Python

Vincent Angladon

ENSEEIHT

8 novembre 2011

Présentation disponible sur
http ://www.bde.enseeiht.fr/clubs/net7/supportFormations/python/2011/python.pdf

Table des matières

Présentation de Python

Definition

Python : un langage interprété orienté objet de haut niveau avec une syntaxe épurée.

Qui utilise Python ?

Qui utilise Python ?

Qui utilise Python ?

Qui utilise Python ?

- Nous ! !

Qui utilise Python ?

Qui utilise Python ?

- Nous ! !
- Reddit

Qui utilise Python ?

Qui utilise Python ?

- Nous ! !
- Reddit
- Google (YouTube)

Qui utilise Python ?

Qui utilise Python ?

- Nous ! !
- Reddit
- Google (YouTube)
- la NASA
- pleins d'autres . . .

Quel usage ?

- scripts, plugins

Qui utilise Python ?

Qui utilise Python ?

- Nous ! !
- Reddit
- Google (YouTube)
- la NASA
- pleins d'autres . . .

Quel usage ?

- scripts, plugins
- applications

Qui utilise Python ?

Qui utilise Python ?

- Nous ! !
- Reddit
- Google (YouTube)
- la NASA
- pleins d'autres . . .

Quel usage ?

- scripts, plugins
- applications
- pages web, cgi
- . . .

Comment coder en Python ?

- dans un shell Python (taper *python* dans une console)
- depuis un éditeur de texte (exécution du fichier créé avec la command *python monFichier.py*)

Remarque

Selon le système d'exploitation, l'exécution du programme python donnera accès à un shell Python2 ou Python3. Pour éviter toute ambiguïté, il est recommandé d'utiliser les commandes python2 ou python3.

Shells Python intéractifs

Il existe un grand nombre de shells Python intéractifs qui proposent des fonctionnalités telles que l'autocomplétion, des tooltips, ...

- pycrust (fenétré)
- ipython (console)
- dreampie (fenétré)
- ...

Avancé : python2 et python3 simultanément

1. ouvrir Terminator
2. diviser verticalement la fenêtre (clic droit)
3. lancer python2 et python3 dans chacun des shells
4. diffuser tout (menu dans le coin en haut à gauche)

Résultat : les commandes tapées dans l'interpréteur python2 le sont également dans l'interpréteur python3.

Déclaration de variables

Déclarations de variables

```
>>> pipo = 8

>>> x, y = 2, 3
>>> a = b = 2
```

Attention

Les noms de variable sont sensibles à la casse
Les accents et caractères spéciaux sont interdits
Une variable ne doit pas commencer par un chiffre

Expressions arithmétiques

Essayez diverses expressions mathématiques du style :

Exemple

```
>>> a = 3*(1+2-1)/2
>>> a += 1
3
```

Autres opérateurs arithmétiques :

division entière	puissance	modulo	comparaison
//	**	%	<, >, <=, =>, ==, !=

Différence Python2 - Python3

Python 2.x : $5/2 = 2$
Python 3.x : $5/2 = 2.5$

Page de référence pour les manipulations d'expressions arithmétiques :
`http://docs.python.org/library/stdtypes.html#numeric-types-int-float-long-complex`

Les booléens

Manipulation des booléens

```
>>> t = True
>>> f = not(t)
>>> f or t
True
>>> f and t
False
```

Les chaînes de caractères

Déclaration des chaînes de caratères

```
>>> s1, s2 = "pipo1", 'pipo2'
>>> s3 = 'des gens disent qu'Emacs est un \"vrai\" éditeur de texte'
>>> s4 = "des gens disent qu\'Emacs est un "vrai" éditeur de texte""
>>> s5 = """Une chaîne avec
deux lignes"""
```

Manipulation des chaînes de caratères

```
>>> s1 + s2
```

```
>>> s1 += s2
>>> s1
```

Page de référence pour la manipulation de chaînes de caractère :
`http://docs.python.org/library/stdtypes.html#string-methods`

Les chaînes de caractères

Déclaration des chaînes de caratères

```
>>> s1, s2 = "pipo1", 'pipo2'
>>> s3 = 'des gens disent qu'Emacs est un \"vrai\" éditeur de texte'
>>> s4 = "des gens disent qu\'Emacs est un "vrai" éditeur de texte""
>>> s5 = """Une chaîne avec
deux lignes"""
```

Manipulation des chaînes de caratères

```
>>> s1 + s2
'pipo1pipo2'
```

```
>>> s1 += s2
>>> s1
'pipo1pipo2'
```

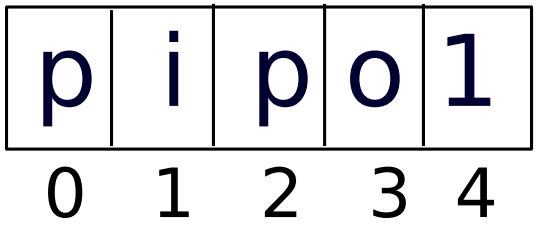
Page de référence pour la manipulation de chaînes de caractère :
`http://docs.python.org/library/stdtypes.html#string-methods`

Les chaînes de caractère suite

Accès au caractères d'une chaîne

>>> **s1 = "pipo1"**

>>> s1[0], s1[1], s1[2], s1[3] **? ?**



Les chaînes de caractère suite

Accès au caractères d'une chaîne

```
>>> s1 = "pipo1"
```

```
>>> s1[0], s1[1], s1[2], s1[3] ??
('p', 'i', 'p', 'o')
```

p	i	p	o	1
0	1	2	3	4

Les chaînes de caractère suite

Accès au caractères d'une chaîne

```
>>> s1 = "pipo1"
```

```
>>> s1[-1], s1[-2], s1[-2], s1[-4] ??
```

-5	-4	-3	-2	-1
p	i	p	o	1
0	1	2	3	4

Les chaînes de caractère suite

Accès au caractères d'une chaîne

```
>>> s1 = "pipo1"
```

```
>>> s1[-1], s1[-2], s1[-2], s1[-4] ? ?
('1', 'o', 'p', 'i')
```

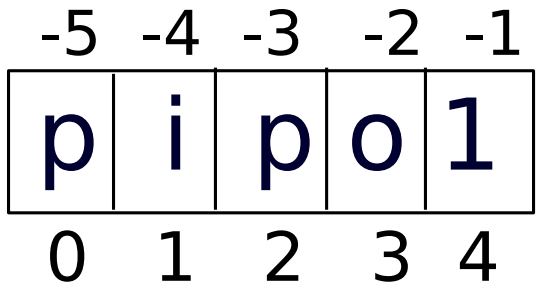
-5	-4	-3	-2	-1
p	i	p	o	1
0	1	2	3	4

Les chaînes de caractère suite

Accès au caractères d'une chaîne

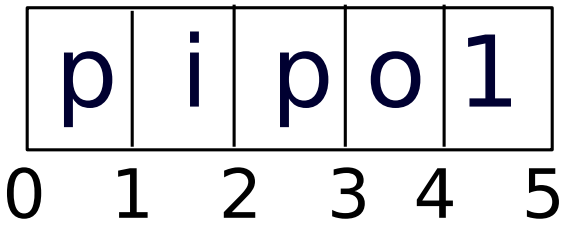
```
>>> s1 = "pipo1"
```

```
>>> s1[-1], s1[-2], s1[-2], s1[-4] ??
('1', 'o', 'p', 'i')
```



Accès à une sous-chaîne

```
>>> s1[0 :2], s1[ :2], s1[2 :]
```

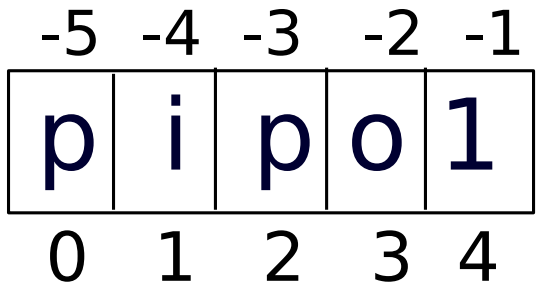


Les chaînes de caractère suite

Accès au caractères d'une chaîne

```
>>> s1 = "pipo1"
```

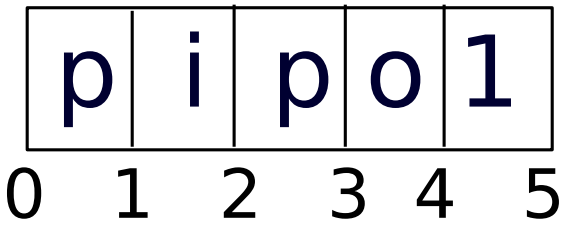
```
>>> s1[-1], s1[-2], s1[-2], s1[-4] ??
('1', 'o', 'p', 'i')
```



Accès à une sous-chaîne

```
>>> s1[0 :2], s1[ :2], s1[2 :]
```

```
('pi', 'pi', 'po1')
```

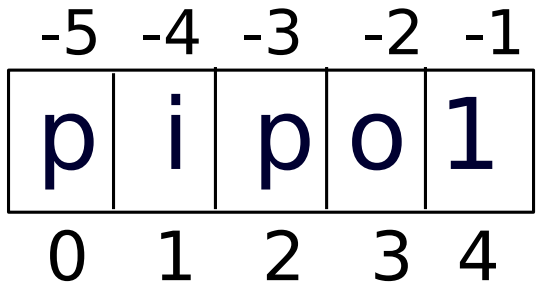


Les chaînes de caractère suite

Accès au caractères d'une chaîne

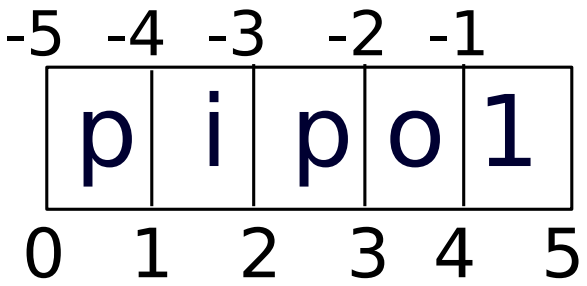
```
>>> s1 = "pipo1"
```

```
>>> s1[-1], s1[-2], s1[-2], s1[-4] ??
('1', 'o', 'p', 'i')
```



Accès à une sous-chaîne

```
>>> s1[ :-1], s1[-3 :]
```

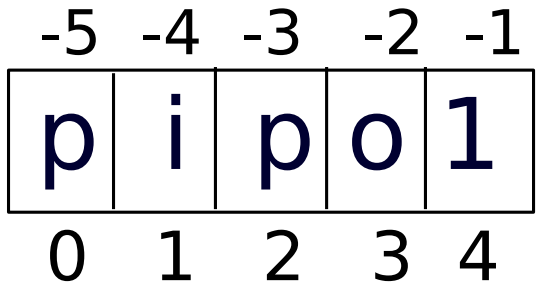


Les chaînes de caractère suite

Accès au caractères d'une chaîne

```
>>> s1 = "pipo1"
```

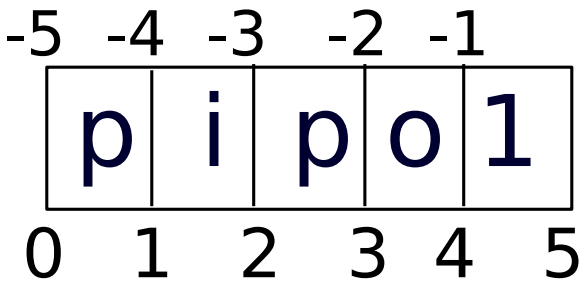
```
>>> s1[-1], s1[-2], s1[-2], s1[-4] ??
('1', 'o', 'p', 'i')
```



Accès à une sous-chaîne

```
>>> s1[ :-1], s1[-3 :]
```

```
('pipo', 'po1')
```



Les chaînes de caractère suite

Manipulation de chaînes

```
>>> len(s1)

>>> int("42"), float("3.14")

>>> "AIE".lower()

>>> "Python 2.7".replace("2.7", "3.2")

>>> "Bonjour net7".split()

>>> "Bonjour net7".find('net7')
```

Les chaînes de caractère suite

Manipulation de chaînes

```
>>> len(s1)
5

>>> int("42"), float("3.14")
(42, 3.14)

>>> "AIE".lower()
"aie"

>>> "Python 2.7".replace("2.7", "3.2")
"Python 3.2"

>>> "Bonjour net7".split()
['Bonjour', 'net7']

>>> "Bonjour net7".find('net7')
8
```

Entrées / Sorties

Sortie

```
>>> age = 10
>>> print ("J'ai ", age, " ans !")
```

Remarque

Les parenthèses ne sont pas obligatoires pour la fonction print si on utilise Python 2.x

Entrées / Sorties

Sortie

```
>>> age = 10
>>> print ("J \'ai ", age, " ans !")
```

Remarque

Les parenthèses ne sont pas obligatoires pour la fonction print si on utilise Python 2.x

Entrées

```
>>> age = input("Quel est votre âge ? ")
>>> job = raw_input("Ou travaillez-vous ? ")
```

Notre premier script

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-

# Ceci est un commentaire
print("Hello World!")
```

Les structures conditionnelles

Si Sinon

```
if age < 18:
    print("Oups, c'est interdit aux mineur")
elif age > 130:
    print("Tttttt")
else:
    print ("Bienvenue")
```

Attention

L'**indentation** est très importante !

Exercices

Exercice

Demander à l'utilisateur son nom et son sexe (chaîne de caractère M ou F). En fonction de ces données, afficher « Cher Monsieur » ou « Chère Mademoiselle » suivi du nom.

Exercice

Écrire un programme qui demande à l'utilisateur de rentrer une année et afficher si elle est bissextile ou non. (Une année A est bissextile si A n'est pas un multiple de 100 et A est divisible par 4 ou 400). 1600 est bissextile mais 1700 ne l'est pas.

Exercice

Demander à l'utilisateur d'entrer trois longueurs a, b, c. A l'aide de ces trois longueurs, déterminer s'il est possible de construire un triangle. Déterminer ensuite si ce triangle est rectangle, isocèle, équilatéral ou quelconque. Attention : un triangle rectangle peut être isocèle.

Les Structures de contrôle

Les boucles For

```
for i in range(debut, fin, pas):
    # corps de la boucle (indentation)
```

Les boucles While

```
while boolean_condition:
    # corps de la boucle (indentation)
```

Exercices

Exercice

Afficher une pyramide d'étoiles du type :
*
**
*** *(astuce : regardez ce que fait "*"*2)*

Exercice

Afficher la table de multiplication de 7

Les Fonctions

Syntaxe des fonctions

```
def nom_de_la_fonction(param_1, param_2, ..., param_n):
    "Documentation de la fonction"
    # corps de la fonction (indentation)
```

Les Fonctions

Syntaxe des fonctions

```
def nom_de_la_fonction(param_1, param_2, ..., param_n):
    "Documentation de la fonction"
    # corps de la fonction (indentation)
```

Un exemple de fonction

```
def degree2radian(angle):
    "Convertit en radian un angle en degre"
    return angle*0.017453292
```

Exercices

Exercice

Recopier la fonction degree2radian, testez là, et observez le résultat de degree2radian.__doc__

Exercice

Écrire la fonction factorielle.

Exercice

Écrire une fonction qui convertit une note sur 20 en une note standardisée suivant le code suivant : [16, 20[=A, [12,16[=B, . . .[0, 4[=F

Exercice

Écrire une fonction qui prend en paramètre les coordonnées de deux planètes ainsi que leur masse et retourne la valeur de la force gravitationnelle exercée par l'une sur l'autre.

Les modules

Importation fougueuse

```
from math import *
n = factorial(1000)
pi*log10(n)
```

Les modules

Importation fougueuse

```
from math import *
n = factorial(1000)
pi*log10(n)
```

Importation parcimonieuse

```
from time import localtime, strptime, strftime, mktime
print ("Tomorrow␣we␣will␣be␣the␣",\
asctime(localtime(mktime(localtime()) + 24 * 3600)))
```

Les modules

Importation fougueuse

```
from math import *
n = factorial(1000)
pi*log10(n)
```

Importation parcimonieuse

```
from time import localtime, strptime, strftime, mktime
print ("Tomorrow␣we␣will␣be␣the␣",\
asctime(localtime(mktime(localtime()) + 24 * 3600)))
```

Importation sage

```
import turtle
for i in range(8):
    turtle.circle(20)
    turtle.right(45)
```

Les objets structurés

- **Les tuples** : n-uplets non modifiables
- **Les listes** : tableaux dynamiques
- **Les dictionnaires** : table hachage qui associe à chaque valeur une clef unique
- **Les ensembles** : contiennent qu'un seul exemplaire d'un objet donné

Les Tuples

Exemple

Création

```
>>> t1 = ()
>>> t2 = ("signleton",)
>>> t3 = ("Bonjour", "net", 7)
```

Accès

```
>>> t3[0]
'Bonjour'
>>> t3[-1]
7
```

Découpage

```
>>> t3[0 :2]
('Bonjour', 'net')
```

Taille

```
>>> len(t3)
3
```

Concaténation

```
>>> t2+t2
('singleton', 'singleton')
```

Multiplication

```
>>> t2*3
('singleton', 'singleton', 'singleton')
```

Appartenance

```
>>> 'net' in t3
True
```

Iteration

```
>>> for element in t3 :
... print(element)
Bonjour
net
7
```

Les Listes

Exemple

Création

```
>>> l1 = []
>>> l2 = ['singleton']
>>> l3 = ["Salut", "net", 7]
```

Accès / Modification

```
>>> l3[0] = 'Bonjour'
>>> l3[-1]
7
```

Découpage

```
>>> l3[0 :2]
['Bonjour', 'net']
```

Taille

```
>>> len(l3)
3
```

Concaténation

```
>>> l2+l2
['singleton', 'singleton']
```

Multiplication

```
>>> l2*3
['singleton', 'singleton', 'singleton']
```

Appartenance

```
>>> 'net' in l3
True
```

Itération

```
>>> for element in l3 :
        print(element)
Bonjour
net
7
```

Les listes suite

Exemple

Ajout

```
>>> l1.append("Bonjour")
>>> l1
['Bonjour']
```

Ajout des éléments d'une liste

```
>>> l1.extend(["net", 7])
>>> l1
['Bonjour', 'net', 7]
```

Insertion

```
>>> l2.insert(1, 'tout')
>>> l2
['Bonjour', 'tout', 'net', 7]
```

Recherche

```
>>> l2.index("net")
1
```

Suppression

```
>>> l1.remove("7")
>>> l1
['Bonjour', 'tout', 'net']
```

Des pointeurs

```
>>> l3 = l1
l3.remove('net')
l3.append('le monde')
>>> l1
['Bonjour', 'tout', 'le monde']
```

Tri

```
>>> l1.sort()
['Bonjour', 'le monde', 'tout']
```

Exercices

Exercice

Écrire une fonction qui analyse une liste d'entiers t renvoie la lists des enters pairs et la liste des entiers impairs..

Exercice

Écrire une fonction qui renvoie les deux plus grands éléments d'une liste. On pourra utiliser la fonction max().

Exercice

Écrire le pendant de la fonction **assoc** *en OCaML. assoc('a',l) devra renvoyer la 2e composante du tuple dont 'a' est la 1e composante. Faire une implémentation itérative et une autre récursive.*

Exercice

Écrire le pendant de la fonction **flatten** *en OCaML qui applatit les listes de listes en listes. Par exemple flatten([8,[[2, [3, [2],4]],[[[]]],3],7]) devra renvoyer [8,2,3,2,4,3,7]. On pourra utiliser la fonction map().*

Les Dictionnaires

Exemple

Création

```
>>> d1 = {}
>>> d2 = {'host' :'titan', 'port' :22}
```

Accès / Modification

```
>>> d2['host']
'titan'
>>> d2['host'] = 'bender'
```

Taille

```
>>> len(d2)
2
```

Appartenance

```
>>> 'host' in d2
True
```

Itération

```
>>> for clef, element in d2.items() :
        print(clef, element)
port 22
host bender
7
```

Recherche de clé

```
>>> d2.has_key('uptime')
False
```

Supression

```
>>> del d2['port']
>>> d2
{'host' :'bender'}
```

Exercices

Exercice

Écrire une fonction qui échange les clefs et les valeurs d'un dictionnaire. Cas d'application : on souhaite construire le dictionnaie anglais-français à partir du dictionnaire français-anglais.

Exercice

Écrire une fonction qui donne la fréquence d'apparition des lettres d'une chaîne donnée.

Exercice

Implémentez une structure de donnée pour un labyrinthe à l'aide de tuples et de dictionnaires. Construisez éventuellement un programme de résolution de votre labyrinthe ainsi qu'un générateur de labyrinthes.

Les ensembles

Exemple

Création

```
>>> s1 = set([1,2,3])
>>> s2 = set((2,3,4))
```

Taille

```
>>> len(s1)
4
```

Union

```
>>> s1| s2
{1,2,3,4}
```

Intersection

```
>>> s1 & s2
{2,3}
```

Difference

```
>>> s1 - s2
{1}
```

Appartenance

```
>>> 2 in s1
True
```

Ajout

```
>>> s1.add(0)
>>> s1
{0, 1, 2, 3}
```

Supression

```
>>> s1.remove(2)
>>> s1
{0, 2, 3}
```

Itération

```
>>> for element in s1 :
        print(element)
0
2
3
```

Une première classe

```
class Point:
    "Point definition"
```

Instanciation

```
>>> p1 = Point()
>>> p2 = Point()
>>> p3 = p1

>>> p1.x = 2
>>> p2.x = 2
```

Une première classe

```
class Point:
    "Point definition"
```

Instanciation

```
>>> p1 = Point()
>>> p2 = Point()
>>> p3 = p1

>>> p1.x = 2
>>> p2.x = 2

# Des pointeurs
>>> p1 == p2
False
>>> p1 == p3
True

>>> p3.x = 3
>>> p1.x
3
```

Les classes : les méthodes, le constructeur

```
class Point:
    "Point definition"

    def __init__(self, x, y):
        self.x = x
        self.y = y

    def afficher(self):
        print('x:', self.x, '\ty:', self.y)
```

Utilisation

```
>>> p1 = Point(3, 2)
>>> p1.afficher()
x : 3    y : 2
```

Les classes : l'héritage

```
class Vecteur(Point):
    "Vecteur definition"

    def __init__(self, x, y):
        super(self.__class__, self).__init__(x, y)
        # ou Point.__init__(self, x, y)

    def __eq__(self, v):
        return (self.x==v.x) and (self.y==v.y)

    def __add__(self, v):
        return Vecteur(v.x+self.x, v.y+self.y)
```

Utilisation

```
>>> v1 = Vecteur(3, 2)
>>> v2 = Vecteur(3, 2)
>>> v1.afficher()
x : 3    y : 2
>>> v1 == v2
True
```

Les exceptions

```
try:
    # Bloc susceptible de lever une exception
except E1:
    # Bloc a executer si l'exception E1 est levee.
except E2:
    # Bloc a executer si l'exception E2 est levee.
except E3 as e3:
    # Bloc a executer si l'exception E3 est levee
    # e3 est un objet Exception E3 permettant d'avoir
    # des details sur l'exception .
except:
    # Bloc a executer qui rattrape les autres exceptions
else:
    # Bloc a executer si aucun exception n'a ete levee (desuet)
finally:
    # Bloc execute exception ou pas (desuet)
```

Les exceptions : un exemple

```
from math import sqrt

i = input("nombre ? ")
try:
    r = 1/sqrt(int(i))
except ZeroDivisionError:
    print("Pas de nombre nul !")
except ValueError as e:
    print ("Erreur mathematique : ", e)
except:
    print("Exception inconnue , tu as gagne le gros lot !")
else:
    print ("calcul : ",r)
```

Lever une exception

Levée une exception prédéfinie

```
>>> raise Exception("Impossible de retirer de l'argent, solde nul")
>>> raise ValueError("log10 defined on R+")
>>> raise RuntimeError("Plus de mémoire")
```

Exception Personalisée :

```
class DecouvertBancaire(Exception):
    def __init__(self, client, valeur):
        self.client = client
        self.valeur = valeur

try:
    raise DecouvertBancaire("Dupont", 420)
except DecouvertBancaire as e:
    print ('M.:', e.client, "a", e.valeur, "euros de decouvert")
```

Manipulation de fichiers

```
# Ecriture
fw = open('fichier_ou_ecrire', 'w')
fw.write("chaine␣que␣l'on␣veut␣ecrire")
fw.close()


# Lecture
fr = open('fichier␣a␣lire', 'r')
while 1:
    # Lecture de la ieme ligne du fichier
    texte=fr.readline()
    if texte == '':
        break
fr.close()
```

Lister le contenu d'un répertoire

```
from glob import glob

# Lire tous les fichiers .tex d'un repertoire
for fichier in glob(dossier_a_parcourir+"*.tex"):
    fr = open(fichier)
    ...
```

Conversions

```
# Convertir un nombre decimal en hexadecimal
hex(10)

# Convertir un nombre decimal en binaire
bin(10)

# Convertir un nombre en decimal
int('42', 8) #convertion de 42 en base 8 vers la base 10

# Obtenir le code ASCII d'un caractere
ord('e')

# Obtenir le caractere d'un code ASCII
chr(65)

# Obtenir le caractere d'un code unicode
unichr(65)
```

Opérations diverses - pot pourri

```
# Executer un programme et recuperer sa sortie
from commands import getouput
output = getoutput("hostname")

# Demander un mot de passe
from getpass import getpass
password = getpass("SSH␣password?␣")

# Connaitre le repertoire de travail
from os import getcwd
print (getcwd())
```

Questions

?

Bibliographie

Gérard Swinnen
Apprendre à programmer avec Python.
http ://inforef.be/swi/python.htm (Creative Commons)

Alexandre Drahon
Plongez au coeur de Python.
http ://diveintopython.adrahon.org (license GPL)
traduit du livre de Mark Pilgrim *Dive Into Python*