

420-436-SH Développement de scripts

# **P6 – Scripts Python**

## **Partie 2**

# Plan

- Fonctions
- Variables globales vs variables locales
- Manipulation de fichiers en Python

# Fonctions

- Définition (création) d'une fonction

```
def calculerSomme(chiffre1, chiffre2):  
    somme = chiffre1 + chiffre2  
    return somme
```

- Appel (utilisation) d'une fonction

```
entree1 = float(input("Veuillez entrer le premier chiffre : "))  
entree2 = float(input("Veuillez entrer le deuxième chiffre : "))  
  
resultat = calculerSomme(entree1, entree2)  
  
print(resultat)  
print(calculerSomme(entree1, entree2))
```

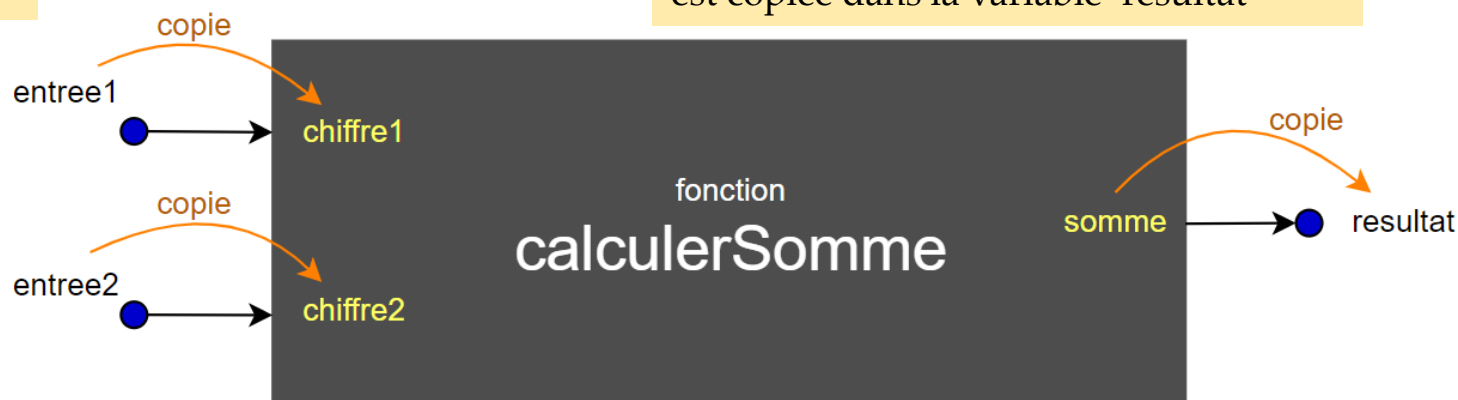
Les valeurs des variables '**entree1**' et '**entree2**' sont **copiées** dans les variables internes '**chiffre1**' et '**chiffre2**'

Si les valeurs de '**chiffre1**' et '**chiffre2**' sont **changées** dans la fonction, ceci **n'affecte pas** les variables '**entree1**' et '**entree2**'

**Attention :** si les variables en entrée sont des **listes** ou des **éléments d'une liste**, les valeurs **ne sont pas copiées** mais utilisées directement.

Si ces **valeurs** changent à l'intérieur de la fonction, elles **changent** également dans la **liste original**

La valeur de la variable interne '**somme**' est copiée dans la variable '**resultat**'



# Fonctions

- Fonction avec deux arguments et aucun retour

```
def genererSalutation(heure, age):  
    salutation = ""  
  
    if heure < 18:  
        salutation += "Bonjour "  
    elif heure < 24:  
        salutation += "Bonsoir "  
    else:  
        salutation += "Salut "  
  
    if age < 12:  
        salutation += "enfant"  
    elif age < 18:  
        salutation += "ado"  
    elif age <= 30:  
        salutation += "jeune"  
    else:  
        salutation += "moins jeune"  
  
    salutation += ", comment ça va ?"  
  
    print (salutation)
```

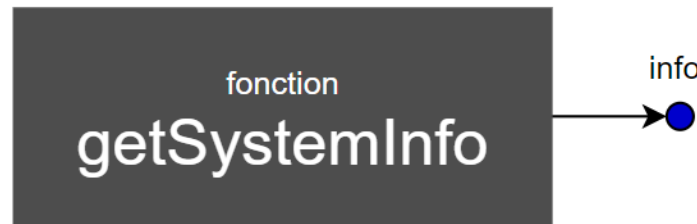


- Appel (utilisation) de cette fonction :

```
info1 = int(input("Veuillez entrer l'heure actuelle (entre 0 et 24, sans minutes ni secondes) : "))  
info2 = int(input("Veuillez entrer votre âge (juste le nombre d'années) : "))  
  
genererSalutation(info1, info2)
```

# Fonctions

- Fonction sans arguments et un retour



```
def getSystemInfo():  
    info = "OS: " + os.name + " - Plate-forme: " + platform.system() + \  
        " - Version: " + platform.release() + " - Utilisateur: " + os.getlogin()  
    return info
```

- Appel (utilisation) de cette fonction :

```
print(getSystemInfo())
```

# Fonctions

- Fonction sans arguments ni retour

```
fonction  
afficherAlea100
```

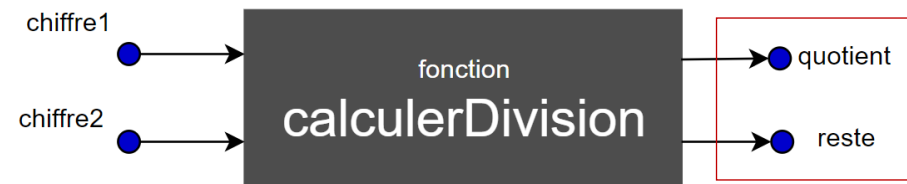
```
def afficherAlea100():  
    print("Voici un chiffre aléatoire entre 0 et 100 : ", round(random()*100))
```

- Appel (utilisation) de cette fonction :

```
afficherAlea100()
```

# Fonctions

- Fonction avec deux arguments et deux sorties



Ces valeurs sont retournées dans un *tuple*  
Si nécessaire, le *tuple* pourrait être converti en liste

```
def calculerDivision(chiffre1, chiffre2):  
    quotient = chiffre1 // chiffre2  
    reste = chiffre1 % chiffre2  
    return quotient, reste
```

- Appel (utilisation) de cette fonction :

```
resultatDivision = calculerDivision(5, 3)  
print("Division 5 / 3. Voici le Quotient et le Reste : ", resultatDivision)
```

- Résultat :

```
Division 5 / 3. Voici le Quotient et le Reste : (1, 2) Tuple
```

# Variables globales vs variables locales

- Variables globales
  - Variables **définies dans le programme**, à l'extérieur des fonctions
  - Ces variables sont visibles (donc utilisables) partout dans le programme
- Variables locales
  - Variables **définies à l'intérieur d'une fonction**
  - Ces variables sont visibles (utilisables) uniquement à l'intérieur de la fonction

```
1 rayonTerre = 6378
2
3 def calculerSurfaceTerre():
4     surfaceTerre = 3.1416 * rayonTerre * rayonTerre
5
6 print(rayonTerre)
7 print(surfaceTerre)
```

Unresolved reference 'surfaceTerre'

Create function 'surfaceTerre' Alt+Shift+Enter More actions... Alt+Enter

**Problème :** la variable locale **surfaceTerre** n'est pas visible dans le programme

```
1 rayonTerre = 6378
2
3 def calculerSurfaceTerre():
4     surfaceTerre = 3.1416 * rayonTerre * rayonTerre
5     return surfaceTerre
6
7 surface = calculerSurfaceTerre()
8
9 print(rayonTerre)
10 print(surface)
```

**Solution 1:** fonction avec retour et assignation dans le programme principal

```
1 rayonTerre = 6378
2 surfaceTerre = 0
3
4 def calculerSurfaceTerre():
5     surfaceTerre = 3.1416 * rayonTerre * rayonTerre
6
7 print(rayonTerre)
8 print(surfaceTerre)
```

**Solution 2:** utilisation de la variable globale 'surfaceTerre'

```
1 rayonTerre = 6378
2
3 def calculerSurface(rayon):
4     surface = 3.1416 * rayon * rayon
5     return surface
6
7 surfaceTerre = calculerSurface(rayonTerre)
8
9 print(rayonTerre)
10 print(surfaceTerre)
```

**Solution 3 (optimale) :** Fonction avec paramètre et retour



# Manipulation de fichiers en Python

- La manipulation se fait avec une variable (un objet) de type **file**.
  - Une série de fonctions de la variable **file** permettent d'ouvrir, lire, écrire et fermer le fichier cible.

Si on n'indique pas un chemin, ce fichier se trouve dans le même répertoire que le programme exécuté

- Accès à un fichier :

```
nomVariable = open(chemin/nom_fichier, [mode d'accès], [buffer])
```

Ce fichier se trouve dans le dossier 'data'

```
fichierEmployes = open("employees.txt")
```

Permet d'accéder au fichier `employees.txt` en mode **lecture**

```
fichierEmployes = open("C:/data/employees.txt")
```

Permet d'accéder au fichier `C:/data/employees.txt` en mode **lecture**

```
fichierEmployes = open("employees.txt", "r" )
```

Permet d'accéder au fichier `employees.txt` en mode **lecture**

```
fichierEmployes = open("employees.txt", "w" )
```

Permet d'accéder au fichier `employees.txt` en mode **écriture**

```
fichierEmployes = open("employees.txt", "a" )
```

Permet d'accéder au fichier `employees.txt` en mode **ajout**

```
fichierEmployes = open("employees.txt", "r+" )
```

Permet d'accéder au fichier `employees.txt` en mode **lecture/écriture**

```
fichierEmployes = open("employees.txt")
```

# Manipulation de fichiers en Python

- Lecture d'un fichier :

```
fichierEmployes.read()
```

Retourne le contenu du fichier **au complet**

```
fichierEmployes.readLine()
```

Retourne le contenu de la **ligne courante** du fichier

```
fichierEmployes.readlines()
```

Retourne une **liste** contenant, dans chaque position, **une ligne** du fichier

- Écriture dans un fichier

```
fichierEmployes.write(valeur)
```

Écrit, dans le fichier, la valeur spécifiée

```
fichierEmployes.writeLines(liste)
```

Écrit, dans le fichier, la **liste** de **valeurs** spécifiées

```
fichierEmployes = open("employees.txt")
```

# Manipulation de fichiers en Python

- Information sur l'état de la manipulation du fichier

```
fichierEmployes.name
```

Indique le nom du fichier pointé par la variable *fichierEmployes*

```
fichierEmployes.mode
```

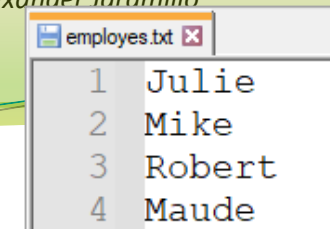
Indique le mode d'accès établi au moment de l'accès au fichier

```
fichierEmployes.closed
```

Indique si l'accès au fichier est maintenant fermé

Tous les attributs : [https://www.w3schools.com/python/python\\_ref\\_file.asp](https://www.w3schools.com/python/python_ref_file.asp)

Toutes les fonctions : [https://www.w3schools.com/python/python\\_ref\\_file.asp](https://www.w3schools.com/python/python_ref_file.asp)



1	Julie
2	Mike
3	Robert
4	Maude

# Manipulation de fichiers texte

Exemple : lecture d'un fichier et placement de son contenu dans une variable

```
1 cheminFichier = "employees.txt"
2
3 # Lecture du contenu du fichier au complet
4 fichierEmployes = open(cheminFichier, "r")
5
6 contenu = fichierEmployes.read()
7
8 fichierEmployes.close()
9
10 print(contenu)
```

```
Julie
Mike
Robert
Maude
```

## Attention :

Le dernier caractère de chaque ligne est un 'saut de ligne' (`\n`)  
C'est pourquoi chaque nom est affiché dans une ligne différente

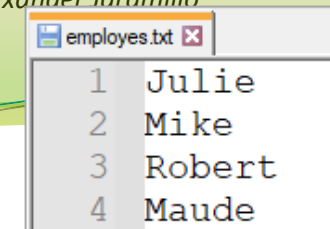
Exemple : lecture de lignes spécifiques d'un fichier

```
1 cheminFichier = "employees.txt"
2
3 fichierEmployes = open(cheminFichier, "r")
4
5 # Lecture de la première ligne du fichier
6 ligne1 = fichierEmployes.readline()
7 print(ligne1, end='')
8
9 # Lecture de la deuxième ligne du fichier
10 ligne2 = fichierEmployes.readline()
11 print(ligne2, end='')
12
13 fichierEmployes.close()
```

```
Julie
Mike
```

## Attention :

`end=''` est utilisé pour éviter un double saut de ligne (`\n`) dans l'affichage. Chaque ligne possède déjà un saut de ligne à la fin



1	Julie
2	Mike
3	Robert
4	Maude

# Manipulation de fichiers texte

Exemple : lecture d'un fichier et placement de son contenu dans une liste

```
1 cheminFichier = "employees.txt"
2
3 listeEmployes = []
4
5 fichierEmployes = open(cheminFichier, "r")
6
7 listeEmployes = fichierEmployes.readlines()
8
9 fichierEmployes.close()
10
11 print(listeEmployes)
```

```
['Julie\n', 'Mike\n', 'Robert\n', 'Maude']
```

**Attention** : des caractères de saut de ligne (`\n`) se trouvent à la fin de chaque ligne du fichier lu (sauf pour la dernière ligne)

Fonction `rstrip()` :  
Pour enlever des caractères 'indésirables' à la fin d'une chaîne de caractères

```
1 message1 = "Bonjour ! Je suis là !!!"
2 message2 = message1.rstrip('!')
3
4 print(message1)
5 print(message2)
```

```
Bonjour ! Je suis là !!!
Bonjour ! Je suis là
```

Exemple : exemple précédent sans les caractères de saut de ligne

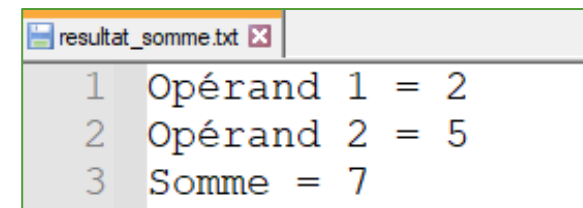
```
1 cheminFichier = "employees.txt"
2
3 listeEmployes = []
4
5 fichierEmployes = open(cheminFichier, "r")
6
7 listeTemporaire = fichierEmployes.readlines()
8
9 fichierEmployes.close()
10
11 for element in listeTemporaire:
12     listeEmployes.append(element.rstrip('\n'))
13
14 print(listeEmployes)
```

```
['Julie', 'Mike', 'Robert', 'Maude']
```

# Manipulation de fichiers texte

## Exemple : écriture dans un fichier

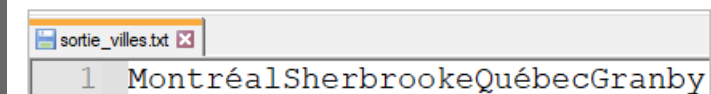
```
1 cheminFichier = "resultat_somme.txt"
2
3 a = 2
4 b = 5
5
6 somme = a + b
7
8 fichierSomme = open(cheminFichier, "w")
9
10 ligne1 = "Opérand 1 = " + str(a) + "\n"
11 ligne2 = "Opérand 2 = " + str(b) + "\n"
12 ligne3 = "Somme = " + str(somme) + "\n"
13
14 fichierSomme.write(ligne1 + ligne2 + ligne3)
15
16 fichierSomme.close()
```



```
1 Opérand 1 = 2
2 Opérand 2 = 5
3 Somme = 7
```

## Exemple : écriture d'une liste dans un fichier

```
1 cheminFichier = "sortie_villes.txt"
2
3 listeVilles = ["Montréal", "Sherbrooke", "Québec", "Granby"]
4
5 fichierVilles = open(cheminFichier, "w")
6
7 fichierVilles.writelines(listeVilles)
8
9 fichierVilles.close()
```



```
1 MontréalSherbrookeQuébecGranby
```

### Attention :

Des caractères de saut de ligne (`\n`) ne sont pas ajoutés après chaque élément de la liste automatiquement

# Manipulation de fichiers csv

Exemple : lecture d'un fichier csv

```
1 import csv
2
3 cheminFichier = "employees.csv"
4
5 listeEmployes = []
6
7 fichierEmployes = open(cheminFichier, "r")
8
9 csvReader = csv.reader(fichierEmployes, delimiter = ',')
10
11 for line in csvReader:
12     print(line)
13     listeEmployes.append(line)
14
15 fichierEmployes.close()
```

employees.csv	
1	Julie,Gagnon,25,Finances
2	Mike,Ford,30,Production
3	Robert,Lacombe,55,Direction
4	Maude,Pratte,35,Direction

```
['Julie', 'Gagnon', '25', 'Finances']
['Mike', 'Ford', '30', 'Production']
['Robert', 'Lacombe', '55', 'Direction']
['Maude', 'Pratte', '35', 'Direction']
```

À remarquer:

- La variable **csvReader** permet de lire le contenu du fichier et placer **chaque ligne dans une liste**
- **listeEmployes** est, à tout fin pratique, une **liste 2D**
- Toutes les données de cette liste sont de type **str** (chaîne de caractères).
- Si on veut faire des calculs/comparaisons avec les **âges**, il faut les **convertir** en **int** ou **float**

# Manipulation de fichiers csv

Exemple précédent avec un affichage plus intéressant :

```
1 import csv
2
3 cheminFichier = "employees.csv"
4
5 listeEmployes = []
6
7 fichierEmployes = open(cheminFichier, "r")
8
9 csvReader = csv.reader(fichierEmployes, delimiter = ',')
10
11 for line in csvReader:
12     listeEmployes.append(line)
13
14 fichierEmployes.close()
15
16 for i in range(len(listeEmployes)):
17     for j in range(len(listeEmployes[i])):
18         print(listeEmployes[i][j], end='\t')
19     print()
```

1	Julie,Gagnon,25,Finances
2	Mike,Ford,30,Production
3	Robert,Lacombe,55,Direction
4	Maude,Pratte,35,Direction

Julie	Gagnon	25	Finances
Mike	Ford	30	Production
Robert	Lacombe	55	Direction
Maude	Pratte	35	Direction



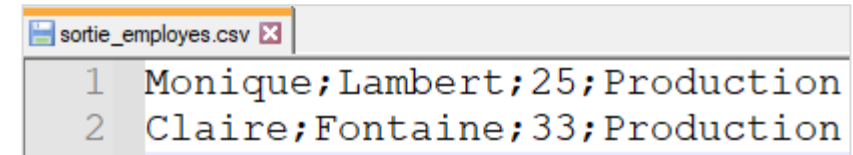
Guide de manipulation de fichiers csv avec Python:

<https://www.programiz.com/python-programming/reading-csv-files>

# Manipulation de fichiers csv

Exemple : écriture de listes dans un fichier csv

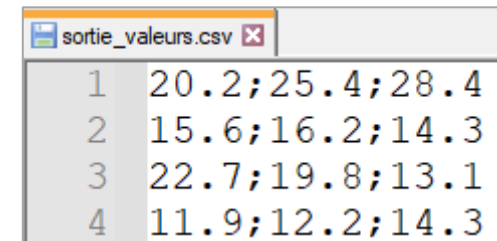
```
1 import csv
2
3 cheminFichier = "sortie_employes.csv"
4
5 employe1 = ["Monique", "Lambert", 25, "Production"]
6 employe2 = ["Claire", "Fontaine", 33, "Production"]
7
8 fichierEmployes = open(cheminFichier, "w")
9
10 csvWriter = csv.writer(fichierEmployes, delimiter=';', lineterminator='\n')
11
12 csvWriter.writerow(employe1)
13 csvWriter.writerow(employe2)
14
15 fichierEmployes.close()
```



1	Monique; Lambert; 25; Production
2	Claire; Fontaine; 33; Production

Exemple : écriture d'une liste 2D dans un fichier csv

```
1 import csv
2
3 cheminFichier = "sortie_valeurs.csv"
4
5 liste2D = [ [20.2, 25.4, 28.4],
6             [15.6, 16.2, 14.3],
7             [22.7, 19.8, 13.1],
8             [11.9, 12.2, 14.3] ]
9
10 fichierValeurs = open(cheminFichier, "w")
11
12 csvWriter = csv.writer(fichierValeurs, delimiter=';', lineterminator='\n')
13
14 csvWriter.writerows(liste2D)
15
16 fichierValeurs.close()
```



1	20.2; 25.4; 28.4
2	15.6; 16.2; 14.3
3	22.7; 19.8; 13.1
4	11.9; 12.2; 14.3

# Références intéressantes

- Types de fichiers et manipulation avec Python
  - .txt ([https://www.tutorialspoint.com/python/python\\_files\\_io.htm](https://www.tutorialspoint.com/python/python_files_io.htm))
  - .csv (<https://www.programiz.com/python-programming/reading-csv-files>)
  - .xml ([xml parser tutorial](#))
  - .json ([https://www.w3schools.com/python/python\\_json.asp](https://www.w3schools.com/python/python_json.asp))