1 Instructions itératives

Exercice 1 :

1. Parmi les conditions suivantes, une seule est vraie. Laquelle ?

   - `30 >21` | `293 <= 176` | `30 < 26` | `33 >= 49`

2. Parmi les conditions suivantes, une seule est vraie. Laquelle ?

   - `26 > 56 or 293 <= 176`
   - `245 >= 154 and 26 > 56`
   - `245 >= 154 and 37 <= 51`
   - `33 >= 49 or 30 < 26`

   Si `a = 5` et `b = 15`, quelle est la seule condition fausse parmi les suivantes ?

   - `a != b` | `a*b > 54` | `a == b` | `a < b`

3. On considère le programme suivant. Quelles valeurs de x permettent d'afficher `"Bonsoir"` ?

```
1 if x >= 13:
2     print("Bonjour")
3 elif x >= 5:
4     print("Bonsoir")
5 else:
6     print("Bonne nuit")
```

Exercice 2 :
Un cinéma pratique trois types de tarifs pour deux personnes.

- Si les deux personnes sont mineures, elles payent 7 euros chacune.
- Si l'une seulement est mineure, elles payent un tarif de groupe de de 15 euros.
- Si les deux personnes sont majeures, elles payent 18 euros en tout.

Ecrire un programme qui demande l'âge de chacune des deux personnes puis qui affiche le prix à payer.

Exercice 3 :

1. Quel mot-clé Python permet de définir une boucle conditionnelle ?
2. Quelle instruction faut-il écrire pour que la variable `i` prenne les valeurs entre 4 et 7 ?
3. Qu'affiche le programme suivant ?

```
1 a = 3
2
3 while a <= 14 :
4     a =a + 4
5
6 print(a)
```

4. Qu'affiche le programme suivant ?

```
1 a = 2
2
3 while a > 14 :
4     a =a + 4
5
6 print(a)
```

5. Qu'affiche le programme suivant ?

```
1 a= 0
2
3 for i in range(3,6) :
4     a =a + 4
5
6 print(a)
```

6. Qu'affiche le programme suivant ?

```
1 s = 0
2 i = 4
3
4 while i <= 7 :
5     s = s + i
6     i = i +1
7
8 print(s)
```

7. Qu'affiche le programme suivant ?

```
s = 0

for i in range(0,3) :
    s = s + i**2

print(s)
```

Exercice 4 :

1. Le programme suivant doit afficher les entiers de 7 à 17 à l'aide d'une boucle `while`. Trouvez les erreurs et corrigez ce programme pour qu'il affiche la valeur demandée.

```
n = 0

while n < 17:
   print(n)
   n = n + 2
```

2. Le programme suivant doit afficher les entiers de 3 à 11 à l'aide d'une boucle `for`. Trouvez les erreurs et corrigez ce programme pour qu'il affiche la valeur demandée.

```
for i in range(2,11):
    print(3)
```

3. Ecrire un programme qui affiche les nombres entiers de 1 à 20.

4. Ecrire un programme qui affiche les nombres entiers de 20 à 37.

Exercice 5 :

1. Que va afficher le programme ci-dessous ?

```
def fonction(y):
    return 3 + y *3

print(fonction(4))
```

2. Ecrire une fonction `multiplication(a,b)` qui renvoie la multiplication de `a` par `b`.

3. On définit la fonction $f : x \mapsto x^2 + 1$. Dans le programme ci-dessous, la fonction f est mal définie. Trouvez les erreurs et corrigez-les.

```
def f(x):
    return x*2 + 1
```

4. Ecrire une fonction `triple(n)` qui renvoie le triple du nombre `n`.

5. Définir la fonction P dont l'expression est $P(x) = x^4 + x^2 + 1$

6. Définir la fonction g dont l'expression est $g(x) = \sqrt{1 + x^2}$

7. Définir la fonction `perimetreCercle(R)` qui retourne le périmètre d'un cercle de rayon `R`.

8. Définir une fonction `moyenne(a,b)` qui renvoie la moyenne des deux nombres `a` et `b`.

9. Définir une fonction `tauxevolution(y_1,y_2)` qui renvoie le taux d'évolution entre y_1 et y_2.

10. Définir la fonction `distance(xA,yA,xB,yB)` qui retourne la distance entre deux points $A(x_A; y_A)$ et $B(x_B; y_B)$ dans un repère orthonormé.

Exercice 6 :

1. Définir une fonction `mini(a,b)` qui renvoie le minimum de ces deux nombres.

2. Compléter la fonction `inverse(x)` afin qu'elle retourne la valeur $\frac{1}{x}$ sir $x \neq 0$ et `"erreur"` dans le cas contraire.

```
def inverse(x):
    if ....:
        return ....
    else:
        return ....
```

3. Ecrire une fonction `estRectangle(a,b,c)` qui retourne `True` si le triangle de côtés `a`, `b` et `c` est rectangle et `False` sinon.

Exercice 7 :

1. Ecrire une fonction `sommeEntiers(n)` qui retourne la somme des entiers naturels inférieurs ou égaux à `n` : $1 + 2 + 3 + ... + n$

2. Ecrire une fonction `sommeInverses(n)` qui retourne la somme des inverses des entiers naturels compris entre `1` et `n` : $1+\frac{1}{2}+\frac{1}{3}+...+\frac{1}{n}$

Exercice 8 :
Une cuve d'eau a une contenance de 1000L. En moyenne, la cuve d'eau se remplit de 3L d'eau par jours. Ecrire une fonction Python `remplir()` qui renvoie le nombre de jours qu'il faudra pour remplir la cuve.

Exercice 9 :

1. Définir une fonction `de(nfaces)` simulant le lancer d'un dé équilibré à `n` faces et retrounant le numéro du dé.
2. Définir une fonction `echantillon(numero,n)` simulant `n` lancers de dé à 6 faces et retournant la fréquence du numéro.
3. Définir une fonction `echantillon(numero,n,m)` simulant `n` lancers de dé à m faces et retournant la fréquence du numéro.

Exercice 10 :

1. Ecrire un programme qui affiche le résultat suivant :

```
0
1
2
3
4
5
```

2. Ecrire un programme qui affiche le résultat suivant :

```
0
2
4
6
8
10
```

3. Ecrire un programme qui affiche le résultat suivant :

```
1
22
333
4444
55555
```

Exercice 11 :

1. Qu'affiche le programme suivant ?

```
c=0
for i in range(10):
    n=int(input("Entrer un chiffre:"))
    if (n==3):
        c=c+1

print("c=",c)
```

2. Que fait le programme suivant ?

```
from random import randint

a=randint(1,10)
b=randint(1,10)
r=int(input(str(a)+"*"+str(b)+"=?"))

if r==a*b:
    print("bien")
```

3. Compléter ce programme pour qu'il affiche un message d'erreur lorsque la réponse donnée n'est pas la bonne.
4. Modifier ce programme : en utilisant une boucle sur ce programme, répéter ce programme 5 fois et poser ainsi 5 questions. Ajouter un compteur au programme qui compte, et affiche à la fin, le nombre de bonnes réponses et la note obtenue.

Exercice 12 :
Sur une carte magnétique est stocké le nombre d'entrées restantes dans une salle de sport. Lorsqu'on présente sa carte à la borne automatique d'entrée, une entrée doit être décomptée.

Écrire un programme qui demande le nombre d'entrées restantes puis affiche le nouveau nombre d'entrées restantes après en avoir retiré une. Le programme affichera de plus un message d'avertissement lorsque le nombre d'entrées restantes est plus petit que 3.

Exercice 13 :

1. Qu'affiche le programme suivant ?

```
1 n=33
2 c=0
3 while n>0:
4     n=n-7
5     c=c+1
6
7 print(c)
8 print(n)
```

2. En s'aidant de ce programme précédent, écrire un programme qui demande à l'utilisateur une somme d'argent et qui la décompose en un minimum de billets et pièces nécessaires.
   Par exemple, la somme de 125 euros sera décomposée en 2 billets de 50 euros, 1 billet de 20 euros, 2 pièces de 2 euros et 1 pièce de 1 euro.

Exercice 14 :
Je souhaite faire des quelques économies. Je mets pour cela de côté les 1000 euros que je possède, sur un compte en banque. Chaque mois je vais en plus y déposer 200 euros.
Écrire un programme qui calcule le montant de mes économies au bout de 10 mois, puis au bout de 2 ans (en s'aidant du programme de l'exercice précédent, ou des autres exercices précédents).

Exercice 15 :
Je dépose 100 euros sur un compte en banque. Ce compte a un taux d'intérêt de 3%, c'est-à-dire que chaque année, en début d'année, la banque ajoute sur ce compte 3% de la somme qui s'y trouve l'année précédente.
Écrire un programme qui calcule le montant de mes économies sur ce compte en banque au bout de 2 ans, puis au bout de 10 ans, puis au bout de 100 ans.

Exercice 16 :

1. Rédiger une fonction `heure_to_sec(h,m,s)` qui prend en argument trois entiers représentant une durée exprimée en heures/minutes/secondes et qui retourne cette durée exprimée en secondes.
2. Rédiger une fonction `sec_to_heure(s)` qui prend en argument un nombre entier de secondes et qui affiche cette durée au format `hh:mm:ss`.
3. En déduire une fonction `durée(h1,m1,s1,h2,m2,s2)` qui prend en arguments six entiers représentant deux dates d_1 et d_2 et qui affiche la durée $d_2 - d_1$ au format `hh:mm:ss` (on supposera $d_1 < d_2$).

Exercice 17 :
La racine carrée entière d'une entier $n \in \mathbb{N}$ est l'unique entier p vérifiant $p^2 \leq n < (p+1)^2$.
Rédiger une fonction baptisée `isqrt` qui calcule la racine carrée entirèe d'un entier passé en paramètre.

Exercice 18 :
Le but de cet exercice est déterminer les mille plus petits nombres premiers. Pour déterminer si un entier est premier, on utilise le critère suivant :
un entier p est premier lorsque $p \geq 2$ et lorsqu'il n'est divisible par aucun entier $k \geq 2$ vérifiant $k^2 \leq p$.

1. Ecrire une fonction nommée `premier`, prenant en un paramètre entier p, et qui retourne le booléen `True` lorsque p est un nombre premier et le booléen `False` dans le cas contraire.
2. Utiliser cette fonction pour afficher les mille plus petits nombres premiers.
3. La conjecture de Goldbach postule que tout entier pair supérieur à 3 peut s'écrire comme somme de deux nombres premiers (éventuellement égaux). Vérifier cette conjecture pour tout entier inférieur ou égal à 1 000.
4. A contrario, montrer que la conjecture suivante est fausse : *Tout nombre impair est la somme d'une puissance de 2 et d'un nombre premier*

Exercice 19 :

1. Définir une fonction `fact(n)` qui prend en argument un entier n et qui renvoie $n!$.
2. On définit la suite de Fibonacci $(F_n)_{n\in\mathbb{N}}$ par $F_0 = 0$, $F_1 = 1$ et $F_{n+2} = F_{n+1} + F_n$. Définir une fonction `fib(n)` qui prend en argument un entier n et qui renvoie F_n.

Exercice 20 :
Les premiers termes de la suite de Conway sont : 1, 11, 21, 1211, 111221, ..., chaque terme étant obtenu en lisant à haute voix le terme précédent (c'est pourquoi Conway avait baptisé cette suite Look and say). Par exemple, le terme 1211 se lit "un 1, un 2, deux 1" donc le terme suivant est 111221.

1. Rédiger en Python une fonction `lookandsay(n)` qui retourne le terme qui suit l'entier `n` dans cette énumération.
2. Afficher à l'aide de cette fonction les 20 premiers termes de la suite de Conway.
3. Il a été démontré que si on note un le nombre de chiffres du ne terme de cette suite alors le rapport $\frac{u_{n+1}}{u_n}$ tendvers une constante, appelée constante de Conway. Calculez-en une valeur approchée.