# TP 1- Proposition de solutions

**Solution 1** Divers calculs : priorité dans les calculs, utilisation des parenthèses !

```
>>> (2-3)/2+0.5
0.0
>>> 6/3*2
4.0
>>> 2/1+1/2
2.5
>>> 2/(1+1)/2
0.5
>>> (2/1+1)/2
1.5
>>> 2/(1+1/2)
1.3333333333333333
```

**Solution 4** Suites d'affectations :

**Méthode :** Afin de suivre le contenu des variables, il est conseillé, dans les cas difficiles, de faire un tableau décrivant leur contenu à chaque étape de la suites d'instructions. Dans le deuxième cas, cela donne :

| a | b |
|---|---|
| 2 | X |
| 2 | 3 |
| $(3-2)\times 2 = 2$ | 3 |
| 2 | $2^2 \times 3 = 12$ |

```
>>> a=2;a=a+2;a=a*a;a=a-(a/4);a;
12.0
>>> a=2;b=3;a=(b-a)*a;b=a**2*b;b;
12
```

**Solution 5** Instruction pour échanger le contenu de deux variables :

1. Avec des affectations simultanées :

```
a=3;b=4;
a,b=b,a
print(a,b)
```

2. Sans affectation simultanée : on utilise une variable auxiliaire pour sauvegarder la valeur d'une des variables.

```
a=3;b=4;
aux=a # variable auxiliaire
a=b
b=aux
print(a,b)
```

**Solution 6** Considérant un nombre :

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

Voici une instruction qui affiche vrai (faux sinon) si :

- le nombre est supérieur à 12 :

```
>>> nb>12
```

- le nombre appartient à $[2,4[$ :

```
>>> nb>=2 and nb<4
```

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

```
>>> (nb>=2 and nb<4) or nb>3
```

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

```
>>> (nb>=2 and nb<4) or (nb==-1 or nb==0)
```

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

```
>>> not ((nb>=2 and nb<4) or (nb==-1 or nb==0))
```

**Solution 7** Assertions logiques :

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

**Solution 10** Calcul de 30! :

```
u=1
for i in range(1,31):
    u=u*i
    print(i,'! = ',u)
```

**Solution 11**

```
a=int(input('Donner la base : a = '))
for i in range(1,13):
    print(a,'*',i,'=',a*i)
```

**Solution 12**

```
a=int(input('Donner la base : a = '))
for i in range(1,13):
    print('{} * {:<2} = {}'.format(a,i,a*i))
```

**Solution 13** Calcul des termes de la suite :

```
u=0
for k in range(1,10):
    u=k+10*u
    print(u)
```

Affichage des identités remarquables :

```
u=0
for k in range(1,10):
    u=k+10*u
    print("8␣*␣{:<9}␣+␣{}␣=␣{}".format(u,k,8*u+k))
```

**Solution 14** Le script est :

```
u=0
for k in range(1,10):
    u=1+10*u
    print("{:>9}*{:<9}␣=␣{:^17}".format(u,u,u**2))
```

**Solution 15** Le script suivant affiche la somme : $S=\sum_{k=2}^{n}\frac{1}{k^3}$

**Complété - corrigé**

```
n=int(input('Donnrt n : '))
s=0 # initialisation
for i in range(2,n+1):
    s=s+1/i**3
print('Pour n =',n,'on a S =',s)
```

Les erreurs étaient :

- Initialisation à 0 de la variable s.
- L'arrêt des itérations est fait pour i=n-1 car la fonction range donne une intervalle d'entiers ouvert à droite.
- L'affection ne cumule pas l'information (on pourrait utiliser +=), mais efface à chaque fois le travail précédent : il faut rajouter le prochain terme à la somme partielle déjà construite.
- L'opération puissance se note ** et non ^.
- Dans la boucle, c'est i qui varie ; le terme rajouté dépend donc de i et non de n.
- L'affiche se fait hors de la boucle sinon l'indentation place l'affichage dans la boucle est donc affiche tous les résultats partiels

**Solution 16**

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

Le script calcul et affiche le produit suivant :

$$P=\prod_{j=1}^{n-1}\exp\left(\frac{1}{j(j+1)}\right)$$

$$\begin{aligned}\text{Or }\ P &= \prod_{j=1}^{n-1}\exp\left(\frac{1}{j}-\frac{1}{j+1}\right)\\ &= \prod_{j=1}^{n-1}\frac{\exp\left(\frac{1}{j}\right)}{\exp\left(\frac{1}{j+1}\right)}\\ &\quad\text{produit téléscopique}\\ &= \frac{\exp(1)}{\exp\left(\frac{1}{n}\right)}\end{aligned}$$

Donc $\ln(P)=1-\frac{1}{n}$ et $\frac{1}{1-\ln(P)}=n$.
La dernière valeur affichée est n.

**Solution 17** ⇛ Calcul de $R=\prod_{k=1}^{23}\sin(k^2)$

```
R=1
for k in range(1,24):
    R=R*np.sin(k*k)
print('R =',R)
```

⇛ Calcul de $S_n=\sum_{1\le 2k+1\le n}2k+1$

On met une astuce pour s'assurer que l'entier donné est bien impair. De plus on utilise la fonction range(a,b,2) pour parcourir les entiers de 2 en 2.

```
n=0
while n%2==0:
    n=int(input('Donner un entier impair n = '))
S=0
for i in range(1,n+1,2):
    S=S+i
print('S =',S)
print('Verification : ',(n+1)**2/4)
```

⇛ Calcul de $T_n=\sum_{1\le 2k+1\le n}\frac{(-1)^k}{2k+1}$

On borne le parcours de la boucle par $k$ tel que $2k+1=n$. On note que n//2 est le quotient entier par 2.
Enfin pour illustrer la valeur de la limite, il suffit de considérer un n grand.

```
n=int(input('Donner un entier impair n = '))
T=0
for i in range(n//2):
    T=T+(-1)**i/(2*i+1)
print('T =',T)
print('Verification : ',np.pi/4)
```

⇛ Calcul de $U_n = \sum_{k=0}^{n-1} (2k+1)^2$

```
n=int(input('Donner un entier impair n = '))
U=0
for i in range(n):
    U=U+(2*i+1)**2
print('U =',U)
print('Verification : ',n*(4*n**2-1)/3)
```

⇛ Calcul de $V = \sum_{k=1}^{123} \prod_{j=k}^{k+2} \frac{k^2+1}{k*j}$

```
V=0
for k in range(1,124):
    p=1
    for j in range(k,k+3):
        p=p*(k**2+1)/k/j
    V=V+p
print('V =',V)
```

Le résultat est : `T=114.00451462019112`

**Solution 18** Le programme ci-dessous calcul le terme $u_{24}$ de la suite définie par :

$$\begin{cases} u_0 = 2 \\ u_{n+1} = \cos u_n \end{cases}$$

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

**Solution 19** Considérons :

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

Il est, au premier abord, naturel d'interpréter ce script comme le calcul d'un terme d'une suite vectorielle :

$$\begin{pmatrix} a_0 \\ b_0 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix} \text{ et } \forall n \in \mathbb{N}, \begin{pmatrix} a_{n+1} \\ b_{n+1} \end{pmatrix} = \begin{pmatrix} b_n \\ a_n + b_n \end{pmatrix}$$

Il affiche $b_{12}$.
On peut aussi interpréter une suite récurrente d'ordre 2 :

$$u_0 = 1 \quad u_1 = 2 \text{ et } \forall n \in \mathbb{N}, \quad u_{n+2} = u_{n+1} + u_n$$

Le script affiche $u_{13}$

**Solution 20** Suite récurrente
En utilisant l'affectation simultanée :

```
a,b,c=1,-2,0
for i in range(1,13):
    a,b,c=a-2*b+c,a-c,2*a-b-c
print(b)
```

En utilisant des affectation séquentielle, il est indispensable d'introduire des variables auxiliaires :

```
a=1;b=-2;c=0
for i in range(1,13):
    A=a-2*b+c
    B=a-c
    C=2*a-b-c
    a=A
    b=B
    c=C
print(b)
```

**Solution 21** On peut simplement proposer le script suivant :

```
n=int(input(' Donner n : '))
u=2 #u(0)
for k in range(n):
    u=np.sqrt(k+u) #u(i+1)
print(u)
```

Comme indiqué en commentaire, lors de la boucle de rang $i$ on calcule le terme $u_{i+1}$.
**Attention !** En général, on peut travailler en calculant le terme $u_i$ lors de la boucle de rang $i$. Dans ce cas, il convient de penser à réécrire la relation de récurrence, en posant $k = n+1$ :

$$\forall k \in \mathbb{N}^*, \ u_k = \sqrt{k-1+u_{k-1}}$$

```
import numpy as np
n=int(input(' Donner n : '))
u=2 #u(0)
for k in range(1,n+1):
    u=np.sqrt(k-1+u) #u(i+1)
print(u)
```

**Solution 22** Le script suivant affiche la parité (True) d'un nombre demandé à l'utilisateur.

**Complété et corrigé**

```
n=int(input('Donner une nombre : n = '))
if n%2==0:
    print(True)
else:
    print(False)
```

ou plus simplement :

```
n=int(input('Donner une nombre : n = '))
print(n%2==0)
```

En effet le résultat du test n%2==0 est déjà un booléen (True ou False).

**Solution 23** Expression conditionnelle :

```
x=float(input('Donner x = '))
print('f(x) = ',end='')
if x<0:
    print(exp(x))
elif x<=1:
    print(1)
else:
    print(exp(x-1))
```

Ce qui donne :

```
Donner x = 12
f(x) = 59874.1417152
```

**Solution 24** Minimum de trois nombres

```
a=float(input("Donner␣a␣:␣"))
b=float(input("Donner␣b␣:␣"))
c=float(input("Donner␣c␣:␣"))
if a<b and a<c:
    print(a)
elif b<c:
    print(b)
else:
    print(c)
```

**Solution 25** La suite de Syracuse

```
print('Suite de Syracuse')
u=int(input('Donner u0 : '))
while u!=1:
    if u%2==0:
        u=u//2
    else:
        u=3*u+1
    print(u)
```

Version modifiée pour afficher le rang du dernier terme calculer :

```
print('Suite de Syracuse')
u=int(input('Donner u0 : '))
n=0
while u!=1:
    if u%2==0:
        u=u//2
    else:
        u=3*u+1
    n=n+1
    print(u)
print('rang :',n)
```

**Solution 26** Série harmonique

**Calcul de $H_n = \sum_{k=1}^{n} \frac{1}{k}$**

```
n=int(input('Donner n = '))
h=0
for j in range(1,n+1):
    h=h+1/j
print('H'+str(n)+' =',h)
```

Le premier rang tel que $H_n \geq 12$ est : 91380

```
n=1
h=1
while h<12:
    n=n+1
    h+=1/n
print(n)
```