Correction TP1 Python ECG2

Exercice 1

a=1024 est de type int

b=6.8 est de type float c=503.0 est de type float, même si c'est un entier. En effet, en Python, la division / renvoie toujours un nombre de type float

d=503, cette fois-ci, c'est de type int, car on calcule le quotient de la division euclidienne de 1515 par 5

e=2 est de type int (reste de la division euclidienne de 37 par 5)

f=False puisque l'inégalité $4 < 3$ est fausse. f est de type bool

g=True puisque l'égalité 1*2*3*4*5=120 est vraie. g est de type bool.

Exercice 2

C'est un programme avec des affectations parallèles.

A l'issue de la première ligne, on a : a=1,b=2 et c=3.

Puis, pour la deuxième ligne, a prend la valeur de c^2, donc a=9 en même temps que c prend la valeur de b-2a (en prenant l'ancienne valeur de a). Donc c=0. A l'issue du programme, on a donc : a=9, b=2 et c=0.

Exercice 3

On obtient successivement :

x=10, y=15, z=25, x=15, y=25, puis x+y+z=15+25+25=65.

Python affiche alors 65.

Exercice 4

Les deux programmes sont identiques au niveau des conditions des tests. Mais, l'un comporte 3 tests `if...`, alors que l'autre comporte un test `if..elif...else...`, cela fait une différence !

Dans le premier programme :

x=1 vérifie la condition du premier test donc x augmente de 5 et vaut 6.

x=6 vérifie la condition du deuxième test donc x diminue de 2 et vaut 4.

x=4 vérifie enfin la condition du troisième test donc x est doublé et vaut 8.

Le programme un affiche 8.

Dans le deuxième programme :

x=1 vérifie la condition du premier test donc x augmente de 5 et vaut 6.

Puis, le programme sort du bloc de tests et affiche 6.

Avec `if..elif...else...`, on n'exécute la condition qu'une seule fois !

Exercice 5

1)Pour tout $i \in \{0, 1, 2, 3, 4\}$, on affiche i^2, c'est-à-dire 0,1,4,9,16.

> Remarque
> Attention au décalage : range(n) est un entier entre 0 et n-1.

2)L'instruction c=2*c permet de construire une suite géométrique de raison 2 (de premier terme 1 du fait de l'initialisation).

Python affiche donc 1,2,4,8,16,32,64 puis s'arrête car la valeur suivante 128 dépasse 100.

Exercice 6

1)On peut utiliser l'affectation parallèle : a,b,k=1,1,0

2)Après passage dans l'affectation parallèle, on obtient : a=21, b=34 et k=7.

3)a)On obtient : $u_0 = 1$, $u_1 = 1$, $u_2 = 2$, $u_3 = 3$, $u_4 = 5$, $u_5 = 8$, $u_6 = 13$, $u_7 = 21$, $u_8 = 34$, $u_9 = 55$ et $u_{10} = 89$.

b)A partir de deux termes consécutifs a et b, on construit le suivant en faisant la somme a+b.
On obtient donc : $u_{n+2} = u_{n+1} + u_n$, c'est la suite de Fibonacci.

Exercice 7

1)programme :

```
def f(x):
    if x<=0:
        return 0
    elif 0<x and x<1:
        return x
    else:
        return 1
```

2)Il suffit d'aller sur la console et de faire un appel de fonction.
On tape f(-1), ce qui donne 0.
Puis, f(0.7), ce qui donne 0.7 et enfin f(2) qui donne 1.

Exercice 8

1)Programme :

```
n=int(input("entrer n"))
u=10
for k in range(1,n+1):
    u=0.5*u-3
    print(u)
```

> Pour des entiers avec a<b, on rappelle que range(a,b) représente tous les entiers entre a et b-1.
> k prend donc ici les n valeurs 1,2,...,n.
> Ce qui compte, c'est que la boucle soit faite n fois. Donc on aurait très bien pu mettre range(n), ce qui donnait un k entre 0 et n-1.

2)On peut conjecturer que $\lim\limits_{n\to+\infty} u_n = -6$.

3)On modifie le programme ci-dessus, en mettant une boucle while.

```
u=10
n=0
while abs(u+6)>10**-3:
    u=0.5*u-3
    n=n+1
print(n)
```

Exercice 9

Programme :

```
n=int(input("entrer n"))
s=0
for k in range(1,n+1):
    s=s+k**3
print(s)
```

Exercice 10

1)Programme :

```
s=0
for k in range(1,2026):
    if k%3==0:
        s=s+k
print(s)
```

On obtient 684450.

2)Programme :

```
s=0
for k in range(1,2026):
    if k%3==0 or k%5==0:
        s=s+k
print(s)
```

On obtient 957825.

Exercice 11

1)Programme :

```
u=2026
for k in range(1,201):
    if u%2==0:
        u=u/2
    else:
        u=3*u+1
    print(u)
```

On obtient : u_1 =1013, u_2 = 3040, u_3 = 1520, u_4 = 760, etc...

A un moment donné, l'un des termes vaut 4.
La suite va alors faire des cycles avec les valeurs 4,2,1.

2)En prenant une autre valeur de u_0, on constate encore à partir d'un certain rang, la présence du cycle 4,2,1.

3)On utilise une boucle while. Python affiche n=112.

```
u=2026
n=0
while u!=1:
    if u%2==0:
        u=u/2
    else:
        u=3*u+1
    n=n+1
print(n)
```

4)Programme :

```
u=2026
v=u
while u!=1:
    if u%2==0:
        u=u/2
    else:
        u=3*u+1
    if u>v:
        v=u
print("altitude=",v)
```

Exercice 12

1)Programme :

```
def factorielle(n):
    f=1
    for k in range(1,n+1):
        f=f*k
    return f
```

2)Programme :

```
def sommechiffres(n):
    s=0
    while n>0:
        r=n%10
        s=s+r
        n=n//10
    return s
```

Par exemple, prenons n=258.
La division euclidienne de n par 10 donne un reste de 8 et un quotient de 25.
A la fin de boucle, on a donc r=8, s=0+8=8 et n=25.
Puis, on recommence la boucle avec n=25 et on effectue la division euclidienne de 25 par 10, ce qui donne r=5, s=8+5=13 et n=2.
On effectue une troisième fois la boucle avec n=2 et on effectue la division euclidienne de 2 par 10, ce qui donne : r=2, s=13+2=15 et n=0.
Comme n=0, on sort de la boucle. La fonction retourne alors 15.

3)Sur la console, on tape : sommechiffres(factorielle(1000)).
On obtient 10539.