

# Memento Python

Auteur : [Antoine Pernot](#)

## • Premiers pas avec Python

### Opérateurs arithmétiques :

Opération	Syntaxe
Addition	a+b
Soustraction	a-b
Multiplication	a*b
Division exacte	a/b
Division entière	a//b
Modulo	a%b
Puissance ( $a^b$ )	a**b
Arrondi de a, b décimales	round(a, b)

## • Le flux d'instruction

### Les conditions :

```
1 | if condition 1:  
2 |     si la condition 1 est vraie  
3 | elif condition 2:  
4 |     si la condition 1 est fausse et la  
   |     condition 2 est vraie  
5 | else:  
6 |     si les deux conditions sont fausses
```

### Opérateurs de comparaison :

Comparateur	Syntaxe
a égal à b	a == b
a différent de b	a != b
a supérieur à b	a > b
a supérieur ou égal à b	a >= b
a inférieur à b	a < b
a inférieur ou égal à b	a <= b
a [pas] dans b	a [not] in b

## • Factoriser le code

### La boucle "Tant que"

```
1 | compteur = 1  
2 | while compteur <= 10:  
3 |     print(compteur)  
4 |     compteur = compteur + 1
```

### Importer un module

```
1 | import csv  
2 | from random import randint  
3 | from json import *
```

## Créer une fonction

```
1 | def addition(a, b):  
2 |     return(a+b)
```

## • Les séquences

### Décomposition d'une séquence

```
1 | variable[indiceDebut:indiceFin:pas]
```

### Modifier la casse Exemple : LoReM iPsUm

Fonction	Exemple
lower()	lorem ipsum
upper()	LOREM IPSUM
title()	Lorem Ipsum
capitalize()	Lorem ipsum
swapcase()	l0rEm IpSuM

### Compter les éléments

```
1 | sequence.count(element)
```

### Remplacer dans un texte

```
1 | chaine.replace(ancien, nouveau)
```

### Ajouter un élément

```
1 | sequence.append(element)
```

### Modifier un élément

```
1 | sequence[indice] = element
```

### Insérer un élément

```
1 | sequence[indice:indice] = [element]
```

### Supprimer un élément

```
1 | sequence.remove(element)  
2 | sequence.pop(indice ou cle)
```

### Compter les éléments

```
1 | len(sequence)
```

### Éclater une chaîne en liste

```
1 | liste = chaine.split(separateur)
```

### Joindre une liste en chaîne

```
1 | separateur.join(liste)
```

### Trier une liste

```
1 | liste.sort()
```

## Inverser une liste

```
1 | liste.reverse()
```

## Mélanger une liste

```
1 | liste.shuffle()
```

## Trouver l'index d'un élément

```
1 | liste.index(element)
```

## Copier une liste

```
1 | from copy import deepcopy  
2 | copieListe = deepcopy(liste)
```

## Créer une liste de nombres

```
1 | range(debut, fin, pas)
```

## La boucle "Pour"

```
1 | for element in liste:  
2 |     print(element)
```

## Récupérer l'élément et son indice

```
1 | for indice, element in enumerate(liste):  
2 |     print(indice, element)
```

## Parcourir deux listes

```
1 | for element1, element2 in zip(liste1, liste2):  
2 |     print(element1, element2)
```

## Lister les clés d'un dictionnaire

```
1 | dictionnaire.keys()
```

## Lister les valeurs d'un dictionnaire

```
1 | dictionnaire.values()
```

## Copier un dictionnaire

```
1 | copie = dictionnaire.copy()
```

## Parcourir un dictionnaire

```
1 | for cle, valeur in dictionnaire.items():  
2 |     print(cle, valeur)
```

## • Manipuler les fichiers

### Navigation dans l'arborescence

```
1 >>> from os import getcwd, chdir, mkdir
2 >>> print(getcwd())
3 /home/antoine
4 >>> chdir('essais')
5 >>> print(getcwd())
6 /home/antoine/essais
7 >>> mkdir('test')
```

### Modes de flux fichier

Lettre	Action
'r'	Ouvrir en lecture seule (défaut)
'w'	Ouvrir en écriture. Écrase l'existant.
'x'	Ouvrir en écriture si inexistant.
'a'	Ouvrir en écriture. Ajoute à l'existant.
'b'	Mode binaire.
't'	Mode texte (défaut).

### Lire un fichier

```
1 fichier = open("texte.txt", 'rt')
2 for ligne in fichier.read():
3     print(ligne)
4 fichier.close()
```

### Écrire un fichier

```
1 fichier = open("texte.txt", 'wt')
2 fichier.write(texte)
3 fichier.close()
```

### Lire un fichier CSV

```
1 import csv
2 fichier = open("fichier.csv", "rt")
3 lecteurCSV = csv.reader(fichier, delimiter=";", quotechar="'')
4 for ligne in lecteurCSV:
5     print(ligne)
6 fichier.close()
```

### Écrire un fichier CSV

```
1 import csv
2 fichier = open("fichier.csv", "wt")
3 ecrivainCSV = csv.writer(fichier, delimiter=";")
4 ecrivainCSV.writerow(liste)
5 fichier.close()
```

### Lire un fichier JSON

```
1 import json
2 fichier = open("fichier.json", "rt")
3 dictionnaire = json.loads(fichier.read())
4 fichier.close()
```

### Écrire un fichier JSON

```
1 import json
2 fichier = open("fichier.json", "wt")
3 fichier.write(json.dumps(dictionnaire))
4 fichier.close()
```

### Gestion des erreurs

```
1 try:
2     # La portion de code à tester
3 except:
4     # Que faire en cas d'erreur
5 else:
6     # Si il n'y a pas d'erreurs
```

### Les chemins de fichiers

```
1 >>> import os.path
2 >>> chemin = "/tmp/dir/dir2/fichier.txt"
3 >>> print(os.path.basename(chemin))
4 fichier.txt
5 >>> print(os.path.dirname(chemin))
6 /tmp/dir/dir2
```

### Différencier les fichiers et les répertoires

Module os.path

Fonction	Action
exists(URI)	Si l'URI existe
isfile(URI)	Si l'URI est un fichier
isdir(URI)	Si l'URI est un dossier

### Lister le contenu d'un répertoire

```
1 os.listdir("/tmp/dir")
```

### Copier un fichier ou un répertoire

```
1 import shutil
2 shutil.copytree("dossier1", "dossier2")
3 shutil.copy("fichier1", "fichier2")
```

### Déplacer un fichier ou un répertoire

```
1 import shutil
2 shutil.move("dossier1", "dossier2")
```

### Supprimer un fichier ou un répertoire

```
1 import os, shutil
2 os.remove("fichier")
3 shutil.rmtree("dossier")
```

### Sauvegarder des variables

```
1 import pickle
2 fichier = open("fichier", "wb")
3 pickle.dump(variable, fichier)
4 fichier.close()
```

```
1 import pickle
2 fichier = open("fichier", "rb")
3 variable = pickle.load(fichier)
4 fichier.close()
```

## • Interagir avec les bases de données

### Exécuter une requête sur SQLite

```
1 import sqlite3
2 base = sqlite3.connect('fichier.db')
3 curseur = base.cursor()
4 curseur.execute(requete[, parametres])
5 base.commit()
6 base.close()
```

### Récupérer des données sur SQLite

```
1 import sqlite3
2 base = sqlite3.connect('fichier.db')
3 curseur = base.cursor()
4 curseur.execute(requete[, parametres])
5 for ligne in curseur.fetchall():
6     print(ligne)
7 base.close()
```

### Exécuter une requête sur MariaDB

```
1 import mysql.connector
2 base = mysql.connector.connect(host=hote,
3                               user=login, password=mdp, database=base)
3 curseur = base.cursor()
4 curseur.execute(requete[, parametres])
5 baseDeDonnees.close()
```

## Récupérer des données sur MariaDB

```
1 import mysql.connector
2 base = mysql.connector.connect(host=hote,
    user=login, password=mdp, database=base)
3 curseur = baseDeDonnees.cursor()
4 curseur.execute(requete[, parametres])
5 for ligne in curseur.fetchall():
6     print(ligne)
```

## • La programmation réseau

### Créer un serveur socket

```
1 import socket
2 serveur = socket.socket(socket.AF_INET,
    socket.SOCK_STREAM)
3 serveur.bind(('', 50000))
4 serveur.listen(5)
5 while True:
6     client, infosClient = serveur.accept()
7     requete = client.recv(255).decode("utf
    -8")
8     client.send(reponse.encode("utf-8"))
9     client.close()
10 serveur.close()
```

### Créer un client socket

```
1 import socket
2 client = socket.socket(socket.AF_INET,
    socket.SOCK_STREAM)
3 client.connect(("127.0.0.1", 50000))
4 client.send(requete.encode("utf-8"))
5 reponse = client.recv(255)
6 client.close()
```

## Le multithread

```
1 thread = threading.Thread(None, fonction,
    nom, arguments, dictionnaireArguments)
2 thread.start()
```

### Créer un serveur Web

```
1 import http.server
2 adresseServeur = ("", 80)
3 serveur = http.server.HTTPServer
4 handler = http.server.
    CGIHTTPRequestHandler
5 handler.cgi_directories = ["/tmp"]
```

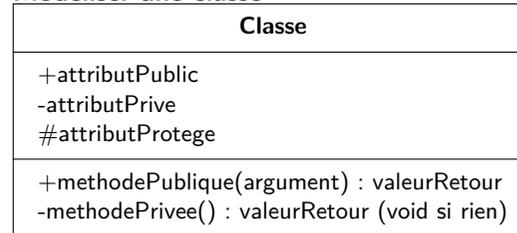
```
6 httpd = serveur(adresseServeur, handler)
7 httpd.serve_forever()
```

## Utiliser des services Web

```
1 import urllib.request
2 import json
3 requete = urllib.request.Request(urlApi)
4 reponse = urllib.request.urlopen(requete)
5 donnees = reponse.read().decode("utf-8")
```

## • Modélisation UML

### Modéliser une classe



### Relations entre classes



## • Programmation orientée objet

### Implémentation d'une classe

```
1 class Classe(ClasseHeritee): # object si
    aucun héritage
2     def __init__(self, argument):
3         # Constructeur
4         ClasseHeritee.__init__(self)
5         self.attributPublic
6         self.__attributPrive
7         variableDeLaFonction
8     def __methodePrivee(self, argument):
9         ...
10    def methodePublique(self):
11        ...
12 instanceClasse = Classe(argument)
13 instanceClasse.methodePublique()
```

## Les méthodes spéciales

Méthode	Action
<code>__str__</code>	Si l'objet est appelé comme texte.
<code>__int__</code>	Si l'objet est appelé comme entier.
<code>__float__</code>	Si l'objet est appelé comme réel.

## Comparaison riche

Méthode	Comparateur
<code>__eq__</code>	a égal à b
<code>__ne__</code>	a différent de b
<code>__gt__</code>	a supérieur à b
<code>__ge__</code>	a supérieur ou égal à b
<code>__lt__</code>	a inférieur à b
<code>__le__</code>	a inférieur ou égal à b