**Programmation 2** Spring, 2026

# TD 3: loops

1. Écrivez un programme Python qui, étant donné un entier positif `n`, imprime tous les nombres dans $\{1, \ldots, n\}$.

**Solution.**

```
n = int(input("Number: "))

i = 1
while i <= n:
    print(i)
    i = i + 1
```

```
n = int(input("Number: "))

for i in range(1, n+1):
    print(i)
```

2. Écrivez un programme Python qui imprime les nombres dans $\{0, \ldots, 100\}$ qui sont divisibles par 5 mais pas par 3.

**Solution 1.**

```
for i in range(0, 101):
    if i % 5 == 0 and i % 3 != 0:
        print(i)
```

3. Utilisez une boucle `while` pour trouver les 20 premiers nombres divisibles par 5, 7 et 11, et imprimez-les.

**Solution.**

```
count = 0
x = 1
while count < 20:
    if x%5 == 0 and x%7 == 0 and x%11 == 0:
        print(x)
        count += 1
    x += 1
```

4. Écrivez un programme Python qui, à partir d'une liste `A`, imprime le plus petit nombre de cette liste.

**Solution 1.**

```
A = [38, 56, 12, 62, 23, 41]

min = A[0]

for i in range(0, len(A)):
    if A[i] < min:
        min = A[i]

print("Minimum is: ", min)
```

**Solution 2.**

```
A = [38, 56, 12, 62, 23, 41]

min = A[0]

for x in A:
    if x < min:
        min = x

print("Minimum is: ", min)
```

5. Le plus petit nombre divisible par 2, 3 et 4 est 12. Trouvez le plus petit nombre divisible par tous les entiers compris entre 1 et 10.

**Solution 1.**

```
x = 1
keep_searching = True
while keep_searching:
    if (x%1==0 and x%2==0 and x%3==0 and x%4==0 and x%5==0 and
        x%6==0 and x%7==0 and x%8==0 and x%9==0 and x%10==0):
        print(x)
        keep_searching = False
    x += 1
```

**Solution 2.**

```
x = 0
keep_searching = True
while keep_searching:
    x += 1
    keep_searching = False
    for i in range(1, 11):
        if x%i != 0 :
            keep_searching = True
print(x)
```

6. Écrivez un programme Python qui, étant donné une liste `A`, trouve toutes les paires de deux nombres dans cette liste dont la somme est soit 14 soit 44.

**Solution.**

```
A = [52, 73, 134, 1, 13, 62, 77, 123, 4, 40, 43, 87]

for i in range(0, len(A)):
    for j in range(i+1, len(A)):
        if ( (A[i] + A[j]) == 14) or ( (A[i]+A[j]) == 44):
            print(A[i], A[j])
```

7. Écrivez un programme Python qui, étant donné une liste `A`, trouve tous les triplets de trois nombres dans cette liste dont la somme est soit 14 soit 44.

**Solution.**

```
A = [0, 52, 73, 134, 1, 13, 62, 77, 123, 4, 40, 43, 87, 2, 15]

for i in range(0, len(A)):
    for j in range(i+1, len(A)):
        for k in range(j+1, len(A)):
            if ( (A[i] + A[j] + A[k] == 14) or
                 (A[i] + A[j] + A[k] == 44) ):
                print(A[i], A[j], A[k])
```

8. Écrivez un programme Python qui imprime tous les triplets ordonnés d'entiers `(a,b,c)` avec $0 \leq a < b < c$ tels que $a + b + c = 15$.

**Solution.**

```
for i in range(0, 16):
    for j in range(i+1, 16):
        for k in range(j+1, 16):
            if (i+j+k) == 15:
                print(i, j, k)
```

9. Écrivez un programme qui demande à l'utilisateur de saisir deux entiers positifs, la hauteur et la longueur d'un parallélogramme et imprime un parallélogramme de cette taille avec des étoiles.

   Par exemple, si la hauteur était de 3 et la longueur de 6, ce qui suit serait imprimé :

```
******
 ******
  ******
```

   Si la hauteur était de 4 et la longueur de 2, ce qui suit serait imprimé :

```
**
 **
  **
   **
```

   **Solution.**

```
h = int(input("Enter height: "))
w = int(input("Enter width: "))

for i in range(0, h):
    for j in range(0, i):
        print(" ", end='')

    for j in range(0, w):
        print("*", end='')

    print()
```

```
w = int(input("Enter w: "))
h = int(input("Enter h: "))


for i in range(0, h):
    print(" "*i, "*"*w, sep='')
```

10. Écrivez un programme Python qui demande à l'utilisateur un entier $n > 1$ et crée et imprime une liste contenant tous les diviseurs de $n$.

**Solution.**

```
n = int(input("Enter a number : "))

divisors = []

for i in range(1, n+1):
    if (n % i) == 0:
        divisors.append(i)

print(divisors)
```

11. Écrivez un programme pour prendre un entier positif $n > 1$ de l'utilisateur et afficher si le nombre est un nombre *parfait*, un nombre *abondant* ou un nombre *déficient* :

**Nombre parfait.** Un nombre parfait est un nombre où la somme de ses diviseurs propres (les nombres qui le divisent de manière égale sans lui-même) est égale au nombre.

**Nombre abondant.** Un nombre abondant est un nombre où cette somme dépasse le nombre lui-même.

**Nombre déficient.** Un nombre déficient est un nombre où cette somme est inférieure au nombre lui-même.

**Exemple:** 28 est parfait: $1 + 2 + 4 + 7 + 14 = 28$.
12 est abondant: $1 + 2 + 3 + 4 + 6 = 16 > 12$.
16 est déficitaire: $1 + 2 + 4 + 8 = 15 < 16$.

**Solution 1.**

```
n = int(input("Enter a number : "))

divisors = []

for i in range(1, n):
    if (n % i) == 0:
        divisors.append(i)

sum = 0
for x in divisors:
    sum = sum + x

if (n < sum):
    print(n, "is an abundant number.")
elif (n == sum):
    print(n, "is a perfect number.")
else:
    print(n, "is a deficient number.")
```

**Solution 2:**

```
n = int(input("Enter a number : "))

sum = 0
for i in range(1, n):
    if (n % i) == 0:
        sum = sum + i

if (n < sum):
    print(n, "is an abundant number.")
elif (n == sum):
    print(n, "is a perfect number.")
else:
    print(n, "!is a deficient number.")
```

12. Écrivez une programme qui prend une liste `A` d'entiers (non trié), et calcule deux indices `i` et `j` tels que `A[j]-A[i]` soit maximisé.

    **Vous devez écrire un algorithme qui utilise une seule boucle.**

    `Example:` pour `A = [3,6,1,5,11,7,1]`, la solution est `j = 4, i = 6`.

    **Solution.** Il est suffisant de trouver la minimum et la maximum.

```
A = [3,6,2,5,11,7,1]

minindex = 0
maxindex = 0

for i in range(0, len(A)):
    if A[i] > A[maxindex]:
        maxindex = i
    if A[i] < A[minindex]:
        minindex = i

print("A: ", A)
print(maxindex, minindex)
```

13. Étant donnée une liste `A` de nombres (ils peuvent être positifs ou négatifs), trouvez les deux indices $i$ et $j$ tels que la somme des nombres compris entre $i$ et $j$ (inclus) en `A` soit maximum.

**Exemple:** pour `A = [1,7,-3,6,-9,2]`, la réponse est $i = 0$ et $j = 3$ car $[1, 7, -3, 6]$ a la somme maximale.

**Solution.**

```
A = [1,7,-3,6,-9,2]

maxv, maxi, maxj = A[0], 0, 0

for i in range(0, len(A)):
    _sum = 0
    for j in range(i, len(A)):
        _sum += A[j]
        if _sum > maxv:
            maxv = _sum
            maxi = i
            maxj = j

print("i: ", maxi, " j: ", maxj, " with sum: ", maxv)
```

14. (**Difficile**) On vous donne une liste `A` de $2n$ entiers, dont la somme vaut $S$. Vous jouez le jeu suivant contre un adversaire :

    Vous commencez. À votre tour, vous choisissez l'un des deux nombres situés aux extrémités actuelles de la liste (`A[0]` ou `A[2n-1]`). Vous ajoutez ce nombre à votre score et le retirez de la liste.

    Puis c'est au tour de votre adversaire, qui fait de meme: il choisit un nombre à l'une des deux extrémités restantes, l'ajoute à son score et le retire de la liste.

    Vous continuez ainsi alternativement jusqu'à ce que la liste soit vide.

    La personne dont les nombres récoltés totalisent au moins S/2 gagne!

    Montrez qu'il existe une algorithme pour vous, telle que quel que soit le jeu de l'adversaire, vous gagnez!

    **Solution.**

    Parmi les $2n$ nombres $A[0], \ldots, A[2n-1]$, soit la somme des nombres d'indices impairs, soit la somme des nombres d'indices pairs, doit être au moins égale à $S/2$. Supposons que la somme des nombres d'indices impairs soit au moins $S/2$. Alors vous suivrez une stratégie où vous obtenez tous les nombres d'indices impairs, et donc vous gagnez.

    En particulier, à partir de $A[0], \ldots, A[2n-1]$, vous prenez $A[2n-1]$, puisque $2n-1$ est impair. Désormais les deux extrémités restantes $A[0]$ et $A[2n-2]$ ont toutes deux un indice pair. Donc, quel que soit l'élément que votre adversaire prend, il y aura à nouveau un indice impair disponible pour vous à votre tour suivant, et ainsi de suite.

    La même stratégie fonctionne si vous devez rassembler tous les éléments d'indice pair.