Rappel :

1) Structures conditionnelles :

* `if condition :` (condition → bool)

* `if condition :`
  `else :`

* `if condition1 :`
  `elif condition2 :`
  `elif condition3 :`

* `if c1 :`
  `elif c2 :`
  `elif c3 :`
  `else :`

Ex :
```
n = ---
if n%2==0 :
    print("pair")
else :
    print("impair")
```

> = : affectation
> == : comparaison (test)

2) Boucle :

* `for x in sequence :`

∀ x ∈ Sequence : - range - liste, tuple, ... - str

ex :
```
for i in range(1,10):
    print(i)
```

range(d, f, pas) : d par défaut = 0 ; pas par défaut = 1

ex :
```
n = int(input("---:"))
for i in range(1, n+1):
    if n%i == 0:
        print(i)
```

Remarque :

continue : va passer vers l'étape suivante

break : sortir de la boucle

ex :
```
for i in range(1,10):
    if i==3:
        break
    else:
        print(i)
```
1
2

```
for i in range(1,10):
    if i==3:
        continue
    else:
        print(i)
```
1 2 4 5 6 7 8 9

* La boucle while :

```
while condition:
```

ex :
```
i = 1                  → Initialisation
while i<=9:            → Condition
    print(i)
    i = i+1            → mise à jour
```

Remarque :

la boucle while est plus générale que la boucle for

ex: n = ---

```
if n < 2 :
    print("Non premier")
else :
    rep = True
    i = 2
    while i < n :
        if n % i == 0 :
            rep = False
            break
    if rep == True :
        print("premier")
    else :
        print("n'est pas premier")
```

```
i = 5
while i < 6 :
    print(i)
    i = i - 1
```
} boucle infinie

3) Fonctions :

```
def nom_fonction(var1, var2, ---) :
    ===
    ===
    ===
    return ... → retourner et Sortir
```
} Corps de la fonction

ex :
```
def Diviseur(n) :
    for i in range(1, n) :
        if n % i == 0 :
            print(i)
```

Remarque :

*
```
def f(x) :
    y = 1 + x
    return y
print(f(2)) → 3
print(y) → Erreur car y est locale
```

```
def f(x) :
    global y
    y = 1 + x
    return y
print(f(2)) → 3
print(y) → 3
```

*
```
def f(x) :
    x = x - 2
    return x

a = 5
print(f(a)) → 3
print(a) → 5
```
a n'est pas modifié

a est dite variable non mutable

```
def f(L) :
    L.append(0)
    return L

M = [1, 2, 3]
print(f(M)) → [1, 2, 3, 0]
print(M) → [1, 2, 3, 0]
```
M sera modifié → M est variable mutable

* Variable mutable : liste, ensemble, dictionnaire

* Variables non mutables : int, float, str, tuple

Exercice : (Voir TP)

Structures de données :

1) listes : indexée + mutable

* `L = [1, 2, 3]` ; `len(L)` → 3

`L = []` ; `L = list()`

Création par compréhension

`L = [ i*2 (3) for i in range(1,1001) (1) if i%2==0 (2) ]` → `[4, 16, ..., 1000000]`

* Accès par indice : `L[i]` avec $0 \leq i < len(L)$
* slicing (tranchage) – sous liste de L –

`L[d:f:p]`

ex : `L = [3,8,5,7,1,4,0]`

`L[2:5]` → `[5,7,1]` ; `L[1:6:2]` → `[8,7,4]`

`L[:3]` → `[3,8,5]` ; `L[2:]` → `[5,7,1,4,0]` ; `L[:]` → `[3,8,5,7,1,4,0]`

* Copie d'une liste :

```
L = [1,2,3,4]
M = L
M[1] = 10
print(M) → [1,10,3,4]
print(L) → [1,10,3,4]
```

Solution :

* `M = L[:]`
* `M = [x for x in L]`
* `M = L.copy()` # une copie d'une seule dimension
* `M = deepcopy(L)` # copie profonde ; avec : `from copy import deepcopy`

* `append(x)` ; `extend(T)` ; `insert(i,x)`
* `remove(x)` ; `pop()` ; `pop(i)` ; `clear()`
* `index(x)` ; `count(x)` ; `sort()` ; `sorted(L)` ; `reverse()`

`sort()` : trie la liste sur place

`sorted(L)` : renvoie une autre liste des éléments de L triés

2) tuple : indexé + non mutable

`T = ()` ; `T = tuple()` ; `T[i]` ✓ ; `T[i] = x` ✗ Erreur

3) Ensembles : non indexé + mutable + les éléments de l'ensemble ne sont pas mutable

- `E = {1,2,3}` ; `E[0]` ✗ Erreur ; `E = set()` — E n'a pas de doublons
- `E.add(x)` ; `E.update(T)` ; `E.remove(x)` ; `E.discard(x)` ; `E.clear()`
- `E.union(F)` ; `E.intersection(F)` ; `E.difference(F)` ; `E.symetric_difference(F)`