



## Présentation ADRAR Pôle Numérique

Suivez-nous... LinkedIn  et Twitter  @Adrar\_Numerique



# ADRAR\o/ PÔLE NUMERIQUE

PÔLE NUMERIQUE DU CENTRE DE FORMATION ADRAR

- > SUPPORT, ADMINISTRATION SYSTEMES & RESEAUX
- > DEVELOPPEMENT D'APPLICATIONS WEB & MOBILES
- > WEBDESIGN & DEVELOPPEMENT INTERFACES
- > TRANSFORMATION NUMERIQUE DES ENTREPRISES

<http://www.adrar-numerique.com>

Suivez-nous sur LinkedIn  Twitter  @Adrar\_Numerique

... et sur le web : [www.adrar-numerique.com](http://www.adrar-numerique.com)



Suivez-nous sur LinkedIn  Twitter  @Adrar\_Numerique

... et sur le web : [www.adrar-numerique.com](http://www.adrar-numerique.com)

# LA LOGIQUE CONDITIONNELLE

Suivez-nous sur LinkedIn  Twitter  @Adrar\_Numerique

... et sur le web : [www.adrar-numerique.com](http://www.adrar-numerique.com)

Python est un langage qui se veut simple à lire et à écrire. De ce fait vous n'avez pas besoin de symbole pour indiquer la fin d'une ligne (comme le « ; » en Php ou Css) ni d'accolades pour définir le début et la fin d'un test, d'une boucle ou d'une fonction. C'est l'indentation qui va délimiter tests, boucles et fonctions.

Pas besoin non plus de parenthèses pour indiquer les éléments d'un test ou d'une boucle.

Un test en python va utiliser le mot clé « if » suivi des conditions du tests et du symbole « : ». Si on veut définir d'autre conditions on utilisera « elif » et pour parler de tous les autres cas on utilisera « else ». C'est la logique du « si, sinon si, sinon ».

```
a = 9
b = 8
if a > b:
    print("a est plus grand")
elif a < b:
    print("a est plus petit")
else:
    print("a et b sont égaux")
```

Suivez-nous sur LinkedIn Twitter @Adrar\_Numerique

... et sur le web : [www.adrar-numerique.com](http://www.adrar-numerique.com)

Il existe 2 boucles en Python : la boucle for et la boucle while.

### La boucle while

La boucle while permet de définir une condition qui mettra fin à la boucle. Elle s'écrit comme un test, c'est-à-dire qu'on indique la condition après le mot clé de la boucle, on termine la condition par « : » et on indente tout le bloc d'action à effectuer pour chaque tour de boucle.

Le compteur qui vous permet de gérer votre boucle doit absolument être initialisé.

Attention à bien faire en sorte de pouvoir sortir de sa boucle pour ne pas se retrouver dans une boucle infinie.

```
i = 0
while i < 10:
    print(i)
```

**ATTENTION BOUCLE INFINI !**  
**i NE SERA JAMAIS >= A 10 !**

```
i = 0
while i < 10:
    print(i)
    i += 1
```

```
continuer = True
while continuer:
    a = input("Dis oui")
    if a == "oui":
        continuer = False
```

Suivez-nous sur LinkedIn  Twitter  @Adrar\_Numerique

... et sur le web : [www.adrar-numerique.com](http://www.adrar-numerique.com)

## La boucle for

La boucle for va nous permettre d'itérer dans un élément, que ce soit une liste, un dictionnaire, une chaîne de caractère ou un intervalle numérique.

Pour la boucle for, le compteur n'a pas besoin d'être initialisé.

Étant donné qu'on parcourt un élément de taille finie la boucle se termine obligatoirement toute seule.

Attention tout de même à ne pas modifier la taille de l'élément qu'on parcourt pendant la boucle

```
tab = [1, 2, 3, 4, 5]
for i in tab:
    print(i)

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

chaîne = "bonjour"
for i in chaîne:
    print(i)
```



# EXERCICES

- Réaliser un programme qui demande un nombre à un utilisateur et qui lui dira si le nombre est pair ou impair.
- Réalisez un programme qui demandera à l'utilisateur son jour, moi et année de naissance et qui indiquera si il est majeur ou non.
- Réalisez un programme qui demande à l'utilisateur le nombre d'articles, puis le prix hors taxes de chaque article et qui affichera le prix total TTC

Suivez-nous sur LinkedIn  Twitter  @Adrar\_Numerique

... et sur le web : [www.adrar-numerique.com](http://www.adrar-numerique.com)

# MANIPULATION DE LISTES ET DE DICTIONNAIRES



Suivez-nous sur LinkedIn  Twitter  @Adrar\_Numerique

... et sur le web : [www.adrar-numerique.com](http://www.adrar-numerique.com)

En python une liste se déclare avec « [ ] » et chaque élément sera séparé par une virgule.

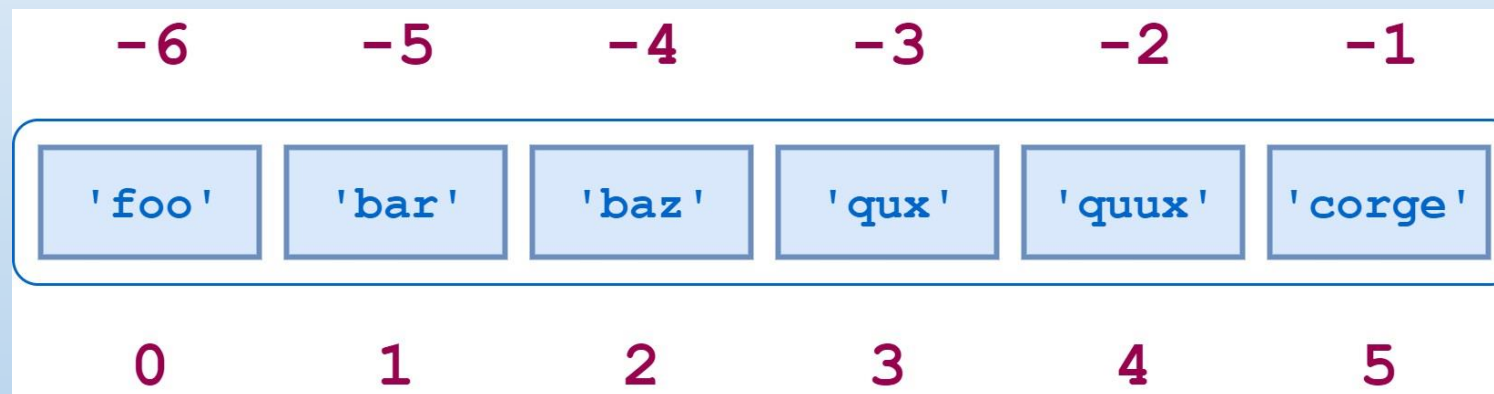
Dans une liste on peut retrouver tout type d'élément : une chaîne de caractères, un entier, un float, ou même une liste ou un dictionnaire. On peut donc se retrouver avec des listes multidimensionnelles dont chaque élément sera lui-même une liste (dont chaque élément pourra également être une liste, dont chaque élément .....).

```
ma_liste = [1, "deux", [1, 2, 3], True, 3.5, {1:"premier", 2:"deuxieme"}]
```

Suivez-nous sur LinkedIn Twitter @Adrar\_Numerique

... et sur le web : [www.adrar-numerique.com](http://www.adrar-numerique.com)

Le premier élément est à l'index 0 et le dernier élément est à l'index longueur - 1 ou -1.



Si la liste ci-dessus s'appelle liste1, la case contenant la chaîne de caractère « bar » sera la case numéro 1. On parlera de liste1[1] (ou bien liste1[-5])

Suivez-nous sur LinkedIn  Twitter  @Adrar\_Numerique

... et sur le web : [www.adrar-numerique.com](http://www.adrar-numerique.com)

Comme je vous l'ai indiqué TOUT est objet en python. Ce qui veut dire qu'une liste est un objet de la classe « list » et qu'elle aura donc des méthodes de classe. La méthode `append()` va nous permettre d'ajouter un élément à une liste et la méthode `pop()` va nous permettre d'en retirer.

```
ma_liste = [1, "deux", [1, 2, 3], True, 3.5, {1:"premier", 2:"deuxieme"}]
print(ma_liste)
ma_liste.append("9")
print(ma_liste)
ma_liste.pop(3)
print(ma_liste)
```

```
[1, 'deux', [1, 2, 3], True, 3.5, {1: 'premier', 2: 'deuxieme'}]
[1, 'deux', [1, 2, 3], True, 3.5, {1: 'premier', 2: 'deuxieme'}, '9']
[1, 'deux', [1, 2, 3], 3.5, {1: 'premier', 2: 'deuxieme'}, '9']
```

Suivez-nous sur LinkedIn Twitter @Adrar\_Numerique

... et sur le web : [www.adrar-numerique.com](http://www.adrar-numerique.com)

Grâce à la boucle for on vous pouvoir itérer dans notre liste (c'est-à-dire naviguer d'élément en élément).

```
liste1 = [1, 2, 3, 4, 5]
for i in liste1:
    print(i)
```

```
1
2
3
4
5
```

Pour récupérer la valeur de l'index de chaque case de la liste on va utiliser la fonction enumerate() qui prend en paramètre un objet. La boucle for prend donc deux valeurs, l'index ainsi que la valeur de la case de l'index.

```
liste1 = [1, 2, 3, 4, 5]
for ind, val in enumerate(liste1):
    print("la valeur :", val, "est à l'index :", ind)
```

```
la valeur : 1 est à l'index : 0
la valeur : 2 est à l'index : 1
la valeur : 3 est à l'index : 2
la valeur : 4 est à l'index : 3
la valeur : 5 est à l'index : 4
```

Suivez-nous sur LinkedIn  Twitter  @Adrar\_Numerique

... et sur le web : [www.adrar-numerique.com](http://www.adrar-numerique.com)

Pour définir à partir de quand est ce qu'on veut itérer dans une liste on a la notation `liste[debut:]`. Et pour définir jusqu'où on veut aller nous avons la notation `liste[:fin]`.

```
liste = [1, 2, 3, 4, 5]
for i in liste[2:]:
    print(i)
```

3  
4  
5

Ici on affiche tous les éléments à partir de l'index numéro 2 (inclus)

```
liste = [1, 2, 3, 4, 5]
for i in liste[:3]:
    print(i)
```

1  
2  
3

Ici on affiche tous les éléments jusqu'à l'index numéro 3 (non inclus)

# EXERCICES

- Réalisez un programme qui demande à l'utilisateur un nombre de notes à inscrire, puis les valeurs de ces notes. Ces valeurs seront enregistrés dans un tableau. Le programme affichera
  - la meilleure note
  - la pire note
  - la moyenne des notes



Suivez-nous sur LinkedIn  Twitter  @Adrar\_Numerique

... et sur le web : [www.adrar-numerique.com](http://www.adrar-numerique.com)

Un dictionnaire se comporte comme une liste mais chaque élément est identifié par un couple clé/valeur.

Un dictionnaire se déclare avec « {} », chaque élément est séparé par un virgule et on met un « : » entre une clé et sa valeur

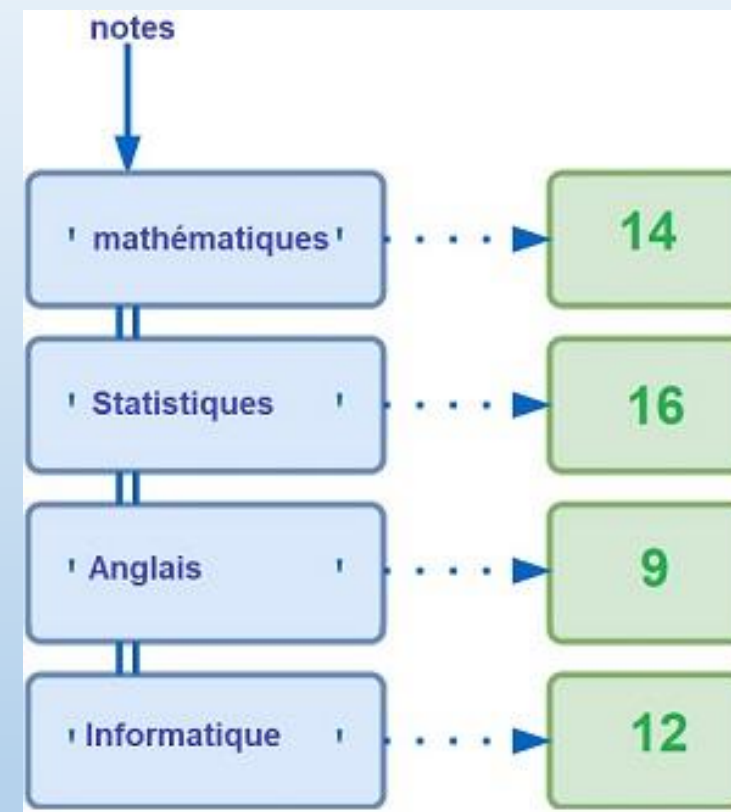
On peut également retrouver tout type d'élément en tant que valeur dans un dictionnaire. Les clés par contre ne peuvent être que des entiers, floats ou chaînes de caractères et chaque clé est unique dans le dictionnaire.

```
dico1 = {1:[1,2,3], "deux":{1:2, 2:3}, 3.5:5}
```

Suivez-nous sur LinkedIn Twitter @Adrar\_Numerique

... et sur le web : [www.adrar-numerique.com](http://www.adrar-numerique.com)

Ici le dictionnaire « notes ». La valeur de la clé « mathématiques » est 14, la clé « Statistiques » a pour valeur 16, la clé « Anglais » a pour valeur 9 et la clé « Informatique » a pour valeur 12.



```
notes = {'mathématiques': 14, 'Statistiques': 16, 'Anglais': 9, 'Informatique': 12}
```

Suivez-nous sur LinkedIn Twitter @Adrar\_Numerique

... et sur le web : [www.adrar-numerique.com](http://www.adrar-numerique.com)

Pour créer une nouvelle entrée dans un dictionnaire on peut simplement déclarer « dico[clé] = valeur »

```
dico = {"maths": 8, "français": 7, "histoire": 18}

print(dico)

dico["anglais"] = 20

print(dico)
```

```
{'maths': 8, 'français': 7, 'histoire': 18}
{'maths': 8, 'français': 7, 'histoire': 18, 'anglais': 20}
```

Suivez-nous sur LinkedIn  Twitter  @Adrar\_Numerique

... et sur le web : [www.adrar-numerique.com](http://www.adrar-numerique.com)

Nous pouvons utiliser les méthodes *values()* et *keys()* de la class « dict » pour afficher les clés et les valeurs du dictionnaire.

```
dico1 = {1:[1,2,3], "deux":{1:2, 2:3}, 3.5:5}
print(dico1.values(), dico1.keys(), sep=" : ")
```

```
dict_values([[1, 2, 3], {1: 2, 2: 3}, 5]) : dict_keys([1, 'deux', 3.5])
```

Suivez-nous sur LinkedIn  Twitter  @Adrar\_Numerique

... et sur le web : [www.adrar-numerique.com](http://www.adrar-numerique.com)

Nous avons également les méthodes *pop()* et *update()* pour supprimer ou ajouter des éléments à notre dictionnaire. La méthode *update()* prend en paramètre un dictionnaire et la méthode *pop()* une clé.

```
dico1 = {1:[1,2,3], "deux":{1:2, 2:3}, 3.5:5}
print(dico1)
ajout = {"add":True}
dico1.update(ajout)
print(dico1)
dico1.pop("deux")
print(dico1)
```

```
{1: [1, 2, 3], 'deux': {1: 2, 2: 3}, 3.5: 5}
{1: [1, 2, 3], 'deux': {1: 2, 2: 3}, 3.5: 5, 'add': True}
{1: [1, 2, 3], 3.5: 5, 'add': True}
```

Suivez-nous sur LinkedIn  Twitter  @Adrar\_Numerique

... et sur le web : [www.adrar-numerique.com](http://www.adrar-numerique.com)

On va également pouvoir itérer dans notre dictionnaire avec la boucle `for`. Le compteur utilisé pour itérer dans le dictionnaire retournera la clé de l'élément. Pour récupérer la valeur d'une clé il faudra considérer le dictionnaire comme un tableau avec pour index la clé.

```
dico1 = {1:[1,2,3], "deux":{1:2, 2:3}, 3.5:5}
for i in dico1:
    print(i, dico1[i], sep=" : ")
```

```
1 : [1, 2, 3]
deux : {1: 2, 2: 3}
3.5 : 5
```

Attention, la fonction `enumerate()` vous retournera l'index de l'élément et sa clé (et pas sa valeur).

```
dico1 = {1:[1,2,3], "deux":{1:2, 2:3}, 3.5:5}
for ind, key in enumerate(dico1):
    print(ind, key, dico1[key], sep=" : ")
```

```
0 : 1 : [1, 2, 3]
1 : deux : {1: 2, 2: 3}
2 : 3.5 : 5
```



# EXERCICES

- Réalisez un programme qui demande à l'utilisateur un nombre de notes à inscrire, puis les matières et enfin les valeurs de ces notes. Ces valeurs seront enregistrés dans un dictionnaire. Le programme affichera
  - la meilleure note et le nom de la matière
  - la pire note et le nom de la matière
  - la moyenne des notes

Suivez-nous sur LinkedIn  Twitter  @Adrar\_Numerique

... et sur le web : [www.adrar-numerique.com](http://www.adrar-numerique.com)

# GESTION DES ERREURS

Suivez-nous sur LinkedIn  Twitter  @Adrar\_Numerique

... et sur le web : [www.adrar-numerique.com](http://www.adrar-numerique.com)

La gestion des erreurs est primordiale dans la réalisation d'un programme. En python plusieurs instructions nous permettent de réaliser ce genre d'opérations : raise, try, except.

Le mot clé "raise" va nous permettre de générer une erreur. Lorsque l'instruction "raise" est exécuté le programme affiche l'erreur et s'arrête.

```
nombreDeRoues = 9

if nombreDeRoues > 4 or nombreDeRoues < 1:
    raise ValueError("Erreur, vous devez indiquer un nombre de roues entre 2 et 4")
else:
    print("Les roues ont été placés")
print("ok")
```

Traceback (most recent call last):

```
File "C:\Users\maximeprzybylo\Documents\Cours\Python\exercices\erreurs.py", line 4, in <module>
    raise ValueError("Erreur, vous devez indiquer un nombre de roues entre 2 et 4")
ValueError: Erreur, vous devez indiquer un nombre de roues entre 2 et 4
```

Suivez-nous sur LinkedIn  Twitter  @Adrar\_Numerique

... et sur le web : [www.adrar-numerique.com](http://www.adrar-numerique.com)

Avec le jeu de mot clés "try/except" on va pouvoir définir des actions pour lesquels on sait qu'il pourrait y avoir des erreurs, indiquer les instructions à effectuer si il n'y a pas de problème et également les actions à effectuer dans le cas où une erreur survient.

```
a = 3
b = 0

try:
    c = a / b
    print("division ok")
except ZeroDivisionError:
    print("Erreur, on ne peut pas diviser par 0")
print("ok")
```

```
Erreur, on ne peut pas diviser par 0
ok
```

Suivez-nous sur LinkedIn  Twitter  @Adrar\_Numerique

... et sur le web : [www.adrar-numerique.com](http://www.adrar-numerique.com)

Dans le "except" il faut préciser de quelle erreur on souhaite paramétrer une action. On peut également gérer l'ensemble des exceptions qui peut arriver en remplaçant le type d'exception par la syntaxe "Exception". Dans ce cas-là il faut absolument indiquer d'afficher le type de l'exception à l'utilisateur.

```
a = 3
b = 0

try:
    c = a / b
    print("division ok")
except Exception as mon_exception:
    print("Erreur, il y a eu un problème: ", mon_exception)
```

```
Erreur, il y a eu un problème: division by zero
```

Par convention, on appelle l'exception « e ». ( except Exception as e)

Suivez-nous sur LinkedIn  Twitter  @Adrar\_Numerique

... et sur le web : [www.adrar-numerique.com](http://www.adrar-numerique.com)

A la suite d'un "try/except" nous pouvons ajouter les mots clés "else" et "finally". "Else" va permettre d'indiquer au programme ce qu'il doit faire s'il a réussi les instructions du try, et "finally" va permettre de définir des actions à effectuer quoi qu'il arrive, qu'on ait réussi ou non les instructions du try.

Vous devez gérer au maximum les exceptions dans votre code ! Un code parfait n'existe pas mais vous pouvez vous en rapprocher au maximum. Essayez d'envisager toutes les erreurs possibles.

```
a = 3
b = 1

try:
    c = a / b
except ZeroDivisionError:
    print("Erreur, on ne peut pas diviser par 0")
else:
    print(c)
finally:
    print("on termine le script quoi qu'il arrive")
```

3.0

on termine le script quoi qu'il arrive



# EXERCICES

Implémentez la gestion des erreurs dans les exercices précédents. Vos programmes doivent redemander l'entrée utilisateur jusqu'à ce qu'il indique une valeur correcte.