

420-436-SH Développement de scripts

P5 – Scripts Python

Partie 1

Plan

- Scripts Python : généralités, installation et configuration
- Scripts Python : programmation
 - Syntaxe
 - Options de débogage
 - Variables
 - Librairies
 - Paramètres d'affichage
 - Saisie d'entrées utilisateur
 - Tableaux (Listes)
 - Opérateurs
 - Arguments
 - Structures de base

Scripts *Python*

- Python
 - C'est un langage de programmation de plus en plus utilisé.
 - Python est un langage hybride (interprété – compilé)
 - Python peut être utilisé pour créer des scripts d'automatisation de tâches du système
 - Python permet la création d'**interfaces graphiques**
 - Python peut être utilisé dans des système Windows, Linux, et MAC
- Comment créer un script Python ?
 - L'interpréteur Python doit être installé dans la machine
 - À l'aide d'un éditeur de texte (ex : *notepad++*, *notepad*, etc.)
 - À l'aide de *VSCode + extension Python*
 - À l'aide d'autres environnements de programmation comme PyCharm
- Comment installer/configurer l'interpréteur Python + VSCode?
 - Voir fichier Lab5/python.pdf

Programmation de scripts *Python*

Exemple 1

```
1 import os
2
3 # Cette commande fait un clear screen (dans un environnement Windows)
4 #os.system('cls')
5
6
7 # Cette commande retourne le chemin du dossier courant
8 print("Dossier courant = ", os.getcwd())
9
10 # Cette commande affiche le contenu du dossier courant
11 contenu = os.listdir(os.getcwd())
12
13 print("Contenu dossier courant = ", contenu)
```

Indique le chemin et le nom du dossier courant

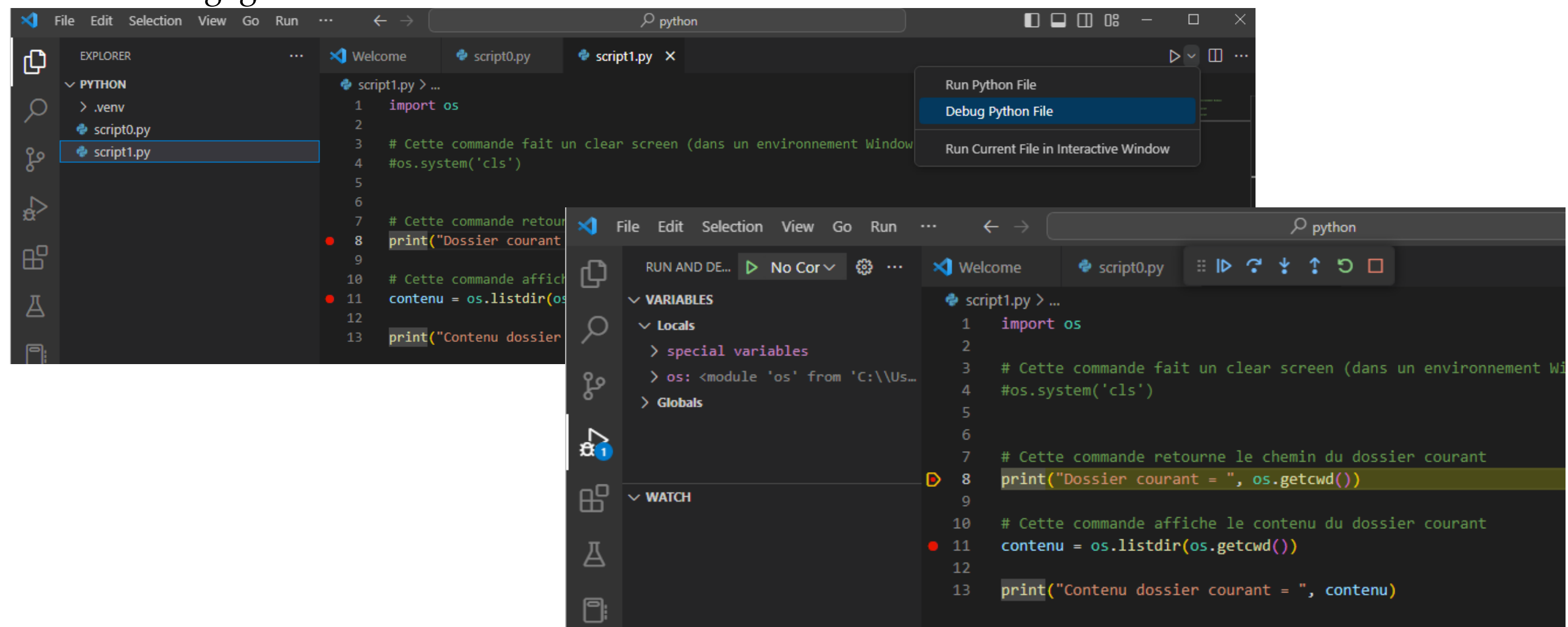
Retourne la liste d'objets qui se trouvent dans le dossier courant

Résultat de l'exécution du script

```
Dossier courant = C:\python
Contenu dossier courant = ['.venv', 'script0.py', 'script1.py']
```

Programmation de scripts *Python*

- Débogage



Src: <https://code.visualstudio.com/docs/editor/debugging#:~:text=To%20bring%20up%20the%20Run,shortcut%20Ctrl%2BShift%2BD>

Programmation de scripts *Python*

- Débogage

Debug actions

Once a debug session starts, the **Debug toolbar** will appear on the top of the editor.



Action	Explanation
Continue / Pause <code>F5</code>	Continue: Resume normal program/script execution (up to the next breakpoint). Pause: Inspect code executing at the current line and debug line-by-line.
Step Over <code>F10</code>	Execute the next method as a single command without inspecting or following its component steps.
Step Into <code>F11</code>	Enter the next method to follow its execution line-by-line.
Step Out <code>Shift+F11</code>	When inside a method or subroutine, return to the earlier execution context by completing remaining lines of the current method as though it were a single command.
Restart <code>Ctrl+Shift+F5</code>	Terminate the current program execution and start debugging again using the current run configuration.
Stop <code>Shift+F5</code>	Terminate the current program execution.

Programmation de scripts *Python*

- Création de variables
 - Python est un langage typé à la base, mais il fonctionne comme un langage **non typé** : il n'est pas obligatoire d'indiquer le type de variable lors de sa **déclaration** (création)
 - On peut indiquer le type de variable lors de sa déclaration
 - Si le type n'est pas déclaré, celui-ci est défini implicitement lors de la première utilisation de la variable

Déclaration typée en Python

```
a: int = 3
b: float = 5.47
c: str = "Bonjour"
d: bool = False
```

Déclaration non-typée en Python

```
a = 3
b = 5.47
c = "Bonjour"
d = False
```

- Utilisation des variables

```
print(a)
print("Voici la valeur de a : ", a)
print(b)
print(c)
print(d)
```

Résultat

```
3
Voici la valeur de a : 3
5.47
Bonjour
False
```

Programmation de scripts *Python*

- Utilisation de variables

```
9  # Variables
10 chiffre1 = 2.3
11 chiffre2 = 2
12 chiffre3 = 3
13
14 somme = chiffre1 + chiffre2 #
15
16 print(somme)
17 print("Le résultat de la somme est", somme)
18
19 # Quelques opérateurs mathématiques
20 soustraction = chiffre1 - chiffre2
21 multiplication = chiffre1 * chiffre2
22 division = chiffre1 / chiffre2
23 quotient = chiffre1 // chiffre2
24 reste = chiffre1 % chiffre2
25 puissance = chiffre3 ** chiffre2
26
27 print("Résultat de la somme = ", somme)
28 print("Résultat de la soustraction = ", soustraction)
29 print("Résultat de la multiplication = ", multiplication)
30 print("Résultat de la division = ", division)
31 print("Quotient de la division = ", quotient)
32 print("Reste de la division = ", reste)
33 print("Résultat de la puissance = ", puissance)
```

Exécution

```
4.3
Le résultat de la somme est 4.3
Résultat de la somme = 4.3
Résultat de la soustraction = 0.29999999999999998
Résultat de la multiplication = 4.6
Résultat de la division = 1.15
Quotient de la division = 1.0
Reste de la division = 0.29999999999999998
Résultat de la puissance = 9
```


Programmation de scripts *Python*

- Utilisation de librairies

```
9  # Importation d'une librairie déjà présente dans l'environnement du projet
10 import math
11
12 # Définition de variables
13 chiffre1 = 5.42618
14 chiffre2 = 9
15
16 # Affichage de PI
17 print("pi = ", math.pi)
18
19 # Fonction puissance
20 print(math.pow(2, 3))
21
22 # fonction arrondi (à deux decimaux)
23 arrondi = round(chiffre1, 2)
24
25 print("Arrondi du chiffre1:", arrondi)
26 print("Arrondi du chiffre1:", round(chiffre1, 2))
27
28 # Calcul d'une racine carrée
29 racine = math.sqrt(chiffre2)
30
31 print("La racine carrée de", chiffre2, "est", racine)
32
33 # Partie entière d'un chiffre décimal
34 print("Partie entière (trunc) de", chiffre1, "est:", math.trunc(chiffre1))
35 print("Partie entière (trunc) de {0} est: {1}".format(chiffre1, math.trunc(chiffre1)))
36
37 # Extraction de l'entier inférieur (floor) d'un chiffre décimal
38 print("Entier inférieur (floor) de", chiffre1, "est:", math.floor(chiffre1))
39
40 # Extraction de l'entier supérieur (ceiling) d'un chiffre décimal
41 print("Entier supérieur (ceil):", chiffre1, "est:", math.ceil(chiffre1))
42
43 nom = "Marco"
44 age = 25
45
46 # Utilisation de paramètres de formatage de texte
47 print("Je m'appelle", nom, "et j'ai", age, "ans")
48 print("Je m'appelle {0} et j'ai {1} ans".format(nom, age))
```

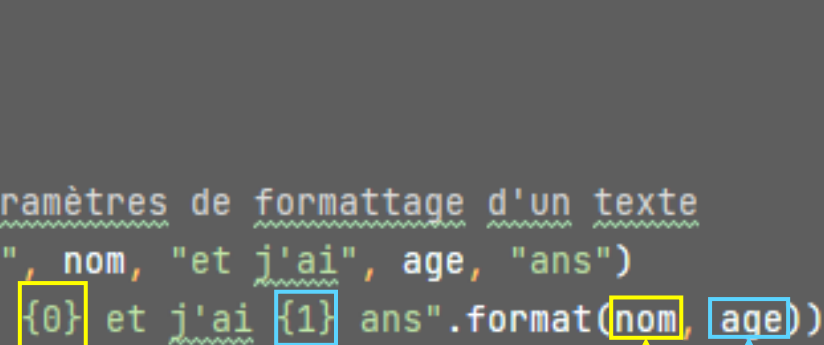
Exécution

```
pi = 3.141592653589793
8.0
Arrondi du chiffre1: 5.43
Arrondi du chiffre1: 5.43
La racine carrée de 9 est 3.0
Partie entière (trunc) de 5.42618 est: 5
Partie entière (trunc) de 5.42618 est: 5
Entier inférieur (floor) de 5.42618 est: 5
Entier supérieur (ceil): 5.42618 est: 6
Je m'appelle Marco et j'ai 25 ans
Je m'appelle Marco et j'ai 25 ans
```

Programmation de scripts *Python*

- Paramètres d'affichage

```
42 nom = "Marco"
43 age = 25
44
45 # Utilisation de paramètres de formatage d'un texte
46 print("Je m'appelle", nom, "et j'ai", age, "ans")
47 print("Je m'appelle {0} et j'ai {1} ans".format(nom, age))
```



Paramètre 0

Paramètre 1

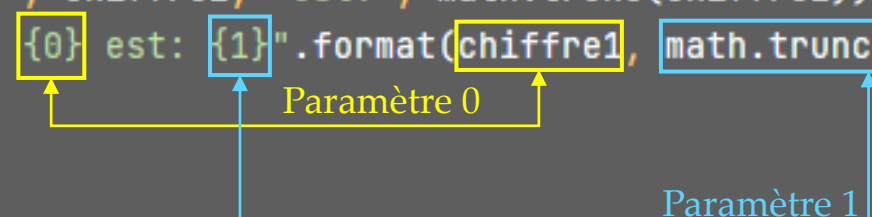
Exécution

```
Je m'appelle Marco et j'ai 25 ans
Je m'appelle Marco et j'ai 25 ans
```

Programmation de scripts *Python*

- Paramètres d'affichage

```
chiffre1 = 5.42618
31 # Partie entière d'un chiffre décimal
32 print("Partie entière (trunc) de", chiffre1, "est:", math.trunc(chiffre1))
33 print("Partie entière (trunc) de {0} est: {1}".format(chiffre1, math.trunc(chiffre1)))
```



Exécution

```
Partie entière (trunc) de 5.42618 est: 5
```

Programmation de scripts Python

- Installation de bibliothèques

```
10 import datetime
11 import pytz
12
13 # Définition de variables de type datetime
14 moment1 = datetime.datetime(2020, 7, 7, 14, 5, 15)
15 moment2 = datetime.datetime(2021, 7, 8, 14, 30, 00)
16 moment3 = datetime.datetime.fromisoformat("2021-07-08 16:30:55")
17
18 print(moment1)
19 print(moment2)
20 print(moment3)
21
22 # Il est possible d'extraire les composants d'une datetime
23 print("Année moment1 :", moment1.year)
24 print("Mois moment1 :", moment1.month)
25 print("Jour moment1 :", moment1.day)
26 print("Heures moment1 :", moment1.hour)
27 print("Minutes moment1 :", moment1.minute)
28 print("Secondes moment1 :", moment1.second)
29
30 # Il est possible de calculer une période de temps
31 periodeA = moment2 - moment1
32 periodeB = moment3 - moment2
33
34 print("Période A : ", periodeA)
35 print("Période B : ", periodeB)
36 print("Nombre de jours période A : ", periodeA.days)
37 print("Nombre de jours période B : ", periodeB.days)
38 print("Nombre de secondes période B : ", periodeB.seconds)
39 print("Nombre d'heures période B : ", periodeB.seconds/3600)
40
41 # Obtenir la date/heure courantes (pour la zone horaire du système)
42 maintenantSysteme = datetime.datetime.now()
43 print("Date/heure courante du système :", maintenantSysteme)
44
45 # Obtenir la date/heure courantes (pour une zone horaire donnée)
46 maintenantPacifique = datetime.datetime.now(tz=pytz.timezone("Canada/Pacific"))
47 # print("Date/heure courante (pacifique) :", maintenantPacifique)
48
49 # Pour obtenir la liste de toutes les zones horaires
50 print(pytz.all_timezones)
```

La bibliothèque **pytz** n'est pas présente dans le système. Il faut l'installer :

```
(.venv) PS C:\python> pip install pytz
Collecting pytz
  Downloading pytz-2022.7.1-py2.py3-none-any.whl (499 kB)
    499.4/499.4 kB 10.4 MB/s eta 0:00:00
Installing collected packages: pytz
Successfully installed pytz-2022.7.1
```

Exécution :

```
2020-07-07 14:05:15
2021-07-08 14:30:00
2021-07-08 16:30:55
Année moment1 : 2020
Mois moment1 : 7
Jour moment1 : 7
Heures moment1 : 14
Minutes moment1 : 5
Secondes moment1 : 15
Période A : 366 days, 0:24:45
Période B : 2:00:55
Nombre de jours période A : 366
Nombre de jours période B : 0
Nombre de secondes période B : 7255
Nombre d'heures période B : 2.015277777777778
Date/heure courante du système : 2023-03-15 16:52:16.992235
['Africa/Abidjan', 'Africa/Accra', 'Africa/Addis_Ababa', 'Africa/Alg
/Blancpain', 'Africa/Brazzaville', 'Africa/Bujumbura', 'Africa/Cairo',
```

Programmation de scripts *Python*

- Saisie d'entrées utilisateur

```
9  # La fonction 'input' permet de saisir une entrée clavier
10 # Dans ce cas, la variable 'valeur' stocke le chiffre entré
11 valeur = input("Veuillez entrer un chiffre : ")
12 print("Vous avez entré : ", valeur)
13
14 # La fonction 'type' permet de connaître le type d'une variable
15 # En principe, l'entrée par clavier est de type text (str)
16 print("Type de valeur entrée : ", type(valeur))
17
18 # Pour convertir l'entrée clavier en chiffre entier, on effectue une conversion de type
19 valeur = int(valeur)
20 print("Type de valeur convertie : ", type(valeur))
21
22 # On peut convertir l'entrée clavier en chiffre entier directement
23 valeur = int(input("Veuillez entrer un chiffre : "))
24 print("Type de valeur convertie : ", type(valeur))
25
26 couleur = input("Veuillez entrer une couleur : ")
27 print("Le ciel est : ", couleur)
```

Exécution :

```
Veillez entrer un chiffre : 4
Vous avez entré : 4
Type de valeur entrée : <class 'str'>
Type de valeur convertie : <class 'int'>
Veillez entrer un chiffre : 5
Type de valeur convertie : <class 'int'>
Veillez entrer une couleur : rouge
Le ciel est : rouge
```

Programmation de scripts *Python*

- Tableaux : Python peut manipuler des tableaux facilement. Les tableaux sont appelés des **Listes**

```
30  tableauFruits = ["Pomme", "Orange", "Banane"]
31
32  print(tableauFruits[0])
33  print(tableauFruits[1])
34  print(tableauFruits[2])
35  print(tableauFruits)
36
37  print("Taille du tableau = ", len(tableauFruits))
38  print( "Dernier élément tableau =", tableauFruits[len(tableauFruits) - 1] )
39  print( "Dernier élément tableau =", tableauFruits[-1] )
```

Résultat

```
Pomme
Orange
Banane
['Pomme', 'Orange', 'Banane']
Taille du tableau = 3
Dernier élément tableau = Banane
Dernier élément tableau = Banane
```

Programmation de scripts Python

Quelques opérations *built-in* sur une liste

```
temperatures = [15.2, 22.3, 25.5, 17.9, 30.14, 25.5, 12.4, 10.8, 26.6, 25.9]
```

`len(temperatures)` Pour obtenir le nombre d'éléments (*length*) de la liste 10

`temperatures[len(temperatures) - 1]` Pour obtenir le dernier élément de la liste 25.9

`temperatures[-1]` Pour obtenir le dernier élément de la liste (méthode abrégée) 25.9

`max(temperatures)` Pour obtenir la valeur maximale dans la liste 30.14

`min(temperatures)` Pour obtenir la valeur minimale dans la liste 10.8

`temperatures[2:5]` Pour obtenir les éléments qui se trouvent entre les positions 2 et 4 de la liste [25.5, 27.9, 30.14]

`temperatures.count(25.5)` Pour obtenir le nombre de fois que l'élément 25.5 se trouve dans la liste 2

`temperatures.index(12.4)` Pour obtenir l'index (la position) de l'élément 12.4 dans la liste 6

`temperatures.append(8.8)` Pour insérer un nouvel élément à la fin de la liste [15.2, 22.3, 25.5, 27.9, 30.14, 25.5, 12.4, 10.8, 26.6, 25.9, 8.8]

`temperatures.insert(3, 16)` Pour insérer un nouvel élément à la position 3 de la liste [15.2, 22.3, 25.5, 16, 27.9, 30.14, 25.5, 12.4, 10.8, 26.6, 25.9, 8.8]

`temperatures.remove(25.5)` Pour supprimer l'élément 25.5 de la liste [15.2, 22.3, 16, 27.9, 30.14, 25.5, 12.4, 10.8, 26.6, 25.9, 8.8]

`del temperatures[3]` Pour supprimer l'élément qui se trouve à la position 3 de la liste [15.2, 22.3, 16, 30.14, 25.5, 12.4, 10.8, 26.6, 25.9, 8.8]

`temperatures.reverse()` Pour inverser l'ordre des éléments dans la liste [8.8, 25.9, 26.6, 10.8, 12.4, 25.5, 30.14, 16, 22.3, 15.2]

`temperatures.sort()` Pour trier les éléments de la liste en ordre croissant [8.8, 10.8, 12.4, 15.2, 16, 22.3, 25.5, 25.9, 26.6, 30.14]

`temperatures.sort(reverse=True)` Pour trier les éléments de la liste en ordre décroissant [30.14, 26.6, 25.9, 25.5, 22.3, 16, 15.2, 12.4, 10.8, 8.8]

Programmation de scripts *Python*

Opérateurs arithmétiques

Operator	Name	Example
+	Addition	<code>x + y</code>
-	Subtraction	<code>x - y</code>
*	Multiplication	<code>x * y</code>
/	Division	<code>x / y</code>
%	Modulus	<code>x % y</code>
**	Exponentiation	<code>x ** y</code>

Src : https://www.w3schools.com/python/python_operators.asp

Programmation de scripts *Python*

Opérateurs booléens

Operator	Name	Example
<code>==</code>	Equal	<code>x == y</code>
<code>!=</code>	Not equal	<code>x != y</code>
<code>></code>	Greater than	<code>x > y</code>
<code><</code>	Less than	<code>x < y</code>
<code>>=</code>	Greater than or equal to	<code>x >= y</code>
<code><=</code>	Less than or equal to	<code>x <= y</code>

Src : https://www.w3schools.com/python/python_operators.asp

Programmation de scripts *Python*

Opérateurs logiques

Operator	Description	Example
and	Returns True if both statements are true	<code>x < 5 and x < 10</code>
or	Returns True if one of the statements is true	<code>x < 5 or x < 4</code>
not	Reverse the result, returns False if the result is true	<code>not(x < 5 and x < 10)</code>

Src : https://www.w3schools.com/python/python_operators.asp

Programmation de scripts *Python*

Paramètres (arguments) d'un Script

- **Paramètre** : une donnée que l'on fournit à un script
- Il est possible de fournir plusieurs paramètres à un script
- Les paramètres sont fournis au moment où le script est exécuté

Code du fichier **script2.py** :

```
1 import sys
2
3 print("Nombre de paramètres passés au script = ", len(sys.argv))
4 print("Premier paramètre passé au script = ", sys.argv[0])
5 print("Tous les paramètres passés au script = ", sys.argv)
```

Appel d'un script **script2.py** sans paramètres

```
c:\scripts>python script2.py
```

Résultat de l'exécution du script

```
Nombre de paramètres passés au script = 1
Premier paramètre passé au script = script2.py
Tous les paramètres passés au script = ['script2.py']
```

Appel d'un script **script2.py** avec plusieurs paramètres

```
c:\scripts>python script2.py param1 param2 param3
```

Résultat de l'exécution du script

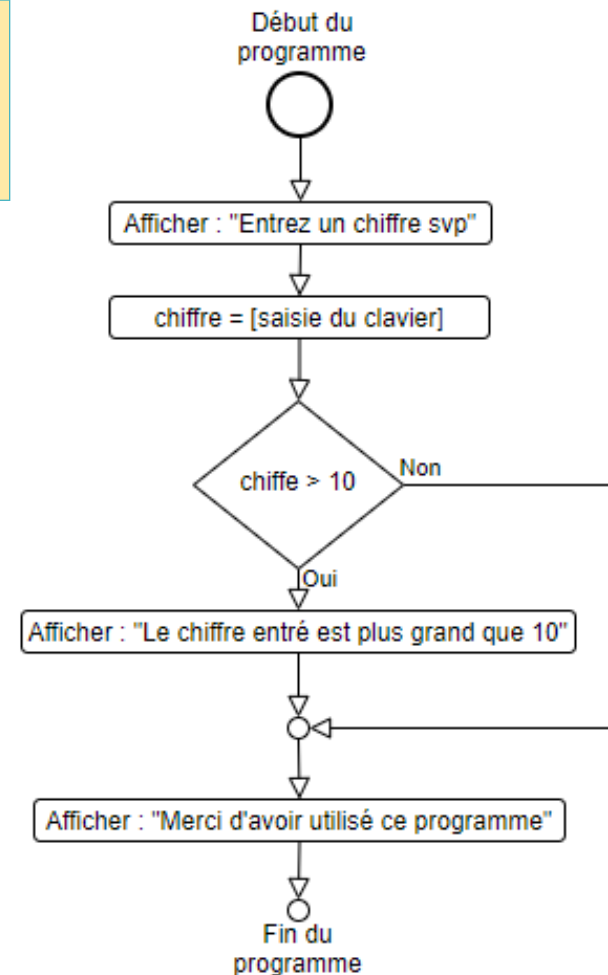
```
Nombre de paramètres passés au script = 4
Premier paramètre passé au script = script2.py
Tous les paramètres passés au script = ['script2.py', 'param1', 'param2', 'param3']
```

Programmation de scripts *Python*

Codage des structures logiques : IF

Exemple :

Afficher un message si le chiffre entré est plus grand que 10



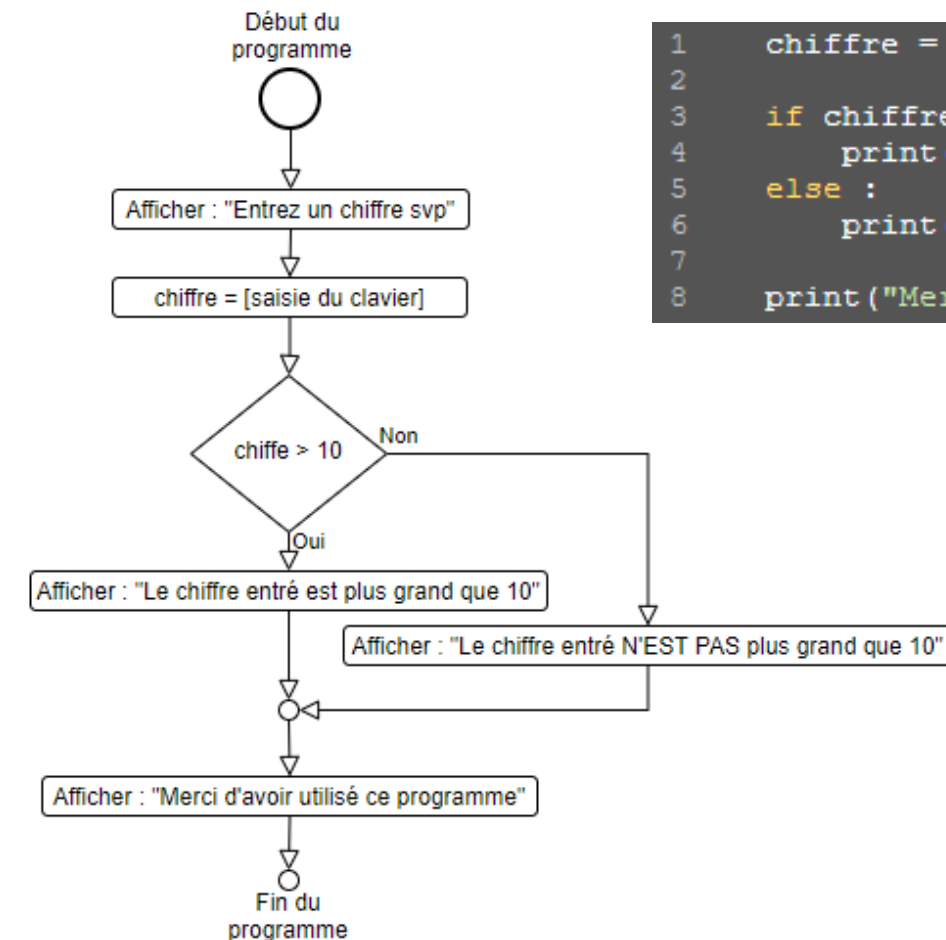
```
1  chiffre = int(input("Entrez un chiffre svp : "))
2
3  if chiffre > 10 :
4      print("Le chiffre entré est plus grand que 10")
5
6  print("Merci d'avoir utilisé ce programme")
```

Programmation de scripts *Python*

Codage des structures logiques : **IF...ELSE**

Exemple :

Afficher un message si le chiffre entré est plus grand que 10. Autrement, afficher un autre message



```
1  chiffre = int(input("Entrez un chiffre svp : "))
2
3  if chiffre > 10 :
4      print("Le chiffre entré est plus grand que 10")
5  else :
6      print("Le chiffre entré N'EST PAS plus grand que 10")
7
8  print("Merci d'avoir utilisé ce programme")
```

Programmation de scripts *Python*

Codage des structures logiques : **IF...ELIF..ELSE**

```
1  chiffre = int(input("Entrez un chiffre svp : "))
2
3  if chiffre == 10 :
4      print("Le chiffre entré est 10")
5  elif chiffre > 10 :
6      print("Le chiffre entré est plus grand que 10")
7  else :
8      print("Le chiffre entré N'EST PAS plus grand que 10")
9
10 print("Merci d'avoir utilisé ce programme")
```

Codage des structures logiques : **IF imbriqués**

```
1  pays = input("Quel est votre pays de naissance ? ")
2  if pays == "Canada":
3      province = input("Quelle est votre province ? ")
4      if province == "Québec":
5          langue = input("Quelle est votre lange de choix ? (fr/en) ")
6          if langue == "fr":
7              print("Bonjour !")
8  print("Fin du programme")
```

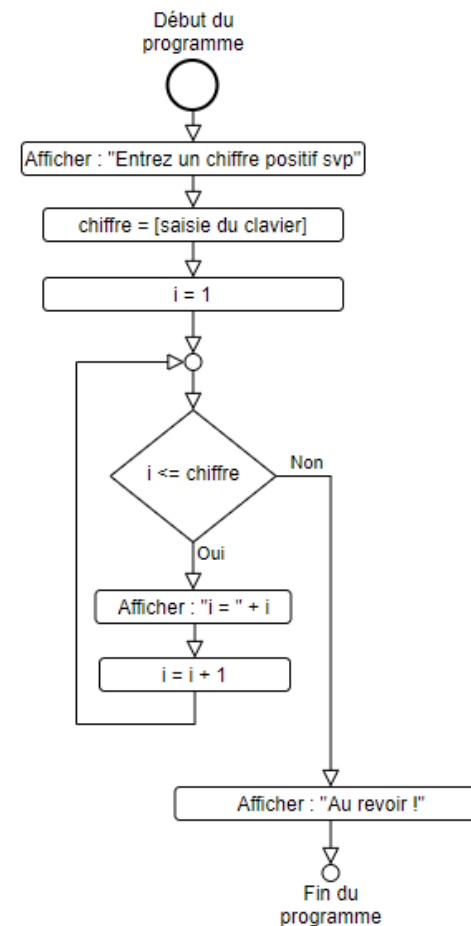
Conditions multiples :

```
1  revenuAnnuel = float(input("Veuillez entrer votre revenu annuel : "))
2  nombreEnfants = int(input("Veuillez entrer le nombre d'enfants à charge : "))
3
4  if revenuAnnuel <= 35000 and nombreEnfants > 0:
5      print("Vous avez le droit au crédit d'impôts maximal")
6  elif revenuAnnuel <= 35000 or nombreEnfants > 0:
7      print("Vous avez le droit à un crédit d'impôts")
8  else:
9      print("Vous n'avez pas le droit à des crédits d'impôts")
```

Programmation de scripts *Python*

Codage des structures logiques : Boucle **WHILE**

Exemple :
Afficher les chiffres
à l'écran **entre 1 et le**
chiffre entré par
l'utilisateur



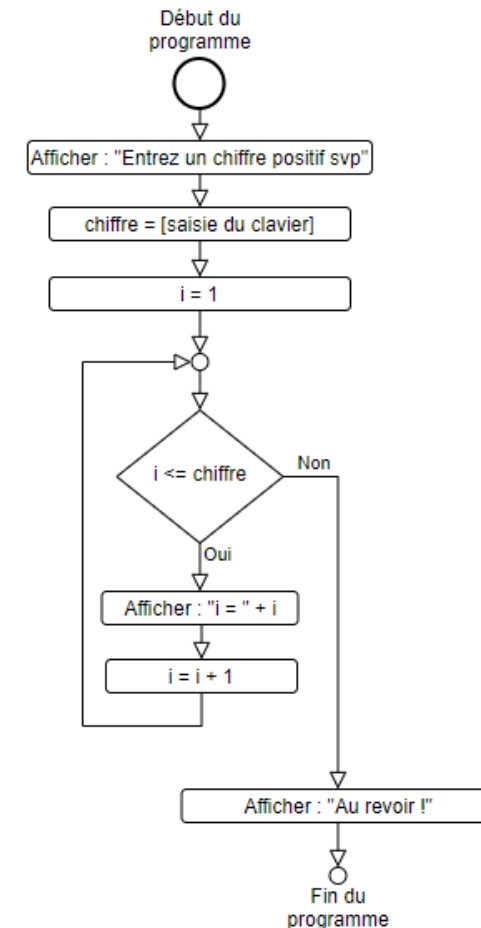
```
1  chiffre = int(input("Entrez un chiffre positif svp "))
2
3  i = 1
4
5  while i <= chiffre:
6      print("i = ", i)
7      i = i + 1;
8  print("Au revoir !")
```

Programmation de scripts *Python*

Codage des structures logiques : Boucle **FOR**

Exemple :

Afficher les chiffres
à l'écran **entre 1 et le
chiffre entré** par
l'utilisateur



```
1  chiffre = int(input("Entrez un chiffre positif svp "))
2
3  for i in range(1, chiffre+1):
4      print("i = ", i)
5
6  print("Au revoir !")
```

chiffre = 5
range(chiffre) → 0, 1, 2, 3, 4

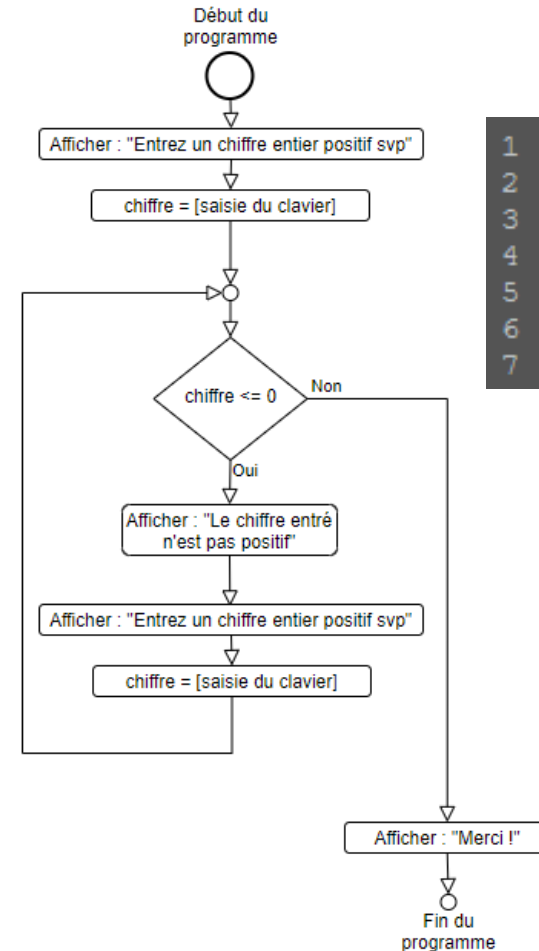
range(1, chiffre+1) → 1, 2, 3, 4, 5

Programmation de scripts *Python*

Codage des structures logiques : Boucle **WHILE**

Exemple :

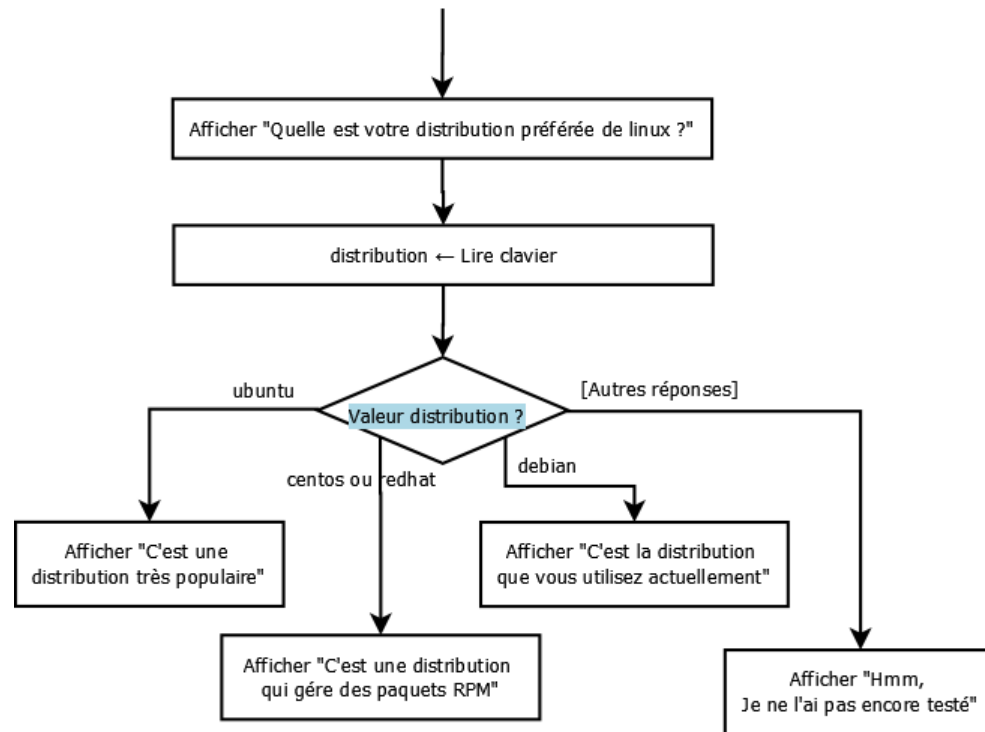
Poser toujours la même question **tant que** l'entrée reçue ne respecte pas les conditions exigées



```
1  chiffre = int(input("Entrez un chiffre entier positif svp "))
2
3  while chiffre <= 0:
4      print("Le chiffre entré n'est pas positif")
5      chiffre = int(input("Entrez un chiffre entier positif svp "))
6
7  print("Merci !")
```

Programmation de scripts *Python*

Codage des structures logiques : **SWITCH...CASE**



```
distribution = "Kali"

match distribution:
    case "Ubuntu":
        print("C'est une distribution très populaire.")

    case "Centos":
        print("C'est une distribution qui gère des paquets RPM.")

    case "Red Hat":
        print("C'est une distribution qui gère des paquets RPM.")

    case "Kali":
        print("C'est une distribution pour les tests en sécurité")

    case _:
        print("Hmm. Je ne l'ai pas encore testée.")
```

SWITCH...CASE est implémenté par un MATCH...CASE en Python

Références intéressantes

- Documentation Python

<https://docs.python.org/fr/3/>

- La référence du langage Python

<https://docs.python.org/fr/3/reference/index.html>

- *Python Tutorial*

https://bugs.python.org/file47781/Tutorial_EDIT.pdf