DAUPHINE
UNIVERSITÉ PARIS

Master Mathématique, Informatique, Décision, Organisation (MIDO)
2ème année - Spécialités ID, MIAGE-IF, MIAGE-SITN

TP de langage Python
2013-2014

Maude Manouvrier

La rédaction de ce TP a été réalisé à l'aide des tutoriels et livres en ligne, cités à la fin de ce document, ainsi que du TP de M. Menceur qui faisait ce cours en 2010-2011.

Ce TP va vous permettre d'apprendre le langage **Python par l'exemple**, à l'aide de petits exercices.

Table des matières

Python, développé depuis 1989 par Guido van Rossum et de nombreux contributeurs bénévoles, est un langage à typage dynamique (i.e. le type des objets manipulés n'est pas forcément connu à l'avance mais est défini à partir de la valeur de la variable) et fortement typé (i.e. qu'il garantit que les types de données employés décrivent correctement les données manipulées). Il est doté 'd'une gestion automatique de la mémoire par ramasse-miettes (pas de gestion de pointeurs ! ! !) et d'un système de gestion d'exceptions.

En Python : tout est objet.

Le langage Python peut être interprété (interprétation du bytecode compilé) ou traduit en *bytecode*, qui est ensuite interprété par une machine virtuelle Python. Il est interfaçable avec des langages comme le C, le C++ ou Java.

Pour la réalisation de ce TP, vous êtes invités à lire ce qui suit dans le document, tout en vous aidant de la documentation en ligne et des ouvrages référencés à la fin du document et dont les liens sont accessibles à partir de la page Web : `http://www.lamsade.dauphine.fr/~manouvri/PYTHON/CoursPython_MM.html`.

1 Prise en main de Python

Cette section vous explique comment exécuter des commandes ou un programme Python soit en ligne de commande soit depuis Eclipse. Il est conseillé de tester les deux environnements.

1.1 En ligne de commandes

Pour la prise en main de Python en ligne de commande (i.e. à l'aide d'un terminal), vous aurez besoin de l'intérpréteur python (`/opt/python2.7/bin/python2.7`) et de votre éditeur de texte préféré.
Attention : si vous utilisez Python 3, la syntaxe de certaines commandes peut être différente.

Pour lancer l'interpréteur : taper uniquement la commande `python`. Le symbole `>>>` correspond au signal d'invite, ou *prompt* principal, lequel vous indique que Python est prêt à exécuter une commande. Les lignes non précédées de ce symbole correspond à l'affichage d'un résultat. Aprés avoir saisi chaque instruction, il suffit de taper sur la touche `Enter` pour que la commande soit exécutée (i.e. interprétée). Pour quitter l'interpréteur de commandes, il faut taper l'instruction `quit()`.

Pour exécuter un programme tapé dans un fichier (d'extension `.py`), il suffit de saisir la commande suivante dans un terminal : `python MonProgramme.py`

1.2 Depuis Eclipse avec Pydev

Vous pouvez programmer en Python depuis Eclipse en utilisant `PyDev`, un plugin Eclipse pour Python. Pour lancez Eclipse avec Pydev depuis la ligne de commande :

`/opt/eclipse-2010/eclipse-pydev/eclipse`

Vous devez d'abord créer un projet Pydev (en demandant qu'un répertoire `src` soit créé) puis un *Pydev Module* pour écrire votre programme. L'exćution se fait par le menu *Run As*, puis *Python Run*.

Aidez-vous du tutoriel *Python Development with PyDev and Eclipse - Tutorial* de Lars Vogel, 2011 - `http://www.vogella.de/articles/Python/article.html`

2 Premiers pas en Python

Cette section présente quelques exemples de code Python, réalisés avec `Python 2.7.1` en ligne de commande.

Les lignes commençant par >>> correspondent aux instructions. Les lignes situées juste en dessous correspondent à l'affichage après exécution de l'instruction (i.e. après avoir tapé <Enter>).

En Python, les commentaires commencent par le symbole #.

Vous êtes invités à taper les exemples ci-dessous pour vous entaîner et à répondre à chaque question associée aux exemples. Vous pouvez soit copier chaque instructions dans l'interpréteur de commandes Python, soit copier les instructions dans un module Pydev pour les exécuter sous Eclipse.

2.1 Faire des calculs avec Python

Exercice 1 : **Quelques exemples de calcul**
Essayez, en les exécutant, de comprendre ce que fait chaque instruction (non commentée) de l'exemple ci-dessous.

Cet exemple est valable en ligne de commande uniquement. En ligne de commande, la valeur d'une variable ou le résultat d'un calcul s'affiche directement après la saisie de cette variable ou de ce calcul. Pour exécuter cet exemple sous Eclipse, il suffit d'ajouter l'instruction `print` devant chaque instruction correspondant à un affichage (i.e. autre que celles du type `variable = valeur`).

```
>>> 5+3 # Taper print 5+3 si vous êtes sous Eclipse
8
>>>5*3
15
>>>5**3
125
>>> x=1 # déclaration d'un variable x de valeur 1 (# pour le commentaire)
>>> x   # affichage de x
1
>>> a,b,c=3,5,7 # déclaration de 3 variables a, b et c de valeurs resp. 3, 5 et 7
>>> a-b/c
3
>>> (a-b)/c
-1
>>> b/c
0
>>> b//c
0
>>> b%c
5
>>> d=1.1
>>> d/c
0.15714285714285717
>>> d//c
0.0
```

Exemple 1 : **Importation de la librairie mathématique et exemple de fonction mathématique**

```
>>> from math import *  # Pour importer la librairie de fonctions mathématiques
>>> sqrt(4) # Pour calculer la racine carrée
2.0
>>>pi
3.141592653589793
```

NB : La liste des fonctions de la librairie `math` est disponible à l'adresse :
`http://docs.python.org/library/math.html?highlight=math#math`

2.2 Affichage

Exemple 2 : Utilisation de la fonction d'affichage `print()`

```
>>> print(a+b) # a et b sont les variable de l'exercice 1
8
>>> print('la valeur de', a,'+',b,'est :', a+b)
('la valeur de', 3, '+', 5, 'est :', 8)
>>> print 'la valeur de', a,'+',b,'est :', a+b
la valeur de 3 + 5 est : 8
```

2.3 Déclaration et initialisation de variables et types

Exemple 3

```
>>> print(type(a)) # a est la variable de l'exercice 1
<type 'int'>
>>> pi=3,14
>>> print(type(pi))
<type 'tuple'>
>>> pi=3.14
>>> print(type(pi))
<type 'float'>
>>> s='exemple de chaine de caracteres'
>>> s
'exemple de chaine de caracteres'
>>> type(s)
<type 'str'>

>>> 2+'1.5'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unsupported operand type(s) for +: 'int' and 'str'
>>> 2+eval('1.5') # Pour éliminer l'erreur précédente
3.5
```

2.4 Chaînes de caractères

Exemple 4 : Manipulation des chaîne de caractères et exemples de fonctions sur les chaînes de caractères

```
>>> s='un exemple de chaine'
>>> s2="un autre exemple"
>>> s[1] # Accés au caractère d'indice 1 (les indices commencent à zéro)
'n'
>>> print(s[0],s2[0])
('u', 'u')
>>> print(s[4],s2[0])
('x', 'u')
>>> print s + ' et ' + s2 # Concaténation de chaînes
un exemple de chaine et un autre exemple
>>> s3=s + ' et ' + s2
```

```
>>> s3
'un exemple de chaine et un autre exemple'
>>> print('La taille de s est :', len(s))
('La taille de s est :', 20)
>>> s3[0:3] # Récupération des caractères de position entre les 0 et 3
'un '
>>> s3[4:8]
'xemp'
>>> print s3[:3] # Récupération des 3 premiers caractères
un
>>> print s3[3:] # Récupération des caractères à partir de la position 3
exemple de chaine et un autre exemple
```

Exemple 5 : **Exemple de récupération des mots d'une chaîne de caractères**

```
>>> sentence = 'It is raining cats and dogs'
>>> words = sentence.split()
>>> print words
['It', 'is', 'raining', 'cats', 'and', 'dogs']
```

2.5 Boucles et conditions

Exercice 2 : **Boucle for**
Tapez le code suivant et observez le résultat.

```
>>> for i in range(10): # Ne pas oublier les deux points!!
...   x = 2 # Attention ne pas oublier une tabulation en d\'ebut de ligne sinon erreur!!!
...   print(x*i) # Ne pas oublier la tabulation en d\'ebut de ligne!!
# Tapez encore une fois <Enter> si vous êtes en ligne de commande
```

Exercice 3 : **Boucle while**
Tapez le code suivant et observez le résultat.

```
>>> a=0
>>> while(a<12):  # Ne pas oublier les deux points!!
...     a=a+1 # Ne pas oublier la tabulation en début de ligne!!
...     print(a, a**2,a**3) # Ne pas oublier la tabulation en début de ligne!!
# Tapez encore une fois <Enter> si vous êtes en ligne de commande
```

Exercice 4 : **Condition If/Then/Else**
Tapez le code suivant et observez le résultat.

```
>>> a=0
>>> if a==0:  # Ne pas oublier les deux points!!
...     print('0')  # Ne pas oublier la tabulation en début de ligne!!
... elif a==1: # Ne pas mettre de tabulation et ne pas oublier les deux points!!
...     print('1') # Ne pas oublier la tabulation en début de ligne!!
... else:  # Ne pas mettre de tabulation et ne pas oublier les deux points!!
...     print('2') # Ne pas oublier la tabulation en début de ligne!!
# Tapez encore une fois <Enter> si vous êtes en ligne de commande
```

2.6 Récupérer des saisies claviers

Exercice 5 : Récupérez, analysez et exécutez le fichier
`http://www.lamsade.dauphine.fr/~manouvri/PYTHON/EXEMPLES/SaisieEntier.py`

Remarque : Pour plus de détails sur le pseudo-dommentaire `# -*- coding:Utf8 -*-` - cf. `http://inforef.be/swi/download/apprendre_python.pdf`, page 43, dont est tiré l'exemple.

Exercice 6 : Ecrivez un script permettant d'obtenir le résultat suivant

```
Saisissez une chaine
une chaine
La chaine inversée est:
eniahc enu
```

Astuces : la fonction `raw_input()` permet de saisir une chaîne de caractères au clavier. La concaténation entre deux chanes `s1` et `s2` se fait par l'instruction `s1 + s2`. Une chaîne vide se définit par `s=""`.

3 Structures de données

3.1 Listes

Exercice 7 : **Manipulation des listes**
Tapez chacune des insctructions suivantes (en ligne de commande) et observez le résultat.

```
>>> list=['lundi', 2, 'janvier']
>>> print list
>>> list[0]
>>> print list[2]
>>> len(list)
>>> list.append(2010) # list est un objet - nous verrons cela plus tard
>>> list
>>> list[3]=list[3]+1
>>> del list[0]
>>> list
>>> list.insert(0,'mardi')
>>> list
>>> 'mardi' in list
>>> 'lundi' in list
>>> list.index(2)
>>> list2=list[1:3]
>>> list2
>>> list3=list[:2]
>>> list3
>>> list4=list[1:]
>>> list4
>>> list3=list3 + [2011]
>>> list3
>>> list5=3*list
>>> list5
>>> list.extend([3,4])
>>> list
>>> list6=list.pop(0)
>>> list6
>>> list
```

Exemple 6 : **Utilisation d'une liste pour échanger deux variables**
Repris de A Quick, Painless Tutorial on the Python Language de Norman Matloff.

```
>>> x = 5
>>> y = 12
>>> [x,y] = [y,x]
>>> x
12
>>> y
5
```

Exercice 8 : Ecrivez une programme qui demande à l'utilisateur d'entrer des notes d'élèves. Si l'utilisateur entre une valeur négative, le programme s'arrête. En revanche, pour chaque note saisie, le programme construit progressivement une liste. Après chaque entrée d'une nouvelle note (et donc à chaque itération de la boucle), il affiche le nombre de notes entrées, la note la plus élevée, la note la plus basse, la moyenne de toutes les notes.
Astuces : Pour créer une liste vide, il suffit de taper la commande `list = []`.

3.2 Dictionnaire

Exercice 9 : Tapez le code suivant [1] et observez le résultat.

```
>>> dico = {}
>>> dico['computer'] = 'ordinateur'
>>> dico['mouse'] ='souris'
>>> dico['keyboard'] ='clavier'
>>> print dico
>>> print dico.keys()
>>> print dico.values()
>>> del dico['mouse']
>>> print dico
>>> print len(dico)
>>> print dico.has_key('computer')
>>> print dico.items()
>>> for clef in dico:
...     print clef # Attention ne pas oublier la tabulation!!
>>> for clef in dico:
...     print clef, dico[clef] # Attention ne pas oublier la tabulation!!
>>> for clef, value in dico.items():
...     print clef, value # Attention ne pas oublier la tabulation!!
>>> dico2={'ordinateur': 'computer', 'souris' : 'mouse'}
>>> print dico2
>>> dico2=dico
>>> print dico
>>> del dico2['computer']
>>> print dico
>>> print dico2
>>> dico3=dico.copy()
>>> del dico3['keyboard']
>>> print dico
>>> print dico3
```

1. Repris et adapté de *Apprendre à programmer avec Python* de Gérard Swinnen, 2009

4 Fichiers

4.1 Instanciation du répertoire courant

Exemple 7 : modifier le répertoire courant de l'interpréteur

```
>>>from os import chdir
>>>chdir("/home/login/exercices") #Mettre ici le répertoire où sont stockés vos exercices
```

La première ligne permet d'importer la commande `chdir()` du module `os` contenant une série de fonctions permettant de dialoguer avec le système d'exploitation (*os = operating system*).
La deuxième ligne permet de spécifier le répertoire courant de l'interpréteur python, i.e. le répertoire où il va récupérer les fichiers.

4.2 Manipulation de fichiers

Exercice 10 : Tapez le code suivant et observez le résultat.

```
>>> f=open('test.txt','w') # Pour ouvrir un fichier en mode écriture
>>> f.write("Bonjour\n") # Pour écrire dans le fichier
>>> f.write("Ceci est un test d'ecriture dans un fichier")
>>> f.close() # Pour fermer le fichier - vérifier le fichier dans un éditeur de texte
>>> f=open('test.txt','r') # # Pour ouvrir un fichier en mode lecture
>>> b=f.read() # Pour lire tout le fichier
>>> print b
>>> f.close() # Pour fermer le fichier
>>> f=open('test.txt','r') # # Pour ouvrir un fichier en mode lecture
>>> b=f.read(3) # Pour lire 3 caractère à partir de la position courante du curseur
>>> print b
>>> f.close()
>>> f=open('test.txt','r')
>>> b = f.readline() # Pour lire une ligne dans le fichier
>>> print b
>>> b = f.readlines() # Pour lire toutes les lignes et les stocker dans une liste
>>> print b
>>> f.close()
>>> f=open('test.txt','a') # Pour ouvrir le fichier en mode ajout
>>> f.write("\n Fin")
>>> f.close()
>>> f=open('test.txt','r')
>>> b = f.readlines()
>>> print len(b)
>>> print b
>>> f.close()
>>> f=open('test.txt','r')
>>> print f.read() # Pour lire tout le fichier - déplace le curseur à la fin du fichier
>>> print f.read()
>>> f.close()
>>> f=open('test.txt','r')
>>> print f.read()
>>> f.seek(0,0) # Déplace le curseur de 0 caractère depuis le début du fichier (codé par 0)
>>> print f.read()
>>> f.seek(-10,2) # Déplace le curseur de 10 caractères en arrière
                  depuis la fin du fichier (codée par 2)
```

```
>>> print f.read()
>>> f.seek(-10,2)
>>> f.seek(-5,1) # Déplace le curseur de 5 caractères en arrière
                 depuis la position courante (codée par 1)
>>> print f.read()
>>> f.close()
```

4.3 Copie de fichiers

Exercice 11 : Récupérez le fichier
`http://www.lamsade.dauphine.fr/~manouvri/PYTHON/EXEMPLES/CopieFichier.py`
Etudiez le code en ouvrant le fichier dans un éditeur de texte et exécutez le.

4.4 Copier des variables dans un fichier

Exercice 12 : Tapez le code suivant [2] et observez le résultat.

```
>>> a = 5
>>> b = 2.83
>>> c = 67
>>> f = open('test.txt', 'w')
>>> f.write(str(a)) # Pour écrire un entier converti en string
>>> f.write(str(b))
>>> f.write(str(c))
>>> f.close()
>>> f = open('test.txt', 'r')
>>> print f.read()
>>> f.close()

>>> import pickle # Pour importer un module permettant de conserver le type des variables
>>> f = open('Monfichier', 'w')
>>> pickle.dump(a, f) # Pour enregistrer dans le fichier
>>> pickle.dump(b, f) # équivalent à write mais en conservant le type de b
>>> pickle.dump(c, f)
>>> f.close()
>>> f = open('Monfichier', 'r')
>>> print f.read()
>>> f.close()
>>> f = open('Monfichier', 'r')
>>> t = pickle.load(f) # Opération inverse de dump
>>> print t, type(t)
>>> t = pickle.load(f) # Equivalent à read mais en conservant le type de b
>>> print t, type(t)
>>> t = pickle.load(f)
>>> print t, type(t)
>>> f.close()
```

2. Repris de *Apprendre à programmer avec Python* de Gérard Swinnen, 2009

5 Fonctions

5.1 Fonctions Python existantes

La liste des fonctions Python existantes est disponible en ligne à l'adresse :
`http://docs.python.org/library/functions.html`

5.2 Fonction simple sans paramètre

Exercice 13 : **Définir une fonction sans paramètre**
Tapez le code suivant et observez le résultat.

```
>>> def function(): # Définition de fonction sans paramtère - Ne pas oublier les :
...     n=10 # Faire une tabulation en début de ligne
...     while n>0: # Faire une tabulation en début de ligne et ne pas oublier les :
...             print n/2, n%2 # Faire 2 tabulations en début de ligne
...             n=n-1  # Faire 2 tabulations en début de ligne
# Tapez encore une fois <Enter> si vous êtes en ligne de commande
>>>function() # Appel de la fonction
```

Exercice 14 : **Définir une fonction sans paramètre qui appelle une autre fonction**
Tapez le code suivant et observez le résultat.

```
>>> def function2(): # Ne pas oublier les : et une tabulation sur la ligne suivante
...     print 'Affichage du résultat et du reste de la division des entiers de 10 à 0 par 2 :'
...     function() # Faire une tabulation en début de ligne
# Tapez encore une fois <Enter> si vous êtes en ligne de commande
>>>function2()
```

5.3 Fonction avec paramètres

Exercice 15 : **Définir une fonction à 1 paramètre**
Tapez le code suivant et observez le résultat.

```
def function(n): # D\'efinition d'une fonction à un paramètre
...     while n>0: # Faire une tabulation en début de ligne et ne pas oublier les :
...             print n/2, n%2 # Faire 2 tabulations en début de ligne
...             n=n-1 # Faire 2 tabulations en début de ligne
# Tapez encore une fois <Enter> si vous êtes en ligne de commande
>>> function(3) # Appel de la fonction avec le paramètre 3
# Tapez encore une fois <Enter> si vous êtes en ligne de commande
>>> function(5) # Appel de la fonction avec le paramètre 5
```

Exercice 16 : **Définir une fonction à 2 paramètres**
Tapez le code suivant et observez le résultat.

```
>>> def function(entree,diviseur): # Définition d'une fonction à plusieurs paramètres
...     while(entree>0):
...             print entree/diviseur, entree%diviseur
...             entree=entree-1
# Tapez encore une fois <Enter> si vous êtes en ligne de commande
>>> function(10,2)
```

5.4 Valeur par défaut des paramètres

Exercice 17 : Tapez le code suivant et observez le résultat.

```
>>> def function(entree,diviseur=2): # fonction avec valeur par défaut pour diviseur
...     while(entree>0):
...             print entree/diviseur, entree%diviseur
...             entree=entree-1
# Tapez encore une fois <Enter> si vous êtes en ligne de commande
>>> function(10)
```

Exercice 18 : Tapez le code suivant et observez le résultat.

```
>>> def function(entree=10,diviseur=2):
...     while(entree>0):
...             print entree/diviseur, entree%diviseur
...             entree=entree-1
# Tapez encore une fois <Enter> si vous êtes en ligne de commande
>>> function()
>>> function(4,2)
>>> function(diviseur=3,entree=9)
```

5.5 Affecter une instance de fonction à une variable

Exercice 19 : Tapez le code suivant et observez le résultat.

```
>>>def multiplication(n,p):
...     return n*p # pour retourner une valeur
# Tapez encore une fois <Enter> si vous êtes en ligne de commande
>>> a=multiplication(3,6)
>>> a
```

5.6 Exemple d'utilisation de fonctions

Exercice 20 : Récupérez le fichier
`http://www.lamsade.dauphine.fr/~manouvri/PYTHON/EXEMPLES/CalculVolumeSphere.py`
étudiez le code en ouvrant le fichier et exécutez le.

Exercice 21 : Tapez le code suivant et observez le résultat.

```
>>> def afficher3fois(arg):
...     print arg, arg, arg
# Tapez encore une fois <Enter> si vous êtes en ligne de commande
>>> afficher3fois(3)
>>> afficher3fois('exemple')
>>> afficher3fois([3,4])
>>> afficher3fois(3*4)
```

5.7 Variable locale/variable globale

Exercice 22 : Récupérez les fichiers `tme.py` et `x` à l'adresse :
`http://www.lamsade.dauphine.fr/~manouvri/PYTHON/EXEMPLES/`
et étudier le commentaire de ce programme dans la section 4 de :
`http://mononeurona.org/files/userfiles/aarkerio_121.pdf`.

5.8 Fonction anonyme (lambda function)

Python permet la création de fonctions anonymes (i.e. sans nom et donc non définie par `def`) à l'aide du mot-clé `lambda`. Une fonction anonyme ne peut pas avoir d'instruction `return` et doit forcément retourner une expression. De telles fonctions permettent de représenter tout sous forme de fonctions sans réllement en définir explicitement.

Exercice 23 : Tapez le code [3] suivant et observez le résultat.

```
>>> def f (x): return x**2
>>> print f(8)
>>> g = lambda x: x**2
>>> print g(8)
>>> (lambda x: x*2)(3) //2
>>> def make_incrementor (n): return lambda x: x + n
>>> f = make_incrementor(2)
>>> g = make_incrementor(6)
>>> print f(42), g(42)
>>> print make_incrementor(22)(33)
>>> sentence = 'It is raining cats and dogs'
>>> words = sentence.split()
>>> print words
>>> lengths = map(lambda word: len(word), words)
>>> print lengths
>>> print map(lambda w: len(w), 'It is raining cats and dogs'.split())
```

6 Gestion des exceptions

Exercice 24 : Récupérez le fichier
`http://www.lamsade.dauphine.fr/~manouvri/PYTHON/EXEMPLES/ExceptionOuvertureFichier.py`
étudiez le code en ouvrant le fichier dans un éditeur de texte et exécutez le.

7 Programmation orientée-objet

7.1 Premier exemple de classe

Exercice 25 : Récupérez le fichier
`http://www.lamsade.dauphine.fr/~manouvri/PYTHON/EXEMPLES/ExempleClasse1.py`
étudiez le code en ouvrant le fichier et exécutez le.

Exercice 26 : Tapez le code suivant et observez le résultat.

```
>>> import ExempleClasse1
>>> c1 = ExempleClasse1.CompteBancaire('Toto', 1000)
>>> c1.depot(350)
>>> c1.retrait(200)
>>> c1.affiche()
>>> print c1.nom
>>> print c1
```

3. Repris et adapté de `http://www.secnetix.de/olli/Python/lambda_functions.hawk`

7.2 Accessibilité

Exercice 27 : Reprendre la classe de l'exercice précédent et remplacer `nom` par `__nom`. Exécutez à nouveau l'exemple de l'exercice précédent. Observez le résultat.

7.3 Objet complexe

Exercice 28 : Récupérez le fichier
`http://www.lamsade.dauphine.fr/~manouvri/PYTHON/EXEMPLES/rect_carre.py`
étudiez le code en ouvrant le fichier et exécutez le.

7.4 Héritage

Exercice 29 : Récupérez le fichier
`http://www.lamsade.dauphine.fr/~manouvri/PYTHON/EXEMPLES/formes.py`
étudiez le code en ouvrant le fichier et exécutez le.

NB : L'héritage multiple est possible en Python.

8 Documentation en ligne et liens importants

- **Tutoriel de la documentation officielle en ligne** : `http://docs.python.org/tutorial/index.html`
- **Livre et exercices corrigés en ligne sur Python 2 et 3** : *Apprendre à programmer avec Python* de Gérard Swinnen, 2009 - `http://inforef.be/swi/python.htm`
- **Livre (format pdf) en ligne sur Python 3 :** *Introduction à Python 3* de Robert CORDEAU, 2010
  `http://www.afpy.org/Members/bcordeau/Python3v1-1.pdf/download`
- **Tutoriel en ligne pour développer en Python sous Eclipse** : *Python Development with PyDev and Eclipse - Tutorial* de Lars Vogel, 2011 - `http://www.vogella.de/articles/Python/article.html`
- **Tutoriel en ligne** : *A Quick, Painless Tutorial on the Python Language* de Norman Matloff
  - Tutoriel en ligne (2008) : `http://heather.cs.ucdavis.edu/~matloff/Python/PythonIntro.html`
  - Fichier pdf (2009) : `http://mononeurona.org/files/userfiles/aarkerio_121.pdf`
- **Tutoriel en ligne** : `http://www.tutorialspoint.com/python/index.htm`
- **Documentation en ligne** : `http://infohost.nmt.edu/tcc/help/lang/python/docs.html` dont *Extending and Embedding Python* de Guido van Rossum, Fred L. Drake, Jr., editor October 06, 2009, Python Software Foundation (`http://infohost.nmt.edu/tcc/help/lang/python/2_6_3/extending.pdf`).
- **Site officiel python** : `http://www.python.org/`
- **Association francophone Python** : `http://www.afpy.org/python/tutoriels`
- **Documentation en ligne de l'intégration C/C++/Python** :
  `http://docs.python.org/extending/extending.html`