TD M1 SPGF - Introduction à Python - Corrections

Maupetit Julien

28 septembre 2007

1 Exercice 1

– *Calculer la surface d'un disque de 10 cm de diamètre en utilisant deux variables : Pi et D, le diamètre.*

```
#!/usr/bin/python

# Formalisons les donnees du probleme
Pi = 3.14
d  = 10
r  = d / 2

# Calcul de la surface
S = Pi * r * r

# Affiche le resultat
print "Surface: %.2f cm2 (rayon: %.2f cm)." % ( S, r )
```

– *Générez une séquence polyA de 100 bases sans taper les 100 caractères.*

```
#!/usr/bin/python

# Generation de la sequence polyA
seq = "a" * 100

# Affiche le resultat
print "Sequence generee: %s" % seq

# -- Pour aller plus loin --
# -- Generation d'une sequence aleatoire --

# Importation du module random
# qui permet de generer des nombres aleatoires
import random

# Les differentes bases azotees
# que ma sequence peut contenir
bases = [ "a", "t", "g", "c" ]

# La taille de ma sequence
size  = 100

# Initialisation de ma sequence
seq   = ""
```

```
# Boucle de creation d'une sequence
for i in range(0, size, 1):

    # Nous tirons un chiffre aleatoirement
    # compris dans l'intervalle de bases
    idx = random.randint( 0, len(bases)-1 )

    # On ajoute une nouvelle base
    # seq = seq + bases[ idx ]
    # ce qui peut etre raccourci en
    seq += bases[ idx ]

# Affiche le resultat
print "Sequence generee: %s" % seq
```

2 Exercice 2

– *Calculez le pourcentage de chaque type de base dans la séquence suivante ATGCTCGCGGCGCTAGCTACTAGCTAGCA.*

Nota bene : pour faire cet exercice, vous devez avoir vu la notion de tests. Avez vous fait le tutoriel de Patrick Fuchs ?

```
#!/usr/bin/python

# La sequence a analyser
seq = "ATGCTCGCGGCGCTAGCTACTAGCTAGCA"

# La longueur de ma sequence
seqLen = len(seq)

# On defini les differents types de bases
bases = [ "A", "T", "G", "C" ]

# On initialise une liste dont chaque item
# sera l'occurence de la base du meme indice
# dans la liste bases
occurences = [0] * len(bases)

# On boucle sur la sequence pour
# determiner le nombre d'occurence
# de chaque base
for base in seq:

    # Si la base courante est connue
    # et donc contenue dans la liste bases
    if base in bases:

        # la fonction index nous permet
        # de determiner l'indice de la premiere
        # occurence d'une valeur
        # donc ici nous voulons l'indice, dans bases
        # de la base courante de la sequence que nous
        # parcourons
        idx = bases.index( base )
```

```
        # On incremente le compteur pour ce type de base
        occurences[ idx ] += 1

    # La base courante n'est pas connue
    else:
        print "La base %s n'est pas d'un type connu !" % base

# On affiche les pourcentages
for base in bases:

    # Indice du type de base courante
    idx = bases.index( base )

    # On calcul le poucentage
    # Attention, si l'on veut diviser deux entiers,
    # il faut en convertir au moins un en flottant
    # pour que python ne retourne pas un resultat de
    # type entier (donc 0 dans notre cas)
    pct = occurences[ idx ] / float(seqLen) * 100.

    # On affiche le pourcentage courant
    # la virgule de fin empeche que python
    # ne revienne a la ligne
    print "%2s:%5.1f %%" % ( base, pct ),

# Nous voulons un retour chariot a la fin de l'affichage
print ""
```

Maintenant changez la séquence en ajoutant un 'U' quelque part. Que se passe-t-il ?

Si maintenant, vous ajoutez la base 'U' à la liste des bases. Que se passe-t-il ?

– *Calculez le barycentre des 10 premiers carbones alpha de la crambine (code PDB 1crn) à partir de la liste suivante :*

Une description du format PDB (ligne ATOM) est disponible à l'adresse suivante :

http ://www.wwpdb.org/documentation/format23/sect9.html#ATOM

```
#!/usr/bin/python

# La trace (carbone alpha) des 10 premiers residus
# de la crambine, au format PDB
trace = ["ATOM      2  CA  THR     1       16.967  12.784   4.338  1.00
    10.80       1CRN  71",
 "ATOM      9  CA  THR     2       13.856  11.469   6.066  1.00  8.31
        1CRN  78",
 "ATOM     16  CA  CYS     3       13.660  10.707   9.787  1.00  5.39
        1CRN  85",
 "ATOM     22  CA  CYS     4       10.646   8.991  11.408  1.00  4.24
        1CRN  91",
 "ATOM     28  CA  PRO     5        9.448   9.034  15.012  1.00  4.25
        1CRN  97",
 "ATOM     35  CA  SER     6        8.673   5.314  15.279  1.00  4.45
        1CRN 104",
 "ATOM     41  CA  ILE     7        8.912   2.083  13.258  1.00  6.33
        1CRN 110",
```

```
 "ATOM      49  CA  VAL      8       5.145   2.209  12.453  1.00  6.93
     1CRN 118",
 "ATOM      56  CA  ALA      9       5.598   5.767  11.082  1.00  3.56
     1CRN 125",
 "ATOM      61  CA  ARG     10       8.496   4.609   8.837  1.00  3.38
     1CRN 130"]

# Remarque:
# le barycentre d'un ensemble de coordonnees est la position
# moyenne ce cet ensemble. Nous allons donc calculer une moyenne
# sur les X, Y et Z de ces 10 carbones alpha
sumX = 0.
sumY = 0.
sumZ = 0.

# Pour chaque item de ma liste trace
# (pour chaque carbone alpha donc)
for CA in trace:

    # On recupere les coordonnees x,y,z dans la ligne
    # du fichier PDB
    x = float( CA[30:37] )
    y = float( CA[38:45] )
    z = float( CA[46:53] )

    # On somme les X, Y, et Z
    sumX += x
    sumY += y
    sumZ += z

# On calcul la position moyenne
baryX = sumX / len(trace)
baryY = sumY / len(trace)
baryZ = sumZ / len(trace)

# Affiche le resultat
print "Coordonnees du barycentre: %8.3f %8.3f %8.3f" % ( baryX, baryY,
    baryZ )

# Le resultat qui doit apparaitre est:
# Coordonnees du barycentre:   10.135    7.291   10.746
```