TP4

BOUCLES FOR

Corrigé

Exercice 1 **Exercice classique : calculer le terme de rang n d'une suite récurrente**

Soit $(u_n)_{n\in\mathbb{N}}$ la suite définie par $\begin{cases} u_0 = 0 \\ \forall n \in \mathbb{N}, \quad u_{n+1} = \sqrt{u_n + 4n} \end{cases}$

1.

Étape	À faire	Instruction python
Initialisation	$u_0 = 0$	`u=0`
$k = 0$	$u_1 = \sqrt{u_0 + 4 \times 0} = 0$	`u=sqrt(u+4*k)`
$k = 1$	$u_2 = \sqrt{u_1 + 4 \times 1} = 2$	`u=sqrt(u+4*k)`
$k = 2$	$u_3 = \sqrt{u_2 + 4 \times 2} = \sqrt{10}$	`u=sqrt(u+4*k)`

2.

```
from math import *
n = int(input('Saisir un entier naturel n : '))
u = 0
for k in range(0,n) :   # ou range(n), k varie de 0 à n-1
    u = sqrt(u + 4*k)
print(u)
```

3. $u_3 \approx 3.16$.
4. Pour afficher tous les termes, il faut indenter le print

```
from math import *
n = int(input('Saisir un entier naturel n : '))
u = 0
print(u)  # affichage de u0
for k in range(0,n) :
    u = sqrt(u + 4*k)
    print(u)
```

Exercice 2 1.

```
s = 0
for i in range(5):
    x = float(input("Saisir un nombre :")) # demande un réel
    s = s + x # ajoute ce reel au reel précédent
# à ce stade, s = somme des nombres reels obtenus
print(s/5) # la somme est divisee par 5,
```

```
# Conclusion: ce programme renvoie la moyenne de 5 nombres demandé à
l'utilisateur..
```

2. **Classique : Somme**. Pour $n \geqslant 1$, on définit : $S_n = \sum_{i=1}^{n} \frac{1}{i^2}$.

   (a) Compléter le tableau suivant donnant les calculs à faire pour obtenir $S = S_3$ en faisant une seule addition par étape.

Étape	**Valeur de S**
Initialisation	$S = 0$
$i = 1$	$S = S_1 = \frac{1}{1^2} = S + \frac{1}{1^2} = 1$
$i = 2$	$S = S_2 = \frac{1}{1^2} + \frac{1}{2^2} = S + \frac{1}{2^2} = \frac{5}{4}$
$i = 3$	$S = S_3 = \frac{1}{1^2} + \frac{1}{2^2} + \frac{1}{3^2} = S + \frac{1}{3^2}$

   (b) Pour $n \geq 1$, exprimer S_{n+1} en fonction de S_n.

$$\forall n \geq 1,\ S_{n+1} = S_n + \frac{1}{(n+1)^2}$$

   (c) Écrire un programme python demandant à l'utilisateur de saisir un entier n puis calculant et affichant S_n.

```
n=int(input("donner un entier"))

S=0
for i in range(1,n+1):
    S=S+1/(i**2)

print(S)
```

   (d) Vérifier que Python donne $S_{100} = 1.6349839001848923$.

3. **Classique : factorielle.** Pour $n \in \mathbb{N}^*$, on définit : $n! = 1 \times 2 \times \cdots \times n$ et $0! = 1$.

   (a) Compléter le tableau suivant donnant les calculs à faire pour obtenir $p = 4!$.

Étape	**Valeur de p**
Initialisation	$p = 1$
$i = 1$	$p = 1 = p \times 1$
$i = 2$	$p = 1 \times 2 = p \times 2$
$i = 3$	$p = 1 \times 2 \times 3 = p \times 3$
$i = 4$	$p = 1 \times 2 \times 3 \times 4 = p \times 4$

   (b) Pour $n \geq 1$, exprimer $(n+1)!$ en fonction de $n!$.

$$\forall n \geq 1,\ (n+1)! = n! \times (n+1)$$

(c) Écrire une fonction python renvoyant $n!$.

```
def factorielle(n):
    '''factorielle(n :int >=0 ) -> n! :int'''
    assert n >= 0 # message d'erreur si n est negatif
    p=1
    for i in range(1,n+1):
        p=p*i
    if n==0:
        p=1
    return p
```

Vérifier : $0! = 1$, $3! = 6$, $5! = 120$.

```
>>> factorielle(0)
1

>>> factorielle(3)
6

>>> factorielle(5)
120
```

Exercice 3

1. Comment simuler le lancer d'une pièce qui tombe sur pile avec une probabilité de $\frac{1}{3}$ en utilisant `randint(1,3)` ?
Le programme doit renvoyer `'P'` (chaîne de caractères) avec probabilité $\frac{1}{3}$ et `'F'` avec probabilité $\frac{2}{3}$.

```
from random import randint
def piece():
    r = randint(1,3)
    if r==1 : # il y a une chance sur 3 d'obtenir l'entier 1 (par exemple)
        return 'P'
    else :
        return 'F'
```

2. Tester la fonction précédente jusqu'à obtenir deux faces différentes.
Combien de lancers t'a t-il fallu ?

```
>>> piece()
'F'

>>> piece()
'F'

>>> piece()
'F'

>>> piece()
'F'

>>> piece()
```

```
    'P'         # personnellement il m'a fallu 5 lancers et toi ?
```

3. **Classique : compteur**. On simule 900 lancers de cette pièce. Compléter le programme pour déterminer le nombre de `'P'` obtenus. Ce nombre doit être proche de 300.

```
compteur = 0

for l in range(900) :

    if piece() == 'P' :

         compteur=compteur +1

print(compteur)
```

4. Combien obtenez-vous de `'P'` ?

```
>>> (executing lines 32 to 35 of "TP FOR.py")
305 # resultat proche de 300 :)
```

5. Simuler 20 lancers de cette pièce. On construira une chaîne de caractères du type `'chaine = PFFPFPPP...'` à l'aide de concaténations (+) successives.

```
chaine = '' # initialisation : chaine de caracteres vide

for i in range(20) :

    chaine = chaine + piece()

print(chaine)
```