« Algorithmique et Langage Python » Formation Liesse professeurs CPGE

Télécom SudParis - ENSIIE

20-21 juin 2013

Plan de la formation

- Jour 1
  - Généralités sur le langage et son environnement
  - Algorithmique de base illustrée en Python :
    - Variables, expressions et instructions
    - Instructions conditionnelles, itératives
    - Fonctions
    - Fichiers
  - Manipulation de quelques structures de données
  - Algorithmes de recherche et de tri
- Jour 2
  - Algèbre linéaire et analyse numérique en Python (résolution, affichage graphique...) avec `numpy`, `scipy`, `matplotlib`
  - Panorama des bibliothèques Python
  - Boîtes à outils Python : installation et configuration de *IDLE* et *IPython*

Horaires 20 juin

- 08:30 : Accueil au forum de Télécom SudParis - Petit déjeuner en salle B313
- 09:00 : **Formation**
- 10:30 : Pause
- 10:45 : **Formation**
- 12:15 : Déjeuner offert dans salle du restaurant
- 13:45 : **Formation**
- 15:15 : Pause
- 15:30 : **Formation**
- 17:00 : Fin de la journée

Horaires 21 juin

- 08:30 : Accueil et Petit déjeuner en salle B313
- 09:00 : **Formation**
- 10:30 : Pause
- 10:45 : **Formation**
- 12:15 : Déjeuner offert dans salle du restaurant
- 13:45 : **Formation**
- 15:15 : Pause
- 15:30 : **Formation**
- 16:45 : Évaluation de la formation
- 17:00 : Fin de la formation

Objectifs

- Principes de conception d'un algorithme
- Syntaxe et sémantique du langage Python
- Manipuler les structures de contrôle de base
- Maîtriser les structures de données de base
- Méthodes de recherche et de tri usuelles
- Manipuler l'environnement Python (IDLE, IPython)
- *numpy*, *scipy* pour l'analyse numérique et le tracé graphique avec *matplotlib*
- Illustrer en python d'autres applications (BD, web . . .)

Bios

- Christophe Mouilleron
  - Professeur agrégé de mathématiques à l'ENSIIE
  - Impliqué dans divers enseignements dont la programmation fonctionnelle (Langage Ocaml)
  - A enseigné la programmation en Python au niveau L1 à l'Université de Perpignan en 2011
- Olivier Berger
  - Ingénieur Recherche à Télécom SudParis
  - Un des premiers Pythonistes en France – Impliqué dans la traduction de documentation Python en français dès 1999
  - Spécialiste Logiciel Libre. Recherche sur l'interopérabilité des outils de développement dans les projets libres.
- Christian Schüller
  - Ingénieur d'études à Télécom SudParis
  - Impliqué dans l'enseignement de l'algorithmique et la programmation (Langage C et Java).
  - Équipe HP2 (Architecture et programmation parallèle)

Sondage

Python

http ://www.python.org/

Présentation du langage (Wikipédia)

Définition Wikipédia :

*Python est un langage de programmation objet, **interprété**, multi-paradigme, et multi-plateformes.*
*Il favorise la **programmation impérative structurée** et orientée objet.*
Il est doté d'un typage dynamique fort, d'une gestion automatique de la mémoire par ramasse-miettes et d'un système de gestion d'exceptions ; il est ainsi similaire à Perl, Ruby, Scheme, Smalltalk et Tcl.

Distribution Python

- Distribution ?
  - le *langage* (syntaxe, interpréteur)
  - les *bibliothèques*
    - bibliothèques « standard » Python
    - bibliothèques additionnelles : par ex. ensemble PyLab : NumPy (tableaux et matrices) + SciPy (ajoute maths avancées, traitement signal, optimisation, statistiques, etc.) + Matplotlib (visualisation).
- Linux : disponible en standard (python 2.6/2.7 ou python 3)
- Windows
  - Canopy Express (Enthought) - Windows 32 bits
  - Anaconda - Windows 32 ou 64 bits
- Mac
  - Anaconda - OSX 64 bits

Versions

Deux versions partiellement incompatibles

- **Python 2** : Python 2.7 en 2010 (à préférer car mieux connue)
- Python 3.x : réservé aux experts - pas rétro-compatible

Différentes implémentations

- CPython : compilé en C : **la version standard**
- Jython (interpréteur en Java + bibliothèques Java)
- IronPython (.Net)
- etc.

Historique du langage

- Créé par Guido van Rossum en 1990

- Maintenu par une communauté de bénévoles, sous couvert de la *Python Software Foundation*

Logiciel libre -> communauté

- License de logiciel libre (proche license BSD)
- Humour spécial : *Monty Pythons* (Spam, Eggs, etc.)
- Association francophone Python : AFPY

Controversé ?

- Les *tabulations*
- Le typage dynamique
- *There's Only One Way To Do It* vs *There's more than one way to do it.* en Perl

Bibliographie

- Documentation officielle : http ://www.python.org/doc/
  - **Tutoriel Python** officiel : `http://www.afpy.org/doc/python/2.7/tutorial/` (en français, mais pas intégralement traduit)
- Livres :
  - **Apprendre à programmer avec Python**, G. Swinnen, http ://inforef.be/swi/python.htm (attention : plusieurs éditions)
  - **Programmation Python**, Tarek Ziadé, Eyrolles, 2009
- Autres :
  - Mémento bases Python 3 : http ://perso.limsi.fr/pointal/python :memento

Windows

- lancement d'un « terminal » / « ligne de commande » : *Menu Démarrer* > *Exécuter*, puis tapez `cmd`.
- changer de répertoire courant :

```
C:[...]\Olivier> cd "Mes Documents"\
C:[...]\Olivier> cd D:\LIESSEPY\
```

- lister les fichiers présents dans le répertoire courant :

```
D:LIESSEPY> dir [*.py]
```

- Lancer un programme Python présent dans le répertoire courant :

```
D:LIESSEPY> python helloworld.py
```

Unix

- lancer un terminal / shell (icône de terminal depuis le menu, visible si souris en haut à gauche de l'écran)
- changer de répertoire courant :

```
$ cd Documents/liessepy
```

- lister les fichiers présents dans le répertoire courant :

```
$ dir [*.py]
...
$ ls [*.py]
```

- Lancer un programme Python présent dans le répertoire courant :

```
$ python helloworld.py
```

Quitter l'interpréteur ?

Classique Unix :

- `Control + C` : interrompt le programme
- `Control + D` : quitte

Deux façons de faire tourner des programmes

- Session interactive
  Interpréteur = Calculatrice améliorée
  Cf. le chapitre « 3. Introduction informelle à Python » du Tutoriel Python officiel (*)

- Lancement de scripts
  - Ecrire un fichier source (`.py`) dans un éditeur
  - Lancer son exécution (pas de compilation nécessaire : langage interprété)
  - Le distribuer à d'autres

Outils de programmation / tests

- interpréteur standard
- IDLE
- IPython

Interpréteur Python

Lancement de l'interpréteur, en mode *interactif* :

```
$ python
Python 2.7.3 (default, Jan  2 2013, 16:53:07)
[GCC 4.7.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Prompt primaire : >>>

```
>>> le_monde_est_plat = 1
>>> if le_monde_est_plat:
...     print "Attention au rebord!"
...
Attention au rebord!
```

Prompt secondaire : . . . , à l'intérieur d'une suite d'instructions.
Retour chariot ↩ pour revenir au prompt primaire.

Outil de développement IDLE

- Editeur + interpréteur (*python shell*)
- Fonctions intéressantes : débogueur, etc.
- 100% Python « standard » (tkinter) : portable
- Ergonomie discutable (ancien)

Copie écran IDLE

Python Shell

File Edit Shell Debug Options Windows Help

```
Python 2.7.3 (default, Jan  2 2013, 16:53:07)
[GCC 4.7.2] on linux2
Type "copyright", "credits" or "license()" for more information.
>>> le_monde_est_plat = 1
>>> if le_monde_est_plat:
        print "Attention au rebord!"


Attention au rebord!
>>> |
```

Ln: 10 Col: 4

Copie écran IDLE (2)

coin.py - /home/olivier/git/fusionforge.int-evry.fr/liessepy/liessepy/coin.py

File Edit Format Run Options Windows Help

```
le_monde_est
if le_monde_
    print "A                bord!"
```

Python Shell
Check Module Alt+X
Run Module F5

Ln: 4 Col: 0

Python Shell

File Edit Shell Debug Options Windows Help

```
Python 2.7.3 (default, Jan  2 2013, 16:53:07)
[GCC 4.7.2] on linux2
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
Attention au rebord!
>>>
```

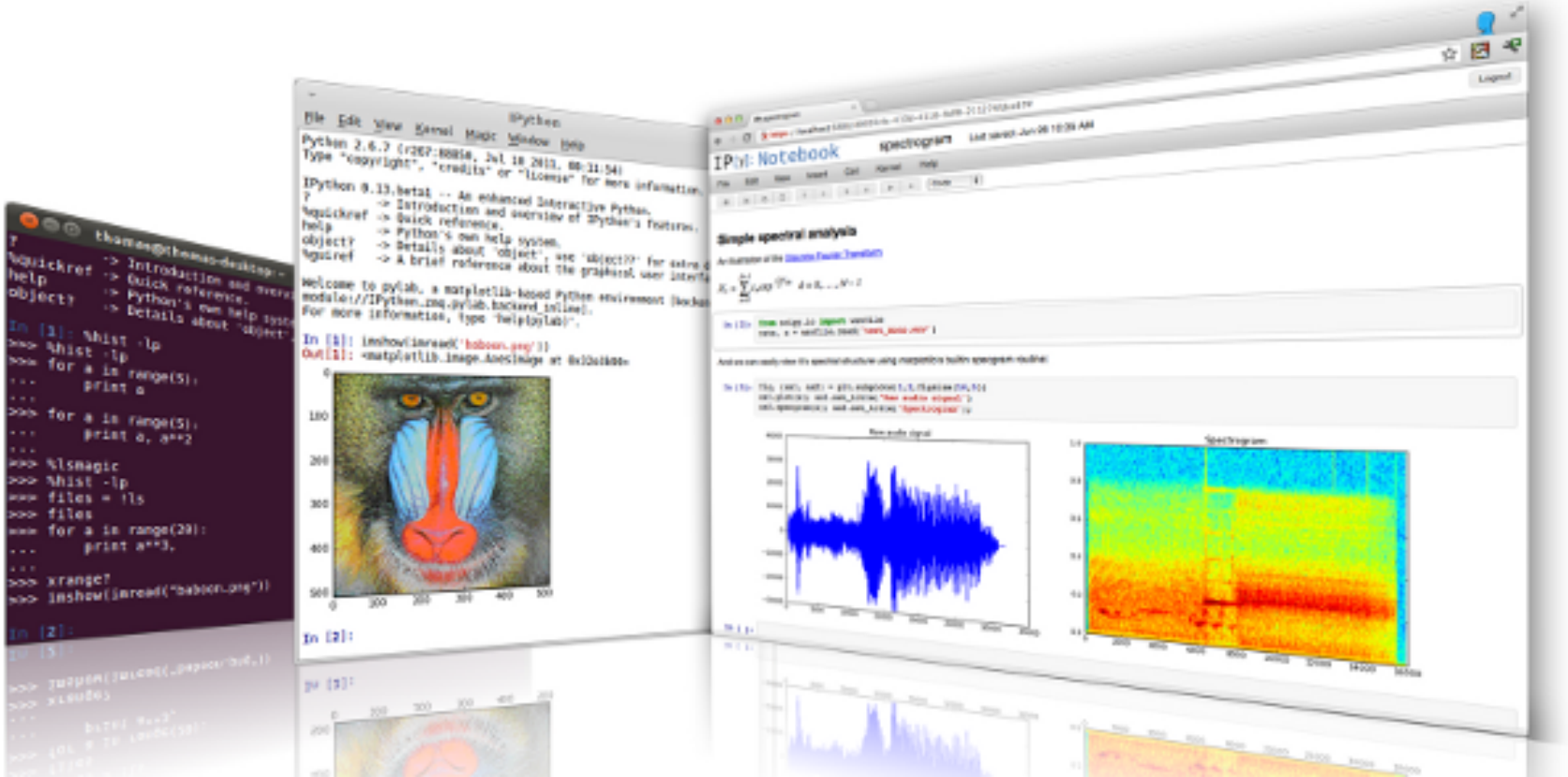
Fonctionnalités IDLE

- colorisation
- complétion
- documentation
- *stack viewer*

IPython

http ://ipython.org/

- Console d'interpréteur amélioré
- *QT Console* : visualisation graphique
- *Notebook* : exerciseur interactif en mode Web



Interpréteur IPython

```
$ ipython
```

olivier@inf-8657: /home/olivier

Fichier Édition Affichage Rechercher Terminal Aide

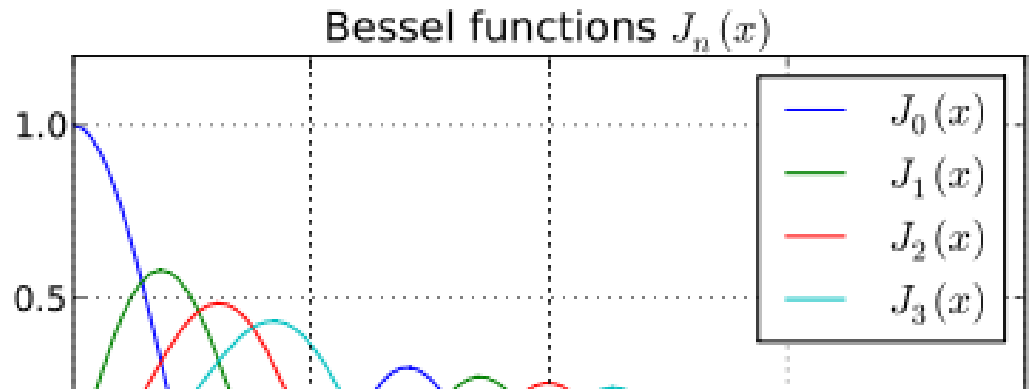
```
olivier@inf-8657:~$ ipython
Python 2.7.3 (default, Jan  2 2013, 16:53:07)
Type "copyright", "credits" or "license" for more information.

IPython 0.13.1 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: print a
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-9d7b17ad5387> in <module>()
----> 1 print a

NameError: name 'a' is not defined

In [2]:
```

Console QT IPython

```
$ ipython qtconsole
```

IPython

```
In [2]: from scipy import special as sp
   ...: x = linspace(0, 20, 100)
   ...: for n in range(4):
   ...:     y = sp.jn(n, x)
   ...:     plot(x, y, label=r'$J_%s(x)$' % n)
   ...: axhline(0, color='green', label='_nolegend_')
   ...: grid()
   ...: legend()
   ...: xlabel('$x$')
   ...: title(r'Bessel functions $J_n(x)$')
Out[2]: <matplotlib.text.Text object at 0x7fcddc1795d0>
```



Notebook IPython

```
$ ipython notebook
[NotebookApp] Using existing profile dir: u'/home/olivier/.config/ipython/profi
[NotebookApp] Serving notebooks from /home/olivier
[NotebookApp] The IPython Notebook is running at: http://127.0.0.1:8888/
[NotebookApp] Use Control-C to stop this server and shut down all kernels.
```

Puis ouverture de `http://127.0.0.1:8888/` dans le navigateur.

Commentaires

Ligne de commentaire :

```
# Ceci est un commentaire
```

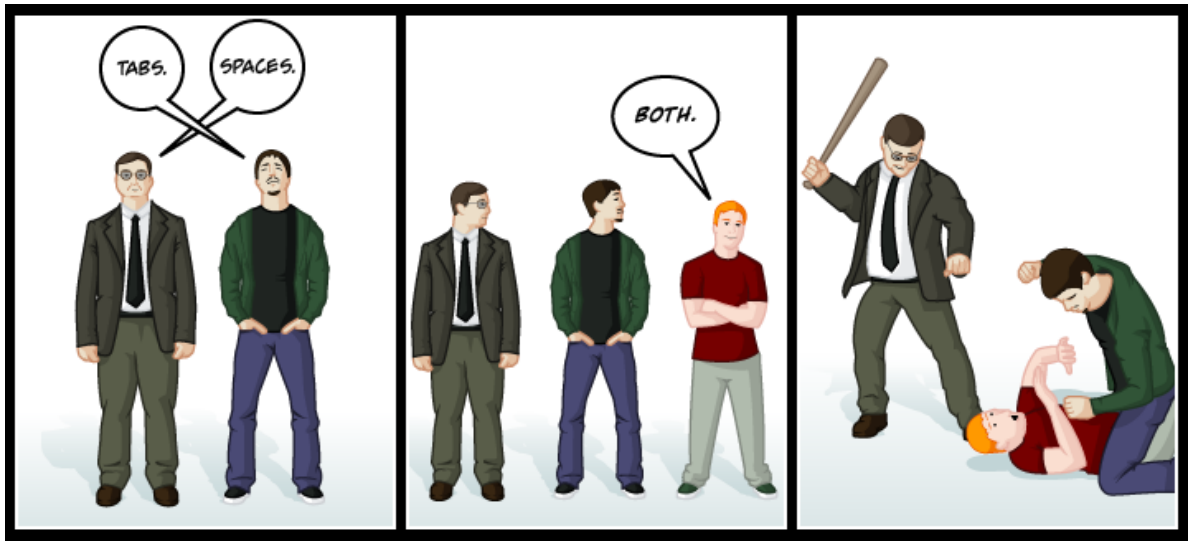
En fin de ligne :

```
a = 2 # Ceci est un commentaire
```

Tabulations

Importance des tabulations => utiliser un éditeur « intelligent »

- appui sur [TAB] (génère des espaces si éditeur bien configuré)
- recommandé : *4 espaces*



source : emacswiki, inspiré d'un dessin de Steve Napierski

Première ligne
Shebang (#!)

Détermine quel interpréteur Python doit être utilisé.

```
#!/usr/bin/python

print 'coucou'
```

ou

```
#!/usr/bin/env python

print 'coucou'
```

Cf. http://fr.wikipedia.org/wiki/Shebang

Encodage des caractères

L'encodage des caractères (accents, etc.) utilisé pour **écrire** un programme Python est important : l'interpréteur construit les chaînes de caractères avec.
UTF-8, marche bien sur Linux.

helloworld.py

```
1  #!/usr/bin/python
2  # -*- coding: utf-8 -*-
3
4  # Ceci est un exemple d'affichage d'une chaîne accentuée
5
6  name = raw_input("Quel est vôtre nom ? ")
7
8  print "J'espère que ça va bien aujourd'hui", name
```

Importance de la 2ème ligne : `-*- coding: utf-8 -*-`
Sur Windows, c'est différent : cp1252 au lieu d'UTF-8, dans XP, par exemple... désolé par avance pour les codes sources accentués.
Cf. Chaînes de caractères Unicode

Apprendre la syntaxe Python

- Survol rapide uniquement pour que les exemples suivants soient compréhensibles
- **Lisez le tutoriel officiel** (ou un bon bouquin)

Identifiants

- Mots clés du langage : `if`, `for`, `while`, `import`, `class`, etc.
- Noms des objets de bibliothèques : déconseillé (`sys`, `string`)
- Tout le reste : identifiants d' « objets » (ASCII uniquement) : variables, fonctions, classes, modules, etc.

Affectations

```
>>> a = 2
>>> a
2
>>> b = a + 3
>>> b
5
```

Pas de déclaration de variables. Typage dynamique.

```
>>> b = 'bonjour'
>>> b
'bonjour'
```

Types de base

- booléens (`True`, `False`)
- nombres
- chaînes de caractères (+ chaînes unicode)

- listes
- ~~tableaux~~
- dictionnaires (tableaux associatifs)

Nombres

- entiers

```
>>> (50-5*6)/4
5
```

- réels

```
>>> 7.0 / 2
3.5
```

- opérateurs arithmétiques de base : +, -, /, *, **, etc
- conversions

```
>>> a = 3
>>> a
3
>>> d = float(a)
>>> d
3.0
```

- complexes, fonctions de base

Chaînes de caractères

Chaînes simples

```
>>> 'spam oeufs'
'spam oeufs'
```

Affichage des chaînes

```
>>> "bonjour"
'bonjour'
>>> print "bonjour"
bonjour
>>> print 'coucou', 'tout', 'le', 'monde'
coucou tout le monde
```

Chaînes longues

Multi-lignes

```
>>> hello = "Ceci est une chaîne assez longue qui contient\n\
... plusieurs lignes de texte comme vous le feriez en C.\n\
...     Notez que les espaces en début de ligne sont\
...  significatifs."
>>> print hello
Ceci est une chaîne assez longue qui contient
plusieurs lignes de texte comme vous le feriez en C.
    Notez que les espaces en début de ligne sont significatifs.
```

Mieux :

```
>>> hello = """Ceci est une chaîne assez longue qui contient
... plusieurs lignes de texte comme vous le ferez en Python."""
>>> print hello
Ceci est une chaîne assez longue qui contient
plusieurs lignes de texte comme vous le ferez en Python.
```

Accentuation
Chaînes de caractères Unicode

unicodestrings.py

```
import sys
print sys.stdin.encoding ## cp1252 sous Windows, UTF-8 sous Linux
machaine = 'äâéç'
print machaine
chunicode = u'äâéç'
print chunicode.encode(sys.stdin.encoding)
```

Sur Linux :

UTF-8
äâéç
äâéç

Sur Windows :

cp1252
aÃ§Ã©Ã
äâéç

Opérations sur les chaînes

- formatage : %

```
>>> a = "mais"
>>> b = "Non"
>>> c = "allo"
>>> d = "quoi"
>>> message = "%s %s %s, %s !" % (b, a, c, d)
>>> message
'Non mais allo, quoi !'
```

- découpage

```
>>> e = message[4:13]
>>> e
'mais allo'
```

```
 +---+---+---+---+---+
 | N | o | n |   | m |
 +---+---+---+---+---+
 0   1   2   3   4   5
-5  -4  -3  -2  -1  -0
```

Listes

Découpage :

```
>>> a = ['spam', 'oeufs', 100, 1234]
>>> a[1:]
['oeufs', 100, 1234]
```

Contenu modifiable :

```
>>> a[2] = a[2] + 23
>>> a
['spam', 'oeufs', 123, 1234]
```

Générateur de liste d'entiers : `range()`

```
>>> range(2, 10)
[2, 3, 4, 5, 6, 7, 8, 9]
```

Contrôles de flux

Instructions `if`, `for`, `while`, `break`, `continue`

Exemple : recherche de nombres premiers

```
>>> for n in range(2, 10):
...     for x in range(2, n):
...         if n % x == 0:
...             print n, 'egal a', x, '*', n/x
...             break
...     else:
...         # la boucle est terminee sans avoir trouve de facteur
...         print n, 'est un nombre premier'
...
2 est un nombre premier
3 est un nombre premier
4 egal a 2 * 2
5 est un nombre premier
6 egal a 2 * 3
7 est un nombre premier
8 egal a 2 * 4
9 egal a 3 * 3
```

Fonctions

Exemple suite de fibonacci :

```
>>> def fib(n):    # Ecrire la suite de fibonacci jusqu'a n
...     """Affiche la suite de Fibonacci jusqu'a n."""
...     a, b = 0, 1
...     while a < n:
...         print a,
...         t = a+b
...         a = b
...         b = t
...         # ou bien : a, b = b, a+b
...
>>> # Maintenant, appelons la fonction qu'on vient de definir :
... fib(2000)
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597
```

Paramètres

Les paramètres peuvent être nommés.

```
>>> def perroquet(voltage, etat='raide', action='wouff'):
...     print "-- Ce perroquet ne va pas faire", action
...     print "si vous le branchez sur du", voltage, "volts."
...     print "Il a l'air", etat, "!"
...
>>> perroquet(voltage = "quatre millions", etat = "sacrement fichu",
...           action = "SCHLACK")
-- Ce perroquet ne va pas faire SCHLACK
si vous le branchez sur du quatre millions volts.
Il a l'air sacrement fichu !
```

Retour de valeurs

Retour d'une liste des valeurs de la suite de Fibonacci

```
>>> def fib2(n): # renvoie la suite de Fibonacci jusqu'a n
...     resultat = []
...     a, b = 0, 1
...     while b < n:
...         resultat.append(b)
...         a, b = b, a+b
...     return resultat
...
>>> # Maintenant, appelons la fonction qu'on vient de definir :
... a = fib2(2000)
>>> print a
[1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597]
```

Chaînes de documentation

Attribut __doc__

```
>>> def ma_fonction():
...     """Ne rien faire, mais le documenter.
...
...     Non, vraiment, ne rien faire.
...     """
...     pass
...
>>> print ma_fonction.__doc__
Ne rien faire, mais le documenter.

    Non, vraiment, ne rien faire.
```

Documentation en ligne

```
>>> help(ma_fonction)

Help on function ma_fonction in module __main__:

ma_fonction()
    Ne rien faire, mais le documenter.
    Non, vraiment, ne rien faire.
```

Listes

```
>>> a = [66.25, 333, 333, 1, 1234.5]
>>> print a.count(333), a.count(66.25), a.count('x')
2 1 0
>>> a.insert(2, -1)
>>> a.append(333)
>>> a
[66.25, 333, -1, 333, 1, 1234.5, 333]
>>> a.index(333)
1
>>> a.remove(333)
>>> a
[66.25, -1, 333, 1, 1234.5, 333]
>>> a.reverse()
>>> a
[333, 1234.5, 1, 333, -1, 66.25]
>>> a.sort()
>>> a
[-1, 1, 66.25, 333, 333, 1234.5]
>>> del a[3]
>>> a
[-1, 1, 66.25, 333, 1234.5]
```

Tuples, séquences

```
>>> t = 12345, 54321, 'hello!'
>>> t[0]
12345
>>> t
(12345, 54321, 'hello!')
>>> # Les tuples ne sont pas modifiables :
... t[0] = 88888
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
>>> # Comme dans : a, b = 1, 2
>>> t = 1, 2
>>> a, b = t
>>> a
1
>>> b
2
```

Dictionnaires

aka *Tableaux associatifs* (clés + valeurs)

```
>>> tel = {'christian': 4098, 'bob': 4139}
>>> tel['olivier'] = 4127
>>> tel
{'bob': 4139, 'olivier': 4127, 'christian': 4098}
>>> tel['christian']
4098
>>> del tel['bob']
>>> tel['christophe'] = 4127
>>> tel
{'olivier': 4127, 'christophe': 4127, 'christian': 4098}
>>> tel.keys()
['olivier', 'christophe', 'christian']
>>> 'olivier' in tel
True
>>> tel['python'] = ['spam', 3.14]
>>> tel
{'python': ['spam', 3.14], 'christian': 4098, 'christophe': 4127, 'olivier': 41
```

Types

- type(x)

```
>>> type(3.14)
<type 'float'>
>>> type('3.14')
<type 'str'>
>>> type(print)
  File "<stdin>", line 1
    type(print)
              ^
SyntaxError: invalid syntax
```

- isinstance()

```
>>> isinstance('3.14', float)
False
>>> isinstance('3.14', str)
True
```

Égalité

- égalité : ==
- identité : `is`

```
>>> '3.14' == '3.14'
True
>>> '3.14' is '3.14'
True
>>> a = ['3.14']
>>> b = ['3.14']
>>> a == b
True
>>> a is b
False
```

Affectation – références

Effet de bord dans une fonction – paye.py

```
def augmenter_mini(salaires, seuilmin, augment):
    """Augmente les salaires inférieurs à un seuil minimal"""

    for i in range(len(salaires)):
        if salaires[i] <= seuilmin:
            salaires[i] += augment

salaires_init = [1000, 1500, 700, 2000, 800, 1010]
salaires_nouv = salaires_init
augmenter_mini(salaires_nouv, 1000, 100)
print salaires_init
print salaires_nouv
```

Affectation – références

Effet de bord dans une fonction – paye.py

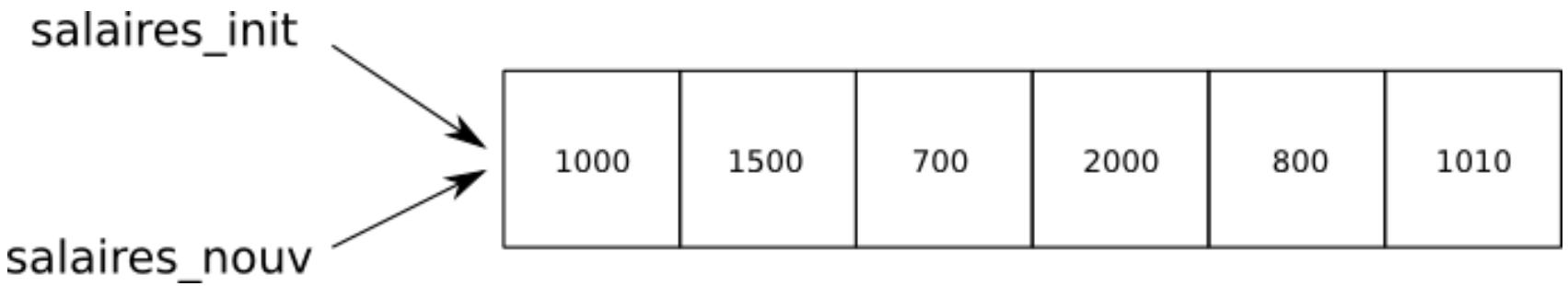
```
def augmenter_mini(salaires, seuilmin, augment):
    """Augmente les salaires inférieurs à un seuil minimal"""

    for i in range(len(salaires)):
        if salaires[i] <= seuilmin:
            salaires[i] += augment

salaires_init = [1000, 1500, 700, 2000, 800, 1010]
salaires_nouv = salaires_init
augmenter_mini(salaires_nouv, 1000, 100)
print salaires_init
print salaires_nouv
```

Résultat

```
[1100, 1500, 800, 2000, 900, 1010]
[1100, 1500, 800, 2000, 900, 1010]
```

Explications

L'affectation associe juste un nom à un objet existant.



Solutions :

- Ne pas modifier l'objet directement dans la fonction, mais renvoyer une copie modifiée
- Utiliser copy plutôt qu'une affectation

Echange de valeurs dans une fonction

echange.py

```
def echange(a, b) :
    aux = a
    a = b
    b = aux

def swap(valeurs) :
    aux = valeurs[0]
    valeurs[0] = valeurs[1]
    valeurs[1] = aux

# Début du programme
x = 5
y = 7

print  "Avant échange x=%d et y=%d" %(x, y)
echange(x , y)
print  "Après échange x=%d et y=%d" %(x, y)

values = [x, y]
print  "Avant swap x=%d et y=%d" %(values[0], values[1])
swap(values)
print  "Après swap x=%d et y=%d" %(values[0], values[1])

# Fin du programme
```

Utiliser le module copy

copy.copy()

```
from copy import copy

a = [1, 2, [30, 40], 5, 6]
b = a
a_copy = copy(a)

a[3] = 50

print a
print b
print a_copy
```

Utiliser le module copy

copy.copy()

```
from copy import copy

a = [1, 2, [30, 40], 5, 6]
b = a
a_copy = copy(a)

a[3] = 50

print a
print b
print a_copy
```

Résultat

```
[1, 2, [30, 40], 50, 6]
[1, 2, [30, 40], 50, 6]
[1, 2, [30, 40], 5, 6]
```

Utiliser le module copy

copy.copy()

```
from copy import copy

a = [1, 2, [30, 40], 5, 6]
b = a
a_copy = copy(a)

a[3] = 50

print a
print b
print a_copy
```

Résultat

```
[1, 2, [30, 40], 50, 6]
[1, 2, [30, 40], 50, 6]
[1, 2, [30, 40], 5, 6]
```

Modifie la sous-liste

```
c = a[2]

c[1] = 42
```

Utiliser le module copy

copy.copy()

```
from copy import copy

a = [1, 2, [30, 40], 5, 6]
b = a
a_copy = copy(a)

a[3] = 50

print a
print b
print a_copy
```

Modifie la sous-liste

```
c = a[2]

c[1] = 42
```

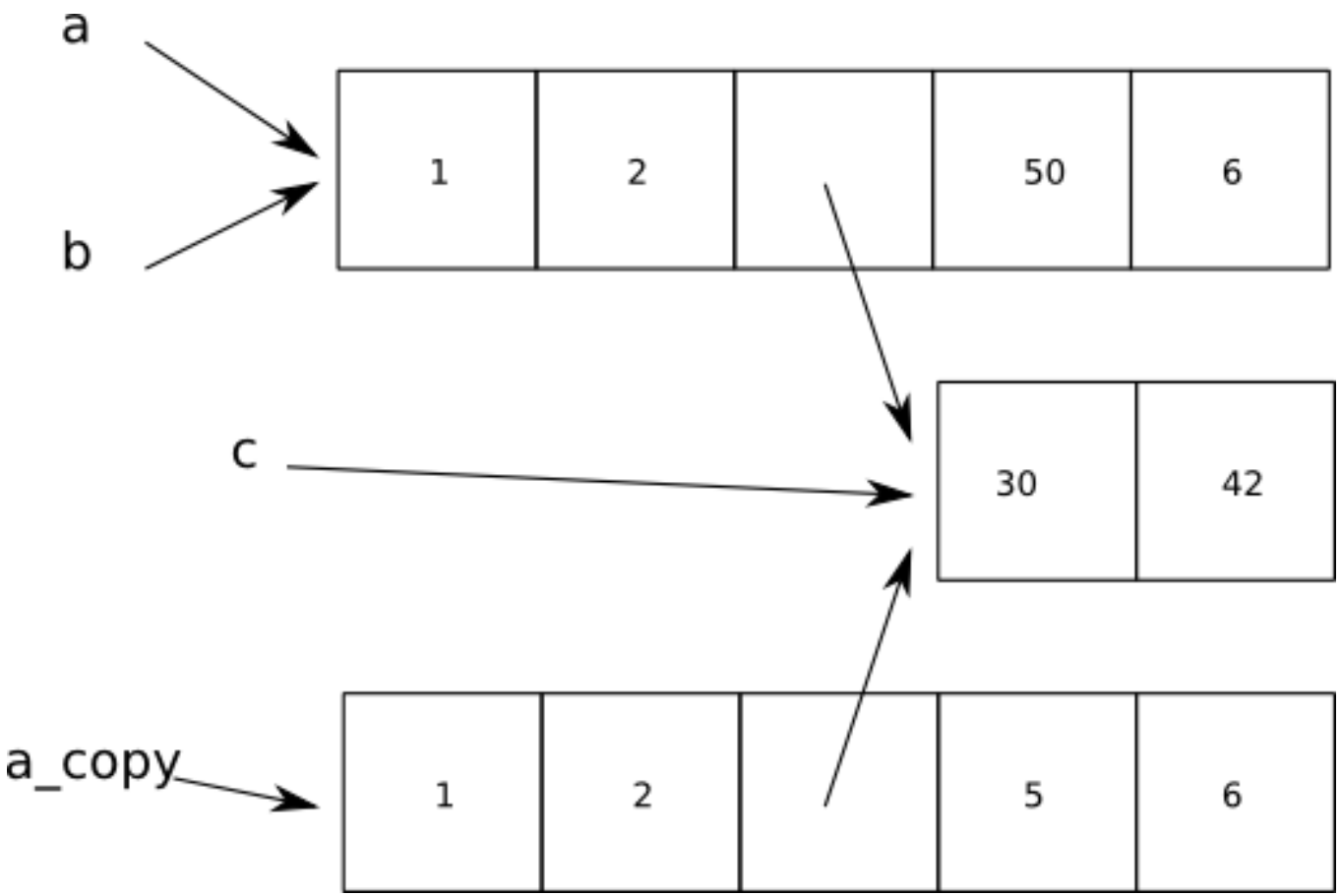
Résultat

```
[1, 2, [30, 40], 50, 6]
[1, 2, [30, 40], 50, 6]
[1, 2, [30, 40], 5, 6]
```

Bug ?

```
[1, 2, [30, 42], 50, 6]
[1, 2, [30, 42], 50, 6]
[1, 2, [30, 42], 5, 6]
```

Explications



Deep Copy

Copie en profondeur

```
from copy import deepcopy

a = [1, 2, [30, 40], 5, 6]
b = a
a_copy = deepcopy(a)

a[3] = 50

c = a[2]
c[1] = 42

print a
print b
print a_copy
```

Résultats

```
[1, 2, [30, 42], 50, 6]
[1, 2, [30, 42], 50, 6]
[1, 2, [30, 40], 5, 6]
```

Portée - Espaces de nommage

Variables locales dans les fonctions, et recherche dynamique en *remontant* vers la portée englobante

```
def bonjour(nom):
    print "Bonjour", prenom, nom

prenom = "Olivier"
nom = "Berger"

bonjour("Christian")
```

```
Bonjour Olivier Christian
```

Modules

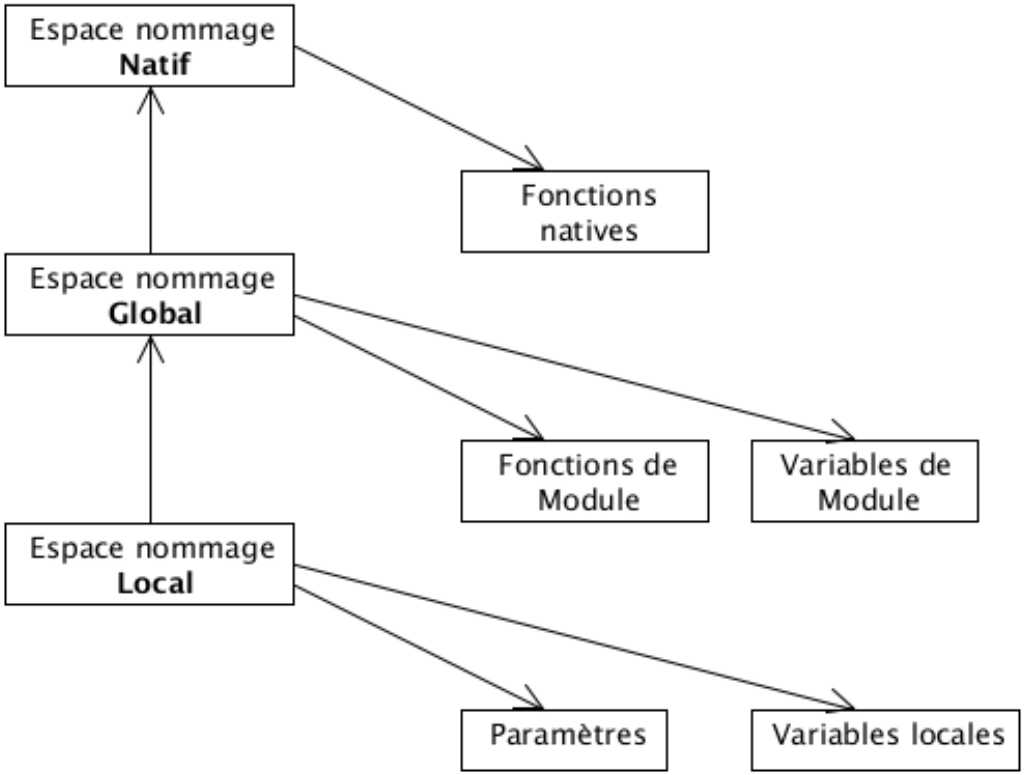
fibonacci.py

```
def fib(n):    # Ecrire la suite de fibonacci jusqu'à n
    a, b = 0, 1
    while b < n:
        print b,
        a, b = b, a+b

def fib2(n): # renvoie la suite de Fibonacci jusqu'à n
    resultat = []
    a, b = 0, 1
    while b < n:
        resultat.append(b)
        a, b = b, a+b
    return resultat
```

```
>>> import fibonacci
>>> fibonacci.fib(1000)
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987
>>> fibonacci.fib2(100)
[1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
>>> fibonacci.__name__
'fibonacci'
```

Principe de dé-référencement des identifiants

Parcours d'un arbre d'espace de noms : de gauche à droite et du bas vers le haut :



Imports

- **import**

```
>>> import fibonacci
>>> fibonacci.fib(1000)
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987
>>> type(fibonacci)
<type 'module'>
>>> dir(fibonacci)
['__builtins__', '__doc__', '__file__', '__name__', '__package__', 'fib',
```

- **from ... import**

```
>>> from fibonacci import fib
>>> fib(1000)
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987
>>> print fib.__module__
fibonacci
>>> print bonjour.__module__
__main__
```

- **import ... as**

```
>>> import numpy as N
>>> N.arange(3)
array([0, 1, 2])
```

Lecture / Ecriture

- ouverture : `open(filename, mode)`
- lecture : `read()`
- fermeture : `close()`

```
>>> f = open('/etc/passwd')
>>> for line in f:
...         print line,
...
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
```

Erreurs

```
>>> while True print 'Hello world'
  File "<stdin>", line 1
    while True print 'Hello world'
                    ^
SyntaxError: invalid syntax
```

Exceptions

```
>>> 10 * (1/0)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZeroDivisionError: integer division or modulo by zero
>>> 4 + spam*3
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'spam' is not defined
>>> '2' + 2
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: cannot concatenate 'str' and 'int' objects
```

Interception des exceptions

```
>>> while True:
...     try:
...         x = int(raw_input("Merci de saisir un nombre : "))
...         break
...     except ValueError:
...         print "Argh !  Ce n'est pas un nombre valide.  Essaye encore..."
...
```

Programmation orientée objets

```
>>> class Complexe:
...     def __init__(self, partiereelle, partieimag):
...         self.r = partiereelle
...         self.i = partieimag
...
>>> x = Complexe(3.0, -4.5)
>>> type(x)
<type 'instance'>
>>> x.r, x.i
(3.0, -4.5)
>>> l = [3.14, 'spam', Complexe(1, -1)]
>>> l
[3.14, 'spam', <__main__.Complexe instance at 0xb74438ec>]
```

Bibliothèque standard

Demain. . .

Démarche d'écriture de programmes

1. Du problème à l'algorithme
2. De l'algorithme au programme

Exercice année bissextile
Première version simple

Énoncé du problème :

Ecrire un programme qui permet de dire si une année est bissextile.

Exercice année bissextile
Première version simple

Énoncé du problème :

Ecrire un programme qui permet de dire si une année est bissextile.

Analyse des données du problème :

- Données du problème :
  - La valeur de `année` de type Entier
- Données en sortie :
  - Booléen indiquant si `Oui` ou `Non` l'année est bissextile

Algorithme

bissextile.txt

```
fonction principale()
    année: Entier
    afficher("Saisir une année: ")
    saisir(année)
    si année mod 4 != 0 alors
        afficher("L'année ", année, "n'est pas bissextile")
    sinon si année mod 100 != 0 alors
            afficher("L'année ", année, "est bissextile")
        sinon si année mod 400 != 0 alors
                afficher("L'année ", année, "n'est pas bissextile")
            sinon
                afficher("L'année ", année, "est bissextile")
            fsi
        fsi
    fsi
ffct principale
```

Programme

bissextile.py

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

# Fichier bissextile.py - Calcul d'année bissextile

annee = input("Entrez une année : ")

if (annee % 4) != 0 :
    print "L'année %d n'est pas bissextile" % annee
elif (annee % 100) != 0 :
    print "L'année %d est bissextile" % annee
elif (annee % 400) != 0 :
    print "L'année %d n'est pas bissextile" % annee
else:
    print "L'année %d est bissextile" % annee
```

Exercice année bissextile

Deuxième version

bissextile2.py

```
def estBissextile(a) :
    if (a % 4) != 0 :
        return False
    elif (a % 100) != 0 :
        return True
    elif (a % 400) != 0 :
        return False
    else:
        return True

# Programme principal
annee = raw_input("Entrez une année : ")
annee = int(annee)
if (annee < 1582) :
    print 'Erreur de saisie !'
else :
    print "L'année %d" % annee,
    if estBissextile(annee) :
        print "est",
    else :
        print "n'est pas",
    print "bissextile."
```

Structures de contrôle

Calcul de factorielle

Énoncé du problème :

Pour tout entier n >= 0, calculer la valeur de n !

Structures de contrôle

Calcul de factorielle

Énoncé du problème :

Pour tout entier n >= 0, calculer la valeur de n !

Analyse des données du problème :

- Données du problème :
  - Un nombre `n` de type Entier
- Données en sortie :
  - La valeur de n !

Algorithme

factorielle.txt

```
fonction factorielle(n:Entier) : Entier
  i = 1, f = 1 : Entier
  tant que i <= n faire
        i <- i + 1
        f <- f x i
  ftq
  retourner f
ffct factorielle

fonction principale()
   nb : Entier
   afficher("Entrez un nombre >= 0 :")
   saisir(nb)
   afficher(nb, factorielle(nb))
ffct principale
```

Programme

factn.py

```
# Fonction
def factorielle(n) :
    f = 1
    for i in range(2, n+1) :
        f *= i
    return f

def factorielle2(n) :
    f = 1
    i = 1
    while i <= n :
        f *= i
        i += 1
    return f

# Programme principal
nb = input("Entrez un nombre >= 0 :")
print "%d! = %d" % (nb, factorielle(nb))
print "%d! = %d" % (nb, factorielle2(nb))
```

Recherche du zéro d'une fonction par dichotomie

Énoncé du problème :

Soit **f** une fonction continue monotone sur l'intervalle **[a, b]** où elle ne s'annule qu'une seule et unique fois. Pour trouver ce zéro, on procède par dichotomie, c'est-à-dire que l'on divise l'intervalle de recherche par deux à chaque étape.
Soit **m** le milieu de **[a, b]**. Si **f(m)** et **f(a)** sont de même signe, le zéro recherché est dans l'intervalle **[m, b]**, sinon il est dans l'intervalle **[a, m]**.
Définir une fonction **zerofonction**, qui calcule le zéro de la fonction **f** donnée sur l'intervalle **[a, b]**, avec une précision **epsilon**. La fonction **f**, de prototype **f(x :Reel) : Reel** est supposée définie par ailleurs et les réels **a**, **b** et **epsilon** sont des données saisies par la fonction principale, à définir également.

Recherche du zéro d'une fonction par dichotomie

Énoncé du problème :

Soit **f** une fonction continue monotone sur l'intervalle **[a, b]** où elle ne s'annule qu'une seule et unique fois. Pour trouver ce zéro, on procède par dichotomie, c'est-à-dire que l'on divise l'intervalle de recherche par deux à chaque étape.
Soit **m** le milieu de **[a, b]**. Si **f(m)** et **f(a)** sont de même signe, le zéro recherché est dans l'intervalle **[m, b]**, sinon il est dans l'intervalle **[a, m]**.
Définir une fonction **zerofonction**, qui calcule le zéro de la fonction **f** donnée sur l'intervalle **[a, b]**, avec une précision **epsilon**. La fonction **f**, de prototype **f(x :Reel) : Reel** est supposée définie par ailleurs et les réels **a**, **b** et **epsilon** sont des données saisies par la fonction principale, à définir également.

Analyse des données du problème :

- Données du problème :
  - `a`, `b` et `epsilon` : Reel et `f` : Reel -> Reel
- Données en sortie :
  - La valeur pour laquelle `f` vaut 0 (à `epsilon` près)

Algorithme (zerofonction.txt)

```
fonction zerofonction(a, b, epsilon: Réel): Réel
    m = (a + b) / 2 : Réel // Définition et initialisation de m
    tant que ( (b-a) > epsilon ) ET ( ABS(f(m)) > epsilon ) faire
        si f(a) x f(m) > 0 alors
            a <- m
        sinon
            b <- m
        fsi
        m <- (a + b) / 2
    ftq
    retourner m
ffct zerofonction

fonction principale()
    début, fin, précision, résultat : Réel
    afficher("Saisir le début et la fin de l'intervalle")
    saisir (début, fin)
    résultat <- zerofonction(début, fin, précision)
    afficher("Le zéro (à", précision, "près) vaut ", résultat)
ffct principale
```

Programme

dicho.py

```
def f(x) :
     return x**2 - 1

def zero(a, b, epsilon) :
    milieu = (b + a)/2
    val = f(milieu)
    while ((b - a) > epsilon) and (abs(val) > epsilon) :
        if ( (f(a) * val) > 0) :
            a = milieu
        else :
            b = milieu
        milieu = (a + b)/2
        val = f(milieu)
    return milieu

# Début du programme
# Calcul du zéro de la fonction x2 - 1 entre 0. et 3. donc 1.
a = 0.
b = 3.
e = 0.001
z = zero(a, b, e)
print  "Le zéro (à %s près) vaut %s " % (e, z)
#Fin du programme
```

Intégration par la méthode des trapèzes

Énoncé :

Soit **f** une fonction continue sur l'intervalle **[a, b]**. L'intégration consiste à découper cet intervalle, en n sous-intervalles de longueur **delta**. L'intégrale d'un sous-intervalle **[x, x+delta]** est approximée au trapèze de base **delta** et de cotés **f(x)** et **f(x+delta)**. Définir une fonction, **intégrale**, qui retourne la valeur de l'intégrale d'une fonction **f** réelle continue sur l'intervalle **[a, b]** . La fonction **f**, de prototype **f(x :Reel) : Reel** est supposée définie par ailleurs et les réels **a**, **b** ainsi que l'entier **n** sont des données saisies par la fonction principale à définir également.

Intégration par la méthode des trapèzes

Énoncé :

Soit **f** une fonction continue sur l'intervalle **[a, b]**. L'intégration consiste à découper cet intervalle, en n sous-intervalles de longueur **delta**. L'intégrale d'un sous-intervalle **[x, x+delta]** est approximée au trapèze de base **delta** et de cotés **f(x)** et **f(x+delta)**. Définir une fonction, **intégrale**, qui retourne la valeur de l'intégrale d'une fonction **f** réelle continue sur l'intervalle **[a, b]** . La fonction **f**, de prototype **f(x :Reel) : Reel** est supposée définie par ailleurs et les réels **a**, **b** ainsi que l'entier **n** sont des données saisies par la fonction principale à définir également.

Analyse des données du problème :

- Données du problème :
  - `a`, `b` : Reel, `n` : Entier et `f` : Reel -> Reel
- Données en sortie :
  - La valeur de l'intégrale de `f` sur `[a, b]`

Algorithme (integrale.txt)

```
fonction integrale(a, b:Réel, n: Entier): Réel
   delta = (b - a) / n : Réel   // Définition et initialisation
   somme = 0 : Réel             //
   x1 = a  : Réel               //
   x2 : Réel
   pour i = 1 : Entier à n
       x2 <- x1 + delta
       somme <- somme + (f(x1) + f(x2)) / 2
       x1 <- x2
   fpour i
   somme <- somme x delta
   retourner somme
ffct integrale
fonction principale()
    début=1., fin=3.14 : Réel
    aire : Réel
    nbintervalles : Entier
    afficher("Nombre d'intervalles: ")
    saisir(nbintervalles)
    aire <- integrale(début, fin, nbintervalles)
    afficher("L'integrale est égale à ",aire)
ffct principale
```

Programme

trapeze.py

```
def f(x) :
    return x**2 - 1

def integrale(a, b, n) :
    delta = (b - a)/n
    somme = 0.
    x1 = a
    i = 1
    while (i <= n) :
        x2 = x1 + delta
        somme += (f(x1) + f(x2))/2
        x1 = x2
        i += 1
    return somme * delta

# Début du programme
a = 1.
b = 3.14
nbinter = input("Nombre d'intervalles: ")
aire = integrale(a, b, nbinter)
print  "L'intégrale est égale à : %f" % aire
```

Gestion des fichiers

fichiers.py

```
f = open('monfichier', 'w') # Ouverture en écriture
f.write("Bonjour à tous") # Ecriture dans le fichier
f.close() # Fermeture du fichier

f = open('monfichier') # Ouverture en lecture par défaut
chaine = f.read(3) # Lecture de 3 caractères
suite = f.read() # Lecture du reste du fichier
print chaine + suite
f.close() # Fermeture du fichier
```

```
Bonjour à tous
```

Manipulation de quelques structures de données

Ecrire un gestionnaire de carnet d'adresses.

Tri à bulles

Énoncé

Définir une fonction **tribul** qui effectue le tri par ordre croissant d'un tableau `tab` d'entiers donné.
Le principe du tri à bulles consiste à comparer deux éléments consécutifs et à les échanger s'ils ne sont pas dans le bon ordre. A l'étape **i**, on parcourt le tableau à partir de la fin de manière à faire remonter le plus petit élément en **t[i]**.
On pourra améliorer les performances de la méthode en gérant une variable booléenne qui va détecter s'il y a eu au moins un échange à une itération donnée.

Tri à bulles

Énoncé

Définir une fonction **tribul** qui effectue le tri par ordre croissant d'un tableau `tab` d'entiers donné.
Le principe du tri à bulles consiste à comparer deux éléments consécutifs et à les échanger s'ils ne sont pas dans le bon ordre. A l'étape **i**, on parcourt le tableau à partir de la fin de manière à faire remonter le plus petit élément en **t[i]**.
On pourra améliorer les performances de la méthode en gérant une variable booléenne qui va détecter s'il y a eu au moins un échange à une itération donnée.

Analyse des données du problème :

- Données du problème :
  - Le tableau `tab[1..N]` : Entier
- Données en sortie :
  - Le même tableau `tab[1..N]` mais trié

Algorithme (tribul.txt)

```
constante N = 10
fonction tribul1(tab: tableau[1..N] de Entier)
  pour i = 1: Entier à N-1
     pour j = N: Entier à i+1 pas -1
        si tab[j] < tab[j-1] alors
           echanger(tab[j], tab[j-1])
        fsi
     fpour j
  fpour i
ffct tribul1
fonction tribul2(tab: tableau[1..N] de Entier)
  i = 1: Entier
  pasdéchange = FAUX: Booléen
  tant que i <= N-1 ET NON pasdéchange faire
       pasdéchange <- VRAI
       pour j = N: Entier à i+1 pas -1
            si tab[j] < tab[j-1] alors
               echanger(tab[j], tab[j-1])
               pasdéchange <- FAUX
            fsi
       fpour j
       i <- i+1
  ftq
ffct tribul2
```

Programme

tribul.py

```
import random

def tribul(tab) :
    taille = len(tab)
    for i in range(taille - 1):
        for j in range(taille-1, i, -1):
            if (tab[j] < tab[j-1]) :
                tab[j], tab[j-1] = tab[j-1], tab[j]
            #print i, j, tab # decommenter pour debug
        #print i, tab # decommenter pour debug

# Début du programme
taille = 20  # Longueur du tableau
# Initialisation du tableau avec des valeurs aléatoires
tab = random.sample(range(101), taille)
print "Affichage du tableau tab non trié"
print tab
# Appel de la fonction tri
tribul(tab)
print "\nAffichage du tableau tab trié"
print tab
```

Tri par sélection

Énoncé :

Définir une fonction **trisel** qui effectue le tri par ordre croissant d'un tableau `tab` d'Entiers donné.
Le principe du tri par sélection consiste à échanger le premier et le plus petit élément de la partie non triée du tableau.

Tri par sélection

Énoncé :

Définir une fonction **trisel** qui effectue le tri par ordre croissant d'un tableau `tab` d'Entiers donné.
Le principe du tri par sélection consiste à échanger le premier et le plus petit élément de la partie non triée du tableau.

Analyse des données du problème :

- Données du problème :
  - Le tableau `tab[1..N]` : Entier
- Données en sortie :
  - Le même tableau `tab[1..N]` mais trié

Algorithme (trisel.txt)

```
constante N = 10
fonction trisel(tab: tableau[1..N] de Entier)
   positionmin: Entier
   pour i = 1: Entier à N-1
        positionmin <- positionduminimum(tab, i)
        échanger(tab[i], tab[positionmin])
   fpour i
ffct trisel
fonction positionduminimum(tab: tableau[1..N] de Entier,
                           borneinf: Entier): Entier
   indmin = borneinf: Entier
   min: Entier
   min <- tab[indmin]
   pour i = indmin+1: Entier à N
        si tab[i] < min alors
           indmin <- i
           min <- tab[i]
        fsi
   fpour i
   retourner indmin
ffct positionduminimum
```

Programme

trisel.py

```
import random

def posmin(tablo, borneinf):
    size = len(tablo)
    indmin = borneinf
    lemin = tablo[indmin]
    for i in range(indmin+1, size):
        if (tablo[i] < lemin) :
              indmin = i
              lemin = tablo[i]
    return indmin

def triparsel(tablo) :
    size = len(tablo)
    for i in range(size-1):
        pos = posmin(tablo, i)
        tablo[i], tablo[pos] = tablo[pos], tablo[i]
        # print i, pos, tablo # decommenter pour debug

taille = 20  # Longueur du tableau
# Initialisation du tableau avec des valeurs aléatoires
tab = random.sample(range(101), taille)
print  "Affichage du tableau non trié"
print tab
# Appel du tri
triparsel(tab)
print  "\nAffichage du tableau trié"
print tab
```

Tri par insertion

Énoncé :

Définir une fonction **triparins** qui effectue le tri par ordre croissant d'un tableau `tab` d'Entiers donné.
Le principe du tri par insertion consiste à placer correctement l'élément courant dans la partie déjà triée du tableau.

Tri par insertion

Énoncé :

Définir une fonction **triparins** qui effectue le tri par ordre croissant d'un tableau `tab` d'Entiers donné.
Le principe du tri par insertion consiste à placer correctement l'élément courant dans la partie déjà triée du tableau.

Analyse des données du problème :

- Données du problème :
  - Le tableau `tab[1..N]` : Entier
- Données en sortie :
  - Le même tableau `tab[1..N]` mais trié

Algorithme

triparins.txt

```
constante N = 10
fonction triparins(tab: tableau[1..N] de Entier)
   pour i= 2: Entier à N
        j= i-1: Entier
        aux = tab[i]: Entier
        tant que j >= 1 ET tab[j] > aux faire
             tab[j+1] <- tab[j]
             j <- j-1
        ftq
   tab[j+1] <- aux
   fpour i
ffct triparins
```

Programme

triins.py

```
import random

def triparins(tablo) :
   for i in range(1, len(tablo)):
       j = i-1
       aux =  tablo[i]
       while ((j >= 0) and (tablo[j] > aux)):
             tablo[j+1] = tablo[j]
             j = j-1
       tablo[j+1] = aux
       # print i, tablo # decommenter pour debug

#Debut du programme
taille = 20  ## Longueur du tableau

# Initialisation du tableau avec des valeurs aléatoires
tab = random.sample(range(101), taille)
print "Affichage du tableau non trié"
print tab
# Appel du tri
triparins(tab)
print "\nAffichage du tableau trié"
print tab
```

Objectifs de cette demi-journée

Sujets abordés durant le jour 1

- Python utilisé comme langage de programmation
- Implantation d'algorithmes simples en Python

Objectifs de cette demi-journée

- Utilisation de Python comme logiciel de calcul scientifique
- Focus certaines bibliothèques fournies avec Python :
  - **Numpy** = vecteurs, matrices, algèbre linéaire
  - **MatPlotLib** = représentations graphiques de données
  - **SciPy** = recherche d'extrema, interpolation, ...
- Réutilisation de ce qui a été vu durant le jour 1

Manipulation de vecteurs/matrices avec NumPy

Cf. « Résumé des principales fonctionnalités apportées par la bibliothèque NumPy » dans le notebook IPython `Notes_sur_Numpy.ipynb` (à télécharger sur https://gist.github.com/olberger/302a61988cd6ecf40d3d ou en version HTML en lecture seule : http://nbviewer.ipython.org/302a61988cd6ecf40d3d)

Exercice de synthese

Énoncé du problème :

Définir les deux matrices carrées suivantes en Python sans utiliser de boucle :

$$A = \begin{bmatrix} 1 & 2 & \dots & 10 \\ 1 & 2 & \dots & 10 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 2 & \dots & 10 \end{bmatrix}, \qquad B = \begin{bmatrix} 0 & & \dots & & 0 \\ 2 & \ddots & & & \\ 0 & 3 & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \\ 0 & \dots & 0 & 9 & 0 \end{bmatrix}.$$

Solution

Pour A, utiliser un produit matrice-vecteur :

```
>>> u = ones( (10,1) )            # vecteur colonne de 1s
>>> v = array( [ range(1,11) ] ) # vecteur ligne [1 2 ... 10]
>>> A = dot(u, v)
```

Pour B, supprimer la première colonne et la dernière ligne d'une matrice diagonale :

```
>>> D = diag( range(1,11) )       # matrice diagonale
>>> tmp = D[0:9]
>>> B = tmp[:, 1:10]
```

Version plus futée :

```
>>> B = diag( range(1,11) )[:-1, 1:]
```

MatPlotLib

Cf. notebook IPython `Notes_sur_Matplotlib.ipynb` (à télécharger sur https://gist.github.com/olberger/4028a96c85cc54135a18 ou en version HTML en lecture seule : http://nbviewer.ipython.org/4028a96c85cc54135a18)
Voir aussi http://matplotlib.org/gallery.html

Exercice de synthèse – Énoncé

1. En utilisant la fonction randint fournie par pylab, écrire une fonction dice() qui renvoie un entier aléatoire entre 1 et 6.
2. Tester la fonction dice() en l'appelant 20 fois. On affichera les différents résultats obtenus sur une seule ligne.
3. Écrire une fonction f qui, sur la donnée d'un entier k, renvoie la somme de chiffres obtenus en lançant k dés.
4. Tester f pour $k = 5$. On fera 100 appels à f, on stockera les résultats dans une liste. Affichera cette liste, la moyenne et l'écart type.
5. Représenter graphiquement à l'aide d'histogrammes les données obtenues à la question précédente. Ajouter la courbe de la Gaussienne de paramètres $5 \times (7/2)$ et $\sqrt{5} \times \sqrt{35/12}$.
6. Généraliser en faisant une fonction qui, étant donnés k et n, fait n appels à $f(k)$, et affiche une représentation en histogramme ainsi que la courbe Gaussienne théorique.

Exercice de synthèse – Corrigé

Cf. notebook IPython `Exercice-Probas_et_Matplotlib.ipynb` (à télécharger sur https://gist.github.com/olberger/06db3a415c98689c60a4 ou en version HTML en lecture seule : http://nbviewer.ipython.org/06db3a415c98689c60a4)

Algèbre linéaire numérique avec NumPy

Cf. notebook IPython `NumPy-Algebre_lineaire.ipynb` (à télécharger sur https://gist.github.com/olberger/5758183 ou en version HTML en lecture seule : http://nbviewer.ipython.org/5758183)

Descriptif de la bibliothèque SciPy

Éléments fournis à l'import du module *scipy* :

- quelques constantes mathématiques (π, e)
- fonctions mathématiques usuelles ($\sqrt{\cdot}$, $\log(\cdot)$, $\cos(\cdot)$, ...)
- accès à plusieurs sous-modules permettant de résoudre divers problèmes mathématiques :
  - optimisation
  - interpolation
  - traitement du signal (transformées de Fourier, ...)
  - integration numérique
  - résolution d'équations aux dérivées partielles

Beaucoup d'exemples disponibles dans le cookbook :
http://www.scipy.org/Cookbook

Illustration de quelques fonctionnalités de SciPy

Cf. notebook IPython `Notes_sur_SciPy.ipynb` (à télécharger sur https://gist.github.com/olberger/187cfab86f8a0dcd20e4 ou en version HTML en lecture seule : http://nbviewer.ipython.org/187cfab86f8a0dcd20e4)

Autres tutoriels

Cf. :

- http ://www.courspython.com/ (Montpellier)
- http ://python-prepa.github.io/ (ENS Paris)

Polyvalence

Python est un langage polyvalent
Pas limité au domaine de l'informatique scientifique

- interface vers le système d'exploitation
- complétion des motifs de noms de fichiers
- gestion des arguments de ligne de commande
- gestion des erreurs et fin d'exécution
- recherche de motifs dans des chaînes
- mathématiques
- accès internet
- dates et heures
- compression de données
- mesure de performances
- contrôle qualité
- etc.

Applications sur le bureau

Exemple d'application wxPython

wxbasic.py

```
import wx

def OnClose(event):
    dlg = wx.MessageDialog(top,
        "Voulez-vous vraiment quitter cette application ?",
        "Confirmer la fermeture", wx.OK|wx.CANCEL|wx.ICON_QUESTION)
    result = dlg.ShowModal()
    dlg.Destroy()
    if result == wx.ID_OK:
        top.Destroy()

app = wx.App(redirect=True)
top = wx.Frame(None, title="Salut tout le monde", size=(300,200))
top.Bind(wx.EVT_CLOSE, OnClose)
top.Show()
app.MainLoop()
```

Client HTTP

```
>>> import urllib2
>>> f = urllib2.urlopen('http://www.python.org/')
>>> print f.read(100)
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.o
```

Applications Web 2.0

Affiche des messages parlant de Python sur le site de micro-blog `identi.ca`

identica.py

```
import urllib

url = urllib.urlopen("http://identi.ca/api/search.json?q=python")

import json

monjson = json.loads(url.read())

for elem in monjson['results']:
    print elem['from_user']+" le "+elem['created_at']
    print elem['text']+"\n"
```

Serveur Web

webserver.py

```
import SocketServer
import SimpleHTTPServer
IP, PORT = '', 8181

def sayHello():
    return 'Hello'

class CustomHandler(SimpleHTTPServer.SimpleHTTPRequestHandler):
    def do_GET(self):
        if self.path=='/hello':
            self.send_response(200)
            self.send_header('Content-type','text/html')
            self.end_headers()
            self.wfile.write(sayHello())
            return
        else:
            SimpleHTTPServer.SimpleHTTPRequestHandler.do_GET(self) #dir listing

httpd = SocketServer.ThreadingTCPServer((IP, PORT),CustomHandler)
httpd.serve_forever()
```

Web sémantique

Exploitation de méta-données RDF avec la bibliothèque rdflib

rdflibex.py

```
import rdflib
g = rdflib.Graph()
result = g.parse("http://www-public.telecom-sudparis.eu/~berger_o/foaf.rdf")

print("le graphe a %s triples." % len(g))

FOAF = rdflib.Namespace('http://xmlns.com/foaf/0.1/')

for triple in g.triples( (None, FOAF.knows, None) ) :
    print triple

s = g.serialize(format='turtle')
print s
```

Bases de données

Stockage de données dans une base SQL locale sqlite

bdsqlite.py

```
import sqlite3
connection = sqlite3.connect('exemple.db')

cursor = connection.cursor()

cursor.execute("CREATE TABLE tel (nom varchar, tel varchar)")

cursor.execute("INSERT INTO tel(nom, tel) VALUES ('christian', '4098')")

cursor.execute("SELECT * FROM tel")

for x in cursor.fetchall():
    print x

# Affiche (u'christian', u'4098')

connection.commit()
```

Installation et configuration de IDLE

- Pour
  - Simple et portable
- Contre
  - Interface en anglais
  - Interface utilisateur mal intégrée dans les environnements (raccourcis clavier, etc.)

Utiliser les Notebooks IPython pour la formation

- Pour
  - Rien à installer (navigateur)
- Contre
  - Sécurité : faire tourner en local : Python permet de tout casser dans un compte !
  - Tester la charge serveur + réseau !
  - Introduit de la distance : Python un vrai langage pour mes programmes sur mon ordinateur ?
  - Surcharge cognitive ?
  - Outil un peu jeune

Autres outils

PythonTurtle

- RoboZZle : http ://robozzle.com/js/index.aspx
- http ://pythonturtle.org/
- https ://github.com/jiffyclub/ipythonblocks/

Turtle : façon logo

À suivre

- Bases de Données
- programme 2ème année

http ://www.virtualenv.org/
Problème : paquetages de distributions Linux pas dispo ou pas à jour.
Solution : Installation Python dans un sous-répertoire sur système Linux (et Windows (?))

- Pour
  - Ne pas installer sauvagement des bibliothèques Python sur une machine où il y a déjà des paquetages (comment désinstaller)
- Contre
  - Encombrement des disques

« **Programme d'informatique** des classes préparatoires scientifiques Mathématiques, physique et sciences de l'ingénieur (MPSI), Physique, chimie et sciences de l'ingénieur (PCSI), Physique, technologie et sciences de l'ingénieur (PTSI), Technologie et sciences industrielles (TSI), Technologie, physique et chimie (TPC), Mathématiques et physique (MP), Physique et chimie (PC), Physique et sciences de l'ingénieur (PSI), Physique et technologie (PT) »

- Bulletin officiel spécial n° 3 du 30 mai 2013 (NOR : ESRS1306084A - arrêté du 4-4-2013 - J.O. du 30-4-2013 - ESR - DGESIP A2)
- annexe : Programme de la discipline informatique

Copyright

À propos de ce document

Ce document a été édité avec Emacs et Org-Mode, pour générer du LaTeX avec le package *beamer*.
Les exemples sont générés avec le module `babel` d'Org-Mode qui offre des fonctions d'intégration de code dans le source Org-Mode.