
EMSI
django
Développement Web avec Python
Pr. Ghazouani Mohamed
Année 2024/2025

1

Partie 3

2

2

Structures de contrôle

Plan

- Enchaînement séquentielle (bloc)
- Traitements conditionnels (**if** . . . **elif** . . . **else**)
- Traitements répétitifs (**while** et **for**)

3

Enchaînement séquentielle (bloc)

4

Enchaînement séquentielle (bloc)

Bloc
Un **bloc** est une suite d'instructions qui sont exécutées dans l'ordre de leur apparition.

Conseil
Ne pas mélanger espace et tabulation dans les indentations. Python recommande de n'utiliser que les espaces et une indentation de 4 espaces.

Règle
Toutes les instructions d'un même bloc doivent avoir exactement la même indentation.

5

5

Instruction if

6

6

Instruction if

```
if condition1 :
    Bloc exécuté si
    condition1 est
    vrai.
elif condition2 :
    Bloc exécuté si condition1 est
    faux et condition2 est vrai.
else :
    Bloc exécuté si aucune
    condition n'est vraie.
```

```
if n > 0:
    resultat = 'positif'
elif n < 0:
    resultat = 'negatif'
else:
    resultat = 'nul'
```

- Les *condition sont des expressions logiques booléennes évaluées à True ou False.*
- Les deux-point (":") en fin de ligne introduit un bloc.

7

7

Instruction if

L'opérateur conditionnel (ternaire) en Python

```
if nombre % 2 == 0:
    resultat = "pair"
else:
    resultat = "impair"
```

```
resultat = "pair" if nombre % 2 == 0 else "impair"
```

8

8

Instruction if

Exercice

Afficher le tarif d'une place de cinéma en fonction de l'âge de la personne qui assistera à la séance. Le tarif normal de la place est de 50 DH. Les enfants (moins de 14 ans) paient 20 DH. Les séniors (65 ans et plus) paient 40 DH.

Avec un if normal puis en if ternaire.

9

Les boucles Répétition PourChaque (for)

10

Répétition PourChaque (for)

Forme

for nom **in** expression:
    bloc

Exemple 1

```
liste = [ 1, 2, 3, 4, 5 ]
for nombre in liste:
    print( nombre ** 2, end=' ' )
```

donne '1 4 9 16 25 '

Exemple 2

```
for nombre in range(0, 10):
    print(nombre, end=' ')
```

affiche '0 1 2 3 4 5 6 7 8 9 '

Sémantique :

- ***expression*** doit être une *séquence a*, par exemple list, tuple, range, str. . .
- ***nom*** prend successivement chaque valeur de l'*itérable*
- ***bloc*** est exécuté pour chaque affectation de *nom*

```
list=['Java','Python','C#']
for item in list:
    print(item)
```

```
for i in range(len(list)):
    print(list[i])
```

Principe :

- On sait combien de fois on exécute *bloc* (autant que d'éléments dans *expression*)
- La terminaison est donc garantie.

11

11

Idiomes en boucle

enumerate

```
liste=['Python', 'Java', 'PHP']
for index, value in enumerate(liste):
    print(index, value)
```

```
0 Python
1 Java
2 PHP
```

zip

```
noms = ['Alice', 'Bob', 'Charlie']
ages = [24, 30, 18]
for nom, age in zip(noms, ages):
    print(f"{nom} a {age} ans.")
```

```
Alice a 24 ans.
Bob a 30 ans.
Charlie a 18 ans.
```

12

12

Répétition PourChaque (for)

Exercice

Trouver les diviseurs : Déterminez tous les diviseurs du nombre donné. Un diviseur est un nombre entier qui divise le nombre donné sans laisser de reste.
Stocker les diviseurs dans une liste
Afficher la liste de diviseurs
Afficher le nombre de diviseurs

13

Les boucles Répétition TantQue (while)

14

Répétition TantQue (while)

Forme :

```
while condition:
    bloc
```

Propriétés

Sémantique : Tant que *condition* est vraie, *bloc* est exécuté.
Remarque : On peut ne pas exécuter ***bloc*** si la ***condition*** est fausse au départ
Règle : Il faut que ***bloc*** modifie ***condition*** (sinon boucle infinie ou code mort)

```
n = 10
i = 1                          # pour parcourir les entiers de 1 à n
somme = 0                      # la somme des entiers
while i <= n:                  # i est à prendre en compte
   somme = somme + i           # ou somme += i
   i = i + 1                   # ou i += 1
print ('la somme des entiers de 1 à 10 est :', somme)
```

Exercice : Exécuter à la main ce programme avec n = 3

15

15

Répétition TantQue (while)

Exercice

Écrire un programme qui, étant donnée une chaîne de caractères, affiche le nombre de chiffres (caractère de '0' à '9') qu'elle contient.
Utiliser la fonction isdigit().
Exemple : 'Le prix de 10 lots de 5 vaut 50.'
Le résultat est : 5.

16

16

Fonctions Procédures

17

Fonctions - Procédures

- Bloc d'instructions =>attention à l'indentation
- Paramètres non typés
- Renvoie une valeur en sortie (ou plus). Ou pas de valeurs: Procédure

```
def affichage(a,b):
  print(a,b)

affichage(a=5,b=4)
affichage(b=4,a=5)
```

```
def divmod(a,b):
    return(a%b)

res=divmod(a=5,b=4)
print (res)
```

```
def divmod(a,b):
    return(a//b, a%b)

div,mod=divmod(a=5,b=4)
print (div, ' ', mod)
```

```
def divmod(a,b):
    return(a//b, a%b)

res=divmod(a=5,b=4)
print (res)
```


18

Fonctions - Procédures

```python
#définition de la fonction
def parite(number):
    '''cette fonction renvoie True si le nombre et pair,
    Sinon False  '''
    pair=False
    if number%2==0:
        pair=True
    return pair
#programme
x=int(input('Entrez x: '))
#appel de la fonction et affichage
if parite(x):
    print(f'{x} est pair')
else:
    print(f'{x} est impair')
```

19

Fonctions - Procédures

```python
def multiply(*numbers):
    ???

multiply(2, 3, 4, 5)
```

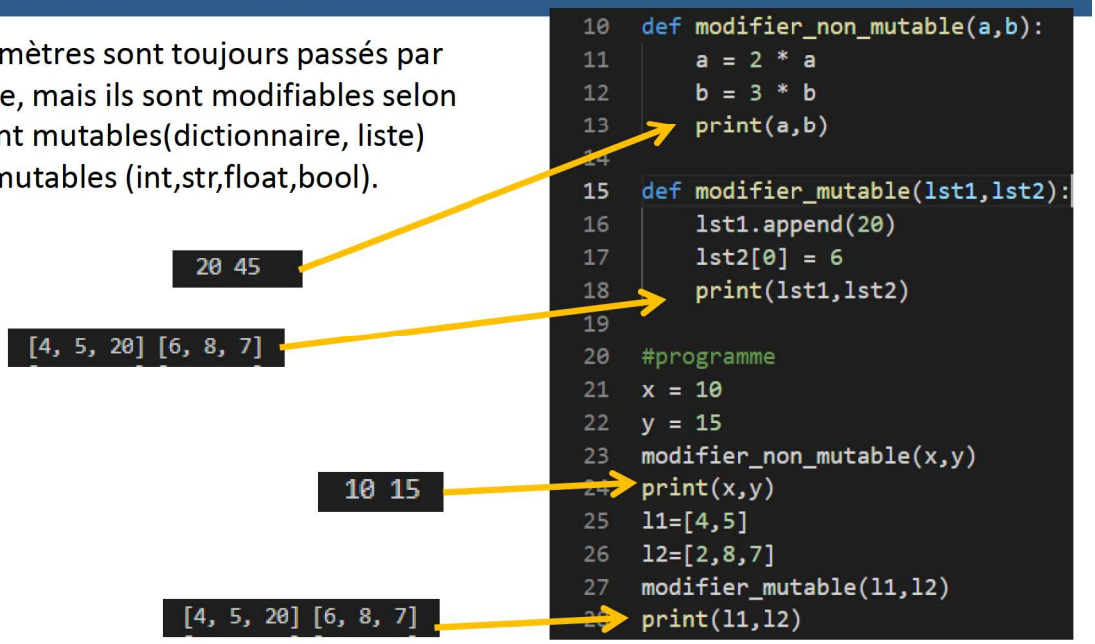
```python
def save_user(**user):
    ???

save_user(id=1, name="John", age=22)
```

20

20

Fonctions - Procédures

Les paramètres sont toujours passés par référence, mais ils sont modifiables selon qu'ils sont mutables(dictionnaire, liste) ou non mutables (int,str,float,bool).




```
10  def modifier_non_mutable(a,b):
11      a = 2 * a
12      b = 3 * b
13      print(a,b)
14
15  def modifier_mutable(lst1,lst2):
16      lst1.append(20)
17      lst2[0] = 6
18      print(lst1,lst2)
19
20  #programme
21  x = 10
22  y = 15
23  modifier_non_mutable(x,y)
24  print(x,y)
25  l1=[4,5]
26  l2=[2,8,7]
27  modifier_mutable(l1,l2)
28  print(l1,l2)
```

21

Défis

Défi 1

# Fonction qui fait la recherche du plus grand élément d'une liste
# Liste fournie au départ :
liste1 = [32, 5, 12, 8, 3, 75, 2, 15]

Défi 2

Fonction qui fait la séparation des nombres pairs et impairs
# Liste fournie au départ :
liste2 = [32, 5, 12, 8, 3, 75, 2, 15]
pairs = []
impairs = []

Défi 3

Fonction qui renvoie l'indice du plus grand élément de la liste liste2

Défi 4

Fonction qui fait l'inversion d'une chaîne de caractères avec la boucle while

22

Fin Partie 3

Fin Partie 3