de la classe Personne

Programme (classes_07)

```
1. # -*- coding=utf-8 -*-
2.
3. import re
4.
5. class Personne:
6.     ...
7.
8. class Enseignant(Personne):
9.     def __init__(self,prenom="x",nom="x",age=0,discipline="x"):
10.         Personne.__init__(self,prenom,nom,age)
11.         self.discipline=discipline
12.
13.     def __str__(self):
14.         return "[{0},{1},{2},{3}]".format(self.prenom,self.nom,self.age,self.discipline)
15.
16.
17. # ---------------------------------- main
18. # création d'une liste d'objets Personne et dérivés
19. groupe=[Enseignant("Paul","Langevin",48,"anglais"), Personne("Sylvie","Lefur",70)]
20. # identité de ces personnes
21. for i in range(len(groupe)):
22.   print "groupe[{0}]={1}".format(i,groupe[i])
```

Notes :

- lignes 5-6 : la classe *Personne* déjà définie ;
- ligne 8 : déclare la classe *Enseignant* comme étant une classe dérivée de la classe *Personne ;*
- ligne 10 : le constructeur de la classe dérivée *Enseignant* doit appeler le constructeur de la classe parent *Personne* ;
- lignes 21-22 : pour afficher groupe[i], l'interpréteur utilise la méthode *groupe[i].__str__()*. La méthode *__str__()* utilisée est celle de la classe réelle de *groupe[i]* comme le montrent les résultats ci-dessous.

Résultats

```
1. groupe[0]=[Paul,Langevin,48,anglais]
2. groupe[1]=[Sylvie,Lefur,70]
```

VI-H - Création d'une seconde classe dérivée de la classe Personne

Programme (classes_08)

```
1. # -*- coding=utf-8 -*-
2.
3. import re
4.
5. class Personne:
6.     ...
7.
8. class Enseignant(Personne):
9.     ...
10.
11. class Etudiant(Personne):
12.     def __init__(self,prenom="x",nom="x",age=0,formation="x"):
13.         Personne.__init__(self,prenom,nom,age)
14.         self.formation=formation
15.
16.     def __str__(self):
17.         return "[{0},{1},{2},{3}]".format(self.prenom,self.nom,self.age,self.formation)
18.
19. # ---------------------------------- main
20. # création d'une liste d'objets Personne et dérivés
```


Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Programme (classes_08)

```
21. groupe=[Enseignant("Paul","Langevin",48,"anglais"), Personne("Sylvie","Lefur",70),
 Etudiant("Steve","Boer",22,"iup2 qualite")]
22. # identité de ces personnes
23. for i in range(len(groupe)):
24.   print "groupe[{0}]={1}".format(i,groupe[i])
```

Notes :

- ce script est analogue au précédent.

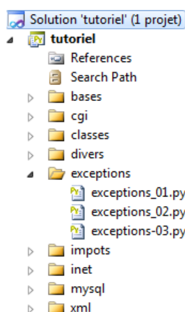
Résultats

```
1. groupe[0]=[Paul,Langevin,48,anglais]
2. groupe[1]=[Sylvie,Lefur,70]
3. groupe[2]=[Steve,Boer,22,iup2 qualite]
```


Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

VII - Les exceptions

Nous nous intéressons maintenant aux exceptions.



Le premier script illustre la nécessité de gérer les exceptions.

Programme (exceptions_01)

```
1. # -*- coding=utf-8 -*-
2.
3. # on provoque une erreur
4. x=4/0
```

On provoque volontairement une erreur pour voir les informations produites par l'interpréteur. Le résultat écran est le suivant :

```
1. Traceback (most recent call last):
2.   File "D:\data\istia-1112\python\tutoriel\exceptions_01.py", line 4, in <module>
3.     x=4/0
4. ZeroDivisionError: integer division or modulo by zero
```

La ligne 4 nous donne :

- le type de l'exception : *ZeroDivisionError ;*
- le message d'erreur associé : *integer division or modulo by zero*. Il est en anglais. C'est quelque chose qu'on peut vouloir changer.

La règle essentielle en programmation est qu'on doit tout faire pour éviter les plantages " sauvages " comme celui ci-dessus. Même en cas d'erreur, le programme doit se terminer proprement en donnant des informations sur l'erreur qui s'est produite.

La syntaxe d'une gestion d'exceptions est la suivante :

```
1. try:
2.     actions susceptibles de lancer une exception
3. except [classe d'exceptions, ...]:
4.     actions de gestion de l'exception
5. finally:
6.    actions toujours exécutées qu'il y ait exception ou non
```

Dans le **try**, l'exécution des actions s'arrête dès qu'une exception survient. Dans ce cas, l'exécution se poursuit avec les actions de la clause **except**.

La clause

except [MyException, ...]:

Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

intercepte les exceptions de type **MyException** ou **dérivé**. Lorsque dans un **try** se produit une exception, l'interpréteur examine les clauses **except** associées au **try** dans l'ordre où elles ont été écrites. Il s'arrête sur la première clause **except** permettant de traiter l'exception qui s'est produite. S'il n'en trouve aucune, l'exception remonte à la méthode appelante. Si celle-ci a un **try** / **except**, l'exception est de nouveau gérée sinon elle continue à remonter la chaîne des méthodes appelées. En dernier ressort, elle arrive à l'interpréteur Python. Celui-ci arrête alors le programme exécuté et affiche un message d'erreur du type montré dans l'exemple précédent. La règle est donc que le programme principal doit arrêter toutes les exceptions qui peuvent remonter des méthodes appelées.

Une exception transporte avec elle des informations sur l'erreur qui s'est produite. On peut les obtenir avec la syntaxe suivante :

except MyException **as** informations:

informations est un tuple qui transporte les informations liées à l'exception.

La syntaxe

except MyException, erreur:

affecte à la variable *erreur*, le message d'erreur lié à l'exception.

Pour lancer une exception, on utilise la syntaxe

raise MyException(param1, param2, ...)

où le plus souvent *MyException* est une classe dérivée de la classe *Exception*. Les paramètres passés au constructeur de la classe seront disponibles à la clause **except** des structures d'interception des exceptions.

Ces concepts sont illustrés par le script suivant qui gère explicitement les erreurs :

Programme (exceptions_02)

```
1. # -*- coding=utf-8 -*-
2.
3. i=0
4. # on provoque une erreur et on la gère
5. x=2
6. try:
7.     x=4/0
8. except ZeroDivisionError, erreur:
9.     print ("%s : %s ") % (i, erreur)
10. # la valeur de x n'a pas changé
11. print "x=%s" % (x)
12.
13. # on recommence
14. i+=1
15. try:
16.     x=4/0
17. except Exception, erreur:
18.     # on intercepte l'exception la plus générale
19.     print ("%s : %s ") % (i, erreur)
20.
21. # on peut intercepter plusieurs types d'exceptions
22. i+=1
23. try:
24.     x=4/0
25. except ValueError, erreur:
26.     # cette exception ne se produit pas ici
27.     print ("%s : %s ") % (i, erreur)
28. except Exception, erreur:
29.     # on intercepte l'exception la plus générale
30.     print ("%s : (Exception) %s ") % (i, erreur)
31. except ZeroDivisionError, erreur:
```


Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Programme (exceptions_02)

```
32.     # on intercepte un type précis
33.     print ("%s : (ZeroDivisionError) %s ") % (i, erreur)
34.
35. # on recommence en changeant l'ordre
36. i+=1
37. try:
38.     x=4/0
39. except ValueError, erreur:
40.     # cette exception ne se produit pas ici
41.     print ("%s : %s ") % (i, erreur)
42. except ZeroDivisionError, erreur:
43.     # on intercepte un type précis
44.     print ("%s : (ZeroDivisionError) %s ") % (i, erreur)
45. except Exception, erreur:
46.     # on intercepte l'exception la plus générale
47.     print ("%s : (Exception) %s ") % (i, erreur)
48.
49. # une clause except sans argument
50. i+=1
51. try:
52.     x=4/0
53. except:
54.     # on ne s'intéresse pas à la nature de l'exception ni au msg d'erreur
55.     print ("%s : il y a eu un probleme ") % (i)
56.
57. # un autre type d'exception
58. i+=1
59. try:
60.     x=int("x")
61. except ValueError, erreur:
62.     print ("%s : %s ") % (i, erreur)
63.
64. # une exception transporte des informations dans un tuple accessible au programme
65. i+=1
66. try:
67.     x=int("x")
68. except ValueError as infos:
69.     print ("%s : %s ") % (i, infos)
70.
71. # on peut lancer des exceptions
72. i+=1
73. try:
74.     raise ValueError("param1","param2","param3")
75. except ValueError as infos:
76.     print ("%s : %s ") % (i, infos)
77.
78. # on peut créer ses propres exceptions
79. class MyError(Exception):
80.     pass
81.
82. # on lance l'exception MyError
83. i+=1
84. try:
85.     raise MyError("info1","info2", "info3")
86. except MyError as infos:
87.     print ("%s : %s ") % (i, infos)
88.
89. # on lance l'exception MyError
90. i+=1
91. try:
92.     raise MyError("mon msg d'erreur")
93. except MyError, erreur:
94.     print ("%s : %s ") % (i, erreur)
95.
96. # on peut lancer n'importe quel type d'objet
97. class Objet:
98.     def __init__(self,msg):
99.         self.msg=msg
100.     def __str__(self):
101.         return self.msg
102.
```


Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Programme (exceptions_02)

```
103. i+=1
104. try:
105.     raise Objet("pb...")
106. except Objet as erreur:
107.     print "%s : %s" % (i, erreur)
108.
109. # la clause finally est toujours exécutée
110. # qu'il y ait exception ou non
111. i+=1
112. x=None
113. try:
114.     x=1
115. except:
116.     print "%s : exception" % (i)
117. finally:
118.     print "%s : finally x=%s" % (i,x)
119.
120. i+=1
121. x=None
122. try:
123.     x=2/0
124. except:
125.     print "%s : exception" % (i)
126. finally:
127.     print "%s : finally x=%s" % (i,x)
```

Notes :

- lignes 6-9 : on gère une division par zéro ;
- ligne 8 : on intercepte l'exception exacte qui se produit ;
- ligne 11 : à cause de l'exception qui s'est produite, *x* n'a pas reçu de valeur et n'a donc pas changé de valeur ;
- lignes 15-19 : on refait la même chose mais en interceptant une exception de plus haut niveau de type *Exception*. Comme l'exception *ZeroDivisionError* dérive de la classe *Exception*, la clause *except* l'arrêtera ;
- lignes 23-33 : on met plusieurs clauses *except* pour gérer plusieurs types d'exception. Une seule clause *except* sera exécutée ou aucune si l'exception ne vérifie aucune clause *except ;*
- lignes 51-55 : la clause *except* peut n'avoir aucun argument. Dans ce cas, elle arrête toutes les exceptions ;
- lignes 59-62 : introduisent l'exception *ValueError ;*
- lignes 66-69 : on récupère les informations transportées par l'exception ;
- lignes 73-76 : introduisent la façon de lancer une exception ;
- lignes 79-84 : illustrent l'utilisation d'une classe d'exception propriétaire *MyError ;*
- lignes 79-80 : la classe *MyError* se contente de dériver la classe de base *Exception*. Elle n'ajoute rien à sa classe de base. Mais maintenant, elle peut être nommée explicitement dans les clauses *except ;*
- lignes 97-107 : illustrent qu'en Python on peut en fait lancer n'importe quel type d'objet et pas simplement des objets dérivés de la classe *Exception ;*
- lignes 109-127 : illustrent l'utilisation de la clause *finally*.

Les résultats écran sont les suivants :

```
1. 0 : integer division or modulo by zero
2. x=2
3. 1 : integer division or modulo by zero
4. 2 : (Exception) integer division or modulo by zero
5. 3 : (ZeroDivisionError) integer division or modulo by zero
6. 4 : il y a eu un probleme
7. 5 : invalid literal for int() with base 10: 'x'
8. 6 : invalid literal for int() with base 10: 'x'
9. 7 : ('param1', 'param2', 'param3')
10. 8 : ('info1', 'info2', 'info3')
11. 9 : mon msg d'erreur
12. 10 : pb...
13. 11 : finally x=1
14. 12 : exception
15. 12 : finally x=None
```


Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Ce nouveau script illustre la remontée des exceptions dans la chaîne des méthodes appelantes :

Programme (exceptions_03)

```
1. # -*- coding=utf-8 -*-
2.
3. # une exception propriétaire
4. class MyError(Exception):
5.     pass
6.
7. # trois méthodes
8. def f1(x):
9.     # on ne gère pas les exceptions - elles remontent automatiquement
10.     return f2(x)
11.
12. def f2(y):
13.     # on ne gère pas les exceptions - elles remontent automatiquement
14.     return f3(y)
15.
16. def f3(z):
17.     if (z % 2) == 0:
18.         # si z est pair, on lance une exception
19.         raise MyError("exception dans f3")
20.     else:
21.         return 2*z
22.
23. #---------- main
24.
25. # les exceptions remontent la chaîne des méthodes appelées
26. # jusqu'à ce qu'une méthode l'intercepte. Ici ce sera main
27. try:
28.     print f1(4)
29. except MyError as infos:
30.     print "type :%s, arguments : %s" % (type(infos),infos)
31.
32. # une méthode peut enrichir les exceptions qu'elle remonte
33. def f4(x):
34.     try:
35.         return f5(x)
36.     except MyError as infos:
37.         # on enrichit l'exception puis on la relance
38.         raise MyError(infos,"exception dans f4")
39.
40. def f5(y):
41.     try:
42.         return f6(y)
43.     except MyError as infos:
44.         # on enrichit l'exception puis on la relance
45.         raise MyError(infos, "exception dans f5")
46.
47. def f6(z):
48.     if (z % 2) == 0:
49.         # on lance une exception
50.         raise MyError("exception dans f6")
51.     else:
52.         return 2*z
53.
54. #---------- main
55. try:
56.     print f4(4)
57. except MyError as infos:
58.     print "type :%s, arguments : %s" % (type(infos),infos)
```

Notes :

- lignes 27-30, dans l'appel *main --> f1 --> f2 --> f3* (ligne 28), l'exception *MyError* lancée par *f3* va remonter jusqu'à *main*. Elle sera alors traitée par la clause *except* de la ligne 29 ;
- lignes 55-58 : dans l'appel *main --> f4 --> f5 --> f6* (ligne 56), l'exception lancée *MyError* par *f6* va remonter jusqu'à *main*. Elle sera alors traitée par la clause *except* de la ligne 29. Cette fois-ci, dans sa remontée de la

Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

chaîne des méthodes appelantes, l'exception *MyError* est enrichie par des informations ajoutées par chaque méthode remontée.

Les résultats écran sont les suivants :

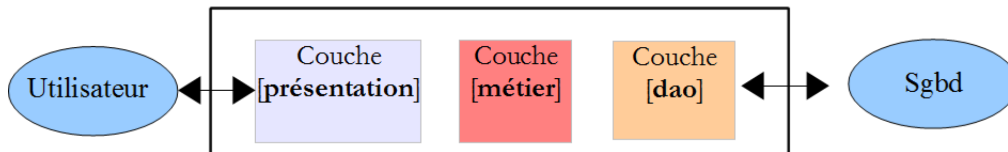
```
1. type :<class '__main__.MyError'>, arguments : exception dans f3
2. type :<class '__main__.MyError'>, arguments : (MyError(MyError('exception dans f6',), 'exception
 dans f5'), 'exception dans f4')
```


Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

VIII - Architecture en couches et programmation par interfaces

VIII-A - Introduction

Nous nous proposons d'écrire une application permettant l'affichage des notes des élèves d'un collège. Cette application aura une architecture multicouche :

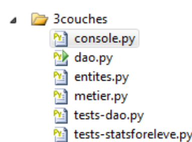


- la couche [**présentation**] est la couche en contact avec l'utilisateur de l'application ;
- la couche [**métier**] implémente les règles de gestion de l'application telles que le calcul d'un salaire ou d'une facture. Cette couche utilise des données provenant de l'utilisateur via la couche [présentation] et du SGBD via la couche [dao]- ;
- la couche [DAO] (**D**ata **A**ccess **O**bjects) gère l'accès aux données du SGBD ;

Nous allons illustrer cette architecture avec une application console simple :

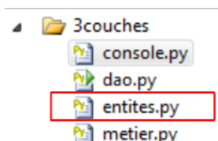
- il n'y aura pas de base de données ;
- la couche [DAO] gèrera des entités *Eleve*, *Classe*, *Matiere*, *Note* permettant de gérer les notes des élèves ;
- la couche [métier] permettra de calculer des indicateurs sur les notes d'un élève précis ;
- la couche [présentation] sera une application console qui affichera les résultats calculés par la couche [métier].

Le projet Visual Studio de l'application est le suivant :



VIII-B - Les entités de l'application

Les entités sont des objets. Nous aurons ici quatre classes pour chacune des entités *Eleve*, *Classe*, *Matiere*, *Note*.



Le fichier [entites.py] regroupe quatre classes.

La classe [Classe] représente une classe du collège :

Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Classe

```
1. class Classe:
2.     # constructeur
3.     def __init__(self,id,nom):
4.         # on mémorise les paramètres
5.         self.id=id
6.         self.nom=nom
7. 
8.     # toString
9.     def __str__(self):
10.         return "Classe[{0},{1}]".format(self.id,self.nom)
```

- lignes 3-6 : une classe est définie par un n° *id* (ligne 5) et un *nom* (ligne 6) ;
- lignes 9-10 : la méthode d'affichage de la classe.

La classe [Matiere] est la suivante :

Matière

```
1. class Matiere:
2.     # constructeur
3.     def __init__(self,id,nom,coefficient):
4.         # on mémorise les paramètres
5.         self.id=id
6.         self.nom=nom
7.         self.coefficient=coefficient
8. 
9.     # toString
10.     def __str__(self):
11.         return "Matiere[{0},{1},{2}]".format(self.id,self.nom,self.coefficient)
```

- lignes 3-7 : une matière est définie par son n° (ligne 5), son nom (ligne 6), son coefficient (ligne 7) ;
- lignes 10-11 : la méthode d'affichage de la matière.

La classe [Eleve] est la suivante :

Eleve

```
1. class Eleve:
2.     # constructeur
3.     def __init__(self,id,nom,prenom,classe):
4.         # on mémorise les paramètres
5.         self.id=id
6.         self.nom=nom
7.         self.prenom=prenom
8.         self.classe=classe
9. 
10.     # toString
11.     def __str__(self):
12.         return "Eleve[{0},{1},{2},{3}]".format(self.id,self.prenom,self.nom,self.classe)
```

- lignes 3-8 : un élève est caractérisé par son n° (ligne 5), son nom (ligne 6), son prénom (ligne 7), sa classe (ligne 8). Ce dernier paramètre est une référence sur un objet [Classe] ;
- lignes 11-12 : la méthode d'affichage de l'élève.

La classe [Note] est la suivante :

Note

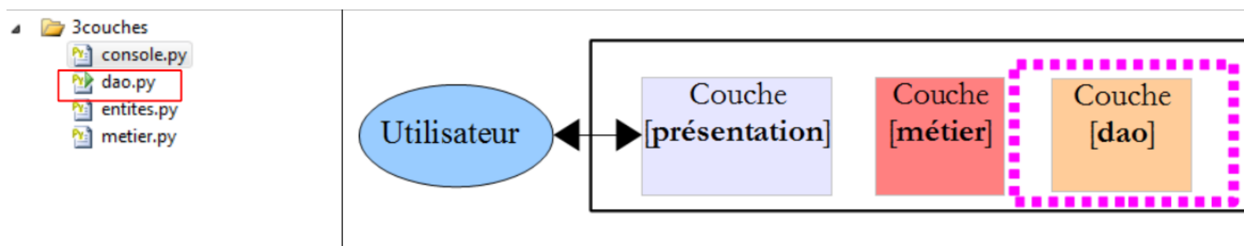
```
1. class Note:
2.     # constructeur
3.     def __init__(self,id,valeur,eleve,matiere):
4.         # on mémorise les paramètres
5.         self.id=id
6.         self.valeur=valeur
7.         self.eleve=eleve
8.         self.matiere=matiere
```


Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Note
```
9.
10.     # toString
11.     def __str__(self):
12.         return "Note[{0},{1},{2},{3}]".format(self.id,self.valeur,self.eleve,self.matiere)
```

- lignes 3-8 : un objet [Note] est caractérisé par son n° (ligne 5), la valeur de la note (ligne 6), une référence sur l'élève qui a cette note (ligne 7), une référence sur la matière objet de la note (ligne 8) ;
- lignes 11-12 : la méthode d'affichage de l'objet [Note].

VIII-C - La couche [dao]



La couche [dao] offrira l'interface suivante à la couche [métier] :

- *getClasses* rend la liste des classes du collège ;
- *getMatieres* rend la liste des matières ;
- *getEleves* rend la liste des élèves ;
- *getNotes* rend la liste des notes.

La couche [métier] n'utilisera que ces méthodes. Elle n'a pas à savoir comment elles sont implémentées. Ces données peuvent alors provenir de différentes sources (en dur, d'une base de données, de fichiers texte...) sans que cela impacte la couche [métier]. On appelle cela la **programmation par interfaces**.

La classe [Dao] implémente cette interface de la façon suivante :

Dao
```
1. # -*- coding=utf-8 -*-
2.
3. # import du module des entités
4. from entites import *
5.
6. class Dao:
7.     # constructeur
8.     def __init__(self):
9.         # on instancie les classes
10.         classe1=Classe(1,"classe1")
11.         classe2=Classe(2,"classe2")
12.         self.classes=[classe1,classe2]
13.         # les matières
14.         matiere1=Matiere(1,"matiere1",1)
15.         matiere2=Matiere(2,"matiere2",2)
16.         self.matieres=[matiere1,matiere2]
17.         # les élèves
18.         eleve11=Eleve(11,"nom1","prenom1",classe1)
19.         eleve21=Eleve(21,"nom2","prenom2",classe1)
20.         eleve32=Eleve(32,"nom3","prenom3",classe2)
21.         eleve42=Eleve(42,"nom4","prenom4",classe2)
22.         self.eleves=[eleve11,eleve21,eleve32,eleve42]
23.         # les notes
24.         note1=Note(1,10,eleve11,matiere1)
25.         note2=Note(2,12,eleve21,matiere1)
26.         note3=Note(3,14,eleve32,matiere1)
```


Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Dao

```
27.         note4=Note(4,16,eleve42,matiere1)
28.         note5=Note(5,6,eleve11,matiere2)
29.         note6=Note(6,8,eleve21,matiere2)
30.         note7=Note(7,10,eleve32,matiere2)
31.         note8=Note(8,12,eleve42,matiere2)
32.         self.notes=[note1,note2,note3,note4,note5,note6,note7,note8]
33.
34.     #-----------
35.     # interface
36.     #-----------
37.
38.     # liste des classes
39.     def getClasses(self):
40.         return self.classes
41.
42.     # liste des matières
43.     def getMatieres(self):
44.         return self.matieres
45.
46.     # liste des élèves
47.     def getEleves(self):
48.         return self.eleves
49.
50.     # liste des notes
51.     def getNotes(self):
52.         return self.notes
```

- ligne 4 : on importe le module qui contient les entités manipulées par la couche [DAO] ;
- ligne 8 : le constructeur n'a pas de paramètres. Il construit en dur quatre listes :
- lignes 10-12 : la liste des classes ;
- lignes 14-16 : la liste des matières ;
- lignes 18-22 : la liste des élèves ;
- lignes 24-32 : la liste des notes.
- lignes 39-52 : les quatre méthodes de l'interface de la couche [dao] se contentent de rendre une référence sur les quatre listes construites par le constructeur.

Un programme de test pourrait être le suivant :

```
1. # -*- coding=utf-8 -*-
2.
3. # import du module des entités et de la couche [dao]
4. from entites import *
5. from dao import *
6.
7. # instanciation couche [dao]
8. dao=Dao()
9.
10. # liste des classes
11. for classe in dao.getClasses():
12.     print classe
13.
14. # liste des classes
15. for matiere in dao.getMatieres():
16.     print matiere
17.
18. # liste des classes
19. for eleve in dao.getEleves():
20.     print eleve
21.
22. # liste des classes
23. for note in dao.getNotes():
24.     print note
```

Les commentaires se suffisent à eux-mêmes. Les lignes 11-24 utilisent l'interface de la couche [DAO]. Il n'y a pas là d'hypothèses sur l'implémentation réelle de la couche. Ligne 8, on instancie la couche [DAO]. Ici on fait des hypothèses : nom de la classe et type de constructeur. Il existe des solutions qui permettent d'éviter cette dépendance.

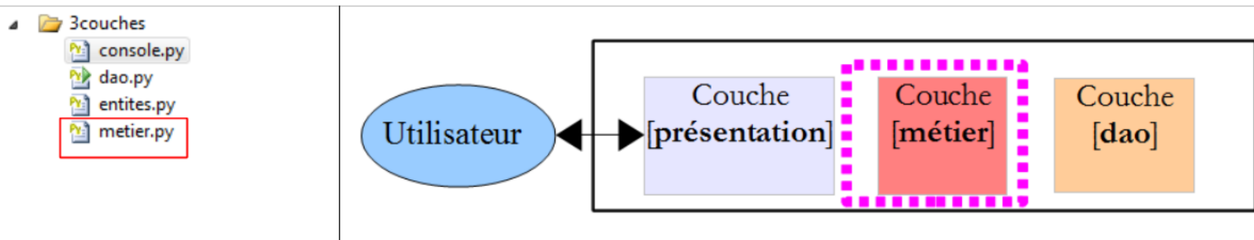
Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Les résultats de l'exécution de ce script sont les suivants :

Résultats

```
1. Classe[1,classe1]
2. Classe[2,classe2]
3. Matiere[1,matiere1,1]
4. Matiere[2,matiere2,2]
5. Eleve[11,prenom1,nom1,Classe[1,classe1]]
6. Eleve[21,prenom2,nom2,Classe[1,classe1]]
7. Eleve[32,prenom3,nom3,Classe[2,classe2]]
8. Eleve[42,prenom4,nom4,Classe[2,classe2]]
9. Note[1,10,Eleve[11,prenom1,nom1,Classe[1,classe1]],Matiere[1,matiere1,1]]
10. Note[2,12,Eleve[21,prenom2,nom2,Classe[1,classe1]],Matiere[1,matiere1,1]]
11. Note[3,14,Eleve[32,prenom3,nom3,Classe[2,classe2]],Matiere[1,matiere1,1]]
12. Note[4,16,Eleve[42,prenom4,nom4,Classe[2,classe2]],Matiere[1,matiere1,1]]
13. Note[5,6,Eleve[11,prenom1,nom1,Classe[1,classe1]],Matiere[2,matiere2,2]]
14. Note[6,8,Eleve[21,prenom2,nom2,Classe[1,classe1]],Matiere[2,matiere2,2]]
15. Note[7,10,Eleve[32,prenom3,nom3,Classe[2,classe2]],Matiere[2,matiere2,2]]
16. Note[8,12,Eleve[42,prenom4,nom4,Classe[2,classe2]],Matiere[2,matiere2,2]]
```

VIII-D - La couche [métier]



La couche [métier] implémente l'interface suivante :

- *getClasses* rend la liste des classes du collège ;
- *getMatieres* rend la liste des matières ;
- *getEleves* rend la liste des élèves ;
- *getNotes* rend la liste des notes ;
- *getStatsForEleve* rend les notes de l'élève n° *idEleve* ainsi que des informations sur celles-ci : *moyenne pondérée*, *note la plus basse*, *note la plus haute*.

La couche [présentation] n'utilisera que ces méthodes. Elle n'a pas à savoir comment elles sont implémentées.

La méthode *getStatsForEleve* rend un objet de type [StatsForEleve] suivant :

StatsForEleve

```
1. class StatsForEleve:
2.     # constructeur
3.     def __init__(self, eleve, notes):
4.         # on mémorise les paramètres
5.         self.eleve=eleve
6.         self.notes=notes
7.         # on s'arrête s'il n'y a pas de notes
8.         if len(notes)==0:
9.             return
10.         # exploitation des notes
11.         sommePonderee=0
12.         sommeCoeff=0
13.         self.max=-1
14.         self.min=21
15.         for note in notes:
16.             valeur=note.valeur
17.             coeff=note.matiere.coefficient
```


Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

StatsForEleve

```
18.             sommeCoeff+=coeff
19.             sommePonderee+=valeur*coeff
20.             if valeur<self.min:
21.                 self.min=valeur
22.             if valeur>self.max:
23.                 self.max=valeur
24.         # calcul de la moyenne de l'élève
25.         self.moyennePonderee=float(sommePonderee)/sommeCoeff
26. 
27.     # toString
28.     def __str__(self):
29.         # cas de l'élève sans notes
30.         if len(self.notes)==0:
31.             return "Eleve={0}, notes=[]".format(self.eleve)
32.         # cas de l'élève avec notes
33.         str=""
34.         for note in self.notes:
35.             str+="{0} ".format(note.valeur)
36.         return "Eleve={0}, notes=[{1}], max={2}, min={3}, moyenne={4}".format(self.eleve, str,
 self.max, self.min, self.moyennePonderee)
```

- ligne 3 : le constructeur reçoit deux paramètres :
- une référence sur l'élève de type [Eleve] pour lequel on calcule des indicateurs,
- une référence sur ses notes, une liste d'objets [Note] ;
- lignes 8-9 : si la liste des notes est vide, on ne va pas plus loin.
- sinon lignes 11-25, on calcule les indicateurs suivants :
- *self.moyennePonderee* : la moyenne de l'élève pondérée par les coefficients des matières,
- *self.min* : la note la plus faible de l'élève,
- *self.max* : sa note la plus haute ;
- ligne 28 : la méthode d'affichage de la classe sous la forme indiquée ligne 36.

Un script de test de cette classe pourrait être le suivant :

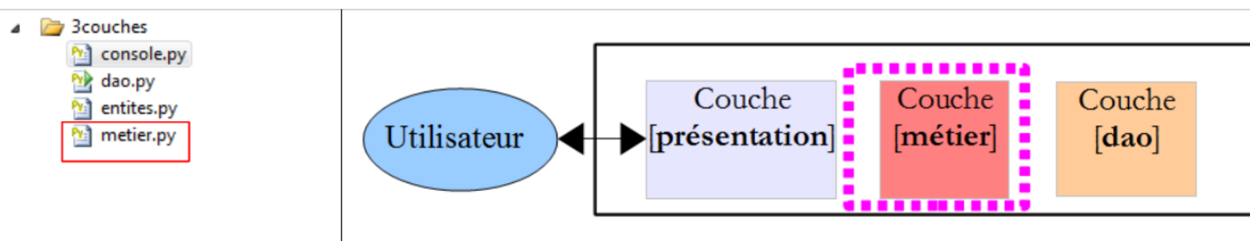
```
1. # -*- coding=utf-8 -*-
2. 
3. # import des modules des entités, de la couche [dao] et de la couche [metier]
4. from entites import *
5. from dao import *
6. from metier import *
7. 
8. # une classe
9. classe1=Classe(1,"6e A")
10. # un élève dans cette classe
11. paul_durand=Eleve(1,"durand","paul",classe1)
12. # trois matières
13. maths=Matiere(1,"maths",1)
14. francais=Matiere(2,"francais",2)
15. anglais=Matiere(3,"anglais",3)
16. # des notes dans ces matières pour l'élève
17. note_maths=Note(1,10,paul_durand,maths)
18. note_francais=Note(2,12,paul_durand,francais)
19. note_anglais=Note(3,14,paul_durand,anglais)
20. # on affiche les indicateurs
21. print StatsForEleve(paul_durand,[note_maths, note_francais,note_anglais])
```

Les résultats écran sont les suivants :

```
1. Eleve=Eleve[1,paul,durand,Classe[1,6e A]], notes=[10 12 14 ], max=14, min=10, moyenne=12.6666666667
```

Revenons à notre architecture en couches :

Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.



La classe [Metier] implémente la couche [métier] de la façon suivante :

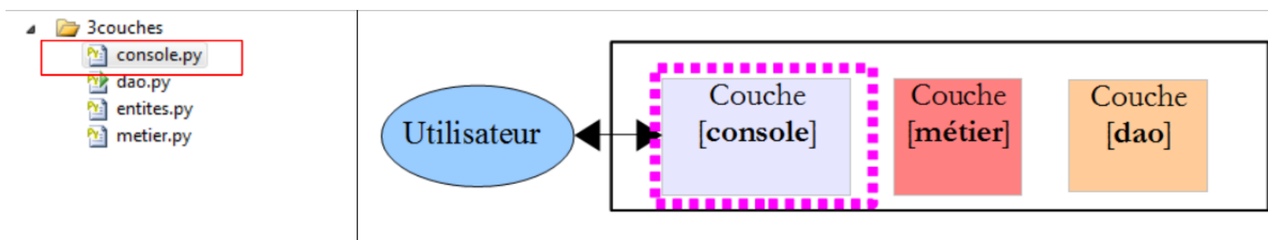
[Metier]
```
1. class Metier:
2.     # constructeur
3.     def __init__(self,dao):
4.         # on mémorise la référence sur la couche [dao]
5.         self.dao=dao
6. 
7.     #-----------
8.     # interface
9.     #-----------
10. 
11.     # les indicateurs sur les notes
12.     def getStatsForEleve(self,idEleve):
13.         # Stats pour l'élève de n° idEleve
14.         # recherche de l'élève
15.         trouve=False
16.         i=0
17.         eleves=self.getEleves()
18.         while not trouve and i<len(eleves):
19.             trouve=eleves[i].id==idEleve
20.             i+=1
21.         # a-t-on trouvé ?
22.         if not trouve:
23.             raise RuntimeError("L'eleve [{0}] n'existe pas".format(idEleve))
24.         else:
25.             eleve=eleves[i-1]
26.         # liste de toutes les notes
27.         notes=[]
28.         for note in self.getNotes():
29.             # on ajoute à notes, toutes les notes de l'élève
30.             if note.eleve.id==idEleve:
31.                 notes.append(note)
32.         # on rend le résultat
33.         return StatsForEleve(eleve,notes)
34. 
35.     # la liste des classes
36.     def getClasses(self):
37.         return self.dao.getClasses()
38. 
39.     # la liste des matières
40.     def getMatieres(self):
41.         return self.dao.getMatieres()
42. 
43.     # la liste des élèves
44.     def getEleves(self):
45.         return self.dao.getEleves()
46. 
47.     # la liste des notes
48.     def getNotes(self):
49.         return self.dao.getNotes()
```

- lignes 3-5 : le constructeur reçoit une référence sur la couche [DAO]. La couche [métier] doit avoir cette référence. Ici, on la lui donne via son constructeur. On pourrait imaginer d'autres solutions. Dans une architecture en couches représentée horizontalement, chaque couche doit avoir une référence sur la couche qui est à sa droite ;
- lignes 36-49 : les méthodes *getClasses*, *getMatieres*, *getEleves*, *getNotes* se contentent de déléguer l'appel aux méthodes de mêmes noms de la couche [DAO] ;

Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.
http://tahe.developpez.com/divers/python/exemples/

- ligne 12 : la méthode *getStatsForEleve* reçoit comme paramètre le n° de l'élève pour lequel on doit rendre des indicateurs.
- ligne 17 : l'élève va être recherché dans la liste de tous les élèves ;
- lignes 18-20 : la boucle de recherche ;
- ligne 23 : si l'élève n'a pas été trouvé, on lève une exception ;
- sinon ligne 25, l'élève trouvé est mémorisé ;
- lignes 28-31 : on cherche parmi l'ensemble des notes du collège, celles qui appartiennent à l'élève mémorisé ;
- lorsqu'on les a trouvées, on peut construire l'objet *StatsForEleve* demandé.

VIII-E - La couche [console]



La couche [console] est implémentée par le script suivant :

[console]

```
1. # -*- coding=utf-8 -*-
2.
3. # import du module des entités, du module [dao], du module [metier]
4. from entites import *
5. from dao import *
6. from metier import *
7.
8. # ----------- couche [console]
9. # instanciation couche [metier]
10. metier=Metier(Dao())
11. # demande /réponse
12. fini=False
13. while not fini:
14.     # question
15.     print "numero de l'eleve (>=1 et * pour arreter) : "
16.     # reponse
17.     reponse=raw_input()
18.     # fini ?
19.     if reponse.strip()=="*":
20.         break
21.     # a-t-on une saisie correcte ?
22.     ok=False
23.     try:
24.         idEleve=int(reponse,10)
25.         ok=idEleve>=1
26.     except:
27.         pass
28.     # donnée correcte ?
29.     if not ok:
30.         print "Saisie incorrecte. Recommencez..."
31.         continue
32.     # calcul
33.     try:
34.         print metier.getStatsForEleve(idEleve)
35.     except RuntimeError,erreur:
36.         print "L'erreur suivante s'est produite : {0}".format(erreur)
```

- ligne 10 : instanciation à la fois des couches [DAO] et [métier]. C'est la seule dépendance de notre code vis-à-vis de l'implémentation de ces couches ;

Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

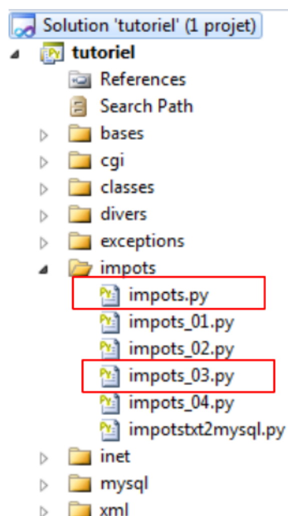
- ligne 34 : on utilise l'interface de la couche [métier] ;
- ligne 19 : la méthode *strip* supprime les espaces de début et fin de chaîne ;
- ligne 20 : *break* permet de sortir d'une boucle ;
- ligne 24 : on tente de convertir la chaîne saisie en entier décimal ;
- ligne 29 : *ok* est vrai seulement si on est passé par la ligne 25 ;
- ligne 31 : *continue* permet de reboucler en milieu de boucle ;
- ligne 34 : calcul des indicateurs ;
- ligne 35 : on intercepte l'exception *RuntimeError* qui peut sortir de la couche [métier].

Voici un exemple d'exécution :

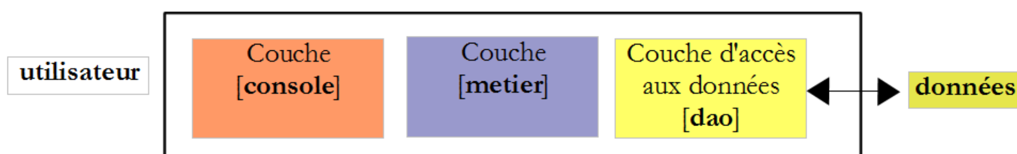
```
1. numero de l'eleve (>=1 et * pour arreter) :
2. xx
3. Saisie incorrecte. Recommencez...
4. numero de l'eleve (>=1 et * pour arreter) :
5. -4
6. Saisie incorrecte. Recommencez...
7. numero de l'eleve (>=1 et * pour arreter) :
8. 11
9. Eleve=Eleve[11,prenom1,nom1,Classe[1,classe1]], notes=[10 6 ], max=10, min=6, mo
10. yenne=7.33333333333
11. numero de l'eleve (>=1 et * pour arreter) :
12. 111
13. L'erreur suivante s'est produite : L'eleve [111] n'existe pas
14. numero de l'eleve (>=1 et * pour arreter) :
15. *
```


Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

IX - Exercice d'application - [IMPOTS] avec objets



Nous reprenons ici l'exercice décrit précédemment. Nous partons de la version avec fichiers texte. Pour traiter cet exemple avec des objets, nous allons utiliser une architecture à trois couches :



- la couche [DAO] (Data Access Object) s'occupe de l'accès aux données. Dans la suite, ces données seront trouvées d'abord dans un fichier texte, puis dans une base de données MySQL ;
- la couche [metier] s'occupe des problèmes métier, ici le calcul de l'impôt. Elle ne s'occupe pas des données. Celles-ci peuvent avoir deux provenances :
- la couche [DAO] pour les données persistantes,
- la couche [console] pour les données fournies par l'utilisateur ;
- la couche [console] s'occupe des interactions avec l'utilisateur.

Dans la suite, les couches [DAO] et [metier] seront chacune implémentée à l'aide d'une classe. La couche [console] sera elle implémentée par le programme principal.

On supposera également que toutes les implémentations de la couche [DAO] offrent la méthode *getData()* qui rend un tuple de trois éléments (*limites, coeffR, coeffN*) qui sont les trois tableaux de données nécessaires au calcul de l'impôt. Dans d'autres langages, on appelle cela une interface. Une interface définit des méthodes (ici *getData*) que les classes implémentant cette interface doivent avoir.

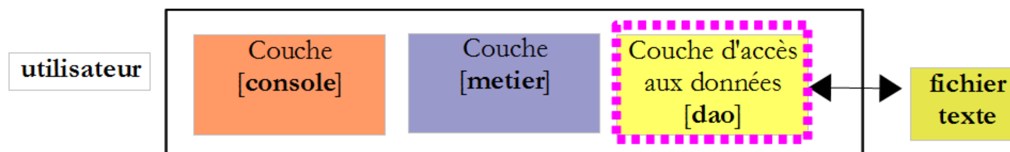
La couche [metier] sera implémentée par une classe dont le constructeur aura pour paramètre une référence sur la couche [DAO] ce qui assurera la communication entre les deux couches.

IX-A - La couche [DAO]

Nous allons réunir les différentes classes nécessaires à l'application dans un même fichier *impots.py*. Les objets de ce fichier seront ensuite importés dans les scripts les nécessitant.

Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Nous commençons par le cas où les données sont dans un fichier texte.



Le code de la classe [ImpotsFile] (*impots.py*) qui implémente la couche [DAO] est le suivant :

[ImpotsFile]

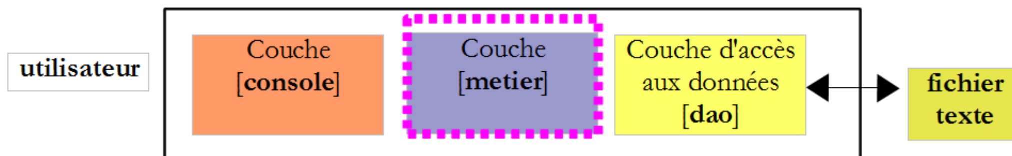
```
1. # -*- coding=utf-8 -*-
2.
3. import math, sys
4.
5. # ---------------------------------------------------------------------------
6. # classe d'exceptions propriétaire
7. class ImpotsError:
8.     pass
9.
10. # ---------------------------------------------------------------------------
11. class Utilitaires:
12.     """classe de fonctions utilitaires"""
13.     def cutNewLineChar(self,ligne):
14.         # on supprime la marque de fin de ligne si elle existe
15.         l=len(ligne)
16.         while(ligne[l-1]=="\n" or ligne[l-1]=="\r"):
17.             l-=1
18.         return(ligne[0:l])
19.
20. # ---------------------------------------------------------------------------
21. class ImpotsFile:
22.     def __init__(self,IMPOTS):
23.         # IMPOTS : le nom du fichier contenant les données des tables limites, coeffR, coeffN
24.         # on ouvre le fichier
25.         data=open(IMPOTS,"r")
26.         # un objet Utilitaires
27.         u=Utilitaires()
28.
  # création des 3 tables - on suppose que les 3 lignes de IMPOTS sont syntaxiquement correctes
29.         # -- ligne 1
30.         ligne=data.readline()
31.         if ligne== '':
32.             raise ImpotsError("La premiere ligne du fichier {0} est absente".format(IMPOTS))
33.         limites=u.cutNewLineChar(ligne).split(":")
34.         for i in range(len(limites)):
35.             limites[i]=int(limites[i])
36.         # -- ligne 2
37.         ligne=data.readline()
38.         if ligne== '':
39.             raise ImpotsError("La deuxieme ligne du fichier {0} est absente".format(IMPOTS))
40.         coeffR=u.cutNewLineChar(ligne).split(":")
41.         for i in range(len(coeffR)):
42.             coeffR[i]=float(coeffR[i])
43.         # -- ligne 3
44.         ligne=data.readline()
45.         if ligne== '':
46.             raise ImpotsError("La troisieme ligne du fichier {0} est absente".format(IMPOTS))
47.         coeffN=u.cutNewLineChar(ligne).split(":")
48.         for i in range(len(coeffN)):
49.             coeffN[i]=float(coeffN[i])
50.         # fin
51.         (self.limites,self.coeffR,self.coeffN)=(limites,coeffR,coeffN)
52.
53.     def getData(self):
54.         return (self.limites, self.coeffR, self.coeffN)
```


Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Notes :

- lignes 7-8 : on définit une classe *ImpotsError* dérivée de la classe *Exception*. Cette classe n'ajoute rien à la classe *Exception*. On l'utilise seulement pour avoir une classe d'exception propriétaire. Ultérieurement cette classe pourrait être enrichie ;
- lignes 11-18 : une classe de méthodes utilitaires. Ici la méthode *cutNewLineChar* supprime l'éventuelle marque de fin de ligne d'une chaîne de caractères ;
- ligne 25 : l'ouverture du fichier peut lancer l'exception *IOError ;*
- lignes 32, 39, 46 : on lance l'exception propriétaire *ImpotsError ;*
- le code est analogue à celui étudié précédemment.

IX-B - La couche [metier]



La classe [ImpotsMetier] (*impots.py*) qui implémente la couche [metier] est la suivante :

[ImpotsMetier]
```
1. class ImpotsMetier:
2.
3.     # constructeur
4.     # on récupère un pointeur sur la couche [dao]
5.     def __init__(self, dao):
6.         self.dao=dao
7.
8.     # calcul de l'impôt
9.     # --------------------------------------------------------------------------
10.     def calculer(self,marie,enfants,salaire):
11.         # marié : oui, non
12.         # enfants : nombre d'enfants
13.         # salaire : salaire annuel
14.
15.         # on demande à la couche [dao] les données nécessaires au calcul
16.         (limites, coeffR, coeffN)=self.dao.getData()
17.
18.        # nombre de parts
19.         marie=marie.lower()
20.         if(marie=="oui"):
21.             nbParts=float(enfants)/2+2
22.         else:
23.             nbParts=float(enfants)/2+1
24.         # une 1/2 part de plus si au moins 3 enfants
25.         if enfants>=3:
26.             nbParts+=0.5
27.         # revenu imposable
28.         revenuImposable=0.72*salaire
29.         # quotient familial
30.         quotient=revenuImposable/nbParts
31.         # est mis à la fin du tableau limites pour arrêter la boucle qui suit
32.         limites[len(limites)-1]=quotient
33.         # calcul de l'impôt
34.         i=0
35.         while quotient>limites[i] :
36.             i=i+1
37.         # du fait qu'on a placé quotient à la fin du tableau limites, la boucle précédente
38.         # ne peut déborder du tableau limites
39.         # maintenant on peut calculer l'impôt
```

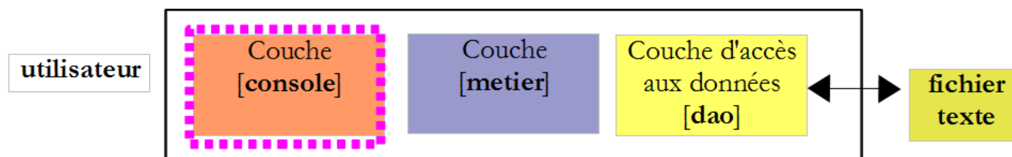

Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

[ImpotsMetier]
```
40.         return math.floor(revenuImposable*(float)(coeffR[i])-nbParts*(float)(coeffN[i]))
```

Notes :

- lignes 5-6 : le constructeur de la classe reçoit comme paramètre une référence sur la couche [DAO] ;
- ligne 16 : on utilise la méthode *getData* de la couche [DAO] pour récupérer les données permettant le calcul de l'impôt ;

IX-C - La couche [console]



Le script implémentant la couche [console] (impots-03) est le suivant :

```
1. # -*- coding=utf-8 -*-
2.
3. # import du module des classes Impots*
4. from impots import *
5.
6. # ------------------------------------------------ main
7. # définition des constantes
8. DATA="data.txt"
9. RESULTATS="resultats.txt"
10. IMPOTS="impots.txt"
11.
12. # les données nécessaires au calcul de l'impôt ont été placées dans le fichier IMPOTS
13. # à raison d'une ligne par tableau sous la forme
14. # val1:val2:val3...
15.
16. # instanciation couche [metier]
17. try:
18.     metier=ImpotsMetier(ImpotsFile(IMPOTS))
19. except (IOError, ImpotsError) as infos:
20.     print ("Une erreur s'est produite : {0}".format(infos))
21.     sys.exit()
22.
23. # lecture des données
24. try:
25.     data=open(DATA,"r")
26. except:
27.     print "Impossible d'ouvrir en lecture le fichier des donnees [DATA]"
28.     sys.exit()
29.
30. # ouverture fichier des résultats
31. try:
32.     resultats=open(RESULTATS,"w")
33. except:
34.     print "Impossible de creer le fichier des résultats [RESULTATS]"
35.     sys.exit()
36.
37. # utilitaires
38. u=Utilitaires()
39.
40. # on exploite la ligne courante du fichier des données
41. ligne=data.readline()
42. while(ligne != ''):
43.     # on enlève l'éventuelle marque de fin de ligne
44.     ligne=u.cutNewLineChar(ligne)
```


Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

```
45.     # on récupère les 3 champs marié:enfants:salaire qui forment la ligne
46.     (marie,enfants,salaire)=ligne.split(",")
47.     enfants=int(enfants)
48.     salaire=int(salaire)
49.     # on calcule l'impôt
50.     impot=metier.calculer(marie,enfants,salaire)
51.     # on inscrit le résultat
52.     resultats.write("{0}:{1}:{2}:{3}\n".format(marie,enfants,salaire,impot))
53.     # on lit une nouvelle ligne
54.     ligne=data.readline()
55. # on ferme les fichiers
56. data.close()
57. resultats.close()
```

Notes :

- ligne 4 : on importe tous les objets du fichier *impots.py* qui contient les définitions des classes. Ceci fait, on peut utiliser ces objets comme s'ils étaient dans le même fichier que le script ;
- lignes 17-21 : on instancie à la fois la couche [DAO] et la couche [metier] avec une gestion des exceptions éventuelles ;
- ligne 18 : on instancie la couche [DAO] puis la couche [metier]. On mémorise la référence sur la couche [metier] ;
- ligne 19 : on gère les deux exceptions qui peuvent se produire ;

IX-D - Résultats

Ceux déjà obtenus dans les versions avec tableaux et fichiers.

Le fichier des données *impots.txt* :

impots.txt
```
1. 12620:13190:15640:24740:31810:39970:48360:55790:92970:127860:151250:172040:195000:0
2. 0:0.05:0.1:0.15:0.2:0.25:0.3:0.35:0.4:0.45:0.5:0.55:0.6:0.65
3. 0:631:1290.5:2072.5:3309.5:4900:6898.5:9316.5:12106:16754.5:23147.5:30710:39312:49062
```

Le fichier des données *data.txt* :

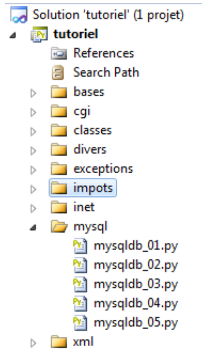
data.txt
```
1. oui,2,200000
2. non,2,200000
3. oui,3,200000
4. non,3,200000
5. oui,5,50000
6. non,0,3000000
```

Les fichier *resultats.txt* des résultats obtenus :

resultats.txt
```
1. oui:2:200000:22504.0
2. non:2:200000:33388.0
3. oui:3:200000:16400.0
4. non:3:200000:22504.0
5. oui:5:50000:0.0
6. non:0:3000000:1354938.0
```

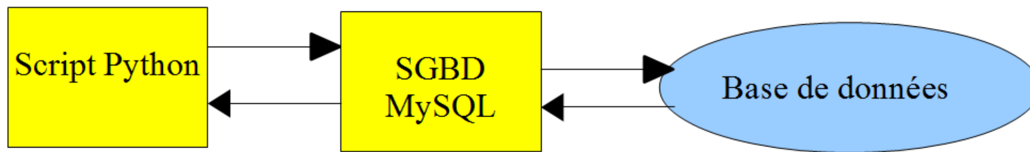

Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

X - Utilisation du SGBD MySQL

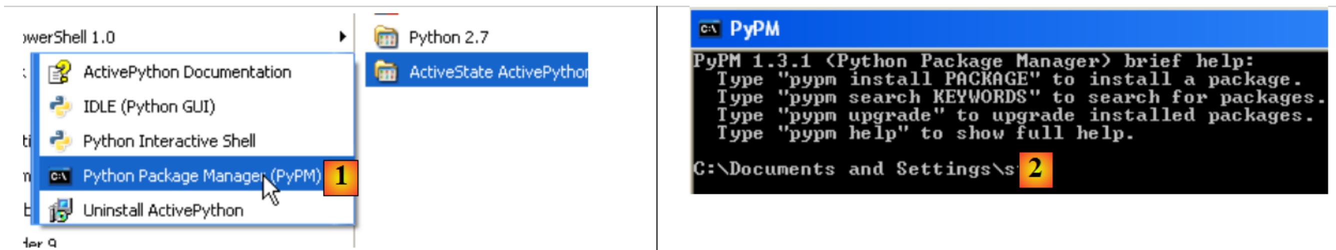


X-A - Installation du module MySQLdb

Nous allons écrire des scripts utilisant une base de données MySQL :



Les fonctions Python de gestion d'une base de données MySQL sont encapsulées dans un module MySQLdb qui n'est pas intégré à la distribution initiale de Python. Il faut alors procéder au téléchargement et à l'installation du module. Voici une façon de procéder :



- dans le menu des programmes, prendre en [1] le gestionnaire de packages Python. La fenêtre de commandes [2] apparaît alors.

On recherche le mot-clé *mysql* dans les packages :

```
1. C:\Documents and Settings\st>pypm search mysql
2. Get: [pypm-be.activestate.com] :repository-index:
3. Get: [pypm-free.activestate.com] :repository-index:
4. autosync: synced 2 repositories
5.   chartio                      Setup wizard and connection client for connecting
6.   chartio-setup                Setup wizard and connection client for connecting
7.   cns.recipe.zmysqlda          Recipe for installing ZMySQLDA
8.   collective.recipe.zmysqlda Recipe for installing ZMySQLDA
9.   django-mysql-manager         DESCRIPTION_DESCRIPTION_DESCRIPTION
10.   jaraco.mysql                 MySQLDB-compatible MySQL wrapper by Jason R. Coomb
11.   lovely.testlayers            mysql, postgres nginx, memcached cassandra test la
12.   mtstat-mysql                 MySQL Plugins for mtstat
13.   mysql-autodoc                Generate HTML documentation from a mysql database
```


Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

```
14.   mysql-python              Python interface to MySQL
15.   mysqldbda                 MySQL Database adapter
16.   products.zmysqlda         MySQL Zope2 adapter.
17.   pymysql                   Pure Python MySQL Driver
18.   pymysql-sa                PyMySQL dialect for SQLAlchemy.
19.   pymysql3                  Pure Python MySQL Driver
20.   sa-mysql-dt               Alternative implementation of DateTime column for
21.   schemaobject              Iterate over a MySQL database schema as a Python o
22.   schemasync                A MySQL Schema Synchronization Utility
23.   simplestore               A datastore layer built on top of MySQL in Python.
24.   sqlbean                   A auto maping ORM for MYSQL and can bind with memc
25.   sqlwitch                  sqlwitch offers idiomatic SQL generation on top of
26.   tiddlywebplugins.mysql    MySQL-based store for tiddlyweb
27.   tiddlywebplugins.mysql2   MySQL-based store for tiddlyweb
28. zest.recipe.mysql         A Buildout recipe to setup a MySQL database.
```

Ont été listés tous les modules dont le nom ou la description ont le mot-clé *mysql*. Celui qui nous intéresse est [mysql-python], ligne 14. Nous l'installons :

```
1. C:\Documents and Settings\st>pypm install mysql-python
2. The following packages will be installed into "%APPDATA%\Python" (2.7):
3.  mysql-python-1.2.3
4. Hit: [pypm-free.activestate.com] mysql-python 1.2.3
5. Installing mysql-python-1.2.3
6.
7. C:\Documents and Settings\st>echo %APPDATA%
8. C:\Documents and Settings\st\Application Data
```

- ligne 5 : le package *mysql-python-1.2.3* a été installé dans le dossier "*%APPDATA%\Python*" où *APPDATA* est le dossier désigné ligne 8.

Cette opération peut échouer si :

- l'interpréteur Python utilisé est une version 64 bits ;
- le chemin %APPDATA% contient des caractères accentués.

X-B - Installation de MySQL

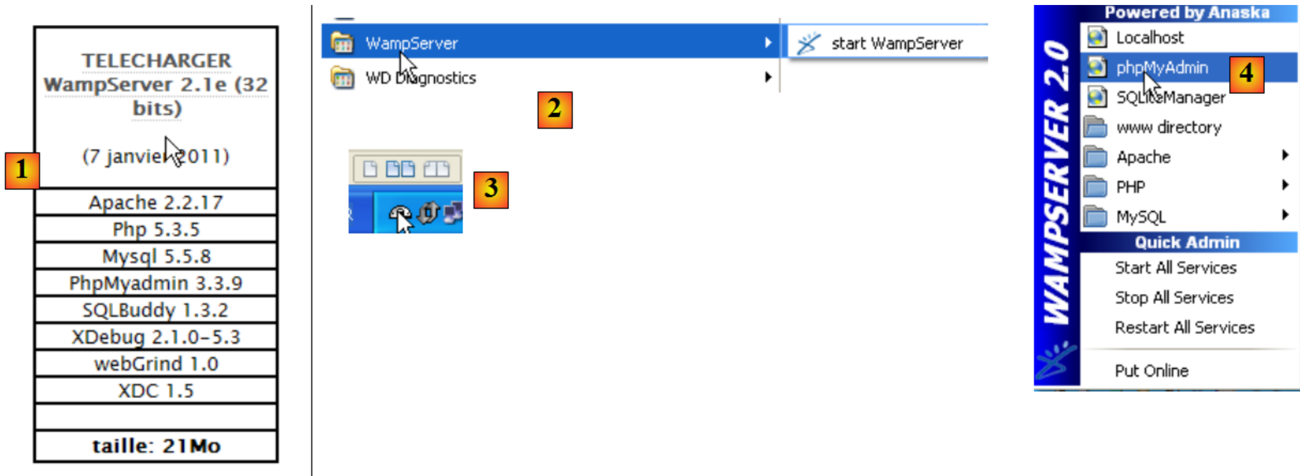
Il existe diverses façons d'installer le SGBD MySQL. Ici nous avons utilisé *WampServer*, un package réunissant plusieurs logiciels :

- un serveur Web Apache. Nous l'utiliserons pour l'écriture de scripts Web en Python ;
- le SGBD MySQL ;
- le langage de script PHP ;
- un outil d'administration du SGBD MySQL écrit en PHP : phpMyAdmin.

WampServer peut être téléchargé (juin 2011) à l'adresse suivante :

http://www.wampserver.com/download.php

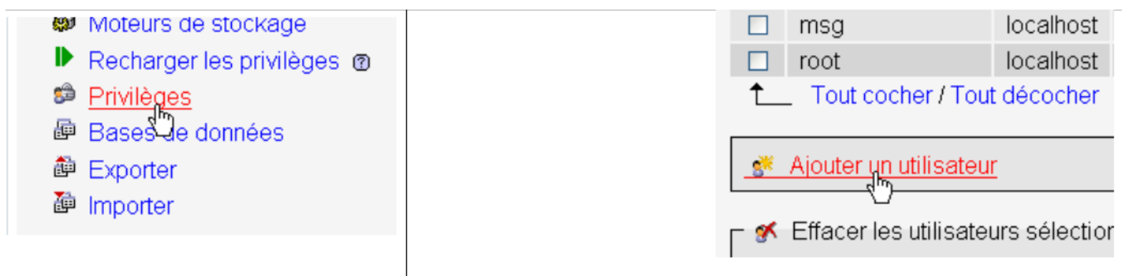
Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.



- en [1], on télécharge la version de *WampServer* qui va bien ;
- en [2], une fois installé, on le lance. Cela va lancer le serveur Web Apache et le SGBD MySQL ;
- en [3], une fois lancé, *WampServer* peut être administré à partir d'une icône [3] placée en bas à droite de la barre des tâches ;
- en [4], on lance l'outil d'administration de MySQL.

On crée une base de données [dbpersonnes] :

On crée un utilisateur [admpersonnes] avec le mot de passe [nobody] :



Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.
http://tahe.developpez.com/divers/python/exemples/

Ajouter un utilisateur

Information pour la connexion

Nom d'utilisateur:	Entrez une valeur:	admpersonnes	1
Serveur:	Local	localhost	2
Mot de passe:	Entrez une valeur:	••••••	3
Entrer à nouveau:		••••••	4
Générer un mot de passe:	Générer Copier		

Base de données pour cet utilisateur

- Aucune
- Créer une base portant son nom et donner à cet utilisateur tous les privilèges sur cette base
- Donner les privilèges passepartout ("%")

Privilèges globaux (Tout cocher / Tout décocher)

Veuillez noter que les noms de privilèges sont exprimés en anglais

Données: SELECT, INSERT, UPDATE, DELETE, FILE

Structure: CREATE, ALTER, INDEX, DROP, CREATE TEMPORARY TABLES, CREATE VIEW, SHOW VIEW, CREATE ROUTINE, ALTER ROUTINE, EXECUTE

Administration: GRANT, SUPER, PROCESS, RELOAD, SHUTDOWN, SHOW DATABASES, LOCK TABLES, REFERENCES, REPLICATION CLIENT, REPLICATION SLAVE, CREATE USER (5)

Limites de ressources

Note: Une valeur de 0 (zero) enlève la limite.

MAX QUERIES PER HOUR 0
MAX UPDATES PER HOUR 0
MAX CONNECTIONS PER HOUR 0
MAX USER_CONNECTIONS 0

6 Exécuter

- en [1], le nom de l'utilisateur ;
- en [2], la machine du SGBD sur laquelle on lui donne des droits ;
- en [3], son mot de passe [nobody] ;
- en [4], idem ;
- en [5], on ne donne aucun droit à cet utilisateur ;
- en [6], on le crée.

localhost / localhost | php
phpMyAdmin
7
Base de données
dbpersonnes (0)

8 echarger les privilèges
Privilèges
Bases de données

A 9 C D E F G H I J K L M N O P Q R S T U

	Utilisateur	Serveur	Mot de passe	Privilèges globaux	"Grant"
☐	admagi0809	localhost	Oui	USAGE	Non
☐	admpersonnes	localhost	Oui	USAGE	Non
☐	admrdvmedecins	localhost	Oui	USAGE	Non

Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

- en [7], on revient sur la page d'accueil de phpMyAdmin ;
- en [8], on utilise le lien [Privileges] de cette page pour aller modifier ceux de l'utilisateur [admpersonnes] [9].

- en [10], on indique qu'on veut donner à l'utilisateur [admpersonnes] des droits sur la base de données [dbpersonnes] ;
- en [11], on valide le choix.

- avec le lien [12] [Tout cocher], on accorde à l'utilisateur [admpersonnes] tous les droits sur la base de données [dbpersonnes] [13] ;
- on valide en [14].

Désormais nous avons :

- une base de données MySQL [dbpersonnes] ;
- un utilisateur [admpersonnes / nobody] qui a tous les droits sur cette base de données.

Nous allons écrire des scripts Python pour exploiter la base de données.

X-C - Connexion à une base MySQL - 1

Programme (mysqldb_01)

```
1. # import du module MySQLdb
2. import sys
3. sys.path.append("D:\Programs\ActivePython\site-packages")
4. import MySQLdb
5.
```


Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.
http://tahe.developpez.com/divers/python/exemples/

```
6. # connexion à une base MySql
7. ....
```

Notes :

- lignes 2-4 : les scripts contenant des opérations avec le SGBD MySQL doivent importer le module **MySQLdb.** On se souvient que nous avons installé ce module dans le dossier [%APPDATA%\Python]. Le dossier [%APPDATA%\Python] est automatiquement exploré lorsqu'un code Python réclame un module. En fait ce sont tous les dossiers déclarés dans **sys.path** qui sont explorés. Voici un exemple qui affiche ces dossiers :

```
1. # -*- coding=utf-8 -*-
2.
3. import sys
4.
5. # affichage des dossiers de sys.path
6. for dossier in sys.path:
7.     print dossier
```

L'affichage écran est le suivant :

```
1. D:\data\istia-1112\python\tutoriel
2. C:\Windows\system32\python27.zip
3. D:\Programs\ActivePython\Python2.7.2\DLLs
4. D:\Programs\ActivePython\Python2.7.2\lib
5. D:\Programs\ActivePython\Python2.7.2\lib\plat-win
6. D:\Programs\ActivePython\Python2.7.2\lib\lib-tk
7. D:\Programs\ActivePython\Python2.7.2
8. C:\Users\Serge TahÚ\AppData\Roaming\Python\Python27\site-packages
9. D:\Programs\ActivePython\Python2.7.2\lib\site-packages
10. D:\Programs\ActivePython\Python2.7.2\lib\site-packages\win32
11. D:\Programs\ActivePython\Python2.7.2\lib\site-packages\win32\lib
12. D:\Programs\ActivePython\Python2.7.2\lib\site-packages\Pythonwin
13. D:\Programs\ActivePython\Python2.7.2\lib\site-packages\setuptools-0.6c11-py2.7.egg-info
```

Ligne 8, le dossier [site-packages] où MySQLdb avait été installé initialement. Nous déplaçons le dossier [site-packages] qui est le dossier où l'utilitaire *pypm* installe les modules Python. Pour ajouter un nouveau dossier que Python explorera à la recherche des modules, on l'ajoute à la liste *sys.path* :

```
1. # import du module MySQLdb
2. import sys
3. sys.path.append("D:\Programs\ActivePython\site-packages")
4. import MySQLdb
5.
6. # connexion à une base MySql
7. ....
```

Ligne 3, on ajoute le dossier où a été déplacé le module MySQLdb.

Le code complet de l'exemple est le suivant :

```
1. # import du module MySQLdb
2. import sys
3. sys.path.append("D:\Programs\ActivePython\site-packages")
4. import MySQLdb
5.
6. # connexion à une base MySql
7. # l'identité de l'utilisateur est (admpersonnes,nobody)
8. user="admpersonnes"
9. pwd="nobody"
10. host="localhost"
11. connexion=None
12. try:
13.     print "connexion..."
14.     # connexion
15.     connexion=MySQLdb.connect(host=host,user=user,passwd=pwd)
```


Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

```
16.     # suivi
17.
  print "Connexion a MySQL reussie sous l'identite host={0},user={1},passwd={2}".format(host,user,pwd)
18. except MySQLdb.OperationalError,message:
19.     print "Erreur : {0}".format(message)
20. finally:
21.     try:
22.         connexion.close()
23.     except:
24.         pass
```

- lignes 8-11 : le script va connecter (ligne 15) l'utilisateur [admpersonnes / nobody] au SGBD MySQL de la machine [localhost]. On ne le connecte pas à une base de données précise ;
- lignes 12-24 : la connexion peut échouer. Aussi la fait-on dans un try / except / finally ;
- ligne 15 : la méthode *connect* du module MySQLdb admet différents paramètres nommés :
- *user* : utilisateur propriétaire de la connexion [admpersonnes],
- *pwd* : mot de passe de l'utilisateur [nobody],
- *host* : machine du SGBD MySQL [localhost],
- *db* : la base de données à laquelle on se connecte. Optionnel ;
- ligne 18 : si une exception est lancée, elle est de type [MySQLdb.OperationalError] et le message d'erreur associé sera trouvé dans la variable [message] ;
- lignes 20-23 : dans la clause [finally], on ferme la connexion. En cas d'exception, on l'intercepte (ligne 23) mais on ne fait rien avec (ligne 24).

Résultats

```
1. connexion...
2. Connexion a MySQL reussie sous l'identite host=localhost,user=admpersonnes,passwd=nobody
```

X-D - Connexion à une base MySQL - 2

Programme (mysqldb_02)

```
1. # import du module MySQLdb
2. import sys
3. sys.path.append("D:\Programs\ActivePython\site-packages")
4. import MySQLdb
5.
6. # ---------------------------------------------------------------------------------
7. def testeConnexion(hote,login,pwd):
8.     # connecte puis déconnecte (login,pwd) du SGBD mysql du serveur hote
9.     # lance l'exception MySQLdb.operationalError
10.     # connexion
11.     connexion=MySQLdb.connect(host=hote,user=login,passwd=pwd)
12.     print "Connexion a MySQL reussie sous l'identite (%s,%s,%s)" % (hote,login,passwd)
13.     # on ferme la connexion
14.     connexion.close()
15.     print "Fermeture connexion MySql reussie\n"
16.
17.
18. # ---------------------------------------------- main
19. # connexion à la base MySql
20. # l'identité de l'utilisateur
21. user="admpersonnes"
22. passwd="nobody"
23. host="localhost"
24. # test connexion
25. try:
26.     testeConnexion(host,user,passwd)
27. except MySQLdb.OperationalError,message:
28.     print message
29. # avec un utilisateur inexistant
30. try:
31.     testeConnexion(host,"xx","xx")
32. except MySQLdb.OperationalError,message:
33.     print message
```


Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Notes :

- lignes 7-15 : une fonction qui tente de connecter puis de déconnecter un utilisateur à un SGBD MySQL. Affiche le résultat ;
- lignes 18-34 : programme principal - appelle deux fois la méthode *testeConnexion* et affiche les éventuelles exceptions.

Résultats

```
1. Connexion a MySQL reussie sous l'identite (localhost,admpersonnes,nobody)
2. Fermeture connexion MySql reussie
3.
4. Echec de la connexion a MySQL : (1045, "Access denied for user 'xx'@'localhost'(using password:
 YES)")
```

X-E - Création d'une table MySQL

Maintenant qu'on sait créer une connexion avec un SGBD MySQL, on commence à émettre des ordres SQL sur cette connexion. Pour cela, nous allons nous connecter à la base créée [dbpersonnes] et utiliser la connexion pour créer une table dans la base.

Programme (mysqldb_03)

```
1. # import du module MySQLdb
2. import sys
3. sys.path.append("D:\Programs\ActivePython\site-packages")
4. import MySQLdb
5.
6. # ---------------------------------------------------------------------------------
7. def executeSQL(connexion,update):
8.     # exécute une requête update de mise à jour sur la connexion
9.     # on demande un curseur
10.     curseur=connexion.cursor()
11.     # exécute la requête SQL sur la connexion
12.     try:
13.         curseur.execute(update)
14.         connexion.commit()
15.     except Exception, erreur:
16.         connexion.rollback()
17.         raise
18.     finally:
19.         curseur.close()
20.
21. # ---------------------------------------------- main
22. # connexion à la base MySql
23. # l'identité de l'utilisateur
24. ID="admpersonnes"
25. PWD="nobody"
26. # la machine hôte du SGBD
27. HOTE="localhost"
28. # identité de la base
29. BASE="dbpersonnes"
30. # connexion
31. try:
32.     connexion=MySQLdb.connect(host=HOTE,user=ID,passwd=PWD,db=BASE)
33. except MySQLdb.OperationalError,message:
34.     print message
35.     sys.exit()
36.
37. # suppression de la table personnes si elle existe
38. # si elle n'existe pas, une erreur se produira
39. # on l'ignore
40. requete="drop table personnes"
41. try:
42.     executeSQL(connexion,requete)
43. except:
44.     pass
```


Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Programme (mysqldb_03)

```
45. # création de la table personnes
46. requete="create table personnes (prenom varchar(30) NOT NULL, nom varchar(30) NOT NULL, age integer NOT NULL,
47. try:
48.     executeSQL(connexion,requete)
49. except MySQLdb.OperationalError,message:
50.     print message
51.     sys.exit()
52. # on se deconnecte et on quitte
53. try:
54.     connexion.close()
55. except MySQLdb.OperationalError,message:
56.     print message
57. sys.exit()
```

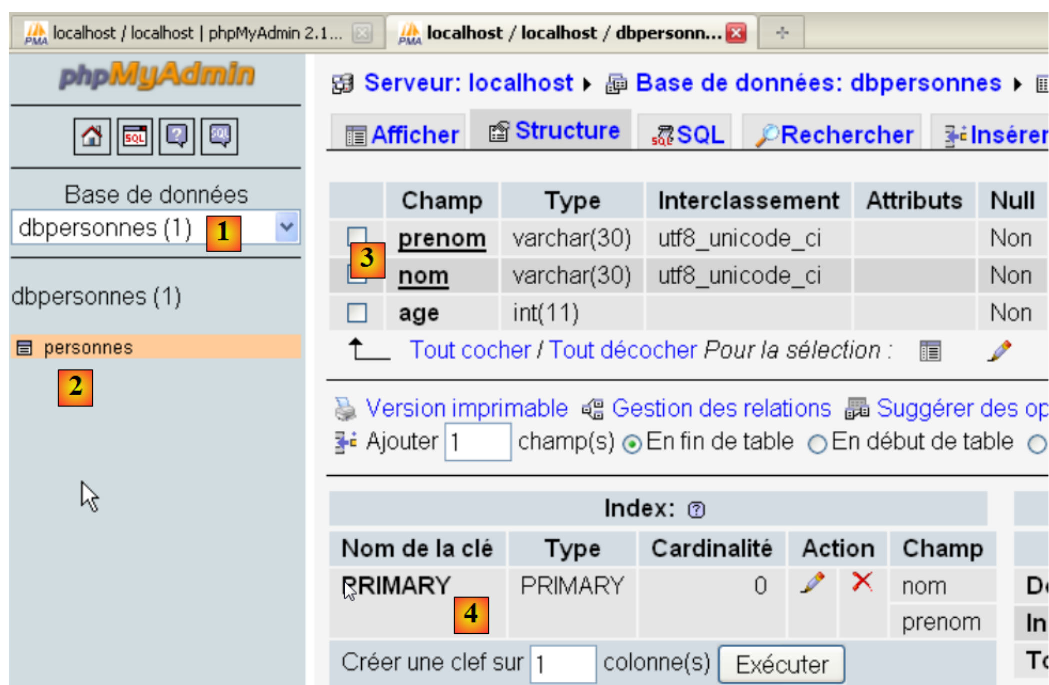
Notes :

- ligne 7 : la fonction *executeSQL* exécute une requête SQL sur une connexion ouverte ;
- ligne 10 : les opérations SQL sur la connexion se font au travers d'un objet particulier appelé *curseur ;*
- ligne 10 : obtention d'un curseur ;
- ligne 13 : exécution de la requête SQL ;
- ligne 14 : la transaction courante est validée ;
- ligne 15 : en cas d'exception, le message d'erreur est récupéré dans la variable *erreur ;*
- ligne 16 : la transaction courante est invalidée ;
- ligne 17 : l'exception est relancée ;
- ligne 19 : qu'il y ait erreur ou non, le curseur est fermé. Cela libère les ressources qui lui sont associées.

Résultats

```
1. create table personnes (prenom varchar(30) NOT NULL, nom varchar(30) NOT NULL, age integer NOT NULL,
 primary key(nom,prenom)) : requete reussie
```

Vérification avec phpMyAdmin :



- la base de données [dbpersonnes] [1] a une table [personnes] [2] qui a la structure [3] et la clé primaire [4].

Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

X-F - Remplissage de la table [personnes]

Après avoir créé précédemment la table [personnes], maintenant nous la remplissons.

Programme (mysqldb_04)

```
1. # import du module MySQLdb
2. import sys
3. sys.path.append("D:\Programs\ActivePython\site-packages")
4. import MySQLdb
5. # autres modules
6. import re
7.
8. # ----------------------------------------------------------------------------------
9. def executerCommandes(HOTE,ID,PWD,BASE,SQL,suivi=False,arret=True):
10.     # utilise la connexion (HOTE,ID,PWD,BASE)
11.     # exécute sur cette connexion les commandes SQL contenues dans le fichier texte SQL
12.     # ce fichier est un fichier de commandes SQL à exécuter à raison d'une par ligne
13.
   # si suivi=True alors chaque exécution d'un ordre SQL fait l'objet d'un affichage indiquant sa réussite ou son
14.
   # si arret=True, la fonction s'arrête sur la 1re erreur rencontrée sinon elle exécute toutes les commandes SQL
15.     # la fonction rend une liste (nb d'erreurs, erreur1, erreur2...)
16.
17.     # on vérifie la présence du fichier SQL
18.     data=None
19.     try:
20.         data=open(SQL,"r")
21.     except:
22.         return [1,"Le fichier %s n'existe pas" % (SQL)]
23.
24.     # connexion
25.     try:
26.         connexion=MySQLdb.connect(host=HOTE,user=ID,passwd=PWD,db=BASE)
27.     except MySQLdb.OperationalError,erreur:
28.         return [1,"Erreur lors de la connexion a MySql sous l'identite (%s,%s,%s,%s) : %s" % (HOTE,
 ID, PWD, BASE, erreur)]
29.
30.     # on demande un curseur
31.     curseur=connexion.cursor()
32.     # exécution des requêtes SQL contenues dans le fichier SQL
33.     # on les met dans un tableau
34.     requetes=data.readlines()
35.     # on les exécute une à une - au départ pas d'erreur
36.     erreurs=[0]
37.     for i in range(len(requetes)):
38.         # on mémorise la requête courante
39.         requete=requetes[i]
40.         # a-t-on une requête vide ? Si oui, on passe à la requête suivante
41.         if re.match(r"^\s*$",requete):
42.             continue
43.         # exécution de la requête i
44.         erreur=""
45.         try:
46.             curseur.execute(requete)
47.         except Exception, erreur:
48.             pass
49.         #y a-t-il eu une erreur ?
50.         if erreur:
51.             # une erreur de plus
52.             erreurs[0]+=1
53.             # msg d'erreur
54.             msg="%s : Erreur (%s)" % (requete,erreur)
55.             erreurs.append(msg)
56.             # suivi écran ou non ?
57.             if suivi:
58.                 print msg
59.             # on s'arrête ?
60.             if arret:
61.                 return erreurs
```


Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.
http://tahe.developpez.com/divers/python/exemples/

Programme (mysqldb_04)

```
62.          else:
63.              if suivi:
64.                  print "%s : Execution reussie" % (requete)
65.      # fermeture connexion et libération des ressources
66.      curseur.close()
67.      connexion.commit()
68.      # on se deconnecte
69.      try:
70.          connexion.close()
71.      except MySQLdb.OperationalError,erreur:
72.          # une erreur de plus
73.          erreurs[0]+=1
74.          # msg d'erreur
75.          msg="%s : Erreur (%s)" % (requete,erreur)
76.          erreurs.append(msg)
77.
78.      # retour
79.      return erreurs
80.
81.
82. # ---------------------------------------------- main
83. # connexion à la base MySql
84. # l'identité de l'utilisateur
85. ID="admpersonnes"
86. PWD="nobody"
87. # la machine hôte du SGBD
88. HOTE="localhost"
89. # identité de la base
90. BASE="dbpersonnes"
91. # identité du fichier texte des commandes SQL à exécuter
92. TEXTE="sql.txt";
93.
94. # création et remplissage de la table
95. erreurs=executerCommandes(HOTE,ID,PWD,BASE,TEXTE,True,False)
96. #affichage nombre d'erreurs
97. print "il y a eu %s erreur(s)" % (erreurs[0])
98. for i in range(1,len(erreurs)):
99.     print erreurs[i]
```

Le fichier sql.txt

```
1. drop table personnes
2. create table personnes (prenom varchar(30) not null, nom varchar(30) not null, age integer not null,
 primary key (nom,prenom))
3. insert into personnes values('Paul','Langevin',48)
4. insert into personnes values ('Sylvie','Lefur',70)
5. xx
6.
7. insert into personnes values ('Pierre','Nicazou',35)
8. insert into personnes values ('Geraldine','Colou',26)
9. insert into personnes values ('Paulette','Girond',56)
```

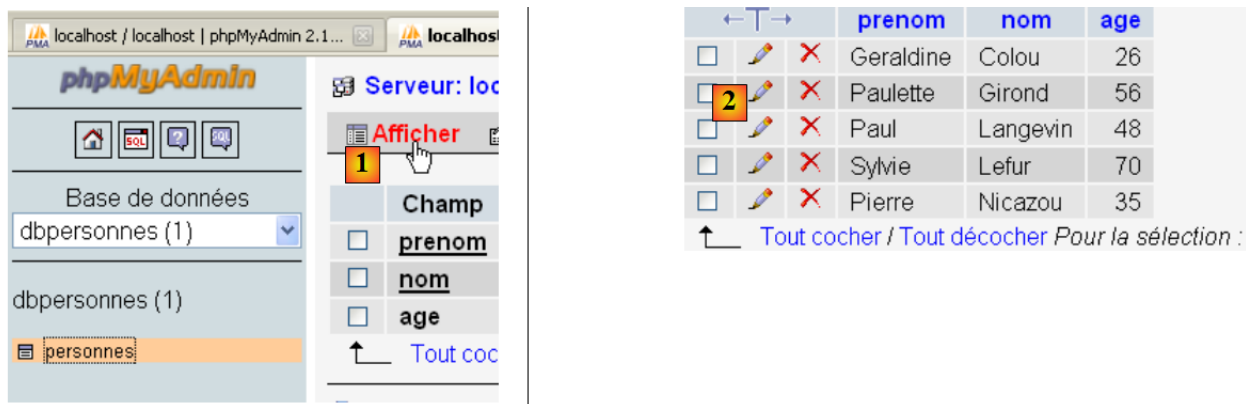
Une erreur a été volontairement insérée en ligne 5.

Les résultats écran

```
1. drop table personnes : Execution reussie
2. create table personnes (prenom varchar(30) not null, nom varchar(30) not null, age integer not null,
 primary key (nom,prenom)) : Execution reussie
3. insert into personnes values('Paul','Langevin',48) : Execution reussie
4. insert into personnes values ('Sylvie','Lefur',70) : Execution reussie
5. xx : Erreur ((1064, "You have an error in your SQL syntax; check the manual that corresponds to your
 MySQL server version for the right syntax to use near 'xx' at
6.  line 1"))
7. insert into personnes values ('Pierre','Nicazou',35) : Execution reussie
8. insert into personnes values ('Geraldine','Colou',26) : Execution reussie
9. insert into personnes values ('Paulette','Girond',56) : Execution reussie
10. il y a eu 1 erreur(s)
11. xx : Erreur ((1064, "You have an error in your SQL syntax; check the manual that corresponds to
 your MySQL server version for the right syntax to use near 'xx' at line 1"))
```


Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est
MCours.com

Vérification avec phpMyAdmin :



- en [1], le lien [Afficher] permet d'avoir le contenu de la table [personnes] [2].

X-G - Exécution de requêtes SQL quelconques

Le script suivant permet d'exécuter un fichier de commandes SQL et d'afficher le résultat de chacune d'elles :

- le résultat du SELECT si l'ordre est un SELECT ;
- le nombre de lignes modifiées si l'ordre est INSERT, UPDATE, DELETE.

Programme (mysqldb_05)

```
1. # import du module MySQLdb
2. import sys
3. sys.path.append("D:\Programs\ActivePython\site-packages")
4. import MySQLdb
5. # autres modules
6. import re
7.
8. def cutNewLineChar(ligne):
9.     # on supprime la marque de fin de ligne de [ligne] si elle existe
10.     l=len(ligne)
11.     while(ligne[l-1]=="\n" or ligne[l-1]=="\r"):
12.         l-=1
13.     return(ligne[0:l])
14.
15. # ---------------------------------------------------------------------------------
16. def afficherInfos(curseur):
17.     # affiche le résultat d'une requête SQL
18.     # s'agissait-il d'un select ?
19.     if curseur.description:
20.         # il y a une description - donc c'est un select
21.         # description[i] est la description de la colonne n° i du select
22.         # description[i][0] est le nom de la colonne n° i du select
23.         # on affiche le nom des champs
24.         titre=""
25.         for i in range(len(curseur.description)):
26.             titre+=curseur.description[i][0]+","
27.         # on affiche la liste des champs sans la virgule de fin
28.         print titre[0:len(titre)-1]
29.         # ligne séparatrice
30.         print "-"*(len(titre)-1)
31.         # ligne courante du select
32.         ligne=curseur.fetchone()
33.         while ligne:
34.             print ligne
35.             # ligne suivante du select
36.             ligne=curseur.fetchone()
37.     else:
```


Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Programme (mysqldb_05)

```
38.         # il n'y a pas de champ - ce n'était pas un select
39.         print "%s lignes(s) a (ont) ete modifiee(s)" % (curseur.rowcount)
40.
41.
42. # --------------------------------------------------------------------------------
43. def executerCommandes(HOTE,ID,PWD,BASE,SQL,suivi=False,arret=True):
44.     # utilise la connexion (HOTE,ID,PWD,BASE)
45.     # exécute sur cette connexion les commandes SQL contenues dans le fichier texte SQL
46.     # ce fichier est un fichier de commandes SQL à exécuter à raison d'une par ligne
47.
  # si suivi=1 alors chaque exécution d'un ordre SQL fait l'objet d'un affichage indiquant sa réussite ou son éch
48.
  # si arret=1, la fonction s'arrête sur la 1re erreur rencontrée sinon elle exécute toutes les commandes SQL
49.     # la fonction rend un tableau (nb d'erreurs, erreur1, erreur2...)
50.
51.     # on vérifie la présence du fichier SQL
52.     data=None
53.     try:
54.         data=open(SQL,"r")
55.     except:
56.         return [1,"Le fichier %s n'existe pas" % (SQL)]
57.
58.     # connexion
59.     try:
60.         connexion=MySQLdb.connect(host=HOTE,user=ID,passwd=PWD,db=BASE)
61.     except MySQLdb.OperationalError,erreur:
62.         return [1,"Erreur lors de la connexion a MySql sous l'identite (%s,%s,%s,%s) : %s" % (HOTE,
 ID, PWD, BASE, erreur)]
63.
64.     # on demande un curseur
65.     curseur=connexion.cursor()
66.     # exécution des requêtes SQL contenues dans le fichier SQL
67.     # on les met dans un tableau
68.     requetes=data.readlines()
69.     # on les exécute une à une - au départ pas d'erreur
70.     erreurs=[0]
71.     for i in range(len(requetes)):
72.         # on mémorise la requête courante
73.         requete=requetes[i]
74.         # a-t-on une requête vide ? Si oui, on passe à la requête suivante
75.         if re.match(r"^\s*$",requete):
76.             continue
77.         # exécution de la requête i
78.         erreur=""
79.         try:
80.             curseur.execute(requete)
81.         except Exception, erreur:
82.             pass
83.         #y a-t-il eu une erreur ?
84.         if erreur:
85.             # une erreur de plus
86.             erreurs[0]+=1
87.             # msg d'erreur
88.             msg="%s : Erreur (%s)" % (requete,erreur)
89.             erreurs.append(msg)
90.             # suivi écran ou non ?
91.             if suivi:
92.                 print msg
93.             # on s'arrête ?
94.             if arret:
95.                 return erreurs
96.         else:
97.             if suivi:
98.                 print "%s : Execution reussie" % (requete)
99.                 # infos sur le résultat de la requête exécutée
100.                 afficherInfos(curseur)
101.
102.     # fermeture connexion et libération des ressources
103.     curseur.close()
104.     connexion.commit()
105.     # on se deconnecte
```


Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Programme (mysqldb_05)

```
106.     try:
107.         connexion.close()
108.     except MySQLdb.OperationalError,erreur:
109.         # une erreur de plus
110.         erreurs[0]+=1
111.         # msg d'erreur
112.         msg="%s : Erreur (%s)" % (requete,erreur)
113.         erreurs.append(msg)
114.
115.     # retour
116.     return erreurs
117.
118.
119. # ---------------------------------------------- main
120. # connexion à la base MySql
121. # l'identité de l'utilisateur
122. ID="admpersonnes"
123. PWD="nobody"
124. # la machine hôte du SGBD
125. HOTE="localhost"
126. # identité de la base
127. BASE="dbpersonnes"
128. # identité du fichier texte des commandes SQL à exécuter
129. TEXTE="sql2.txt"
130.
131.
132. # création et remplissage de la table
133. erreurs=executerCommandes(HOTE,ID,PWD,BASE,TEXTE,True,False)
134. #affichage nombre d'erreurs
135. print "il y a eu %s erreur(s)" % (erreurs[0])
136. for i in range(1,len(erreurs)):
137.     print erreurs[i]
```

Notes :

- la nouveauté est ligne 100 du script : après exécution d'un ordre SQL, on demande des informations sur le curseur utilisé par cette requête. Ces informations sont données par la fonction *afficheInfos* des lignes 16-40 ;
- lignes 19 et 39 : si la requête SQL exécutée était un SELECT, l'attribut [curseur.description] est un tableau dont l'élément n° i décrit le champ n° i du résultat du SELECT. Sinon, l'attribut [curseur.rowcount] (ligne 39) est le nombre de lignes modifiées par la requête INSERT, UPDATE ou DELETE ;
- lignes 32 et 36 : la méthode [curseur.fetchone] permet de récupérer la ligne courante du SELECT. Il existe une méthode [curseur.fetchall] qui permet de les récupérer d'un seul coup.

Le fichier des requêtes exécutées

```
1. select * from personnes
2. select nom,prenom from personnes order by nom asc, prenom desc
3. select * from personnes where age between 20 and 40 order by age desc, nom asc, prenom asc
4. insert into personnes values('Josette','Bruneau',46)
5. update personnes set age=47 where nom='Bruneau'
6. select * from personnes where nom='Bruneau'
7. delete from personnes where nom='Bruneau'
8. select * from personnes where nom='Bruneau'
9. xselect * from personnes where nom='Bruneau'
```

Les résultats écran

```
1. select * from personnes : Execution reussie
2. prenom,nom,age
3. ---------------
4. ('Geraldine', 'Colou', 26L)
5. ('Paulette', 'Girond', 56L)
6. ('Paul', 'Langevin', 48L)
7. ('Sylvie', 'Lefur', 70L)
8. ('Pierre', 'Nicazou', 35L)
9. select nom,prenom from personnes order by nom asc, prenom desc : Execution reussie
10. nom,prenom
11. -----------
12. ('Colou', 'Geraldine')
```


Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Les résultats écran

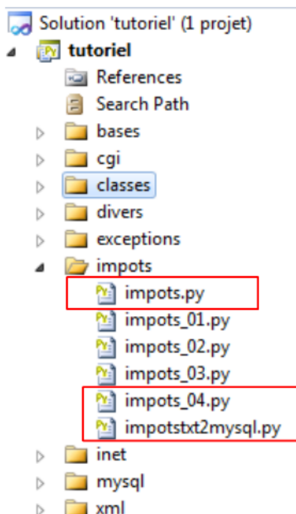
```
13. ('Girond', 'Paulette')
14. ('Langevin', 'Paul')
15. ('Lefur', 'Sylvie')
16. ('Nicazou', 'Pierre')
17. select * from personnes where age between 20 and 40 order by age desc, nom asc,prenom asc :
 Execution reussie
18. prenom,nom,age
19. ---------------
20. ('Pierre', 'Nicazou', 35L)
21. ('Geraldine', 'Colou', 26L)
22. insert into personnes values('Josette','Bruneau',46) : Execution reussie
23. 1 lignes(s) a (ont) ete modifiee(s)
24. update personnes set age=47 where nom='Bruneau' : Execution reussie
25. 1 lignes(s) a (ont) ete modifiee(s)
26. select * from personnes where nom='Bruneau' : Execution reussie
27. prenom,nom,age
28. ---------------
29. ('Josette', 'Bruneau', 47L)
30. delete from personnes where nom='Bruneau' : Execution reussie
31. 1 lignes(s) a (ont) ete modifiee(s)
32. select * from personnes where nom='Bruneau' : Execution reussie
33. prenom,nom,age
34. ---------------
35. xselect * from personnes where nom='Bruneau' : Erreur ((1064, "You have an error in your SQL
 syntax; check the manual that corresponds to your MySQL server version for the right syntax to use
 near 'xselect * from personnes where nom='Bruneau'' at line 1"))
```

Vérification avec PhpMyadmin :

prenom	nom	age
Geraldine	Colou	26
Paulette	Girond	56
Paul	Langevin	48
Sylvie	Lefur	70
Pierre	Nicazou	35

Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

XI - Exercice [IMPOTS] avec MySQL



XI-A - Transfert d'un fichier texte dans une table MySQL

Le script à venir va transférer les données du fichier texte suivant :

```
1. 12620:13190:15640:24740:31810:39970:48360:55790:92970:127860:151250:172040:195000:0
2. 0:0.05:0.1:0.15:0.2:0.25:0.3:0.35:0.4:0.45:0.5:0.55:0.6:0.65
3. 0:631:1290.5:272.5:3309.5:4900:6898.5:9316.5:12106:16754.5:23147.5:30710:39312:49062
```

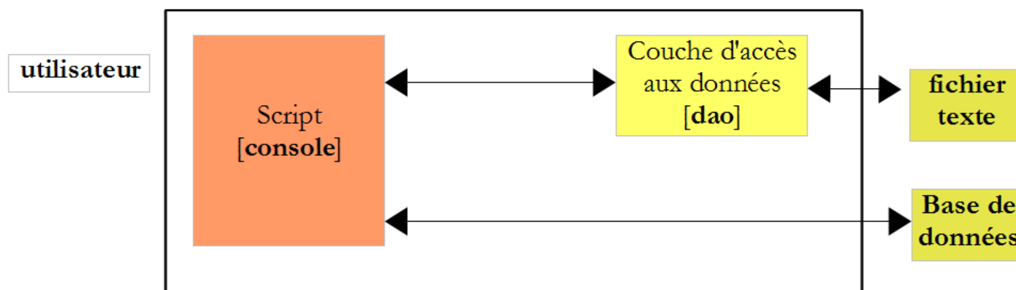
dans la table [impots] de la base MySQL [dbimpots] suivante :

limites	coeffR	coeffN
12620.00	0.00	0.00
13190.00	0.05	631.00
15640.00	0.10	1290.50
24740.00	0.15	2072.50
31810.00	0.20	3309.50
39970.00	0.25	4900.00
48360.00	0.30	6898.50
55790.00	0.35	9316.50
92970.00	0.40	12106.00
127860.00	0.45	16754.50
151250.00	0.50	23147.50
172040.00	0.55	30710.00
195000.00	0.60	39312.00
0.00	0.65	49062.00

La connexion à la base [dbimpots] se fera avec l'identité (root,"").

Nous allons utiliser l'architecture suivante :

Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.



Le script [console] que nous allons écrire va utiliser la classe [ImpotsFile] pour accéder aux données du fichier texte. L'accès en écriture à la base de données se fera avec les méthodes vues précédemment.

Le code du script est le suivant :

Programme (impotstxt2mysql)

```
1. # -*- coding=utf-8 -*-
2.
3. # import du module des classes Impots
4. from impots import *
5. import re
6.
7. # --------------------------------------------------------------------------
8. def copyToMysql(limites,coeffR,coeffN,HOTE,USER,PWD,BASE,TABLE):
9.     # copie les 3 tableaux numériques limites, coeffR, coeffN
10.     # dans la table TABLE de la base mysql BASE
11.     # la base mysql est sur la machine HOTE
12.     # l'utilisateur est identifié par USER et PWD
13.
14.     # on met les requêtes SQL dans une liste
15.     # suppression de la table
16.     requetes=["drop table %s" % (TABLE)]
17.     # création de la table
18.     requete="create table %s (limites decimal(10,2), coeffR decimal(6,2), coeffN decimal(10,2))" %
 (TABLE)
19.     requetes.append(requete)
20.     # remplissage
21.     for i in range(len(limites)):
22.         # requête d'insertion
23.         requetes.append("insert into %s (limites,coeffR,coeffN) values (%s,%s,%s)" %
 (TABLE,limites[i],coeffR[i],coeffN[i]))
24.     # on exécute les ordres SQL
25.     return executerCommandes(HOTE,USER,PWD,BASE,requetes,False,False)
26.
27.
28. def executerCommandes(HOTE,ID,PWD,BASE,requetes,suivi=False,arret=True):
29.     # utilise la connexion (HOTE,ID,PWD,BASE)
30.     # exécute sur cette connexion les commandes SQL contenues dans la liste requetes
31.
  # si suivi=True alors chaque exécution d'un ordre SQL fait l'objet d'un affichage indiquant sa réussite ou son
32.
  # si arret=False, la fonction s'arrête sur la 1re erreur rencontrée sinon elle exécute toutes les commandes SQL
33.     # la fonction rend une liste (nb d'erreurs, erreur1, erreur2...)
34.
35.     # connexion
36.     try:
37.         connexion=MySQLdb.connect(host=HOTE,user=ID,passwd=PWD,db=BASE)
38.     except MySQLdb.OperationalError,erreur:
39.         return [1,"Erreur lors de la connexion a MySql sous l'identite (%s,%s,%s,%s) : %s" % (HOTE,
 ID, PWD, BASE, erreur)]
40.
41.     # on demande un curseur
42.     curseur=connexion.cursor()
43.     # exécution des requêtes SQL contenues dans la liste requetes
44.     # on les exécute - au départ pas d'erreur
45.     erreurs=[0]
```


Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.
http://tahe.developpez.com/divers/python/exemples/

Programme (impotstxt2mysql)

```
46.     for i in range(len(requetes)):
47.         # on met la requête dans une variable locale
48.         requete=requetes[i]
49.         # a-t-on une requête vide ?
50.         if re.match(r"^\s*$",requete):
51.             continue
52.         # exécution de la requête i
53.         erreur=""
54.         try:
55.             curseur.execute(requete)
56.         except Exception, erreur:
57.             pass
58.         # y a-t-il eu une erreur ?
59.         if erreur:
60.             # une erreur de plus
61.             erreurs[0]+=1
62.             # msg d'erreur
63.             msg="%s : Erreur (%s)" % (requete[0:len(requete)-1],erreur)
64.             erreurs.append(msg)
65.             # suivi écran ou non ?
66.             if suivi:
67.                 print msg
68.             # on s'arrête ?
69.             if arret:
70.                 return erreurs
71.         else:
72.             if suivi:
73.                 # on affiche la requête sans sa marque de fin de ligne
74.                 print "%s : Execution reussie" % (cutNewLineChar(requete))
75.                 # infos sur le résultat de la requête exécutée
76.                 afficherInfos(curseur)
77.     # fermeture connexion
78.     try:
79.         connexion.commit()
80.         connexion.close()
81.     except MySQLdb.OperationalError,erreur:
82.         # une erreur de plus
83.         erreurs[0]+=1
84.         # msg d'erreur
85.         msg="%s : Erreur (%s)" % (requete,erreur)
86.         erreurs.append(msg)
87. 
88.     # retour
89.     return erreurs
90. 
91. 
92. # ------------------------------------------------ main
93. # l'identité de l'utilisateur
94. USER="root"
95. PWD=""
96. # la machine hôte du SGBD
97. HOTE="localhost"
98. # identité de la base
99. BASE="dbimpots"
100. # identité de la table des données
101. TABLE="impots"
102. # le fichier des données
103. IMPOTS="impots.txt"
104. 
105. # instanciation couche [dao]
106. try:
107.     dao=ImpotsFile(IMPOTS)
108. except (IOError, ImpotsError) as infos:
109.     print ("Une erreur s'est produite : {0}".format(infos))
110.     sys.exit()
111. 
112. # on transfère les données récupérées dans une table mysql
113. erreurs=copyToMysql(dao.limites,dao.coeffR,dao.coeffN,HOTE,USER,PWD,BASE,TABLE)
114. if erreurs[0]:
115.     for i in range(1,len(erreurs)):
116.         print erreurs[i]
```


Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Programme (impotstxt2mysql)

```
117. else:
118.     print "Transfert opere"
119. # fin
120. sys.exit()
```

Notes :

- lignes 106-110 : on instancie la classe [ImpotsFile] présentée page ;
- ligne 113 : les tableaux *limites*, *coeffR*, *coeffN* sont transférés dans une base MySQL ;
- ligne 8 : la fonction *copyToMysql* exécute ce transfert. La fonction *copyToMysql* crée un tableau des requêtes à exécuter et les fait exécuter par la fonction *executerCommandes*, ligne 25 ;
- lignes 28-89 : la fonction *executerCommandes* est celle déjà présentée précédemment avec une différence : au lieu d'être dans un fichier texte, les requêtes sont dans une liste ;

Le fichier texte impots.txt

```
1. 12620:13190:15640:24740:31810:39970:48360:55790:92970:127860:151250:172040:195000:0
2. 0:0.05:0.1:0.15:0.2:0.25:0.3:0.35:0.4:0.45:0.5:0.55:0.6:0.65
3. 0:631:1290.5:272.5:3309.5:4900:6898.5:9316.5:12106:16754.5:23147.5:30710:39312:49062
```

Les résultats écran :

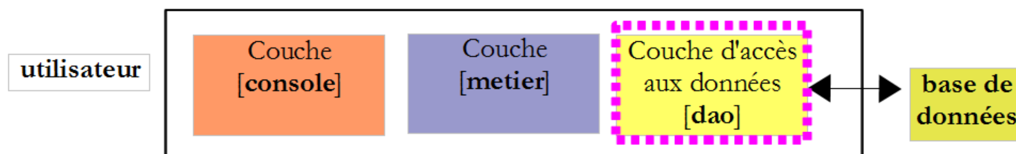
Transfert opéré

Vérification avec phpMyAdmin :

limites	coeffR	coeffN
12620.00	0.00	0.00
13190.00	0.05	631.00
15640.00	0.10	1290.50
24740.00	0.15	2072.50
31810.00	0.20	3309.50
39970.00	0.25	4900.00
48360.00	0.30	6898.50
55790.00	0.35	9316.50
92970.00	0.40	12106.00
127860.00	0.45	16754.50
151250.00	0.50	23147.50
172040.00	0.55	30710.00
195000.00	0.60	39312.00
0.00	0.65	49062.00

XI-B - Le programme de calcul de l'impôt

Maintenant que les données nécessaires au calcul de l'impôt sont dans une base de données, nous pouvons écrire le script de calcul de l'impôt. Nous utilisons de nouveau une architecture trois couches :



La nouvelle couche [DAO] sera connectée au SGBD MySQL et sera implémentée par la classe [ImpotsMySQL]. Elle offrira à la couche [metier] la même interface que précédemment constituée de l'unique méthode *getData* qui rend le tuple (*limites, coeffR, coeffN*). Ainsi la couche [metier] ne changera pas vis-à-vis de la version précédente.

Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

XI-C - La classe [ImpotsMySQL]

La couche [DAO] est maintenant implémentée par la classe [ImpotsMySQL] suivante (fichier impots.py) :

ImpotsMySQL

```
1. class ImpotsMySQL:
2.
3.     # constructeur
4.     def __init__(self,HOTE,USER,PWD,BASE,TABLE):
5.         # initialise les attributs limites, coeffR, coeffN
6.         # les données nécessaires au calcul de l'impôt ont été placées dans la table mysqL TABLE
7.         # appartenant à la base BASE. La table a la structure suivante
8.         # limites decimal(10,2), coeffR decimal(10,2), coeffN decimal(10,2)
9.         # la connexion à la base mysql de la machine HOTE se fait sous l'identité (USER,PWD)
10.         # lance une exception si éventuelle erreur
11.
12.         # connexion à la base mysql
13.         connexion=MySQLdb.connect(host=HOTE,user=USER,passwd=PWD,db=BASE)
14.         # on demande un curseur
15.         curseur=connexion.cursor()
16.
17.         # lecture en bloc de la table TABLE
18.         requete="select limites,coeffR,coeffN from %s" % (TABLE)
19.         # exécute la requête [requete] sur la base [base] de la connexion [connexion]
20.         curseur.execute(requete)
21.         # exploitation du résultat de la requête
22.         ligne=curseur.fetchone()
23.         self.limites=[]
24.         self.coeffR=[]
25.         self.coeffN=[]
26.         while(ligne):
27.             # ligne courante
28.             self.limites.append(ligne[0])
29.             self.coeffR.append(ligne[1])
30.             self.coeffN.append(ligne[2])
31.             # ligne suivante
32.             ligne=curseur.fetchone()
33.         # déconnexion
34.         connexion.close()
35.
36.     def getData(self):
37.         return (self.limites, self.coeffR, self.coeffN)
```

Notes :

- ligne 18 : la requête SQL SELECT qui demande les données de la base de données MySql. Les lignes résultant du SELECT sont ensuite exploitées une par une par [curseur.fetchone] (lignes 22 et 32) pour créer les tableaux *limites*, *coeffR*, *coeffN* (lignes 28-30) ;
- la méthode *getData* de l'interface de la couche [DAO].

XI-D - Le script console

Le code du script console (impots_04) est le suivant :

script console (impots_04)

```
1. # -*- coding=utf-8 -*-
2.
3. # import du module des classes Impots*
4. from impots import *
5.
6. # ------------------------------------------------ main
7. # l'identité de l'utilisateur
```


Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

script console (impots_04)

```
8. USER="root"
9. PWD=""
10. # la machine hôte du sgbd
11. HOTE="localhost"
12. # identité de la base
13. BASE="dbimpots"
14. # identité de la table des données
15. TABLE="impots"
16. # fichier d'entrées
17. DATA="data.txt"
18. # fichier de sorties
19. RESULTATS="resultats.txt"
20.
21. # instanciation couche [metier]
22. try:
23.     metier=ImpotsMetier(ImpotsMySQL(HOTE,USER,PWD,BASE,TABLE))
24. except (IOError, ImpotsError) as infos:
25.     print ("Une erreur s'est produite : {0}".format(infos))
26.     sys.exit()
27.
28. # les données nécessaires au calcul de l'impôt ont été placées dans le fichier IMPOTS
29. # à raison d'une ligne par tableau sous la forme
30. # val1:val2:val3...
31.
32. # lecture des données
33. try:
34.     data=open(DATA,"r")
35. except:
36.     print "Impossible d'ouvrir en lecture le fichier des donnees [DATA]"
37.     sys.exit()
38.
39. # ouverture fichier des résultats
40. try:
41.     resultats=open(RESULTATS,"w")
42. except:
43.     print "Impossible de creer le fichier des résultats [RESULTATS]"
44.     sys.exit()
45.
46. # utilitaires
47. u=Utilitaires()
48.
49. # on exploite la ligne courante du fichier des données
50. ligne=data.readline()
51. while(ligne != ''):
52.     # on enlève l'éventuelle marque de fin de ligne
53.     ligne=u.cutNewLineChar(ligne)
54.     # on récupère les 3 champs marié:enfants:salaire qui forment la ligne
55.     (marie,enfants,salaire)=ligne.split(",")
56.     enfants=int(enfants)
57.     salaire=int(salaire)
58.     # on calcule l'impôt
59.     impot=metier.calculer(marie,enfants,salaire)
60.     # on inscrit le résultat
61.     resultats.write("{0}:{1}:{2}:{3}\n".format(marie,enfants,salaire,impot))
62.     # on lit une nouvelle ligne
63.     ligne=data.readline()
64. # on ferme les fichiers
65. data.close()
66. resultats.close()
```

Notes :

- ligne 23 : instanciation des couches [DAO] et [metier] ;
- le reste du code est connu.

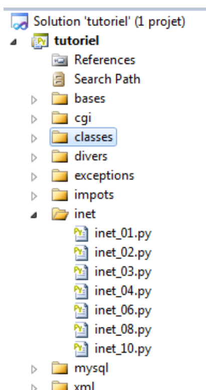
Résultats

Les mêmes qu'avec les versions précédentes de l'exercice.

Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

XII - Les fonctions réseau de Python

Nous abordons maintenant les fonctions réseau de Python qui nous permettent de faire de la programmation TCP / IP (Transfer Control Protocol / Internet Protocol).



XII-A - Obtenir le nom ou l'adresse IP d'une machine de l'Internet

Programme (inet_01)

```
1. import sys, socket
2.
3. #-----------------------------------------------
4. def getIPandName(nomMachine):
5.     #nomMachine : nom de la machine dont on veut l'adresse IP
6.     # nomMachine-->adresse IP
7.     try:
8.         ip=socket.gethostbyname(nomMachine)
9.         print "ip[%s]=%s" % (nomMachine,ip)
10.     except socket.error, erreur:
11.         print "ip[%s]=%s" % (nomMachine,erreur)
12.         return
13.
14.     # adresse IP --> nomMachine
15.     try:
16.         name=socket.gethostbyaddr(ip)
17.         print "name[%s]=%s" % (ip,name[0])
18.     except socket.error, erreur:
19.         print "name[%s]=%s" % (ip,erreur)
20.         return
21.
22.
23. # ---------------------------------------- main
24.
25. # constantes
26. HOTES=["istia.univ-angers.fr","www.univ-angers.fr","www.ibm.com","localhost","","xx"]
27.
28. # adresses IP des machines de HOTES
29. for i in range(len(HOTES)):
30.     getIPandName(HOTES[i])
31. # fin
32. sys.exit()
```

Notes :

- ligne 1 : les fonctions réseau de Python sont encapsulées dans le module *socket*.

Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Résultats

```
1. ip[istia.univ-angers.fr]=193.49.146.171
2. name[193.49.146.171]=istia.istia.univ-angers.fr
3. ip[www.univ-angers.fr]=193.49.144.40
4. name[193.49.144.40]=ametys-fo.univ-angers.fr
5. ip[www.ibm.com]=129.42.58.216
6. name[129.42.58.216]=[Errno 11004] host not found
7. ip[localhost]=127.0.0.1
8. name[127.0.0.1]=Gportpers3.ad.univ-angers.fr
9. ip[xx]=[Errno 11004] getaddrinfo failed
```

XII-B - Un client Web

Un script permettant d'avoir le contenu de la page index d'un site Web.

Programme (inet_02)

```
1. import sys,socket
2.
3.   #----------------------------------------------------------------------
4. def getIndex(site):
5.     # lit l'URL site/ et la stocke dans le fichier site.html
6.
7.     # au départ pas d'erreur
8.     erreur=""
9.     # création du fichier site.html
10.     try:
11.         html=open("%s.html" % (site),"w")
12.     except IOError, erreur:
13.         pass
14.     # erreur ?
15.     if erreur:
16.         return "Erreur (%s) lors de la création du fichier %s.html" % (erreur,site)
17.
18.     # ouverture d'une connexion sur le port 80 de site
19.     try:
20.         connexion=socket.create_connection((site,80))
21.     except socket.error, erreur:
22.         pass
23.     # retour si erreur
24.     if erreur:
25.         return "Echec de la connexion au site (%s,80) : %s" % (site,erreur)
26.
27.     # connexion représente un flux de communication bidirectionnel
28.     # entre le client (ce programme) et le serveur Web contacté
29.     # ce canal est utilisé pour les échanges de commandes et d'informations
30.     # le protocole de dialogue est HTTP
31.
32.     # le client envoie la commande get pour demander l'URL /
33.     # syntaxe get URL HTTP/1.0
34.     # les entêtes (headers) du protocole HTTP doivent se terminer par une ligne vide
35.     connexion.send("GET / HTTP/1.0\n\n")
36.
37.     # le serveur va maintenant répondre sur le canal connexion. Il va envoyer toutes
38.     # ses données puis fermer le canal. Le client lit tout ce qui arrive de connexion
39.     # jusqu'à la fermeture du canal
40.     ligne=connexion.recv(1000)
41.     while(ligne):
42.         html.write(ligne)
43.         ligne=connexion.recv(1000)
44.
45.     # le client ferme la connexion à son tour
46.     connexion.close()
47.     # fermeture du fichier html
48.     html.close()
49.     # retour
50.     return "Transfert reussi de la page index du site %s" % (site)
51.
52. # --------------------------------------------- main
```


Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Programme (inet_02)

```
53.
54. # obtenir le texte HTML d'URL
55.
56. # liste de sites Web
57. SITES=("istia.univ-angers.fr","www.univ-angers.fr","www.ibm.com","xx")
58.
59. # lecture des pages index des sites du tableau SITES
60. for i in range(len(SITES)):
61.       # lecture page index du site SITES[i]
62.       resultat=getIndex(SITES[i])
63.       # affichage résultat
64.       print resultat
65.
66. # fin
67. sys.exit()
```

Notes :

- ligne 57 : la liste des URL des sites Web dont on veut la page index. Celle-ci est stockée dans le fichier texte [nomsite.html] ;
- ligne 62 : la fonction *getIndex* fait le travail ;
- ligne 4 : la fonction *getIndex ;*
- ligne 20 : la méthode *create_connection((site,port))* permet de créer une connexion avec un service TCP / IP travaillant sur le port *port* de la machine *site ;*
- ligne 35 : la méthode *send* permet d'envoyer des données au travers d'une connexion TCP / IP. Ici, c'est du texte qui est envoyé. Ce texte obéit au protocole HTTP (HyperText Transfer Protocol) ;
- ligne 40 : la méthode *recv* permet de recevoir des données au travers d'une connexion TCP / IP. Ici, la réponse du serveur Web est lue par blocs de 1000 caractères et enregistrée dans le fichier texte [nomsite.html].

Résultats

```
1. Transfert reussi de la page index du site istia.univ-angers.fr
2. Transfert reussi de la page index du site www.univ-angers.fr
3. Transfert reussi de la page index du site www.ibm.com
4. Echec de la connexion au site (xx,80) : [Errno 11001] getaddrinfo failed
```

Le fichier reçu pour le site [www.ibm.com] :

```
1. HTTP/1.1 302 Found
2. Date: Wed, 08 Jun 2011 15:43:56 GMT
3. Server: IBM_HTTP_Server
4. Content-Type: text/html
5. Expires: Fri, 01 Jan 1990 00:00:00 GMT
6. Pragma: no-cache
7. Cache-Control: no-cache, must-revalidate
8. Location: http://www.ibm.com/us/en/
9. Content-Length: 209
10. Kp-eeAlive: timeout=10, max=14
11. Connection: Keep-Alive
12.
13. <!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
14. <html><head>
15. <title>302 Found</title>
16. </head><body>
17. <h1>Found</h1>
18. <p>The document has moved <a href="http://www.ibm.com/us/en/">here</a>.</p>
19. </body></html>
```

- les lignes 1-11 sont les entêtes HTTP de la réponse du serveur ;
- ligne 1 : le serveur demande au client de se rediriger vers l'URL indiquée ligne 8 ;
- ligne 2 : date et heure de la réponse ;
- ligne 3 : identité du serveur Web ;
- ligne 4 : contenu envoyé par le serveur. Ici une page HTML qui commence ligne 13 ;

Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

- ligne 12 : la ligne vide qui termine les entêtes HTTP ;
- lignes 13-19 : la page HTML envoyée par le serveur Web.

XII-C - Un client SMTP

Parmi les protocoles TCP / IP, SMTP Simple Mail Transfer Protocol) est le protocole de communication du service d'envoi de messages.

Programme (inet_03)

```
1. # -*- coding=utf-8 -*-
2.
3. import sys,socket
4.
5. #-----------------------------------------------------------------------
6. def getInfos(fichier):
7.
  # rend les informations (smtp,expéditeur,destinataire,message) prises dans le fichier texte [fichier]
8.     # ligne 1 : smtp, expéditeur, destinataire
9.     # lignes suivantes : le texte du message
10.
11.     # ouverture de [fichier]
12.     erreur=""
13.     try:
14.         infos=open(fichier,"r")
15.     except IOError, erreur:
16.         return ("Le fichier %s n'a pu etre ouvert en lecture : %s" % (fichier, erreur))
17.
18.     # lecture de la 1re ligne
19.     ligne=infos.readline()
20.     # suppression de la marque de fin de ligne
21.     ligne=cutNewLineChar(ligne)
22.     # récupération des champs smtp,expéditeur,destinataire
23.     champs=ligne.split(",")
24.     # a-t-on le bon nombre de champs ?
25.     if len(champs)!=3 :
26.         return
 ("La ligne 1 du fichier %s (serveur smtp, expediteur, destinataire) a un nombre de champs incorrect" %
 (fichier))
27.     # "traitement" des informations récupérées
28.     # on enlève à chacun des 3 champs les "blancs" qui précèdent ou suivent l'information utile
29.     for i in range(3):
30.         champs[i]=champs[i].strip()
31.     # récupération des champs
32.     (smtpServer,expediteur,destinataire)=champs
33.     message=""
34.     # lecture reste du message
35.     ligne=infos.readline()
36.     while ligne!='':
37.         message+=ligne
38.         ligne=infos.readline()
39.     infos.close()
40.     # retour
41.     return ("",smtpServer,expediteur,destinataire,message)
42.
43. #-----------------------------------------------------------------------
44. def sendmail(smtpServer,expediteur,destinataire,message,verbose):
45.     # envoie message au serveur smtp smtpserver de la part de expéditeur
46.     # pour destinataire. Si verbose=True, fait un suivi des échanges client-serveur
47.
48.     # on récupère le nom du client
49.     try:
50.         client=socket.gethostbyaddr(socket.gethostbyname("localhost"))[0]
51.     except socket.error, erreur:
52.         return "Erreur IP / Nom du client : %s" % (erreur)
53.
54.     # ouverture d'une connexion sur le port 25 de smtpServer
55.     try:
```


Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.
http://tahe.developpez.com/divers/python/exemples/

Programme (inet_03)

```
56.         connexion=socket.create_connection((smtpServer,25))
57.     except socket.error, erreur:
58.         return "Echec de la connexion au site (%s,25) : %s" % (smtpServer,erreur)
59.
60.     # connexion représente un flux de communication bidirectionnel
61.     # entre le client (ce programme) et le serveur smtp contacté
62.     # ce canal est utilisé pour les échanges de commandes et d'informations
63.
64.     # après la connexion le serveur envoie un message de bienvenue qu'on lit
65.     erreur=sendCommand(connexion,"",verbose,1)
66.     if(erreur) :
67.         connexion.close()
68.         return erreur
69.     # cmde ehlo:
70.     erreur=sendCommand(connexion,"EHLO %s" % (client),verbose,1)
71.     if erreur :
72.         connexion.close()
73.         return erreur
74.     # cmde mail from:
75.     erreur=sendCommand(connexion,"MAIL FROM: <%s>" % (expediteur),verbose,1)
76.     if erreur :
77.         connexion.close()
78.         return erreur
79.     # cmde rcpt to:
80.     erreur=sendCommand(connexion,"RCPT TO: <%s>" % (destinataire),verbose,1)
81.     if erreur :
82.         connexion.close()
83.         return erreur
84.     # cmde data
85.     erreur=sendCommand(connexion,"DATA",verbose,1)
86.     if erreur :
87.         connexion.close()
88.         return erreur
89.     # préparation message à envoyer
90.     # il doit contenir les lignes
91.     # From: expéditeur
92.     # To: destinataire
93.     # ligne vide
94.     # Message
95.     # .
96.     data="From: %s\r\nTo: %s\r\n%s\r\n.\r\n" % (expediteur,destinataire,message)
97.     # envoi message
98.     erreur=sendCommand(connexion,data,verbose,0)
99.     if erreur :
100.         connexion.close()
101.         return erreur
102.     # cmde quit
103.     erreur=sendCommand(connexion,"QUIT",verbose,1)
104.     if erreur :
105.         connexion.close()
106.         return erreur
107.     # fin
108.     connexion.close()
109.     return "Message envoye"
110.
111. # --------------------------------------------------------------------------
112. def sendCommand(connexion,commande,verbose,withRCLF):
113.     # envoie commande dans le canal connexion
114.     # mode verbeux si verbose=1
115.     # si withRCLF=1, ajoute la séquence RCLF à commande
116.
117.     # données
118.     RCLF="\r\n" if withRCLF else ""
119.     # envoie cmde si commande non vide
120.     if commande:
121.         connexion.send("%s%s" % (commande,RCLF))
122.         # écho éventuel
123.         if verbose:
124.             affiche(commande,1)
125.         # lecture réponse de moins de 1000 caractères
126.         reponse=connexion.recv(1000)
```


Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Programme (inet_03)

```
127.         # écho éventuel
128.         if verbose:
129.             affiche(reponse,2)
130.         # récupération code erreur
131.         codeErreur=reponse[0:3]
132.         # erreur renvoyée par le serveur ?
133.         if int(codeErreur) >=500:
134.             return reponse[4:]
135.         # retour sans erreur
136.         return ""
137.
138. # ---------------------------------------------------------------------------
139. def affiche(echange,sens):
140.     # affiche échange ? l'écran
141.     # si sens=1 affiche -->echange
142.     # si sens=2 affiche <-- échange sans les 2 derniers caractères RCLF
143.     if sens==1:
144.         print "--> [%s]" % (echange)
145.         return
146.     elif sens==2:
147.         l=len(echange)
148.         print "<-- [%s]" % echange[0:l-2]
149.         return
150.
151.
152. # ---------------------------------------------------------------------------
153. def cutNewLineChar(ligne):
154.     # on supprime la marque de fin de ligne de [ligne] si elle existe
155.     l=len(ligne)
156.     while(ligne[l-1]=="\n" or ligne[l-1]=="\r"):
157.         l-=1
158.     return(ligne[0:l])
159.
160. # main ------------------------------------------------------------------
161.
162.  # client SMTP (SendMail Transfer Protocol) permettant d'envoyer un message
163. # les infos sont prises dans un fichier INFOS contenant les lignes suivantes
164. # ligne 1 : smtp, expéditeur, destinataire
165. # lignes suivantes : le texte du message
166.
167. # expéditeur:email expéditeur
168. # destinataire: email destinataire
169. # smtp: nom du serveur smtp à utiliser
170.
171.
172. # protocole de communication SMTP client-serveur
173. # -> client se connecte sur le port 25 du serveur smtp
174. # <- serveur lui envoie un message de bienvenue
175. # -> client envoie la commande EHLO: nom de sa machine
176. # <- serveur répond OK ou non
177. # -> client envoie la commande mail from: <expéditeur>
178. # <- serveur répond OK ou non
179. # -> client envoie la commande rcpt to: <destinataire>
180. # <- serveur répond OK ou non
181. # -> client envoie la commande data
182. # <- serveur répond OK ou non
183. # -
> client envoie toutes les lignes de son message et termine avec une ligne contenant le seul caractère .
184. # <- serveur répond OK ou non
185. # -> client envoie la commande quit
186. # <- serveur répond OK ou non
187.
188. # les réponses du serveur ont la forme xxx texte où xxx est un nombre à 3 chiffres. Tout nombre xxx >=500
189. # signale une erreur. La réponse peut comporter plusieurs lignes commençant toutes par xxx- sauf la dernière
190. # de la forme xxx(espace)
191.
192. # les lignes de texte échangées doivent se terminer par les caractères RC(#13) et LF(#10)
193.
194. # # les paramètres de l'envoi du courrier
195. MAIL="mail2.txt"
196. # on récupère les paramètres du courrier
```

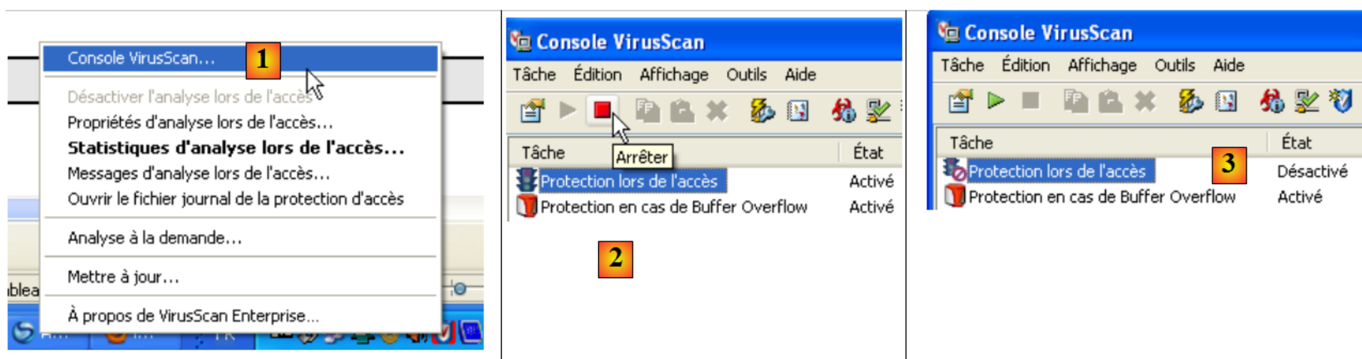

Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Programme (inet_03)

```
197. res=getInfos(MAIL)
198. # erreur ?
199. if res[0]:
200.       print "%s" % (erreur)
201.       sys.exit()
202. # envoi du courrier en mode verbeux
203. (smtpServer,expediteur,destinataire,message)=res[1:]
204. print "Envoi du message [%s,%s,%s]" % (smtpServer,expediteur,destinataire)
205. resultat=sendmail(smtpServer,expediteur,destinataire,message,True)
206. print "Resultat de l'envoi : %s" % (resultat)
207. # fin
208. sys.exit()
```

Notes :

- sur une machine Windows possédant un antivirus, ce dernier empêchera peut-être le script Python de se connecter au port 25 d'un serveur SMTP. Il faut alors désactiver l'antivirus. Pour McAfee par exemple, on peut procéder ainsi :



- en [1], on active la console VirusScan ;
- en [2], on arrête le service [Protection lors de l'accès] ;
- en [3], il est arrêté.

Résultats

Le fichier *infos.txt* :

```
1. smtp.univ-angers.fr, serge.tahe@univ-angers.fr , serge.tahe@univ-angers.fr
2. Subject: test
3.
4. ligne1
5. ligne2
6.
7. ligne3
```

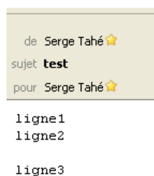
Les résultats écran :

```
1. Envoi du message [smtp.univ-angers.fr,serge.tahe@univ-angers.fr,serge.tahe@univ-angers.fr]
2. --> [EHLO Gportpers3.ad.univ-angers.fr]
3. <-- [220 smtp.univ-angers.fr ESMTP Postfix
4. 250-smtp.univ-angers.fr
5. 250-PIPELINING
6. 250-SIZE 20480000
7. 250-VRFY
8. 250-ETRN
9. 250-ENHANCEDSTATUSCODES
10. 250-8BITMIME
11. 250 DSN]
12. --> [MAIL FROM: <serge.tahe@univ-angers.fr>]
```


Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

```
13. <-- [250 2.1.0 Ok]
14. --> [RCPT TO: <serge.tahe@univ-angers.fr>]
15. <-- [250 2.1.5 Ok]
16. --> [DATA]
17. <-- [354 End data with <CR><LF>.<CR><LF>]
18. --> [From: serge.tahe@univ-angers.fr
19. To: serge.tahe@univ-angers.fr
20. Subject: test
21.
22. ligne1
23. ligne2
24.
25. ligne3
26.
27. .
28. ]
29. <-- [250 2.0.0 Ok: queued as 63412114203]
30. --> [QUIT]
31. <-- [221 2.0.0 Bye]
32. Resultat de l'envoi : Message envoye
```

Le message lu par le lecteur de courrier Thunderbird :



XII-D - Un second client SMTP

Ce second script fait la même chose que le précédent mais en utilisant les fonctionnalités du module [smtplib].

Programme (inet_04)

```
1. # -*- coding=utf-8 -*-
2.
3. import sys,socket, smtplib
4.
5. #-----------------------------------------------------------------------
6. def getInfos(fichier):
7.
  # rend les informations (smtp,expéditeur,destinataire,message) prises dans le fichier texte [fichier]
8.     # ligne 1 : smtp, expéditeur, destinataire
9.     # lignes suivantes : le texte du message
10.
11.     # ouverture de [fichier]
12.     erreur=""
13.     try:
14.         infos=open(fichier,"r")
15.     except IOError, erreur:
16.         return ("Le fichier %s n'a pu etre ouvert en lecture : %s" % (fichier, erreur))
17.
18.     # lecture de la 1re ligne
19.     ligne=infos.readline()
20.     # suppression de la marque de fin de ligne
21.     ligne=cutNewLineChar(ligne)
22.     # récupération des champs smtp,expéditeur,destinataire
23.     champs=ligne.split(",")
24.     # a-t-on le bon nombre de champs ?
25.     if len(champs)!=3 :
26.         return
 ("La ligne 1 du fichier %s (serveur smtp, expediteur, destinataire) a un nombre de champs incorrect" %
 (fichier))
```


Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Programme (inet_04)

```
27.
  # "traitement" des informations récupérées - on leur enlève les "blancs" qui les précèdent ou les suivent
28.     for i in range(3):
29.         champs[i]=champs[i].strip()
30.     # récupération des champs
31.     (smtpServer,expediteur,destinataire)=champs
32.     # lecture message à envoyer
33.     message=""
34.     ligne=infos.readline()
35.     while ligne!='':
36.         message+=ligne
37.         ligne=infos.readline()
38.     infos.close()
39.     # retour
40.     return ("",smtpServer,expediteur,destinataire,message)
41.
42. #-------------------------------------------------------------------------
43. def sendmail(smtpServer,expediteur,destinataire,message,verbose):
44.     # envoie message au serveur smtp smtpserver de la part de expéditeur
45.     # pour destinataire. Si verbose=True, fait un suivi des échanges client-serveur
46.
47.     # on utilise la bibliothéque smtplib
48.     try:
49.         server = smtplib.SMTP(smtpServer)
50.         if verbose:
51.             server.set_debuglevel(1)
52.             server.sendmail(expediteur, destinataire, message)
53.             server.quit()
54.     except Exception, erreur:
55.         return "Erreur envoi du message : %s" % (erreur)
56.     # fin
57.     return "Message envoye"
58.
59. # -------------------------------------------------------------------------
60. def cutNewLineChar(ligne):
61.     # on supprime la marque de fin de ligne de [ligne] si elle existe
62.     l=len(ligne)
63.     while(ligne[l-1]=="\n" or ligne[l-1]=="\r"):
64.         l-=1
65.     return(ligne[0:l])
66.
67. # main ---------------------------------------------------------------
68.
69.  # client SMTP (SendMail Transfer Protocol) permettant d'envoyer un message
70. # les infos sont prises dans un fichier INFOS contenant les lignes suivantes
71. # ligne 1 : smtp, expéditeur, destinataire
72. # lignes suivantes : le texte du message
73.
74. # expéditeur:email expéditeur
75. # destinataire: email destinataire
76. # smtp: nom du serveur smtp à utiliser
77.
78.
79. # protocole de communication SMTP client-serveur
80. # -> client se connecte sur le port 25 du serveur smtp
81. # <- serveur lui envoie un message de bienvenue
82. # -> client envoie la commande EHLO: nom de sa machine
83. # <- serveur répond OK ou non
84. # -> client envoie la commande mail from: <expéditeur>
85. # <- serveur répond OK ou non
86. # -> client envoie la commande rcpt to: <destinataire>
87. # <- serveur répond OK ou non
88. # -> client envoie la commande data
89. # <- serveur répond OK ou non
90. # -
> client envoie toutes les lignes de son message et termine avec une ligne contenant le seul caractère .
91. # <- serveur répond OK ou non
92. # -> client envoie la commande quit
93. # <- serveur répond OK ou non
94.
95. # les réponses du serveur ont la forme xxx texte où xxx est un nombre à 3 chiffres. Tout nombre xxx >=500
```


Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Programme (inet_04)

```
96. # signale une erreur. La réponse peut comporter plusieurs lignes commençant toutes par xxx- sauf la dernière
97. # de la forme xxx(espace)
98.
99. # les lignes de texte échangées doivent se terminer par les caractères RC(#13) et LF(#10)
100.
101. # # les paramètres de l'envoi du courrier
102. MAIL="mail2.txt"
103. # on récupère les paramètres du courrier
104. res=getInfos(MAIL)
105. # erreur ?
106. if res[0]:
107.       print "%s" % (erreur)
108.       sys.exit()
109. # envoi du courrier en mode verbeux
110. (smtpServer,expediteur,destinataire,message)=res[1:]
111. print "Envoi du message [%s,%s,%s]" % (smtpServer,expediteur,destinataire)
112. resultat=sendmail(smtpServer,expediteur,destinataire,message,True)
113. print "Resultat de l'envoi : %s" % (resultat)
114. # fin
115. sys.exit()
```

Notes :

- ce script est identique au précédent à la fonction *sendmail* près. Cette fonction utilise désormais les fonctionnalités du module [smtplib] (ligne 3).

Le fichier mail2.txt

```
1. smtp.univ-angers.fr, serge.tahe@univ-angers.fr , serge.tahe@univ-angers.fr
2. From: serge.tahe@univ-angers.fr
3. To: serge.tahe@univ-angers.fr
4. Subject: test
5.
6. ligne1
7. ligne2
8.
9. ligne3
```

Les résultats écran :

```
1. Envoi du message [smtp.univ-angers.fr,serge.tahe@univ-angers.fr,serge.tahe@univ-
2. angers.fr]
3. send: 'ehlo Gportpers3.ad.univ-angers.fr\r\n'
4. reply: '250-smtp.univ-angers.fr\r\n'
5. reply: '250-PIPELINING\r\n'
6. reply: '250-SIZE 20480000\r\n'
7. reply: '250-VRFY\r\n'
8. reply: '250-ETRN\r\n'
9. reply: '250-ENHANCEDSTATUSCODES\r\n'
10. reply: '250-8BITMIME\r\n'
11. reply: '250 DSN\r\n'
12. reply: retcode (250); Msg: smtp.univ-angers.fr
13. PIPELINING
14. SIZE 20480000
15. VRFY
16. ETRN
17. ENHANCEDSTATUSCODES
18. 8BITMIME
19. DSN
20. send: 'mail FROM:<serge.tahe@univ-angers.fr> size=99\r\n'
21. reply: '250 2.1.0 Ok\r\n'
22. reply: retcode (250); Msg: 2.1.0 Ok
23. send: 'rcpt TO:<serge.tahe@univ-angers.fr>\r\n'
24. reply: '250 2.1.5 Ok\r\n'
25. reply: retcode (250); Msg: 2.1.5 Ok
26. send: 'data\r\n'
27. reply: '354 End data with <CR><LF>.<CR><LF>\r\n'
28. reply: retcode (354); Msg: End data with <CR><LF>.<CR><LF>
```


Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

```
29. data: (354, 'End data with <CR><LF>.<CR><LF>')
30. send: 'From: serge.tahe@univ-angers.fr\r\nTo: serge.tahe@univ-angers.fr\r\nSubje
31. ct: test\r\n\r\nligne1\r\nligne2\r\n\r\nligne3\r\n.\r\n'
32. reply: '250 2.0.0 Ok: queued as D4977114358\r\n'
33. reply: retcode (250); Msg: 2.0.0 Ok: queued as D4977114358
34. data: (250, '2.0.0 Ok: queued as D4977114358')
35. send: 'quit\r\n'
36. reply: '221 2.0.0 Bye\r\n'
37. reply: retcode (221); Msg: 2.0.0 Bye
38. Resultat de l'envoi : Message envoye
```

XII-E - Client / serveur d'écho

On crée un service d'écho. Le serveur renvoie en majuscules toutes les lignes de texte que lui envoie le client. Le service peut servir plusieurs clients à la fois grâce à l'utilisation de threads.

Le programme du serveur (inet_08)

```
1. # -*- coding=utf-8 -*-
2.
3. # serveur tcp générique sur Windows
4.
5. # chargement des fichiers d'en-tête
6. import re,sys,SocketServer,threading
7.
8. # serveur tcp multithreadé
9. class ThreadedTCPRequestHandler(SocketServer.StreamRequestHandler):
10.
11.   def handle(self):
12.     # thread courant
13.     cur_thread = threading.currentThread()
14.     # données du client
15.     self.data="on"
16.     # arrêt sur chaine vide
17.     while self.data:
18.       # les lignes de texte du client sont lues avec la méthode readline
19.       self.data =self.rfile.readline().strip()
20.       # suivi console
21.       print "client %s : %s (%s)" % (self.client_address[0],self.data,cur_thread.getName())
22.       # envoi réponse au client
23.       response = "%s: %s" % (cur_thread.getName(), self.data.upper())
24.       # self.wfile est le flux d'écriture vers client
25.       self.wfile.write(response)
26.
27. class ThreadedTCPServer(SocketServer.ThreadingMixIn, SocketServer.TCPServer):
28.     pass
29.
30. # ------------------------------------------------------ main
31. # syntaxe d'appel : argv[0] port
32. # le serveur est lancé sur le port nommé
33.
34. # Données
35. syntaxe="syntaxe : %s port" % (sys.argv[0])
36.
37. #------------------------------------- on vérifie l'appel
38. # il doit y avoir un argument et un seul
39. nbArguments=len(sys.argv)
40. if nbArguments!=2:
41.     print syntaxe
42.     sys.exit(1)
43.
44. # le port doit être numérique
45. port=sys.argv[1]
46. modele=r"^\s*\d+\s*$"
47. if not re.match(modele,port):
48.     print "Le port %s n'est pas un nombre entier positif\n" % (port)
49.     sys.exit(2)
50.
51. # lancement du serveur - sert chaque client sur un thread
```


Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Le programme du serveur (inet_08)

```
52. host="localhost"
53. server = ThreadedTCPServer((host,int(port)), ThreadedTCPRequestHandler)
54.
55. # le serveur est lancé dans un thread
56. # chaque client sera servi dans un thread supplémentaire
57. server_thread = threading.Thread(target=server.serve_forever)
58. # lance le serveur - boucle infinie d'attente des clients
59. server_thread.start()
60. # suivi
61. print "Serveur d'echo a l'ecoute sur le port % s" % (port)
```

Notes :

- ligne 39 : *sys.argv* représente les paramètres du script. Ils doivent être ici de la forme : *nom_du_script port*. Il doit donc y en avoir deux. *sys.argv[0]* sera alors *nom_du_script* et *sys.argv[1]* sera *port* ;
- ligne 53 : le serveur d'écho est une instance de la classe *ThreadedTcpServer*. Le constructeur de cette classe attend deux paramètres :
- paramètre 1 : un tuple de deux éléments (host,port) qui fixe la machine et le port d'écoute du serveur ;
- paramètre 2 : le nom de la classe chargée de traiter les requêtes d'un client.
- ligne 57 : un thread est créé (mais pas encore lancé). Ce thread exécute la méthode [serve_forever] du serveur TCP. Cette méthode est une boucle d'écoute des connexions clientes. Dès qu'une connexion cliente est détectée, une instance de la classe *ThreadedTCPRequestHandler* sera créée. Sa méthode *handle* est chargée du dialogue avec le client ;
- ligne 59 : le thread du service d'écho est lancé. À partir de ce moment des clients peuvent se connecter au service ;
- ligne 27 : la classe du serveur d'écho. Elle dérive de deux classes : *SocketServer.ThreadingMixIn* et *SocketServer.TCPServer*. Cela en fait un serveur TCP multithreadé : le serveur s'exécute dans un thread et chaque client est servi dans un thread supplémentaire ;
- ligne 9 : la classe qui traite les demandes des clients. Elle dérive de la classe *SocketServer.StreamRequestHandler*. Elle hérite alors de deux attributs :
  - *rfile* : qui est le flux de lecture des données envoyées par le client - peut être traité comme un fichier texte,
  - *wfile* : qui est le flux d'écriture qui permet d'envoyer des données au client - peut être traité comme un fichier texte ;
- ligne 11 : la méthode *handle* traite les demandes des clients ;
- ligne 13 : le thread qui exécute cette méthode *handle ;*
- ligne 17 : boucle de traitement des demandes du client. La boucle se termine lorsque le client envoie une ligne vide ;
- ligne 19 : lecture de la demande du client ;
- ligne 21 : *self.client_address[0]* représente l'adresse IP du client. *cur_thread.getName()* est le nom du thread qui exécute la méthode *handle ;*
- ligne 23 : la réponse au client a deux composantes - le nom du thread qui sert le client et la commande qu'il a envoyée, passée en majuscules.

Le programme du client (inet_06)

```
1. # -*- coding=utf-8 -*-
2. import re,sys,socket
3.
4. # ------------------------------------------------------ main
5. # client tcp générique
6. # syntaxe d'appel : argv[0] hote port
7. # le client se connecte au service d'écho (hote,port)
8. # le serveur renvoie les lignes tapées par le client
9.
10. # syntaxe
11. syntaxe="syntaxe : %s hote port" % (sys.argv[0])
12.
13. #-------------------------------------- on vérifie l'appel
14. # il doit y avoir deux arguments
15. nbArguments=len(sys.argv)
16. if nbArguments!=3:
```


Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Le programme du client (inet_06)

```
17.     print syntaxe
18.     sys.exit(1)
19.
20. # on récupère les arguments
21. hote=sys.argv[1]
22. # le port doit être numérique
23. port=sys.argv[2]
24. modele=r"^\s*\d+\s*$"
25. if not re.match(modele,port):
26.     print "Le port %s foit être un nombre entier positif" % (port)
27.     sys.exit(2)
28.
29. try:
30.   # connexion du client au serveur
31.   connexion=socket.create_connection((hote,int(port)))
32. except socket.error, erreur:
33.   print "Echec de la connexion au site (%s,%s) : %s" % (hote,port,erreur)
34.   sys.exit(3)
35.
36. try:
37.     # boucle de saisie
38.     ligne=raw_input("Commande (rien pour arreter): ").strip()
39.     while ligne!="":
40.         # on envoie la ligne au serveur
41.         connexion.send("%s\n" % (ligne))
42.         # on attend la reponse
43.         reponse=connexion.recv(1000)
44.         print "<-- %s" % (reponse)
45.         ligne=raw_input("Commande (rien pour arreter): ")
46. except socket.error, erreur:
47.     print "Echec de la connexion au site (%s,%s) : %s" % (hote,port,erreur)
48.     sys.exit(3)
49. finally:
50.     # on clôt la connexion
51.     connexion.close()
```

Le serveur est lancé dans une première fenêtre de commandes :

```
1. cmd>%python% inet_08.py 100
2. Serveur d'echo a l'ecoute sur le port 100
```

Un premier client est lancé dans une seconde fenêtre :

```
1. cmd>%python% inet_06.py localhost 100
2. Commande (rien pour arreter): cmde 1 du client 1
3. [cmde 1 du client 1]
4. <-- Thread-2: CMDE 1 DU CLIENT 1
5. Commande (rien pour arreter): cmde 2 du client 1
6. <-- Thread-2: CMDE 2 DU CLIENT 1
7. Commande (rien pour arreter):
```

Le client reçoit bien en réponse à la commande qu'il envoie au serveur, cette même commande est mise en majuscules. Un second client est lancé dans une troisième fenêtre :

```
1. cmd>%python% inet_06.py localhost 100
2. Commande (rien pour arreter): cmde 1 du client 2
3. <-- Thread-3: CMDE 1 DU CLIENT 2
4. Commande (rien pour arreter): cmde 2 du client 2
5. <-- Thread-3: CMDE 2 DU CLIENT 2
6. Commande (rien pour arreter):
```

La console du serveur est alors la suivante :

```
1. cmd>%python% inet_08.py 100
2. Serveur d'echo a l'ecoute sur le port 100
3. client 127.0.0.1 : cmde 1 du client 1 (Thread-2)
```


Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

```
4. client 127.0.0.1 : cmde 2 du client 1 (Thread-2)
5. client 127.0.0.1 : cmde 1 du client 2 (Thread-3)
6. client 127.0.0.1 : cmde 2 du client 2 (Thread-3)
```

Les clients sont bien servis dans des threads différents. Pour arrêter un client, il suffit de taper une commande vide.

XII-F - Serveur Tcp générique

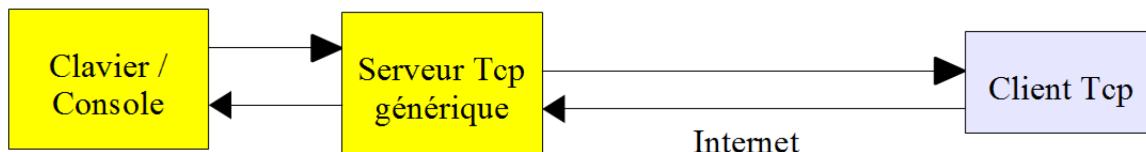
Nous nous proposons d'écrire un script Python qui

- serait un serveur Tcp capable de servir un client à la fois,
- acceptant des lignes de texte envoyées par le client,
- acceptant des lignes de texte venant du clavier et qui sont envoyées en réponse au client.

Ainsi c'est l'utilisateur au clavier qui fait office de serveur :

- il voit sur sa console les lignes de texte envoyées par le client ;
- il répond à celui-ci en tapant la réponse au clavier.

Il peut ainsi s'adapter à toute sorte de clients. C'est pour cela qu'on l'appellera "serveur Tcp générique". C'est un outil pratique pour découvrir des protocoles de communication Tcp. Dans l'exemple qui suit, le client Tcp sera un navigateur Web ce qui nous permettra de découvrir le protocole HTTP utilisé par les clients Web.



Le programme (inet_10)

```
1. # -*- coding=utf-8 -*-
2.
3. # serveur tcp générique
4.
5. # chargement des fichiers d'en-tête
6. import re,sys,SocketServer,threading
7.
8. # serveur tcp générique
9. class MyTCPHandler(SocketServer.StreamRequestHandler):
10.     def handle(self):
11.         # on affiche le client
12.         print "client %s" % (self.client_address[0])
13.         # on crée un thread de lecture des commandes du client
14.         thread_lecture = threading.Thread(target=self.lecture)
15.         thread_lecture.start()
16.         # arrêt sur cmde 'bye'
17.         # lecture cmde tapée au clavier
18.         cmde=raw_input("--> ")
19.         while cmde!="bye":
20.             # envoi cmde au client. self.wfile est le flux d'écriture vers le client
21.             self.wfile.write("%s\n" % (cmde))
22.             # lecture cmde suivante
23.             cmde=raw_input("--> ")
24.
25.     def lecture(self):
26.         # on affiche toutes les lignes envoyées par le client jusqu'à recevoir la commande bye
27.         ligne=""
28.         while ligne!="bye":
29.             # les lignes de texte du client sont lues avec la méthode readline
30.             ligne = self.rfile.readline().strip()
31.             # suivi console
```


Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Le programme (inet_10)

```
32.             print "<--- %s : %s" % (self.client_address[0], ligne)
33.
34. # ------------------------------------------------------ main
35. # syntaxe d'appel : argv[0] port
36. # le serveur est lancé sur le port nommé
37.
38. # Données
39. syntaxe="syntaxe : %s port" % (sys.argv[0])
40.
41. #------------------------------------- on vérifie l'appel
42. # il doit y avoir un argument et un seul
43. nbArguments=len(sys.argv)
44. if nbArguments!=2:
45.     print syntaxe
46.     sys.exit(1)
47.
48. # le port doit être numérique
49. port=sys.argv[1]
50. modele=r"^\s*\d+\s*$"
51. if not re.match(modele,port):
52.     print "Le port %s n'est pas un nombre entier positif\n" % (port)
53.     sys.exit(2)
54.
55. # lancement du serveur
56. host="localhost"
57. server = SocketServer.TCPServer((host, int(port)), MyTCPHandler)
58. print "Service tcp generique lance sur le port %s. Arret par Ctrl-C" % (port)
59. server.serve_forever()
```

Notes :

- ligne 57 : le serveur Tcp sera une instance de la classe *SocketServer.TCPServer*. Le constructeur de celle-ci admet deux paramètres :
- le 1er paramètre est un tuple de deux éléments (*host, port*) où *host* est la machine sur laquelle opère le service (généralement *localhost*) et *port*, le port sur lequel le service attend (écoute) les demandes des clients,
- le second paramètre précise la classe de service d'un client. Lorsqu'un client se connecte, une instance de la classe de service est créée et c'est sa méthode *handle* qui doit gérer la connexion avec le client.

Le serveur Tcp *SocketServer.TCPServer* n'est pas multithreadé. Il sert un client à la fois ;

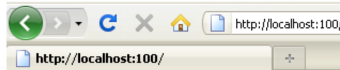
- ligne 59 : la méthode *serve_forever* du serveur Tcp est exécutée. C'est une boucle sans fin d'attente des clients ;
- ligne 9 : le serveur Tcp est ici une classe dérivée de la classe *SocketServer.StreamRequestHandler*. Cela permet de considérer les flux de données échangées avec le client comme des fichiers texte. Nous avons déjà rencontré cette classe. Nous disposons des méthodes :
- *readline* pour lire une ligne de texte venant du client,
- *write* pour lui renvoyer des lignes de texte ;
- ligne 12 : *self.client_address[0]* est l'adresse Ip du client ;
- ligne 14 : le serveur Tcp va échanger avec le client à l'aide de deux threads :
- un thread de lecture des lignes du client,
- un thread d'écriture de lignes au client ;
- ligne 14 : le thread de lecture des lignes du client est créé. Son paramètre *target* fixe la méthode exécutée par le thread. C'est celle définie ligne 25 ;
- ligne 25 : le thread de lecture est lancé ;
- lignes 25-32 : la méthode exécutée par le thread de lecture ;
- ligne 28 : le thread de lecture lit toutes les lignes de texte envoyées par le client jusqu'à la réception de la ligne "bye" ;
- ligne 18 : on est là dans le thread d'écriture au client. Le principe est d'envoyer au client toutes les lignes de texte tapées au clavier par l'utilisateur.

Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Les résultats

Le serveur
```
1. dos>%python% inet_10.py 100
2. Service tcp generique lance sur le port 100. Arret par Ctrl-C
```

Le navigateur client :



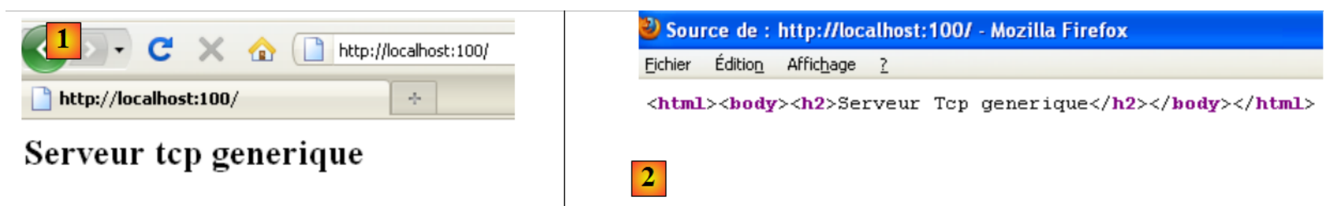
La demande reçue par le serveur
```
1. dos>%python% inet_10.py 100
2. Service tcp generique lance sur le port 100. Arret par Ctrl-C
3. client 127.0.0.1
4. <--- 127.0.0.1 : GET / HTTP/1.1
5. <--> --- 127.0.0.1 : Host: localhost:100
6. <--- 127.0.0.1 : User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; fr; rv:1.9.2.10)
 Gecko/20100914 Firefox/3.6.10 GTB7.1 ( .NET CLR 3.5.30729)
7. <--- 127.0.0.1 : Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
8. <--- 127.0.0.1 : Accept-Language: fr,fr-fr;q=0.8,en;q=0.5,en-us;q=0.3
9. <--- 127.0.0.1 : Accept-Encoding: gzip,deflate
10. <--- 127.0.0.1 : Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
11. <--- 127.0.0.1 : Keep-Alive: 115
12. <--- 127.0.0.1 : Connection: keep-alive
13. <--- 127.0.0.1 : Cache-Control: max-age=0
14. <--- 127.0.0.1 :
```

La réponse du serveur tapée par l'utilisateur au clavier (sans le signe -->)

```
1. HTTP/1.1 200 OK
2. --> Server: serveur tcp generique
3. --> Connection: close
4. --> Content-Type: text/html
5. -->
6. --> <html><body><h2>Serveur tcp generique</h2></body></html>
7. --> bye
```

- lignes 1-5 : réponse HTTP envoyée au client ;
- ligne 6 : la page HTML envoyée au client ;
- ligne 7 : fin du dialogue avec le client. Le service au client va se terminer et la connexion va être fermée, ce qui va interrompre brutalement le thread de lecture des lignes de texte envoyées par le client ;
- les lignes 1-5 de la réponse Http faite au client ont la signification suivante :
- ligne 1 : la ressource demandée par le client a été trouvée,
- ligne 2 : identification du serveur,
- ligne 3 : le serveur va fermer la connexion après envoi de la ressource,
- ligne 4 : nature de la ressource envoyée par le serveur : un document HTML,
- ligne 5 : une ligne vide.

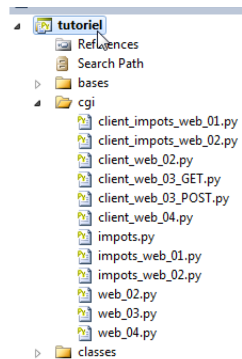
La page affichée par le navigateur [1] :



Si on fait afficher le code source reçu par le navigateur [2], on s'aperçoit que le code HTML reçu par le navigateur Web est bien celui qu'on lui avait envoyé.

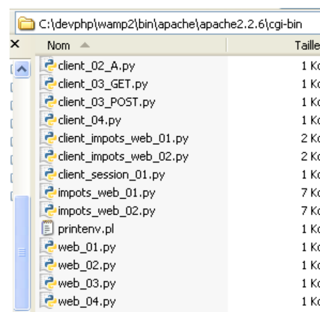
Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

XIII - Des services Web en Python



Les scripts Python peuvent être exécutés par un serveur Web. C'est ce dernier qui sera à l'écoute des demandes clientes. Du point de vue du client, appeler un service Web revient à demander l'URL de ce service. Le client peut être écrit avec n'importe quel langage, notamment en Python. Il nous faut savoir "converser" avec un service Web, c'est-à-dire comprendre le protocole *HTTP* de communication entre un serveur Web et ses clients. C'est le but des programmes qui suivent.

Les scripts de service Web seront exécutés par le serveur Web Apache de *WampServer*. Il faut les placer dans un répertoire particulier : *<WampServer>\bin\apache\apachex.y.z\cgi-bin* où *<WampServer>* est le dossier d'installation de *WampServer* et *x.y.z* est la version du serveur Web Apache.



Placer les scripts Python dans le dossier <cgi-bin> n'est pas suffisant. Il faut que le script indique en première ligne le chemin de l'interpréteur Python à utiliser. Ce chemin est mis sous la forme d'un commentaire :

```
#!D:\Programs\ActivePython\Python2.7.2\python.exe
```

Le lecteur adaptera ce chemin à son propre environnement.

XIII-A - Application client/ serveur de date/heure

Notre premier service Web sera un service de date et heure : le client reçoit la date et l'heure courantes.

XIII-A-1 - Le serveur

Le programme (web_02)

```
1. #!D:\Programs\ActivePython\Python2.7.2\python.exe
```


Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Le programme (web_02)

```
2.
3. import time
4.
5. # headers
6. print "Content-Type: text/plain\n"
7.
8. # envoi heure au client
9.   # localtime : nb de millisecondes depuis 01/01/1970
10.   # "format affichage date-heure
11.   # d: jour sur 2 chiffres
12.   # m: mois sur 2 chiffres
13.   # y : année sur 2 chiffres
14.   # H : heure 0,23
15.   # M : minutes
16.   # S: secondes
17.
18. print time.strftime('%d/%m/%y %H:%M:%S',time.localtime())
```

Notes :

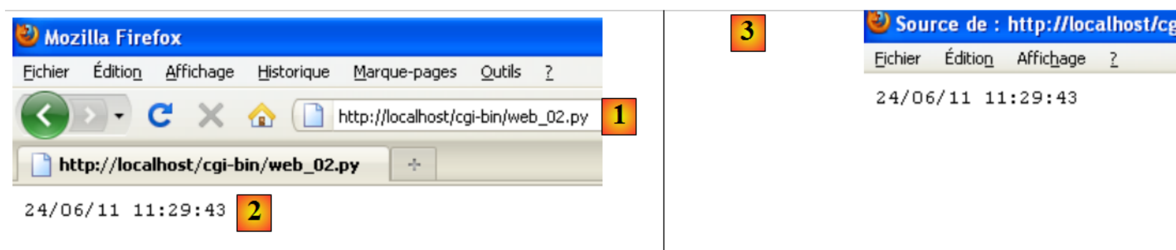
- ligne 6 : le script doit générer lui-même certains des entêtes HTTP de la réponse au client. Ils viendront s'ajouter aux entêtes HTTP générés par le serveur Apache lui-même. L'entête HTTP de la ligne 6 indique au client qu'on va lui envoyer une ressource au format *text/plain*, c'est-à-dire du texte non formaté. On notera le "\n" à la fin de l'entête qui va générer une ligne vide derrière l'entête. C'est obligatoire : c'est cette ligne vide qui signale au client HTTP la fin des entêtes HTTP de la réponse. Vient ensuite la ressource demandée par le client, ici un texte non formaté ;
- ligne 18 : la ressource envoyée au client est un texte affichant la date et l'heure courantes.

XIII-A-2 - Deux tests

Le script précédent peut être exécuté directement par l'interpréteur Python dans une fenêtre de commandes comme nous l'avons fait jusqu'à maintenant. Cela permet d'éliminer d'éventuelles erreurs de syntaxe ou de fonctionnement. On obtient le résultat suivant :

```
1. cmd>%python% web_02.py
2. Content-Type: text/plain
3.
4. 24/06/11 11:16:55
```

Lorsque le script a été ainsi testé, on peut le placer dans le dossier <*cgi-bin*> du serveur Apache. Lançons l'application *WampServer*. Cela lance à la fois un serveur Web Apache et un SGBD MySQL. Nous n'utiliserons pour le moment que le serveur Web. Puis avec un navigateur, demandons l'URL suivante : *http://localhost/cgi-bin/web_02.py* :



- en [1] : l'URL demandée ;
- en [2] : la réponse affichée par le navigateur ;
- en [3] : le code source reçu par le navigateur Web. C'est bien celui envoyé par le script Python.

Avec certains outils (ici Firebug, un plugin du navigateur Firefox) on peut avoir accès aux entêtes HTTP échangés avec le serveur. Ci-dessus, le navigateur Web a reçu les entêtes HTTP suivants :

Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

```
1. HTTP/1.1 200 OK
2. Date: Fri, 24 Jun 2011 09:35:02 GMT
3. Server: Apache/2.2.6 (Win32) PHP/5.2.5
4. Keep-Alive: timeout=5, max=100
5. Connection: Keep-Alive
6. Transfer-Encoding: chunked
7. Content-Type: text/plain
```

On reconnaît lignes 7-8, l'entête HTTP envoyé par le script Python. Ceux qui précèdent ont été générés par le serveur Web Apache.

XIII-A-3 - Un client programmé

On écrit maintenant un script qui sera client du service Web précédent. On utilise les fonctionnalités du module *httplib* qui facilite l'écriture de clients HTTP.

Le programme (client_web_02)

```
1. # -*- coding=utf-8 -*-
2.
3. import httplib,re
4.
5. # constantes
6. HOST="localhost"
7. URL="/cgi-bin/web_02.py"
8. # connexion
9. connexion=httplib.HTTPConnection(HOST)
10. # suivi
11. connexion.set_debuglevel(1)
12. # envoi de la requête
13. connexion.request("GET", URL)
14. # traitement de la réponse
15. reponse=connexion.getresponse()
16. # contenu
17. contenu=reponse.read()
18. # fermeture connexion
19. connexion.close()
20. print "------\n",contenu,"-----\n"
21. # récupération des éléments de l'heure
22. elements=re.match(r"^(\d\d)/(\d\d)/(\d\d)  (\d\d):(\d\d):(\d\d)\s*$",contenu).groups()
23. print "Jour=%s,Mois=%s,An=%s,Heures=%s,Minutes=%s,Secondes=%s" %
 (elements[0],elements[1],elements[2],elements[3],elements[4],elements[5])
```

Notes :

- ligne 3 : le module *re* est nécessaire aux expressions régulières, le module *httplib* aux fonctions des clients HTTP ;
- ligne 9 : une connexion HTTP est créée avec le port 80 de HOST défini ligne 6 ;
- ligne 11 : le suivi permet de voir les entêtes HTTP de la demande du client et de la réponse du serveur ;
- ligne 13 : l'URL du service Web est demandée. Il y a deux façons de la demander : à l'aide d'une commande HTTP GET ou POST. La différence entre les deux est expliquée plus loin. Ici elle sera demandée avec la commande HTTP GET ;
- ligne 15 : la réponse du serveur est lue. C'est l'ensemble de la réponse qui est ici obtenue : entêtes HTTP et ressources demandées par le client. Dans sa réponse, le serveur a pu demander au client de se rediriger. Dans ce cas, le client *httplib* fait automatiquement la redirection. La réponse obtenue est donc celle issue de la redirection ;
- ligne 17 : la réponse est composée des entêtes Http et du document demandé par le client. Pour obtenir seulement les entêtes HTTP, on utilisera *[reponse].getHeaders()*. Pour obtenir le document, on utilise *[reponse].read() ;*
- ligne 19 : une fois obtenue la réponse du serveur Web, la connexion à celui-ci est fermée ;
- on sait que le document envoyé par le serveur est une ligne de texte de la forme 15/06/11 14:56:36. Lignes 22-26, on utilise une expression régulière pour récupérer les différents éléments de cette ligne.

Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.
http://tahe.developpez.com/divers/python/exemples/

Résultats

```
1. send: 'GET /cgi-bin/web_02.py HTTP/1.1\r\nHost: localhost\r\nAccept-Encoding: identity\r\n\r\n'
2. reply: 'HTTP/1.1 200 OK\r\n'
3. header: Date: Tue, 14 Feb 2012 14:51:07 GMT
4. header: Server: Apache/2.2.17 (Win32) PHP/5.3.5
5. header: Transfer-Encoding: chunked
6. header: Content-Type: text/plain
7. ------
8. 14/02/12 15:51:07
9. -----
10.
11. Jour=14,Mois=02,An=12,Heures=15,Minutes=51,Secondes=07
```

Notes :

- ligne 1 : les entêtes HTTP envoyés par le client au serveur Web ;
- lignes 2-6 : les entêtes HTTP de la réponse du serveur Web ;
- ligne 8 : le document envoyé par le serveur ;
- ligne 11 : le résultat de son exploitation ;

XIII-B - Récupération par le serveur des paramètres envoyés par le client

Dans le protocole HTTP, un client a deux méthodes pour passer des paramètres au serveur Web :

1. Il demande l'URL du service sous la forme

GET url?param1=val1¶m2=val2¶m3=val3\x{2026} HTTP/1.0

où les valeurs *vali* doivent au préalable subir un encodage afin que certains caractères réservés soient remplacés par leur valeur hexadécimale ;

2. Il demande l'URL du service sous la forme

POST url HTTP/1.0

puis, parmi les entêtes HTTP envoyés au serveur, place l'entête suivant :

Content-length: N

La suite des entêtes envoyés par le client se termine par une ligne vide. Il peut alors envoyer ses données sous la forme

val1¶m2=val2¶m3=val3\x{2026}

où les *vali* doivent, comme pour la méthode GET, être préalablement encodées. Le nombre de caractères envoyés au serveur doit être N où N est la valeur déclarée dans l'entête

Content-length: N

XIII-B-1 - Le service Web

Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Le service Web qui suit reçoit trois paramètres de son client : *nom*, *prenom*, *age*. Il les récupère dans une sorte de dictionnaire nommé *cgi.FieldStorage* fourni par le module *cgi*. La valeur *vali* d'un paramètre *parami* est obtenue par *vali=cgi.FieldStorage().getlist("parami")*. On obtient un tableau de :

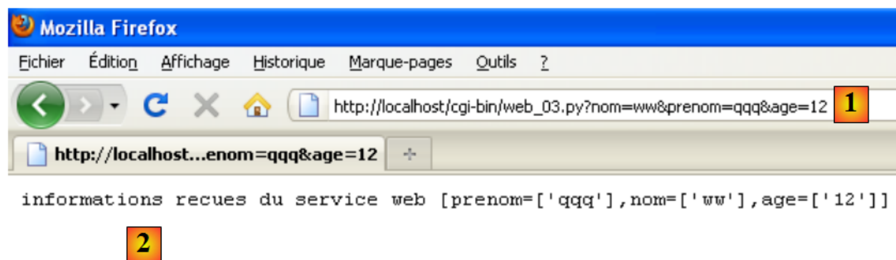
- 0 élément si le paramètre *parami* n'est pas présent dans la requête du client ;
- 1 élément si le paramètre *parami* est présent une fois dans la requête du client ;
- n éléments si le paramètre *parami* est présent *n* fois dans la requête du client.

Une fois qu'il a récupéré les paramètres, le script les renvoie au client.

Le programme (web_03)

```
1. #!D:\Programs\ActivePython\Python2.7.2\python.exe
2.
3. import cgi
4.
5. # headers
6. print "Content-Type: text/plain\n"
7.
8. # récupération par le serveur des informations envoyées par le client
9. # ici prenom=P&nom=N&age=A
10. formulaire=cgi.FieldStorage()
11.
12. # on les renvoie au client
13. print "informations recues du service web [prenom=%s,nom=%s,age=%s]" %
 (formulaire.getlist("prenom"),formulaire.getlist("nom"),formulaire.getlist("age"))
```

Un test peut être fait avec un navigateur Web :



En [1], l'URL du service Web. On notera la présence des trois paramètres *nom*, *prenom*, *age*. En [2] la réponse du service Web.

XIII-B-2 - Le client GET

Le programme (client_web_03_GET)

```
1. # -*- coding=utf-8 -*-
2.
3. import httplib,urllib
4.
5. # constantes
6. HOST="localhost"
7. URL="/cgi-bin/web_03.py"
8. PRENOM="Jean-Paul"
9. NOM="de la Huche"
10. AGE=42
11.
12. # les paramètres doivent être encodés avant d'être envoyés au serveur
13. params = urllib.urlencode({'nom': NOM, 'prenom': PRENOM, 'age': AGE})
14. # les paramètres sont mis à la fin de l'URL
15. URL+="?"+params
16. # connexion
17. connexion=httplib.HTTPConnection(HOST)
18. # suivi
```


Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Le programme (client_web_03_GET)

```
19. connexion.set_debuglevel(1)
20. # envoi de la requête
21. connexion.request("GET",URL)
22. # traitement de la réponse
23. reponse=connexion.getresponse()
24. # contenu
25. contenu=reponse.read()
26. print contenu,"\n"
27. # fermeture de la connexion
28. connexion.close()
```

Notes :

- lignes 8-10 : les valeurs des trois paramètres envoyés au service Web ;
- ligne 13 : il faut les encoder. Cela se fait à l'aide de la méthode *urlencode* du module *urllib*. Ce module est importé ligne 3. La méthode admet comme paramètre un dictionnaire *{param1:val1, param2:val2...} ;*
- ligne 15 : dans une commande GET (ligne 21), le client doit mettre les paramètres encodés à la fin de l'URL du service Web ;
- les lignes suivantes ont déjà été vues.

Les résultats

```
1. dos>%python% client_03_GET.py
2. send: 'GET /cgi-bin/web_03.py?nom=de+la+Huche&age=42&prenom=Jean-Paul HTTP/1.1\r\nHost: localhost\r
\nAccept-Encoding: identity\r\n\r\n'
3. reply: 'HTTP/1.1 200 OK\r\n'
4. header: Date: Wed, 15 Jun 2011 13:22:15 GMT
5. header: Server: Apache/2.2.6 (Win32) PHP/5.2.5
6. header: Transfer-Encoding: chunked
7. header: Content-Type: text/plain
8. informations recues du client [['Jean-Paul'],['de la Huche'],['42']]
```

Notes :

- ligne 2 : noter l'encodage des paramètres (nom, prenom, age) ;
- ligne 8 : la réponse du service Web.

XIII-B-3 - Le client POST

Le client POST est analogue au client GET si ce n'est que les paramètres encodés ne font plus partie de l'URL cible. Ils sont passés comme troisième argument de la requête POST (ligne 19).

Le programme (client_web_03_POST)

```
1. # -*- coding=utf-8 -*-
2.
3. import httplib,urllib
4.
5. # constantes
6. HOST="localhost"
7. URL="/cgi-bin/web_03.py"
8. PRENOM="Jean-Paul"
9. NOM="de la Huche"
10. AGE=42
11.
12. # les paramètres doivent être encodés avant d'être envoyés au serveur
13. params = urllib.urlencode({'nom': NOM, 'prenom': PRENOM, 'age': AGE})
14. # connexion
15. connexion=httplib.HTTPConnection(HOST)
16. # suivi
17. connexion.set_debuglevel(1)
18. # envoi de la requête
19. connexion.request("POST",URL,params)
```


Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.
http://tahe.developpez.com/divers/python/exemples/

Le programme (client_web_03_POST)

```
20. # traitement de la réponse
21. reponse=connexion.getresponse()
22. # contenu
23. contenu=reponse.read()
24. print contenu,"\n"
25. # fermeture de la connexion
26. connexion.close()
```

Les résultats

```
1. dos>%python% client_03_POST.py
2. send: 'POST /cgi-bin/web_03.py HTTP/1.1\r\nHost: localhost\r\nAccept-Encoding:identity\r\nContent-Length: 39\r\n\r\nnom=de+la+Huche&age=42&prenom=Jean-Paul'
3. reply: 'HTTP/1.1 200 OK\r\n'
4. header: Date: Fri, 24 Jun 2011 12:03:31 GMT
5. header: Server: Apache/2.2.6 (Win32) PHP/5.2.5
6. header: Transfer-Encoding: chunked
7. header: Content-Type: text/plain
8. informations recues du service web [prenom=['Jean-Paul'],nom=['de la Huche'],age=['42']]
```

Notes :

- on notera ligne 2, la méthode utilisée par le client POST pour envoyer les paramètres encodés :
- l'entête HTTP *Content-Length* indique le nombre de caractères qui vont être envoyés au service Web,
- cet entête HTTP est ensuite suivi d'une ligne vide indiquant la fin des entêtes HTTP,
- ensuite les 39 caractères des paramètres encodés sont envoyés ;
- ligne 8 : la réponse du service Web.

XIII-C - Récupération des variables d'environnement d'un service Web

XIII-C-1 - Le service Web

Le script cgi Python s'exécute dans un environnement système qui possède des attributs. Ceux-ci et leurs valeurs sont disponibles dans un dictionnaire *os.environ*.

Le programme (web_04)

```
1. #!D:\Programs\ActivePython\Python2.7.2\python.exe
2.
3. import os
4.
5. # headers
6. print "Content-Type: text/plain\n"
7. # infos d'environnement
8. for (cle,valeur) in os.environ.items():
9.     print "%s : %s" % (cle,valeur)
```

Notes :

- ligne 3 : il faut importer le module *os* pour disposer des variables "système".

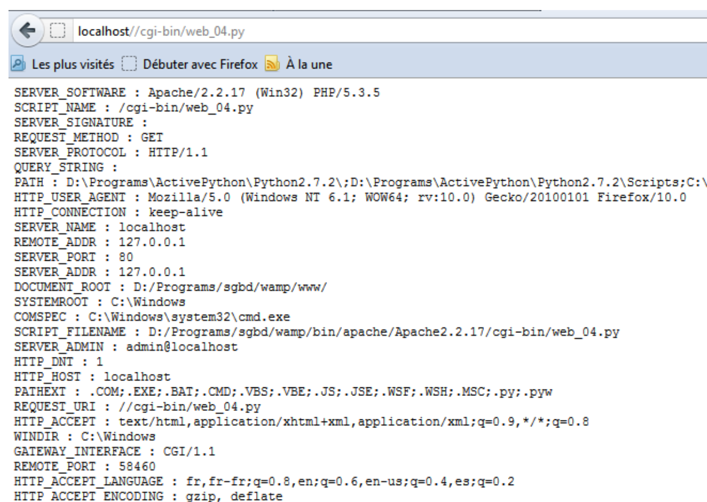
Si on exécute directement le script ci-dessus (c'est-à-dire en script console et non cgi), on obtient dans la console les résultats suivants :

```
1. Content-Type: text/plain
2.
3. TMP : C:\Users\SERGET~1\AppData\Local\Temp
4. COMPUTERNAME : GPORTPERS3
5. USERDOMAIN : Gportpers3
```


Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

```
6. VS100COMNTOOLS : D:\Programs\dotnet\Visual Studio 10\Common7\Tools\
7. VISUALSTUDIODIR : D:\Documents\Visual Studio 2010
8. PSMODULEPATH : C:\Windows\system32\WindowsPowerShell\v1.0\Modules\
9. COMMONPROGRAMFILES : C:\Program Files (x86)\Common Files
10. PROCESSOR_IDENTIFIER : Intel64 Family 6 Model 42 Stepping 7, GenuineIntel
11. PROGRAMFILES : C:\Program Files (x86)
12. PROCESSOR_REVISION : 2a07
13. SYSTEMROOT : C:\Windows
14. PATH : D:\Programs\ActivePython\Python2.7.2\;D:\Programs\ActivePython\Python2.7.2\Scripts;C:
\Program Files\Common Files\Microsoft Shared\Windows Live;...
15. PROGRAMFILES(X86) : C:\Program Files (x86)
16. WINDOWS_TRACING_FLAGS : 3
17. TEMP : C:\Users\SERGET~1\AppData\Local\Temp
18. COMMONPROGRAMFILES(X86) : C:\Program Files (x86)\Common Files
19. PROCESSOR_ARCHITECTURE : x86
20. ALLUSERSPROFILE : C:\ProgramData
21. LOCALAPPDATA : C:\Users\Serge TahÚ\AppData\Local
22. HOMEPATH : \Users\Serge TahÚ
23. PROGRAMW6432 : C:\Program Files
24. USERNAME : Serge TahÚ
25. LOGONSERVER : \\GPORTPERS3
26. PROMPT : $P$G
27. SESSIONNAME : Console
28. PROGRAMDATA : C:\ProgramData
29. PATHEXT : .COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC;.py;.pyw
30. FP_NO_HOST_CHECK : NO
31. WINDIR : C:\Windows
32. PYTHON : D:\Programs\python\python2.7.2\python
33. WINDOWS_TRACING_LOGFILE : C:\BVTBin\Tests\installpackage\csilogfile.log
34. HOMEDRIVE : C:
35. SYSTEMDRIVE : C:
36. COMSPEC : C:\Windows\system32\cmd.exe
37. NUMBER_OF_PROCESSORS : 8
38. VBOX_INSTALL_PATH : D:\Programs\systeme\Oracle\VirtualBox\
39. APPDATA : C:\Users\Serge TahÚ\AppData\Roaming
40. PROCESSOR_LEVEL : 6
41. PROCESSOR_ARCHITEW6432 : AMD64
42. COMMONPROGRAMW6432 : C:\Program Files\Common Files
43. OS : Windows_NT
44. PUBLIC : C:\Users\Public
45. USERPROFILE : C:\Users\Serge TahÚ
```

Dans un navigateur Web (c'est alors le script cgi qui est exécuté), on obtient les résultats suivants :



On notera que selon le contexte d'exécution, l'environnement obtenu n'est pas le même.

Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

XIII-C-2 - Le client programmé

Le programme (client_web_04)

```
1. # -*- coding=utf-8 -*-
2.
3. import httplib
4.
5. # constantes
6. HOST="localhost"
7. URL="/cgi-bin/web_04.py"
8. # connexion
9. connexion=httplib.HTTPConnection(HOST)
10. # envoi de la requête
11. connexion.request("GET", URL)
12. # traitement de la réponse
13. reponse=connexion.getresponse()
14. # contenu
15. print reponse.read()
```

Résultats

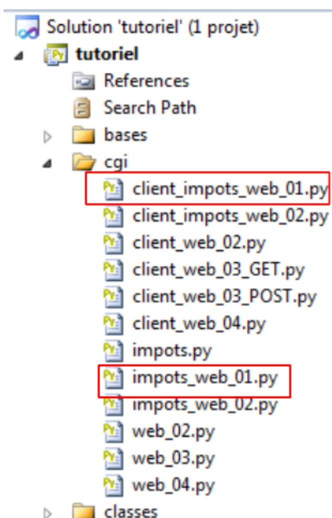
```
1. SERVER_SOFTWARE : Apache/2.2.17 (Win32) PHP/5.3.5
2. SCRIPT_NAME : /cgi-bin/web_04.py
3. SERVER_SIGNATURE :
4. REQUEST_METHOD : GET
5. SERVER_PROTOCOL : HTTP/1.1
6. QUERY_STRING :
7. SYSTEMROOT : C:\Windows
8. SERVER_NAME : localhost
9. REMOTE_ADDR : 127.0.0.1
10. SERVER_PORT : 80
11. SERVER_ADDR : 127.0.0.1
12. DOCUMENT_ROOT : D:/Programs/sgbd/wamp/www/
13. COMSPEC : C:\Windows\system32\cmd.exe
14. SCRIPT_FILENAME : D:/Programs/sgbd/wamp/bin/apache/Apache2.2.17/cgi-bin/web_04.py
15. SERVER_ADMIN : admin@localhost
16. PATH : D:\Programs\ActivePython\Python2.7.2\;D:\Programs\ActivePython\Python2.7.
17. 2\Scripts;C:\Program Files\Common Files\Microsoft Shared\Windows Live;...
18. HTTP_HOST : localhost
19. PATHEXT : .COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC;.py;.pyw
20. REQUEST_URI : /cgi-bin/web_04.py
21. WINDIR : C:\Windows
22. GATEWAY_INTERFACE : CGI/1.1
23. REMOTE_PORT : 58468
24. HTTP_ACCEPT_ENCODING : identity
```

On notera que le client programmé ne reçoit pas exactement la même réponse que le navigateur Web. C'est parce que ce dernier a envoyé au serveur Web des informations qui ont été utilisées par le serveur Web pour créer sa réponse. Ici, le client programmé n'a envoyé aucune information sur lui-même.

Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

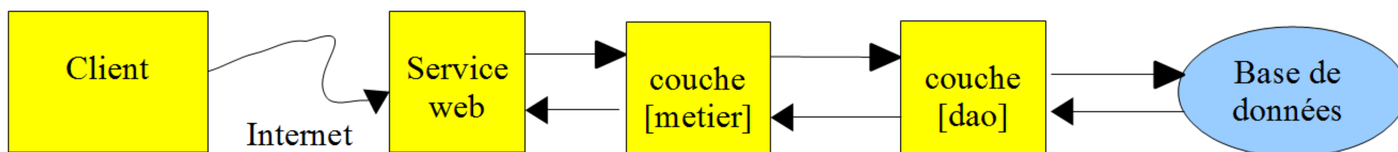
XIV - Exercice [IMPOTS] avec un service Web

Nous revenons à l'exercice [IMPOTS].



XIV-A - Le serveur

Nous nous plaçons dans l'architecture client-serveur suivante :



Côté serveur, nous allons utiliser de nouveau une architecture trois couches. La couche [DAO] sera implémentée par la classe *ImpotsMySQL* utilisée dans la version avec le Sgbd MySQL. La couche [metier] sera implémentée par la classe [ImpotsMetier] déjà étudiée. Il ne reste donc qu'à écrire le service Web. Celui-ci reçoit de ses clients un paramètre *params* sous la forme *params= oui,2,200000* où le premier élément indique si le contribuable est marié ou non, le deuxième son nombre d'enfants, le troisième son salaire annuel.

Le service Web est le suivant (impots_web_01) :

impots_web_01
```
1. #!D:\Programs\ActivePython\Python2.7.2\python.exe
2.
3. # -*- coding=Utf-8 -*-
4. # import du module des classes Impots*
5. from impots import *
6. import cgi,cgitb,re
7.
8. # on autorise l'affichage d'informations de débogage
9. cgitb.enable()
10. sys.stderr=sys.stdout
11.
12. # ------------------------------------------------
13. # le service Web des impôts
14. # ------------------------------------------------
15.
```


Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

impots_web_01

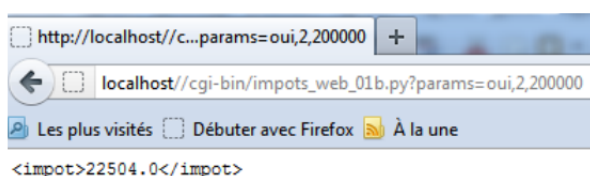
```
16. # les données nécessaires au calcul de l'impôt ont été placées dans la table mysqL TABLE
17. # appartenant à la base BASE. La table a la structure suivante
18. # limites decimal(10,2), coeffR decimal(6,2), coeffN decimal(10,2)
19. # les paramètres des personnes imposables (statut marital, nombre d'enfants, salaire annuel)
20. # sont envoyés par le client sous la forme params=statut marital, nombre d'enfants, salaire annuel
21. # les résultats (statut marital, nombre d'enfants, salaire annuel, impôt à payer) sont renvoyés au client
22. # sous la forme <impot>valeur</impot>
23. # ou sous la forme <erreur>msg</erreur>, si les paramètres sont invalides
24.
25. # le serveur renvoie au client du texte non formaté
26. print "Content-Type: text/plain\n"
27.
28. # définition des constantes
29. USER="root"
30. PWD=""
31. HOTE="localhost"
32. BASE="dbimpots"
33. TABLE="impots"
34.
35.
36. # on crée la couche [metier]
37. # Python ne semble pas avoir de support pour les sessions
38. # en PHP, on aurait utilisé une session pour mémoriser l'objet metier
39. # ici, on le construit systématiquement à partir des données de la base de données MySQL
40.
41. # instanciation couche [metier]
42. try:
43.     metier=ImpotsMetier(ImpotsMySQL(HOTE,USER,PWD,BASE,TABLE))
44. except (IOError, ImpotsError) as infos:
45.     print ("<erreur>Une erreur s'est produite : {0}</erreur>".format(infos))
46.     sys.exit()
47.
48. # on récupère la ligne envoyée par le client au serveur
49. params=cgi.FieldStorage().getlist('params')
50. # si pas de paramètre alors erreur
51. if not params:
52.   print "<erreur>Le parametre [params] est absent<erreur>"
53.   sys.exit()
54.
55. # on exploite le paramètre params
56. items=params[0].strip().lower().split(',')
57. # il ne doit y avoir que 3 champs
58. if len(items)!=3:
59.   print "<erreur>[%s] : nombre de parametres invalide</erreur>" % (params[0])
60.   sys.exit()
61. # le premier paramètre (statut marital) doit être oui/non
62. marie=items[0].strip()
63. if marie!="oui" and marie != "non":
64.   print "<erreur>[%s] : 1er parametre invalide</erreur>\n"% (params[0])
65.   sys.exit()
66. # le second paramètre (nbre d'enfants) doit être un nombre entier
67. match=re.match(r"^\s*(\d+)\s*$",items[1])
68. if not match:
69.   print "<erreur>[%s] : 2e parametre invalide</erreur>\n"% (params[0])
70.   sys.exit()
71. enfants=int(match.groups()[0])
72. # le troisième paramètre (salaire) doit être un nombre entier
73. match=re.match(r"^\s*(\d+)\s*$",items[2])
74. if not match:
75.   print "<erreur>[%s] : 3e parametre invalide</erreur>\n"% (params[0])
76.   sys.exit()
77. salaire=int(match.groups()[0])
78. # on calcule l'impôt
79. impot=metier.calculer(marie,enfants,salaire)
80. # on renvoie le résultat
81. print "<impot>%s</impot>\n" % (impot)
82. # fin
```

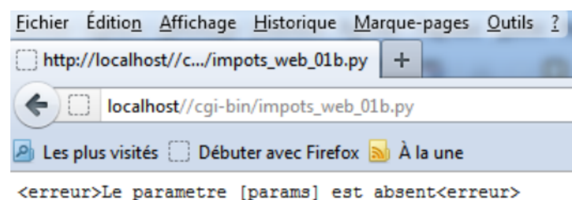
Notes :

Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

- ligne 5 : on importe les objets du fichier *impots.py ;*
- ligne 9 : déboguer un script Cgi peut être problématique. Le module *cgitb* envoie, en cas d'erreur, la raison du plantage. Pour cela, il faut rediriger la sortie erreur standard (*sys.stderr*) vers la sortie standard (*sys.stdout*) (ligne 10) ;
- ligne 26 : l'entête HTTP qui fixe la nature du document envoyé, ici un texte non formaté. Le service Web renvoie sa réponse sous forme d'une unique ligne de texte :
- *<erreur>message</erreur>* s'il y a une erreur ;
- *<impot>valeur</impot>* où *valeur* est le montant de l'impôt.
- ligne 43: instanciation des couches [DAO] et [metier] ;
- lignes 44-46 : en cas d'erreur de l'instanciation, le script envoie sa réponse et se termine ;
- lignes 49-53 : le paramètre 'params' est récupéré. S'il est absent, le script envoie sa réponse et se termine ;
- ligne 56 : le paramètre 'oui,2,200000' est décomposé en 3 champs dans le tableau *items ;*
- lignes 58-77 : la validité des 3 champs est vérifiée. S'il y a une erreur, le script envoie sa réponse et se termine ;
- ligne 79 : l'impôt est calculé ;
- ligne 81 : et envoyé au client.

Dans un navigateur Web, on obtient les résultats suivants :





XIV-B - Un client programmé

Le programme (client_impots_web_01)

```
1. # -*- coding=utf-8 -*-
2.
3. import httplib,urllib,re
4.
5. # constantes
6. HOST="localhost"
7. URL="/cgi-bin/impots_web_01b.py"
8. data=("oui,2,200000","non,2,200000","oui,3,200000","non,3,200000","x,y,z,t","x,2,200000", "oui,x,200000","oui,
9.
10. # connexion
11. connexion=httplib.HTTPConnection(HOST)
12. # suivi
13. #connexion.set_debuglevel(1)
14. # on boucle sur les données à envoyer au serveur
15. for params in data:
16.     # les paramètres doivent être encodés avant d'être envoyés au serveur
17.     parametres = urllib.urlencode({'params': params})
18.     # envoi de la requête
19.     connexion.request("POST",URL,parametres)
20.     # traitement de la réponse (ligne de texte)
21.     reponse=connexion.getresponse().read()
22.     lignes=reponse.split("\n")
23.     # la ligne de texte est de la forme
24.     # <impot>xxx</impot>
25.     # ou <erreur>xxx</erreur>
26.     impot=re.match(r"^<impot>(\S+)</impot>\s*$",lignes[0])
27.     if impot:
28.         print "params=%s, impot=%s" % (params,impot.groups()[0])
29.     else:
30.         erreur=re.match(r"^<erreur>(.+)</erreur>\s*$",lignes[0])
31.         if erreur:
32.             print "params=%s, erreur=%s" % (params,erreur.groups()[0])
```


Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Le programme (client_impots_web_01)

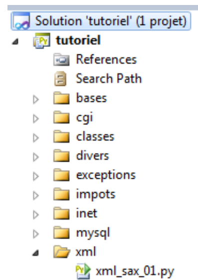
```
33.         else:
34.             print lignes
35. # fermeture connexion
36. connexion.close()
```

Résultats

```
1. params=oui,2,200000, impot=22504.0
2. params=non,2,200000, impot=33388.0
3. params=oui,3,200000, impot=16400.0
4. params=non,3,200000, impot=22504.0
5. params=x,y,z,t, erreur=[x,y,z,t] : nombre de parametres invalide
6. params=x,2,200000, erreur=[x,2,200000] : 1er parametre invalide
7. params=oui,x,200000, erreur=[oui,x,200000] : 2e parametre invalide
8. params=oui,2,x, erreur=[oui,2,x] : 3e parametre invalide
```


Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

XV - Traitement de documents XML



Nous considérons le document XML suivant :

```
1. <tribu>
2.   <enseignant>
3.     <personne sexe="M">
4.       <nom>dupont</nom>
5.       <prenom>jean</prenom>
6.       <age>28</age>
7.       ceci est un commentaire
8.     </personne>
9.     <section>27</section>
10.   </enseignant>
11.   <etudiant>
12.     <personne sexe="F">
13.       <nom>martin</nom>
14.       <prenom>charline</prenom>
15.       <age>22</age>
16.     </personne>
17.     <formation>dess IAIE</formation>
18.   </etudiant>
19. </tribu>
```

Nous analysons ce document pour produire la sortie console suivante :

```
1. tribu
2.   enseignant
3.    (personne,(sexe,M) )
4.     nom
5.      [dupont]
6.     /nom
7.     prenom
8.      [jean]
9.     /prenom
10.     age
11.      [28]
12.     /age
13.    /personne
14.    section
15.     [27]
16.    /section
17.   /enseignant
18.   etudiant
19.    (personne,(sexe,F) )
20.     nom
21.      [martin]
22.     /nom
23.     prenom
24.      [charline]
25.     /prenom
26.     age
27.      [22]
28.     /age
29.    /personne
```


Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

```
30.     formation
31.      [dess IAIE]
32.     /formation
33.    /etudiant
34.   /tribu
```

Il nous faut savoir comment reconnaître :

- une balise de début telle *<formation> ;*
- une balise de fin telle *</enseignant> ;*
- une balise de début avec attributs telle *<personne sexe="F"> ;*
- le corps d'une balise telle *martin* dans *<nom>martin</nom>.*

Le programme d'analyse d'un code XML est appelé un **parseur** XML. Deux modules nous fournissent les fonctionnalités pour analyser un code XML : *xml.sax* et *xml.sax.handler*.

Le module [xml.sax] nous fournit un parseur XML par l'instruction :

xml_parser=xml.sax.make_parser()

Ce parseur analyse le texte XML séquentiellement. Il appelle des méthodes de l'utilisateur sur des événements :

- la méthode *startElement* sur un début de balise ;
- la méthode *endElement* sur une fin de balise ;
- la méthode *characters* sur le corps d'une balise.

Il nous faut indiquer au parseur, la classe implémentant ces méthodes :

xml_parser.setContentHandler(XmlHandler())

Nous passons à la méthode *setContentHandler* du parseur, une instance de classe implémentant les méthodes *startElement*, *endElement*, *characters*. La classe utilisée est une classe dérivée de la classe *xml.sax.handler.ContentHandler*. Les méthodes précédentes sont appelées avec des paramètres :

def startElement(self,name,attributs):

- *name* est le nom de la balise de début. *attributs* est le dictionnaire des attributs de la balise. Ainsi pour la balise *<personne sexe="M">* on aura *name="personne"* et *attributs={'sexe':'M'}*

def endElement(self,name):

- *name* est le nom de la balise de fin. Ainsi pour la balise *</etudiant>* on aura *name='etudiant'*.

def characters(self,data):

- data est le corps de la balise. Ainsi si la balise est

<nom>

dupont

</nom>

on aura *data='\r\n dupont\r\n '*. En général, on se débarrassera des blancs qui précèdent et suivent la donnée.

Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Ceci expliqué, on peut passer au script d'analyse d'un document XML.

Le programme (xml_sax_01)

```
1. # -*- coding=utf-8 -*-
2.
3. import xml.sax, xml.sax.handler,re
4.
5. # classe de gestion XML
6. class XmlHandler(xml.sax.handler.ContentHandler):
7.
8.     # fonction appelée lors de la rencontre d'une balise de début
9.     def startElement(self,name,attributs):
10.         global depth
11.         # une suite d'espaces (indentation)
12.         print " " * depth,
13.         # attributs
14.         precisions=""
15.         for (attrib,valeur) in attributs.items():
16.             precisions+="(%s,%s) " % (attrib,valeur)
17.         # on affiche le nom de la balise et les éventuels attributs
18.         if precisions :
19.             print "(%s,%s)" % (name,precisions)
20.         else :
21.             print name
22.         # un niveau d'arborescence en plus
23.         depth+=1
24.         # est-ce une balise de données ?
25.         global balisesDonnees,baliseDeDonnees
26.         if balisesDonnees.has_key(name.lower()):
27.             baliseDeDonnees=1
28.
29.      # la fonction appelée lors de la rencontre d'une balise de fin
30.      def endElement(self,name):
31.         # fin de balise
32.         # niveau d'indentation
33.         global depth
34.         depth-=1
35.         # une suite d'espaces (indentation)
36.         print " " * depth,
37.         # le nom de la balise
38.         print "/%s" % (name)
39.
40.      # la fonction d'affichage des données
41.      def characters(self,data):
42.         # données
43.         global baliseDeDonnees
44.
45.         # la balise courante est-elle une balise de données ?
46.         if not baliseDeDonnees :
47.             return
48.         # niveau d'indentation
49.         global depth
50.         # une suite d'espaces (indentation)
51.         print " " * depth,
52.         # les données sont affichées
53.         match=re.match(r"^\s*(.*)\s*$",data)
54.         if match:
55.             print "[%s]" % (match.groups()[0])
56.         # fin de la balise de données
57.         baliseDeDonnees=False
58.
59. # ------------------------------------------- main
60. # le programme
61. # données
62. file="data.xml"       # le fichier xml
63. depth=0               # niveau d'indentation=profondeur dans l'arborescence
64. balisesDonnees={"nom":1,"prenom":1,"age":1,"section":1,"formation":1}
65. baliseDeDonnees=True     # à vrai, indique qu'on a affaire à une balise de données
66.
67. # on crée un objet d'analyse de texte xml
68. xml_parser=xml.sax.make_parser()
```


Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Le programme (xml_sax_01)

```
69. # le gestionnaire de balises
70. xml_parser.setContentHandler(XmlHandler())
71. # exploitation du fichier xml
72. xml_parser.parse(file)
```

Notes :

- le script utilise la bibliothèque de fonctions des modules *xml.sax*, *xml.sax.handler* (ligne 3) ;
- ligne 62 : le fichier XML analysé ;
- ligne 68 : le parseur XML;
- ligne 70 : le gestionnaire des événements émis par le parseur sera une instance de la classe *XmlHandler ;*
- ligne 72 : l'analyse du document XML est lancée ;
- ligne 6 : la classe implémentant les méthodes *startElement*, *endElement*, *characters*. Elle est dérivée de la classe *xml.sax.handler.ContentHandler* qui implémente des méthodes utilisées par le parseur ;
- ligne 9 : la méthode *startElement ;*
- ligne 30 : la méthode *endElement ;*
- ligne 41 : la méthode *characters.*

Les résultats sont ceux présentés au début de ce paragraphe.

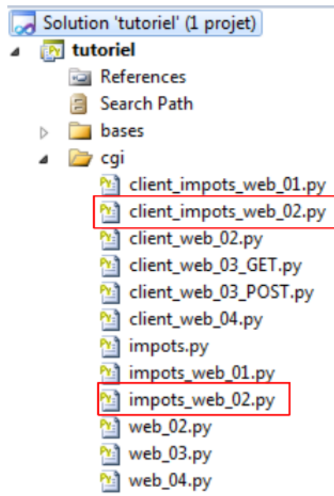
Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

XVI - Exercice [IMPOTS] avec XML

Dans cet exercice déjà étudié de nombreuses fois, le serveur renvoie les résultats au client sous la forme d'un flux XML :

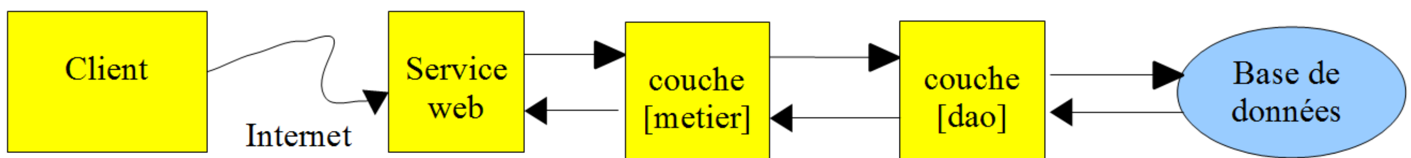
- *<reponse><erreur>msg</erreur></reponse>* en cas d'erreur ;
- *<reponse><impot>valeur</impot></reponse>* si l'impôt a pu être calculé.

Nous utilisons ce que nous venons d'apprendre sur l'analyse d'un document XML.



XVI-A - Le service Web

Le service Web n'est pas différent du service Web étudié précédemment si ce n'est que la réponse XML envoyée au client est légèrement différente. L'architecture reste la même :



Le service web (impots_web_02)

```
1. #!D:\Programs\ActivePython\Python2.7.2\python.exe
2.
3. # -*- coding=Utf-8 -*-
4. # import du module des classes Impots*
5. from impots import *
6. import cgi,cgitb,re
7.
8. # on autorise l'affichage d'informations de débogage
9. cgitb.enable()
10.
11. # -------------------------------------------------
12. # le service Web des impôts
13. # -------------------------------------------------
14.
15. # les données nécessaires au calcul de l'impôt ont été placées dans la table mysqL TABLE
16. # appartenant à la base BASE. La table a la structure suivante
17. # limites decimal(10,2), coeffR decimal(6,2), coeffN decimal(10,2)
```


Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Le service web (impots_web_02)

```
18. # les paramètres des personnes imposables (statut marital, nombre d'enfants, salaire annuel)
19. # sont envoyés par le client sous la forme params=statut marital, nombre d'enfants, salaire annuel
20. # les résultats (statut marital, nombre d'enfants, salaire annuel, impôt à payer) sont renvoyés au client
21. # sous la forme <impot>valeur</impot>
22. # ou sous la forme <erreur>msg</erreur>, si les paramètres sont invalides
23.
24. # le serveur renvoie au client du texte non formaté
25. print "Content-Type: text/plain\n"
26. # début de la réponse
27. print "<reponse>"
28.
29. # définition des constantes
30. USER="root"
31. PWD=""
32. HOTE="localhost"
33. BASE="dbimpots"
34. TABLE="impots"
35.
36. # instanciation couche [metier]
37. try:
38.     metier=ImpotsMetier(ImpotsMySQL(HOTE,USER,PWD,BASE,TABLE))
39. except (IOError, ImpotsError) as infos:
40.     print ("<erreur>Une erreur s'est produite : {0}</erreur>".format(infos))
41.     sys.exit()
42.
43. # on récupère la ligne envoyée par le client au serveur
44. params=cgi.FieldStorage().getlist('params')
45. # si pas de paramètres alors erreur
46. if not params:
47.     print "<erreur>Le parametre [params] est absent<erreur></reponse>"
48.     sys.exit()
49.
50. # on exploite le paramètre params
51. #print "parametres recus --> %s\n" %(params)
52. items=params[0].strip().lower().split(',')
53. # il ne doit y avoir que 3 champs
54. if len(items)!=3:
55.     print "<erreur>[%s] : nombre de parametres invalide</erreur></reponse>" % (params[0])
56.     sys.exit()
57. # le premier paramètre (statut marital) doit être oui/non
58. marie=items[0].strip()
59. if marie!="oui" and marie != "non":
60.     print "<erreur>[%s] : 1er parametre invalide</erreur></reponse>\n"% (params[0])
61.     sys.exit()
62. # le second paramètre (nbre d'enfants) doit être un nombre entier
63. match=re.match(r"^\s*(\d+)\s*$",items[1])
64. if not match:
65.     print "<erreur>[%s] : 2e parametre invalide</erreur></reponse>\n"% (params[0])
66.     sys.exit()
67. enfants=int(match.groups()[0])
68. # le troisième paramètre (salaire) doit être un nombre entier
69. match=re.match(r"^\s*(\d+)\s*$",items[2])
70. if not match:
71.     print "<erreur>[%s] : 3e parametre invalide</erreur></reponse>\n"% (params[0])
72.     sys.exit()
73. salaire=int(match.groups()[0])
74. # on calcule l'impôt
75. impot=metier.calculer(marie,enfants,salaire)
76. # on renvoie le résultat
77. print "<impot>%s</impot></reponse>\n" % (impot)
78. # fin
```

Notes :

Ce service Web ne diffère du précédent que par la nature de sa réponse :

<reponse><erreur>msg</erreur></reponse> en cas d'erreur au lieu de *<erreur>msg</erreur>*

Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.
http://tahe.developpez.com/divers/python/exemples/

<reponse><impot>valeur</impot></reponse> si l'impôt a pu être calculé au lieu de *<impot>valeur</impot>*

XVI-B - Le client programmé

Notre client doit analyser la réponse XML envoyée par le service Web. Nous appliquons ce qui a été vu dans l'analyse d'un document XML.

Le programme (client_impots_web_02)

```
1. # -*- coding=utf-8 -*-
2.
3. import httplib,urllib,re
4. import xml.sax, xml.sax.handler
5. # classe de gestion XML
6. class XmlHandler(xml.sax.handler.ContentHandler):
7.
8.     # fonction appelée lors de la rencontre d'une balise de début
9.     def startElement(self,name,attributes):
10.         # on note l'élément courant
11.         global elementcourant
12.         elementcourant=name.strip().lower()
13.
14.     # la fonction appelée lors de la rencontre d'une balise de fin
15.     def endElement(self,name):
16.         # on ne fait rien
17.         pass
18.
19.     # la fonction de gestion des données
20.     def characters(self,data):
21.         # données
22.         global elementcourant,elements
23.
24.         # les données sont récupérées
25.         match=re.match(r"^\s*(.+?)\s*$",data)
26.         if match:
27.             elements[elementcourant]=match.groups()[0].lower()
28.
29. def getResultatsXml(reponse):
30.     # on analyse la reponse XML
31.     xml.sax.parseString(reponse,XmlHandler())
32.     # on rend les résultats
33.     if elements.has_key('erreur'):
34.         return (elements['erreur'],"")
35.     else:
36.         return ("",elements['impot'])
37.
38. # ------------------------------------------------------------ main
39. # constantes
40. HOST="localhost"
41. URL="/cgi-bin/impots_web_02b.py"
42. data=("oui,2,200000","non,2,200000","oui,3,200000","non,3,200000","x,y,z,t","x,2,200000", "oui,x,200000","oui
43. # variables globales
44. elementcourant=""
45. elements={}
46.
47. # connexion
48. connexion=httplib.HTTPConnection(HOST)
49. # suivi
50. #connexion.set_debuglevel(1)
51. # on boucle sur les données à envoyer au serveur
52. for params in data:
53.     # les paramètres doivent être encodés avant d'être envoyés au serveur
54.     parametres = urllib.urlencode({'params': params})
55.     # envoi de la requête
56.     connexion.request("POST",URL,parametres)
57.     # traitement de la réponse (flux Xml)
58.     reponse=connexion.getresponse().read()
59.     # exploitation du fichier xml
60.     (erreur,impot)=getResultatsXml(reponse)
```


Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Le programme (client_impots_web_02)

```
61.     if not erreur:
62.         print "impot[%s]=%s" % (params,impot)
63.     else:
64.         print "erreur[%s]=%s" % (params,erreur)
```

Notes :

- le flux XML du service Web est exploité par la fonction *getResultatsXml* (ligne 60) ;
- ligne 29 : la fonction *getResultatsXml* ;
- ligne 31 : la réponse XML du service Web est analysée par une instance de la classe *XmlHandler* définie ligne 6 ;
- ligne 6 : la classe *XmlHandler* implémente les trois méthodes *startElement*, *endElement*, *characters*. À l'aide de ces trois méthodes, on crée un dictionnaire. Les clés sont les noms des balises *<erreur>* et *<impot>* et les valeurs sont les données associées à ces deux balises ;
- lignes 33-36 : la fonction *getResultatsXml* retourne un tuple à deux éléments :
- *(erreur,"")* si l'analyse du flux XML a montré l'existence de la balise *<erreur>*. *erreur* représente alors le contenu de cette balise ;
- *("",impot)* si l'analyse du flux XML a montré l'existence de la balise *<impot>*. *impot* représente alors le contenu de cette balise.
- ligne 60 : le résultat de la fonction *getResultatsXml* est récupéré puis exploité lignes 61-64.

Les résultats

```
1. impot[oui,2,200000]=22504.0
2. impot[non,2,200000]=33388.0
3. impot[oui,3,200000]=16400.0
4. impot[non,3,200000]=22504.0
5. erreur[x,y,z,t]=[x,y,z,t] : nombre de parametres invalide
6. erreur[x,2,200000]=[x,2,200000] : 1er parametre invalide
7. erreur[oui,x,200000]=[oui,x,200000] : 2e parametre invalide
8. erreur[oui,2,x]=[oui,2,x] : 3e parametre invalide
```

MCours.com

Copyright ® 2012 Serge Tahé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.