INFO-H-100 - Programmation
TP 6 — Boucles `while`

Lorsque l'exercice demande d'écrire une fonction, écrivez la fonction demandée et testez-la avec plusieurs valeurs pertinentes.

Cette séance portant sur les boucles `while`, veillez à ne pas utiliser de boucle `for`.

Ex. 1. Qu'affiche le programme suivant ?

```
a = 1
while a <= 5:
    print(a,end=" ")
    a += 1
print(a)
```

Ex. 2. Qu'affiche le programme suivant ?

```
a = 0
while a < 5:
    a += 1
    print(a,end=" ")
print(a)
```

Ex. 3. Qu'affiche le programme suivant ?

```
a = 0
while a <= 5:
    a -= 1
    print(a,end=" ")
print(a)
```

Ex. 4. Écrire une fonction `my_range(a,b)` qui produit une liste ordonnée contenant tous les entiers dans $[a, b[$.

Ex. 5. Écrire une fonction `my_range_step(a,b,step)` qui produit une liste ordonnée contenant tous les entiers dans $[a, b[$ séparés de `step`. On considère que $a < b$ et que `step` est strictement positif.

Ex. 6. Écrire une fonction `my_range_step(a,b,step)` qui produit une liste ordonnée contenant tous les entiers dans $[a, b[$ séparés de `step`. a et b ne sont pas nécessairement dans l'ordre croissant et `step` peut être négatif.

```
print(my_range_step(1,10,2))     # [1, 3, 5, 7, 9]
print(my_range_step(10,1,-2))    # [10, 8, 6, 4, 2]
print(my_range_step(1,10,-2))    # []
print(my_range_step(10,1,2))     # []
```

Ex. 7. Écrire une fonction qui renvoie l'indice de la première apparition d'un caractère dans une chaîne, ou `-1` si le caractère n'apparaît pas.

Avec une boucle `for`, cette fonction peut s'écrire de la manière suivante :

```
def find1(string, char):
    for i in range(len(string)):
        if string[i] == char:
            return i
    return -1
```

Réécrivez-la en utilisant une boucle `while` et en ne faisant qu'un seul `return`.

Ex. 8. Écrire une fonction qui reçoit en argument une chaîne de caractères et deux caractères, et qui renvoie une nouvelle chaîne de caractères où la première occurrence du premier caractère a été remplacée par le second caractère.

Ex. 9. Écrire une fonction qui détermine si une séquence est croissante.

```
>>> increasing_func([1, 2, 5, 8]) # -> True
>>> increasing_func([1, 8, 5, 2]) # -> False
>>> increasing_func([1, 2, 2, 5]) # -> False
>>> increasing_func([5])          # -> True
>>> increasing_func([])           # -> True
```

Ex. 10. Écrire une fonction qui détecte si un mot passé en argument est un palindrome. Pour rappel, un palindrome est une phrase qui peut se lire dans les deux sens : l'ordre des lettres est symétrique.

```
>>> palindrome("hello")
False
>>> palindrome("radar")
True
```

INFO-H-100 - Programmation
TP 6 — Boucles `while`
Corrections

Solution de l'exercice 1:

Le programme affiche `1 2 3 4 5 6`

Solution de l'exercice 2:

Le programme affiche `1 2 3 4 5 5`

Solution de l'exercice 3:

Le programme affiche `-1 -2 -3 -4 -5 -6 -7` ... et ne s'arrête jamais (boucle infinie).

Solution de l'exercice 4:

```
def my_range(a,b):
        _list = []
        while (a < b):
                _list.append(a)
                a += 1

        return _list
```

Solution de l'exercice 5:

```
def my_range_step(a,b,step):
        _list = []
        while (a < b):
                _list.append(a)
                a += step

        return _list
```

Solution de l'exercice 6:

```
def my_range_step(a,b,step):
    _list = []
    if a<b and step>0:
        while a<b:
            _list.append(a)
            a += step
    elif a>b and step<0:
        while a>b:
            _list.append(a)
            a += step
    return _list

print(my_range_step(1,10,2))     # [1, 3, 5, 7, 9]
print(my_range_step(10,1,-2))    # [10, 8, 6, 4, 2]
print(my_range_step(1,10,-2))    # []
print(my_range_step(10,1,2))     # []
```

Solution de l'exercice 7:

```
def find1(ls, c):
        i = 0
        while i < len(ls) and ls[i] != c:
                i += 1
        if i == len(ls):
                i = -1
        return i
```

Solution de l'exercice 8:

Voir slides...

```
def replace1(st, old, new):
    i = 0
    while i < len(st) and st[i] != old:
        i += 1
    #i == lg or st[i] == old
    if i < len(st):
        st = st[:i] + new + st[i+1:]
    return st
```

Solution de l'exercice 9:

```
def increasing_func(ls):
    result = True
    if len(ls) >= 2:
        i = 1
        while i < len(ls) and result:
            result = ls[i] > ls[i - 1]
            i += 1
    return result
```

Solution de l'exercice 10:

```
def palindrome(mot):
    g = 0
    d = len(mot)-1
    while g<d and mot[g] == mot[d]:
        g+=1
        d-=1
    return mot[g] == mot[d]
```

Solution alternative...

```
def palindrome(mot):
    i = 0
    ok = True
    while i < len(mot)/2 and ok:
        if mot[i] != mot[len(mot)-i-1]:
            ok = False
        i+=1

    return ok

print(palindrome('sos'))
print(palindrome('abba'))
print(palindrome('thierry'))
```