```
## Introduction a la programmation avec Python

## Eric Normandeau
## 2010 10 29


#### 0. Table des Matieres

##  I    - Introduction Generale
##  II   - Comment utiliser Python
##  III  - Types d'objets
##  IV   - Structures de controle (for, while, if)
##  V    - List comprehensions
##  VI   - Creer des fonctions et des classes
##  VII  - Copier des variables
##  VIII - Numpy
##  IX   - Lectures suggerees


## I. Introduction generale

## 1 - Qu'est-ce que Python?

## - Langage de programmation
## - Tres puissant et flexible
## - Un des plus simples a apprendre
## - Language tout-usage
## - Quantite de modules disponibles
## - Gratuit

## 2 - Qu'est-ce que ca fait?
## - Manipuler de larges fichiers
## - Recherche et modification de texte
## - Programmer applications pour serveurs et sites internets
## - A peu pres n'importe quoi

## 3 - Limitations: pas evident de:
## - Faire des graphiques
## - Operations mathematiques (surtout matricielles) moins directe que dans R
## - Faire des statistiques
## - Il existe des modules pour presque tout, mais ils demandent de l'apprentissage supplementaire


## II. Comment utiliser Python

## 0 - Repertoire courant

## Si on travaille avec des fichiers, il faut les placer dans un repertoire de travail
## pour que python sache ou les trouver. Pour cela, il y a 3 options :

## a - Creer un dossier de travail dans c:\python26\ (Solution la plus simple sous windows)
## b - Changer de dossier avant d'appeller Python
## c - Ajouter le dossier de travail dans le "PATH"

## Interface graphique :
## - Interface graphique (GUI) VS Ligne de commande

## Specificites :
## - Python IMPOSE l'indentations correcte du code
## - Convention: Utiliser 4 espaces (PAS des tabulations)
## - NE PAS utiliser a la fois des espaces et des tabulations


## 1 - Calcul : Python peut etre utilise comme une calculatrice

1+2
2*3
3**2 ## 3 a la puissance 2
3/5  ## ! Division entiere! Retourne : 0
3./5 ## Division 'float' (decimale) Retourne: 0.599999999999998
```

```
import math ## Module "math" pour fonctions mathematiques supplementaires
dir(math)  ## Affiche les fonctions disponibles dans math
help(math.log)  ## Affiche l'aide pour la fonction math.log
math.log(4,10)  ## Log de 4 en base 10
math.sqrt(13)   ## Log naturel de 13


## 2 - Assignation : on peut assigner des valeurs a des variables avec le signe "="

a = 2
b = 3
print a*b
print a/b ## Division entiere
float(a)/b ## Dividion decimale
a = 2.
print a/b ## a est un 'float', donc dividion decimale

c = 3 - 2*a + b**2
print c

d = math.log(float(b)/a, 10)    ## log() calcule le logarithme naturel
print d

## Python peut generer de tres gros nombres entiers exacts
print 9**999
print 9**81 + 2.2  ## Les nombres decimaux sont tronques


## 3 - Aide : pour chaque fonction, il existe un fichier d'aide : help()

help(math.log)
help(math.sqrt)


## 4 - Pour connaitre les fonctions dans un module

dir()
dir(math)
help(math.sin)


## III. Types de variables

## 1 - Nombres
## 2 - Chaines de characteres
## 3 - Listes
## 4 - Tuples
## 5 - Dictionnaires
## 6 - Fichiers
## 7 - Objets et tests logiques


## 1 - Nombres

a = 2   ## Nombre entier
b = 4.3 ## Nombre decimal
print a
print b

## On peut acceder au type de la variable avec la fonction type()

type(a)
type(b)


## 2 - Chaines de characteres

text = "Texte\nSur deux lignes"  ## "\n" indique un changement de ligne
raw_text = r"Texte brut\nSur deux lignes"  ## le texte 'raw' n'interprete pas "\n"
```

```
print text
print raw_text

t1 = "ABCD"
t2 = "EFGH"
t3 = t1 + t2  ## Concatenation de texte
print t3

## Methodes (fonctions propres a un type de variable)

dir(t3)  ## Affiche les methodes disponibles pour le texte

t3.lower()
print t3 ## La methode .lower() n'a pas change t3 en place
t3.find("E")
t3.replace("DE", "D_^_E") ## La methode .replace() n'a pas change t3 en place

t4 = "Saumon Contig3244 22sequences"
t5 = t4.split() ## Cree une liste contenant les mots separes par un espace

## Recherches avancees dans du texte

import re   ## Importer le module Regular Expression
dir(re)

numero = re.findall("Contig([0-9]+)", t4)

numero[0]  ## Premier element (0) de la liste cree par re.findall


## - 3 Listes
## Les listes sont des collections d'objets qui peuvent etre de plusieurs types
## (int, string, long, fonctions...)

L1 = [1,2,3,4,5]
L2 = [1, "abc", L1, math.log]

## Pour acceder aux elements, on utilise un index entre crochets

L1[0]   ## Le 1er element se trouve a l'indice 0
L1[2:4] ## On peut acceder a une tranche (slice) de la liste
L1[-1]  ## Pour atteindre le dernier element de la liste

L2[3](33)    ## On peut meme mettre une fonction dans une liste et l'utiliser
math.log(33) ## Equivalent

## Concatener des listes

L3 = L1 + L2

## Quelques methodes disponibles pour les listes :

dir(L3)

L3.append("XYZ")
L3.pop()
L3.pop()
L3.pop()

L3
L3.reverse()
L3

## Changer les objets en place

L3
L3[3] = "Remplace 4"
L3
```

```
## 4 - Tuples
## Ressemble a une liste, mais les objects a l'interieur ne peuvent pas etre changes
## Utile pour des donnees qui doivent demeurer constantes

T1 = (1,2,3)
T2 = (L1,L2)

print T1[1]

#T1[1] = 2  ## Message d'erreur, on ne peut changer l'objet une fois qu'il a ete cree



## 5 - Dictionnaires
## Contient des couples de donnees avec une clef (key) et une valeur (value)
## Utile pour retrouver des donnees rapidement, faire des decomptes...

D1 = {"Contig22":"34sequences", "Contig1212":"3434sequences","Contig224":
    "2334sequences","Contig2":"32sequences","Contig224":"534sequences",
    "Contig1":"44sequences","Contig34":"4sequences"}

D1["Contig1212"] ## Sortir la valeur associee a la clef "Contig1212"

D1["ABC"] = 123  ## Ajouter des elements
D1["ABC"]

## Quelques methodes disponibles pour les listes

dir(D1)

D1.keys()
D1.values()
D1.items()


## Deuxieme rencontrede Bioinformatique sur Pyhon

## Rappell

# Nous avons vu plusieurs types d'objects:
# - nombres
# - chaines de caracteres
# - listes
# - tuples
# - dictionnaires


## 6 - Objects et tests logiques
## Permettent de faire des tests et des operations conditionnelement au resultat du test

a = 2
a == 2  ## Est-ce que a = 2?
a < 2   ## Est-ce que a est plus petit que 2
a != 3  ## Est-ce que a est different (non egal) a 3


## IV - Stuctures de controle
# Crucial pour tous les programmes non-triviaux

## Quelques fonctions d'interet:

range(10)
range(10,20,1)
range(0,100,10)

## 1 - Conditions avec "if"

a = 2
b = 3
```

```
if a == b:
    print "a = 0"
elif a < b:
    print "a < 0"
else:
    print "a > 0"

# Note sur l'indentation
#
# L'indentation est crucial en Python.  Elle fait partie de la syntaxe
# du  lagage  meme.  La   facon  correcte  d'indenter  en  Python  est
# d'utiliser  4  espaces par  niveau  d'indentation.  IL  NE FAUT  PAS
# UTILISER DE TABULATIONS (convention) ET  IL NE FAUT PAS MELANGER LES
# ESPACES ET LES TABULATIONS (constitue une erreur).

## 2 - Boucles "for"

a = range(10)

for i in a:
    print i,

b = [22, 34, 41, 324, 234, 23, 45]

for number in b:
    c = math.sqrt(number)
    print number, c

## Imprimer la table des sommes des lances de deux des :

for i in range(1,7):
    for j in range(1,7):
        print i + j,
    print


## 3 - Boucles "while"

x = 1
while x <= 30:
    x *= 2  ## equivalent a : x = x * 2
    print x,


## 7 - Fichiers
## On peut lire et ecrire dans des fichiers a l'aide de la fonction "open"

file_in = open("example.txt", "r")

for line in file_in:
    print line.strip()

file_in.close()

# Methode appropriee

with open("example.txt") as f:
    for line in f:
        print line.strip()

# Fermeture automatique du lien vers le fichier en sortant de la
# boucle 'with'

with open("example.txt") as f:
    a = f.readlines()
    print a

with open("example.txt") as f:
    a = f.readline()
```

```
    print a
    b = f.readline()
    print b
    print "etc..."


##  V - List comprehensions

## Outil puissant et rapide de construire des listes. Ideale pour
## generer des listes a partir de larges objets en fonction de critere
## et en modifiant les valeurs


a = [1,2,3,4,5,6,7,8,9,10]

b = [x**2 for x in a]
print b


## equivalent a:
b = []
for x in a:
    b.append(x**2)

print b


c = [(x, x**2) for x in a if x%2 == 0] ## carre des nombres pairs
print c

# Example plus avance :

texte = [x.strip() for x in open("example.txt").readlines() if x.strip() != ""]


##  VI - Creer des fonctions et des classes

## Modele :

def add(num_1, num_2):
    """Documentation: cette fonction additionne par1 et par2"""
    add = num_1 + num_2
    return add

print add(2, 3.1)

def readfile(path):
    """Read file and return a list containing the file lines"""
    out = []
    with open(path, "r") as file:
        for line in file:
            out.append(line.rstrip())
    return out

def writefile(var, path):
    """Write lines from a list of character strings into a file"""
    with open(path, "w") as file:
        for line in var:
            file.write("".join([line, "\n"]))
        file.write("\n\n--> Ecrit avec la fonction writefile <--\n")

my_file = readfile("example.txt")
writefile(my_file, "output_example.txt")


##  VII - Copier des variables

import copy

a = [1, 2, 3]
```

```
b = a
c = copy.copy(a)

print a, b, c

a[2] = -99
print a; print b; print c


##  VIII - Numpy

## Vaste module (doit etre installe pour utiliser Biopython) de de
## calcul matriciel et de fonctionnalites avancees de math

import numpy
dir(numpy) ## Montre toutes les fonctions disponibles

a = numpy.array([[1,2,3],[4,5,6],[7,8,9]])
print a

b = a -22.2234
print b

c = numpy.round(b, 1)
print c

print c[0, :] ## Premiere ligne
print c[:, 0] ## Premiere colonne

ran = numpy.random.uniform
ran()
ran(0.0, 1.0, 100)

numpy.sort(ran(0.0, 1.0, 100)*100)

## Biopython

## Module pour traiter des donnees biologiques. Specialement
## interessant pour manipuler des fichiers de sequences fasta.


##  IX - Lectures suggerees :

## 1 - python tutorial (version en ligne)
## 2 - Learning Python (O'Reilly books)
```