TP 0 : Corrigé

3 À vous de jouer !

Exercice 1. Prévoir le type des expressions suivantes, et le résultat (vous pouvez taper dans la console pour vérifier).

1. `4+2**3.0` Flottant, `12.0`. (Opérations mélangeant entiers et flottants : flottant)
2. `2>=3 or 2**4==16` booléen, `True`
3. `4>3 and 3>4 or True` booléen, `True` (`and` prioritaire sur `or`)
4. `5*5%3` Entier. En fait, `%`, `*` et `//` ont la même priorité ! Opérations évaluées de gauche à droite : `1`
5. `not False and False` booléen, `not` prioritaire donc `False`
6. `int(8.6)+2` Entier, `12`. `int` convertit en entier en tronquant la partie décimale.
7. `float(2)**3` Flottant, `8.0`. `float` convertit en flottant.

Exercice 2. *Expressions booléennes.*

1. Affecter 4 valeurs numériques aux variables `a`, `b`, `c` et `d`. Écrire un morceau de code affectant à la variable `x` un booléen, valant `True` si et seulement si `a` est supérieur ou égal aux autres valeurs. Tester (rajouter `print(x)` après l'expression définissant `x`), en changeant éventuellement les valeurs de `a`, `b`, `c` et `d`.

```
x = a>=b and c>=b and d>=b
```

ou de manière équivalente :

```
if a>=b and c>=b and d>=b:
    x=True
else:
    x=False
```

2. Prévoir la valeur de `y` à la fin du code suivant, en fonction de sa valeur de départ.

```
y = ... #un entier
if y>5:
    y=y+2
elif y%2==0:
    y=y-1
```

Si y est strictement supérieur à 5, on lui rajoute deux. S'il est inférieur à 5 est pair, on lui enlève 1. Sinon, il ne se passe rien !

Exercice 3. *Quelques boucles `for`.* Voici les calculs de $\sum_{i=1}^{9999} \frac{1}{i^2}$, $\sum_{i=0}^{9999} \frac{(-1)^i}{i+1}$ et $\sum_{i=0}^{9999} \frac{(-1)^i}{2i+1}$

```
s1 = 0
for i in range(1,10000): #de 1 à 9999
    s1 = s1 + 1/i**2

s2 = 0
for i in range(10000): #de 0 à 9999
    s2 = s2 + (-1)**i/(i+1)

s3 = 0
for i in range(10000): #de 0 à 9999
    s3 = s3 + (-1)**i/(2*i+1)
```

On vérifie « à l'œil » que ces sommes sont très proches des réels proposés.

Exercice 4. *La conjecture de Syracuse*. Pour $u_0 \in \mathbb{N}^*$, on considère la suite définie par :

$$u_{n+1} = \begin{cases} 3u_n + 1 & \text{si } u_n \text{ est impair.} \\ u_n/2 & \text{sinon.} \end{cases}$$

On conjecture que pour tout $u_0 \in \mathbb{N}^*$, il existe un indice n tel que $u_n = 1$ (à partir de cet entier n, la suite prend périodiquement les valeurs $1, 4$ et 2).

— Écrire une boucle `while` permettant de vérifier cette conjecture pour u_0 fixé. Testez-là ! Remarque : on n'a ici besoin que d'une variable pour calculer successivement les termes de la suite, qu'on pourra appeler u. Pour la division par 2, utilisez `//` : on veut rester sur les entiers, et ne pas utiliser de flottants ici.

— On appelle temps de vol à partir de u_0 le plus petit n tel que u_n vaut 1. Calculez ce temps de vol pour $u_0 = 15$ et $u_0 = 127$. On affichera les valeurs de la suite obtenues avant d'atteindre 1, à l'aide de `print`.

```
u = 127 #par exemple, valeur de départ
nb = 0 #un compteur pour le nombre d'itérations effectuées
while u>1:
    if u%2==0:
        u=u//2
    else:
        u=3*u+1
    nb = nb + 1
    print(u)
print("temps de vol : ",nb)
```

Exercice 5. *Quelques boucles imbriquées*. 1. Éxécuter le code suivant, bien comprendre le processus :

```
for i in range(10):
    for j in range(10):
        print(i,j)
```

2. Calculer (sans le faire mathématiquement) la somme $\sum_{i=0}^{100} \sum_{j=i}^{100} ij$ (réponse : 12920425).
3. En reprenant le code de l'exercice précédent dans une boucle `for`, calculer l'entier inférieur à 1000 ayant le plus grand temps de vol, et le temps de vol correspondant (réponse : 871, ayant un temps de vol de 178).

Pour la somme indiquée en question 2, il suffit de suivre bêtement la formule. Pour un calcul de somme, on initialise une valeur à 0 et on rajoute les termes un par un.

```
S = 0
for i in range(101): #de 0 à 100
    for j in range(i,101): #de i à 101
        S = S + i*j
#On vérifie que S vaut 12920425
```

Pour la question 3, il s'agit d'un calcul de maximum. Il faut pour cela utiliser une variable pour le maximum, qu'on met à jour dès qu'on trouve plus grand. Il faut aussi une variable pour stocker l'entier atteignant ce maximum

```
imax = 0
tps_max = 0
for i in range(1,1000):
    u=i
    nb=0
    while u>1:
        if u%2==0:
            u=u//2
        else:
            u=3*u+1
        nb+=1
    if nb>tps_max:
        tps_max = nb
        imax=i
print(imax, tps_max)
```