TP 1 - Objets, opérations et structures

I. Objets manipulés, opérations élémentaires

Le shell Python permet de faire des opérations numériques élémentaires.
➜ appuyez sur la touche *entrée* à la fin de chaque ligne :

À écrire dans le shell

```
>>> 2*3+4
>>> 3**2
>>> 1/2+1/3+1/6
```

Par défaut, Python ne connaît pas les fonctions usuelles exp, sqrt, log (logarithme nepérien)...ou les constantes usuelles comme le nombre pi : les lignes suivantes retournent un message d'erreur !

À écrire dans le shell

```
>>> pi
>>> sqrt(3)/2
```

⇛ Fonction usuelles

Les fonctions usuelles sont définies dans des modules complémentaires, comme le module math ou les modules numpy ou scipy.
Tester les commandes suivantes :

À écrire dans le shell

```
>>> import math
>>> pi
>>> math.pi
>>> math.sqrt(3)/2
>>> math.sin(pi/3)
>>> math.sin(math.pi/3)
```

Remarque : Importance du préfixe du module
Il convient d'utiliser le préfixe du module pour accéder aux fonctions ou aux constantes définies par ce module. Il est possible d'utiliser une abréviation pour simplifier l'utilisation d'un module. L'abréviation standard pour le module numpy est np, pour le module scipy : sp.

À écrire dans la shell

```
>>> import numpy as np
>>> np.pi
>>> np.sqrt(3)/2
>>> np.sin(np.pi/3)
```

Calculs

Exercice 1 Pré-calculer le résultat mentalement, puis vérifier le résultat avec le shell :

```
>>>(2-3)/2+0.5
?
>>>6/3*2
?
>>>2/1+1/2
?
>>>2/(1+1)/2
?
>>>(2/1+1)/2
?
>>>2/(1+1/2)
?
```

Exercice 2 *Calcul avec des nombres complexes*
Testez les commandes suivantes :

À écrire dans la shell

```
a=1+1j
a**2
a**3
a.real
a.imag
abs(a)
a.conjugate()
np.exp(a)
b=complex(3,4)
b.real
b.imag
abs(b)
b/a
```

Exercice 3 *Manipulation de caractères*
Testez les commandes suivantes :

À écrire dans la shell

```
mot="bonjour"
mot[0]
len(mot)
mot[6]
mot[2:5]
mot[:2]
mot[2:]
mot+"␣Patrick␣:-)"
```

Affection

Exercice 4 Compléter en donnant le résultat attendu :

```
>>> a=2;a=a+2;a=a*a;a=a-(a/4);a
?
>>> a=2;b=3;a=(b-a)*a;b=a**2*b;b
?
```

Exercice 5 *Échange de variable*
Donner des instructions permettant d'échanger le contenu de deux variables (écrire le script dans une feuille) :

1. Avec des affectations simultanées :

```
1 a=3;b=4;
2 ?
3
4
5 print(a,b)
```

2. Sans affectation simultanée :

```
1 a=3;b=4;
2 ?
3
4
5 print(a,b)
```

Variables booléennes

Exercice 6 Un nombre nb est demandé à l'utilisateur.

```
>>> nb=float(input('Donner un nombre : '))
```

Donner une instruction qui retourne True (False sinon) si :

- le nombre est supérieur à 12 :

```
>>> ?
```

- le nombre appartient à $[2,4[$:

```
>>> ?
```

- le nombre appartient à $[2,4[\cap]3,+\infty[$:

```
>>> ?
```

- le nombre appartient à $[2,4[\cup\{-1,0\}$:

```
>>> ?
```

- le nombre n'appartient pas à $[2,4[\cup\{-1,0\}$:

```
>>> ?
```

Exercice 7 Compléter en donnant le résultat attendu :

```
>>> a=3.1;
>>> (a>2 and a<3) or a<0
?
>>> (a+1<5 or a<0) and (not a<0)
?
>>> (False or True) and (True or True)
?
>>> not (True or False)
?
>>> (False and True) or (False or True)
?
```

Création de scripts

Exercice 8 Créez puis exécutez un fichier script dans lequel vous aurez recopié :

```
print("Premiere boucle")
for i in range(5):
    print(i)
print("Deuxieme boucle")
for i in range(2,5):
    print(i)
```

Exercice 9 Créez puis exécutez le script suivant qui calcule et affiche les termes u_n pour n variant entre 0 et 10 de la suite définie le relation de récurrence suivante :

$$\begin{cases} & u_0 = 0 \\ \forall n \in \mathbb{N}^* & u_n = u_{n-1} + \frac{1}{n} \end{cases}$$

```
u=0
for i in range(1,11):
    u=u+1/i
    print("u(",i,")= ",u)
```

Exercice 10 Écrire un script qui calcule et affiche la suite $(n!)_{n\in\mathbb{N}}$ pour n variant de 1 à 30.

II. Strutures répétitives : for

Affichage

Exercice 11 Écrire un script qui demande une base et affiche la table de multiplication de 1 à 12 dans cette base.

Par exemple, pour la base 8, on obtient :

```
8 * 1 = 8
8 * 2 = 16
8 * 3 = 24
...
8 * 12 = 96
```

➜ La méthode **.format()** s'applique à une chaîne de caractères dans la quelle on incorpore des {} aux endroits des valeurs que l'on met en argument de la fonction.

```
n,d=8,3
print('Le reste de la division de {} par {}
                        est {}'.format(n,d,n%d))
```

Cela donne :

```
Le reste de la division de 8 par 3 est 2
```

De plus, on peut spécifier des options d'affichage des valeurs :

	Explication
{:>n}	Aligné à droite sur n caractères
{:<n}	Aligné à gauche sur n caractères
{:^n}	Centré sur n caractères

Exercice 12 Reprendre l'exercice ci-dessus en utilisant la méthode `.format()` afin d'aligner les symboles =.

```
8 * 1  = 8
8 * 2  = 16
8 * 3  = 24
...
8 * 12 = 96
```

Exercice 13 On définit une suite $(u_n)_{n\in\mathbb{N}}$ par :

$$\begin{cases} & u_0 = 0 \\ \forall n \in \mathbb{N}^* & u_n = n + 10 \times u_{n-1} \end{cases}$$

Ainsi $u_0 = 0$, $u_1 = 1$, $u_2 = 12$, $u_3 = 123$ et ainsi de suite jusqu'à $u_9 = 123456789$.

1. Écrire un script qui calcule et affiche u_n pour tout $n \in [\![0,9]\!]$
2. Écrire un script qui affiche :

Identités remarquables

```
8 * 1         + 1 = 9
8 * 12        + 2 = 98
8 * 123       + 3 = 987
8 * 1234      + 4 = 9876
8 * 12345     + 5 = 98765
8 * 123456    + 6 = 987654
8 * 1234567   + 7 = 9876543
8 * 12345678  + 8 = 98765432
8 * 123456789 + 9 = 987654321
```

Exercice 14 En utilisant la suite $(u_n)_{n\in\mathbb{N}}$ définie par :

$$\begin{cases} & u_0 = 0 \\ \forall n \in \mathbb{N}^* & u_n = 1 + 10 \times u_{n-1} \end{cases}$$

écrire un script qui affiche :

Identités remarquables (suite)

```
       1*1       =         1
      11*11      =       121
     111*111     =     12321
    1111*1111    =   1234321
   11111*11111   =  123454321
  111111*111111  = 12345654321
 1111111*1111111 = 1234567654321
11111111*11111111 = 123456787654321
111111111*111111111 = 12345678987654321
```

Calcul de sommes, de produits

Exercice 15 *Compléter/corriger* Le script suivant devrait afficher la somme :

$$S = \sum_{k=2}^{n} \frac{1}{k^3}$$

Le compléter, le corriger.

Compléter - corriger

```
1 n=...
2 s=1 # initialisation
3 for i in range(2,n):
4     s=1/n^3
5     print('Pour n =',n,'on a S =',s)
```

Exercice 16 Qu'affiche le programme suivant :

```
1 import numpy as np
2 n=int(input('Donner n = '))
3 p=1
4 for j in range(1,n):
5     p*=np.exp(1/j/(j+1))
6 print(p)
7 print(1/(1-np.log(p)))
```

Exercice 17 Pour chaque expression suivant, donner un script qui la calcule et vérifie, le cas échéant, l'expression donnée :

$$R = \prod_{k=1}^{23} \sin(k^2) \qquad R \approx 3.05e-07$$

$$S_n = \sum_{1\leq 2k+1\leq n} 2k+1 \ (n \text{ impair}) \qquad S_n = \frac{(n+1)^2}{4}$$

$$T_n = \sum_{1\leq 2k+1\leq n} \frac{(-1)^k}{2k+1} \qquad 4T_n \to \pi$$

$$U_n = \sum_{k=0}^{n-1} (2k+1)^2 \qquad U_n = \frac{n(4n^2-1)}{3}$$

$$V = \sum_{k=1}^{123} \prod_{j=k}^{k+2} \frac{k^2+1}{k*j} \qquad V \approx 114.004$$

Calcul des termes d'une suites

Il n'y a pas de variable indicée en Python. En général, c'est la même variable qui sert à stocker successivement les différentes valeurs des termes d'une suite.

Méthode : Lorsque la suite est vectorielle, ou récurrente d'ordre au moins 2, penser à utiliser des variables auxiliaires pour effectuer les calculs intermédiaires. Un tableau de suivi des variables est pertinent pour comprendre ce qui est fait à chaque étape.

Exercice 18 Que fait le programme suivant :

```
1 u=2
2 for i in range(25):
3     u=np.cos(u)
4 print(u)
```

Exercice 19 Que fait le programme suivant ?

```
1 a,b=1,2
2 for i in range(12):
3     a,b=b,a+b
4 print(b)
```

III. Structures répétitives et conditionnelles

Calcul des termes d'une suites

Exercice 20 Donner un script qui affiche b_{12} défini par : $a_0 = 1$, $b_0 = -2$, $c_0 = 0$

$$\forall n \in \mathbb{N}, \quad \begin{cases} a_{n+1} = a_n - 2b_n + c_n \\ b_{n+1} = a_n - c_n \\ c_{n+1} = 2a_n - b_n - c_n \end{cases}$$

Réponse : 32

Exercice 21 Donner un script qui affiche le terme de rang n de la suite définie par

$$\begin{cases} u_0 = 2 \\ \forall n \in \mathbb{N},\ u_{n+1} = \sqrt{n + u_n} \end{cases}$$

Réponse : $u_{12} = 3.8309...$

Structure conditionnelle

Exercice 22 *Compléter/corriger* Le script suivant doit afficher `True` si l'entier entré est pair et `False` sinon.

Compléter - corriger

```
n=input('Donner une nombre : n = ')
if ...
    print(True)
else:
    print('False')
```

Exercice 23 Créer un script qui demande un nombre x et calcul

$$f(x) = \begin{cases} \exp(x) & \text{si } x < 0 \\ 1 & \text{si } x \in [0,1] \\ \exp(x-1) & \text{si } x > 1 \end{cases}$$

Exercice 24 Donner un script qui demande trois entiers et retourne le plus petit.

Structure répétitive : `while`

Exercice 25 *Suite de Syracuse*

1. Compléter le script suivant qui calcul et affiche les termes de la suite de Syracuse définie par son premier terme $u_0 \in \mathbb{N}^*$ et le relation de récurrence :

$$\forall n \in \mathbb{N}, \quad u_{n+1} = \begin{cases} \dfrac{u_n}{2} & \text{si } u_n \text{ est pair} \\ 3u_n + 1 & \text{sinon} \end{cases}$$

jusqu'à ce que $u_n = 1$.

A compléter

```
print('Suite de Syracuse')
u=int(input('Donner u0 : '))
while u!=1:
    if u%2==0: ...
    else: ...
    print(u)
```

2. Maintenant, modifier le script pour qu'il affiche le rang du premier terme qui prend la valeur 1.

Exercice 26 *Série harmonique* Pour $n \in \mathbb{N}^*$, $H_n = \sum_{k=1}^{n} \frac{1}{k}$.

1. Écrire un script calculant H_n où n est demandé à l'utilisateur.

2. La suite $\left(\sum_{k=1}^{n} \frac{1}{k} \right)$ diverge vers $+\infty$.
Donner le rang à partir duquel $H_n \geq 12$.

Réponse 91380