Formulaire Python en BCPST

Types et opérations

Chaîne de caractère

```
prenom = input('Quel est votre prenom ?')
message = 'Votre prenom est ' + prenom
print('Votre prenom est', prenom, sep=' ')
# commentaire dans le code
```

Transtypage

On créé une variable de type voulu en faisant : variable2 = type_voulu(variable1)

```
entier = 2
flottant = float(entier)
chaine = str(flottant)
entier = int(chaine)
type(entier)
```

Opérations de base

+, - addition, soustraction

***, /** multiplication, division

****** puissance

% reste de la division entière

Booléens (True/False)

==, != teste l'égalité ou la différence

<, >, <=, >= teste l'ordre

in teste l'appartenance à une liste

```
"A" in "chAton"    # renvoie True
2 in [1,3,8]       # renvoie False
```

not() négation d'un booléen

and, or relations logiques

Les listes

Indices d'une liste/chaîne de caractère :

```
liste, chaine = [1,2,3], "mot"
print(liste[1])     # affiche "2"
print(chaine[0:3]) # affiche "mot"
print(liste[:2])    # affiche "[1,2]"
print(chaine[1:])   # affiche "ot"
```

Informations sur la liste :

```
len(liste)
sum(liste)
```

Méthodes de modification :

```
liste.append(9)
liste.extend([0,5,3])
# insere 5 a la position 2
liste.insert(2, 5)
liste.sort()     # trie
liste.reverse() # retourne
liste.count(1)  # nombre de valeurs 1
liste.index(1)  # index de la valeur 1
# supprime le 1er element de valeur 2
liste.remove(2)
# sort l'element d'indice 2
liste.pop(2)
```

Pour copier une liste et ne pas faire un alias :

```
liste2 = list(liste1).
```

Obtenir une liste à partir de *range* :

```
entierspairs = list(range(0,20,2))
```

Manipulation compacte des listes :

```
liste = [1,2,3,4,5,6,7,8]
pairs = [x for x in liste if x%2 == 0]
carre = [x**2 for x in liste]
```

Structure du code

Condition

```
if booleen1:
    execution1
elif booleen2:
    execution2
else:
    execution3
```

Boucle while

```
while booleen:
    execution
```

Boucle for

Avec range()

range(start, stop, step) va de la valeur start (inclue), à la valeur stop (exclue) par pas step. Par défaut, range(n) fait prendre les valeurs entières de 0 à $n-1$.

```
for i in range(1, n+1):
  print("On en est au terme " + str(i))
```

Avec une liste/chaîne de caractères

```
liste = [2, 4, 10, 7, 8]
chaine = "python"
for valeur in liste:
  print(valeur)
for lettre in chaine:
  print(lettre)
```

Définir une fonction

```
def moyenne(a,b):
    return (a+b)/2.0
```

Importer une bibliothèque

```
# Importer et changer le nom
import matplotlib as plt
# Utiliser un sous-module uniquement
from numpy import random
# Tout importer (non-recommande)
from pylab import *
```

Module Numpy

Initialisation de vecteurs et matrices

A partir d'une liste (ou d'une liste de listes) :

```
a = np.array([2,4,6])
b = np.array([[1,2,3],[7,8,9]])
```

Fonctions de séquences numériques :

```
a = np.arange(start, stop, step)
b = np.linspace(start, stop, size)
c = np.ones(10)
```

Initialiser une matrice :

```
# 6 lignes et 2 colonnes de 0
A = np.zeros((6,2))
# matrice identite 10*10
I = np.eye(10)
```

Opérations

Mathématiques

(+,-,*,/) se font élément par élément. Ex : $\text{dot}(A,B)$ fait le produit matriciel, $A*B$ multiplie terme à terme.

np.exp(), np.log() fonctions exp et ln.

np.sin(), np.cos(), np.arccos(), ... trigo.

np.pi la valeur π

np.sqrt() racine carrée.

np.ceil(), np.floor() entier supérieur, inférieur.

np.round(x, 2) arrondi de x à la 2eme décimale

Acceder aux éléments

Les indices de lignes et colonnes commencent à 0.

```
vligne = A[i,:]     # ligne d'indice i
vcolonne = A[:,j]   # colonne d'indice j
element = A[i,j]    # element d'indice i,j
```

Methodes pratiques

A.shape renvoie (6,2) ici

A.sum() somme des termes

A.min(), A.max() min et max

Simulation de nombre aléatoire

Dans le sous-module *random*.

np.random.random(size) renvoie un vecteur de longueur size, avec des réalisations de variable uniforme sur $[0,1[$

np.random.binomial(n, p, size) loi $\mathcal{B}(n,p)$

np.random.geometric(p, size) loi $\mathcal{G}_{\mathbb{N}^*}(p)$

np.random.exponential(λ, size) loi $\mathcal{E}(\frac{1}{\lambda})$

np.random.normal(μ, σ, size) loi $\mathcal{N}(\mu,\sigma)$

Module Scipy

Package qui comprennant :

scipy.integrate quad(), odeint(), ode().

scipy.optimize avec minimize(), leastsq(), curve_fit().

scipy.stats distributions de v.a..