INF 2 - Introduction à Python (2)

1. QUELQUES TYPES SUPPLÉMENTAIRES

1.1. Les chaines de caractères

Le type **str** (pour *string* ou *chaîne de caractère*). Les caractères peuvent être des lettres, des symboles, des chiffres. On crée ou affecte une chaine de caractère en l'encadrant de guillemets ou d'apostrophes.

```
a = "bonjour"
b =","
c =" les ECG sont "
d ="les meilleurs."
```

Remarque 1.1 — Les caractères de la chaînes sont numérotés à partir de 0. Ainsi b est le caractère de place 0 de a. En toute généralité, une chaines de n caractères verra ses caractères numérotés de 0 à $n-1$. Différentes méthodes sont associées aux chaines de caractère :

- **la concaténation** : `a+b+c` renvoie la chaîne de caractère `"bonjour, les ECG sont "`,
- **l'extraction de sous-chaîne** `d[2:5]` renvoie la sous chaîne des caractères de `d` qui sont compris entre 2 et 4, donc `"s m"`,
- **la longueur** : `len(c)` renvoie le nombre de caractères de `c`, c'est à dire l'entier 14 (attention aux espaces),
- **le test d'une sous chaîne** : `"ECG" in c` renvoie un booléen qui vaut True si et seulement si ECG est une sous-chaîne de `c`.

Exemple 1 —

1. que donne `a+b+c+d`?
2. que donnent `"ecg" in c` et `"ECG" in c`?
3. calculer les longueurs des différentes chaînes.

Il existe d'autres méthodes intéressantes, les methodes de conversions entre chaînes et autres types.

- si la chaîne contient uniquement un flottant ou un entier, on peut utiliser les commandes **int** ou **float** pour récupérer l'objet dans le type correspondant.
- si on a un entier ou un flottant, la commande **str** le transforme en chaîne de caractère.

On peut tester les commandes suivantes

```
a = str(5)
b = str(5) + str(5)
b2 =  int(str(5) + str(5))
c = str(5+5)
z = float("5.5")
y = int(float("5.5"))
t= str(int(float("5.5")))
```

Attention

Il n'y a pas de méthode pour modifier directement un caractère donné de la suite. On peut quand même s'en sortir en rusant un peu. Que fait le programme :

```
chaine = "ABDDEFGHIJKLMNOPQRSTUVWXYZ"
chaine = chaine[0:2]+"C"+chaine[3:26]
print(chaine)
```

1.2. Les listes

Attention

Python ne duplique pas les listes. Il leur donne une deuxième adresse. Par exemple le code suivant montre que si on "copie" une liste et qu'on change un terme de la copie, cela change aussi la liste de base.

```
1 M = [0,1,2,3,4,5]
2 N = M
3 N[0] = 1000
4 print(M)
```

Pour dupliquer une liste, on peut plutôt faire ainsi

```
M = [0,1,2,3,4,5]
N = M[:]
```

ou

```
M = [0,1,2,3,4,5]
N = L[0:len(M)]
```

2. INSTRUCTIONS CONDITIONNELLES

Si on veut coder une fonction toute simple mais qui n'est pas définie complètement explicitement, comme

$$f : x \in \mathbf{R} \mapsto \begin{cases} 1 \text{ x } < 0 \\ e^x \text{ si } x \geqslant 0 \end{cases}$$

on peut avoir besoin d'une disjonction de cas. Les commandes suivantes permettent de le faire

- **if** : c'est la commande *si*, elle est suivie d'une expression qui rend un booléen. Si ce booléen vaut True, on effectue la suite des instruction sinon ...
- **else** : c'est la commande *sinon* qui nous dit ce qu'on fait ensuite si la condition n'est pas vérifiée.

L'exemple suivant nous permet de présenter la syntaxe.

```
1 import numpy as np
2
3 def F(x):
4     if x < 0:
5         return 1
6     else:
7         return np.exp(x)
```

Remarque 2.1 —

- Remarquez bien où sont situés les " :" à chaque fois!
- L'indentation est très importante (on s'en rendra compte quand il y aura beaucoup de tests) : tous les connecteurs de la structure conditionnelles doivent être indentés au même endroit!

Si on a plus de deux cas à traités, on est amenée à utiliser la commande **elif** (contraction de **else** et **if**) qui permet de faire un autre test.

Exemple 2 — Que fait le programme suivant?

```
def maximum(a,b):
    if a < b:
        print("le plus grand nombre est le deuxieme", b)
    elif b< a:
        print("le plus grand nombre est le premier", a)
    else:
        print("les deux nombres sont égaux à",a )
```

3. BOUCLES FINIES (FOR)

3.1. La commande Range.

La commande **range** produit un ensemble d'entiers qu'un **boucle** va parcourir à l'aide d'un indice muet.

- **range(n)** crée la liste des entiers de 0 à $n-1$,
- **range(a,b)** crée la liste des entiers de a à $b-1$,
- **range(a,b,r)** crée une progression arithmétique d'entiers de premier terme a, de raison r et qui s'arrête avant b. Par exemple **range(0,15,2)** renvoie la liste des entiers pairs de 0 à 14.

Exemple 3 — Pour visualiser ces ensembles d'entier, on peut les transformer en liste avec la commande `list`. Taper les commandes suivantes :

```
list(range(12))
list(range(12,22)
list(range(5,0))
liste(range(0,15,2))
list(range(20,2,-1))
```

3.2. La boucle for

La boucle permet de répéter un bloc d'instruction un nombre fini de fois. Le format d'une boucle est le suivant.

```
import numpy as np

for k in range(n):

        bloc instructions qui peuvent dependre de k
        bloc instructions

```

Remarque 3.1 —

1. La variable **k** va parcourir la liste **range(n)** c'est à dire tous les entiers entre 0 et n-1. Au premier passage dans le bloc d'instructions, **k** vaut 0. Au deuxieme, la variable vaut 1 et ainsi de suite. Au n-ieme tour, la variable vaut n-1.
2. Si on voulait faire varier **k** de 1 à n, on aurait pu l'appeler dans **range(1,n+1)**. Mais on rappelle qu'en informatique on commence souvent à compter à partir de zéro.
3. Attention à l'indentation et aux deux points.

Exemple 4 — Que font les programmes suivants ?

```
res = 0
for k in range(100):
    res += k
print(res)
```

```
def F(n):
    res = 1
    for i in range(n):
        res = res*(i+1)
    return (res)
```

3.3. Exemples à connaître

Étude d'une suite récurrente. Soit (u_n) la suite définie par

$$u_0 = -2 \text{ et } \forall n \in \mathbf{N}, u_{n+1} = u_n^2 - u_n.$$

On souhaite écrire une fonction prend un entier n comme argument et renvoie u_n.

```
def U(n):
    # entrée : un entier n
    # sortie : le terme Un

    res = -2#on initialise le resultat à u0 = 1
    for k in range(n): #on va calculer n nouveaux termes
        res = res*res - res #on remplace un par u(n+1)
    return (res)
```

On peut aussi modifier légèrement le code pour que la fonction affiche tous les termes de la suite qui sont calculés, ou un terme sur deux, etc.

```
def U(n):
    # entrée : un entier n
    # sortie : le terme Un

    res = -2#on initialise le resultat à u0 = 1
    for k in range(n): #on va calculer n nouveaux termes
        res = res*res - res #on remplace un par u(n+1)
        print(res) #affiche le terme calculé
    return (res)
```

Exemple 5 — Modifier le code précédent pour qu'il affiche uniquement un terme sur deux. On utilisera un test sur l'entier k.

Somme des termes d'une suite. Calculs de somme. On peut aussi légèrement modifier les codes précédents pour obtenir non pas le terme u_n mais la somme $\sum_{k=0}^{n} u_k$.

```
def Somme(n):
    # entrée : un entier n
    # sortie : la somme Sn = u0 + u1 + u2 + un

    somme = -2 #on initialise le calcul de la somme
    res = -2#on initialise le calcul de la suite à u0 = 1
    #dans cette fonction c'est une variable auxiliaire
    for k in range(n): #on va calculer n nouveaux termes
        res = res*res - res #on remplace un par u(n+1)
```

```
10          somme += res #on ajoute le terme de la suite à la somme
11      return (somme)
```

Exemple 6 — Modifier légèrement la fonction Somme pour sommer uniquement sur les entiers impairs.

EXERCICES

Exercice 1 Écrire un programme qui prend en entrée trois flottants a, b, c et renvoie une liste contenant les racines réelles du polynôme $ax^2 + bx + c$. (S'il n'y en a pas, la réponse attendue est une liste vide.)

Exercice 2 Combien y a-t-il d'entiers dans chacun des listes d'entiers suivantes ?

```
range(100)
range(5)
range(1,50)
range(6,20)
range(0,2,40)
range(0,3,25)
```

Que se passe-t-il si on veut boucler sur

```
range(10,1)
range(20,2,0)
```

Exercice 3 **Exemple à connaître.**

Coder une fonction qui prend un entier n et renvoie $n!$.

Exercice 4 Soit $a \in \mathbf{R}_+$ et u_n la suite définie par

$$u_0 = 1 \text{ et } u_{n+1} = \frac{u_n + \frac{a}{u_n}}{2}.$$

Écrire une fonction qui prend un entier un réel a et un entier n puis renvoie le n-ième terme de la suite u_n et affiche tous les termes précédents. Tester la fonction pour $a = 4$, $a = 16$. Faire une conjecture sur cette fonction.

Exercice 5 Écrire une fonction qui prend un entier n en entrée et renvoie la somme

$$\sum_{k=0}^{n} k^2.$$

Exercice 6 A l'aide d'une boucle for, coder une fonction qui prend un entier n en entrée et renvoie F_n où F_n est le n-ième terme de la suite de Fibonacci définie par

$$F_0 = F_1 = 1 \text{ et } \forall n \in \mathbf{N}, F_{n+2} = F_{n+1} + F_n.$$