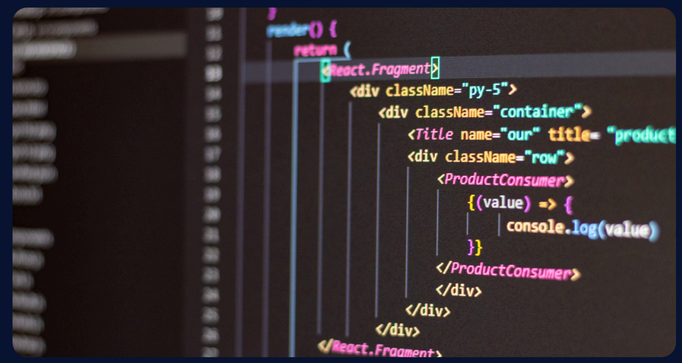


Seance 2 : Python

Laffet takwa

Variables Python : Attribuer plusieurs valeurs

Python vous permet d'attribuer des valeurs à plusieurs variables sur une seule ligne :

```
x, y, z = "Orange", "Banana", "Cherry"
print(x)
print(y)
print(z)
```

```
x = y = z = "Orange"

print(x)
print(y)
print(z)
```

Types de séquences :

Si vous disposez d'une collection de valeurs dans un list, tupleetc., Python vous permet d'extraire ces valeurs dans des variables. C'est ce qu'on appelle le déballage .

```
fruits = ["apple", "banana", "cherry"]
x, y, z = fruits

print(x)
print(y)
print(z)
```

Variables Python : Attribuer plusieurs valeurs

- Le premier élément est à l’index 0.
- On peut accéder avec un index positif ou négatif.

```
print(fruits[0])   # pomme
print(fruits[-1])  # kiwi
```

Manipulation des listes

- Ajouter un élément

```
fruits.append("mangue")      # ajoute à la fin
fruits.insert(1, "fraise")   # insère à l’index 1
```

-

```
fruits.remove("banane")  # supprime par valeur
fruits.pop(2)            # supprime par index
del fruits[0]            # supprime l’élément à l’index 0
```

Variables Python : Attribuer plusieurs valeurs

```
#Modifier un élément
fruits[0] = "ananas"

# Parcourir une liste
for f in fruits:
    print(f)
```

Les Tuples

Un tuple est une collection ordonnée mais immuable (non modifiable).
 On utilise des parenthèses ().

```
coordonnees = (10, 20)
informations = ("Ali", 25, "Tunis")

#Accès aux éléments

print(coordonnees[0])  # 10
```

Différence avec les listes :

Caractéristique	Liste (list)	Tuple (tuple)
Définition	Collection ordonnée et **modifiable**	Collection ordonnée et **immuable**
Syntaxe	Crochets []	Parenthèses ()
Modification	On peut ajouter, supprimer, modifier des éléments	Impossible de modifier (ajout, suppression, remplacement interdits)
Taille	Peut changer (dynamique)	Fixe après la création
Performance	Moins rapide que les tuples	Plus rapide (optimisé car immuable)
Utilisation typique	Quand on a besoin d'une séquence flexible (liste de courses, données modifiables)	Quand on veut des données fixes (coordonnées (x, y), couleurs RGB, dates)
Méthodes disponibles	Beaucoup de méthodes : .append(), .remove(), .insert(), etc.	Peu de méthodes : .count(), .index() uniquement
Exemple	fruits = ["pomme", "banane"]	coord = (10, 20)

Structures conditionnelles

1. Conditions Python et instructions If

Python prend en charge les conditions logiques habituelles des mathématiques :

- Égal à : a == b
- Pas égal à : a != b
- Moins de : a < b
- Inférieur ou égal à : a <= b
- Supérieur à : a > b
- Supérieur ou égal à : a >= b

Ces conditions peuvent être utilisées de plusieurs manières, le plus souvent dans des « instructions if » et des boucles.
Une « instruction if » est écrite en utilisant le mot-clé if

```
if condition:
    instructions
elif autre_condition:
    instructions
else:
    instructions
```

2. Boucles et itérations

a boucle for
On l’utilise avec **range()** ou pour parcourir une liste.

```
for i in range(5):
    print(i)
```

break = arrête la boucle immédiatement
continue = saute à l’itération suivante

Sortez de la boucle lorsque x c'est "banane", mais cette fois la pause vient avant l'impression
: fruits = ["apple", "banana", "cherry"]

```
for x in fruits:
if x == "banana":
  break
print(x)
```

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
if x == "banana":
  continue
print(x)
```

```
for x in range(2, 6):
print(x)
```

- La **range()** fonction prend par défaut 0 comme valeur de départ, mais il est possible de spécifier la valeur de départ en ajoutant un paramètre : range(2, 6), ce qui signifie des valeurs de 2 à 6 (mais sans inclure 6)

if dans la boucle For

```
for x in range(6):
    print(x)
else:
    print("Finally finished")
```

```
for x in range(6):
    if x == 3: break
        print(x)
else:
    print("Finally finished!")
```

Boucles imbriquées

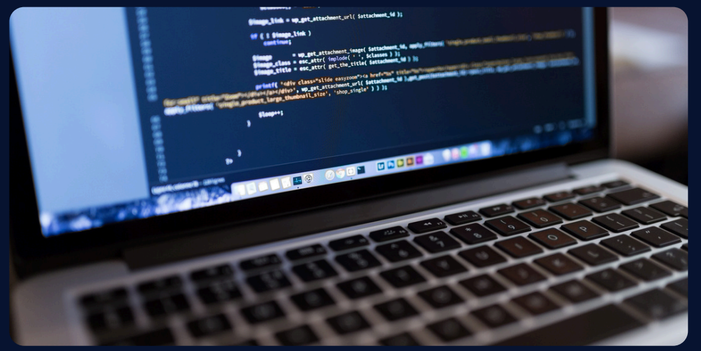
```
adj = ["red", "big", "tasty"]
fruits = ["apple", "banana", "cherry"]

for x in adj:
    for y in fruits:
        print(x, y)
```

La boucle while :Elle s’exécute tant que la condition est vraie.

```
compteur = 0
while compteur < 5:
    print("Compteur:", compteur)
    compteur += 1
```

```
i = 1
while i < 6:
  print(i)
 if i == 3:
   break
 i += 1
```



Thank You