

## INTRODUCTION

**Chap. 1 :** *Installation de Pycharm ; Version Community de l'édition la plus récente*

**Chap. 2 :** *Premier programme en langage python*

**Chap. 3 :** *Variables et Opérateurs*

**Chap. 4 :** *Les entrées au clavier*

**Chap. 5 :** *Instructions Conditionnelles*

**Chap. 6 :** *Instructions Itératives*

**Chap. 7 :** *Mon premier jeu avec le langage Python*

**Chap. 1** : Installation de Pycharm ; Version Community  
de l'édition la plus récente

## **Chap. 2 : Premier programme en langage python**

Pour commencer à écrire un programme, il faudra avoir quelques instructions de départ.

### **1. Insertion d'un commentaire**

C'est une partie du code qui n'est pas interpréter par la machine. Cela permet d'apporter des explications aux programme et aux lignes de code, de sorte à toujours comprendre le programme et même permettre à un novice ou un groupe de personne de comprendre plus facilement le programme.

Alors, en python pour insérer un commentaire, on utilise le symbole **#** suivi du commentaire

**Exemple :**    **# Ceci est un commentaire sur une ligne en python**

### **2. Affichage dans la console**

Pour afficher les informations liées au programme, on utilise l'instruction ***print***

Syntaxe : `print(' l'information à afficher ')`

Avec l'instruction `print`, on peut afficher des informations (entre guillemets) dans la console

**Exemple :**    **# Ceci est une instruction qui permet d'afficher une info en python**  
`print (" J'aime coder en python ")`

### **3. Cas pratique**

Suivez les étapes décrites :

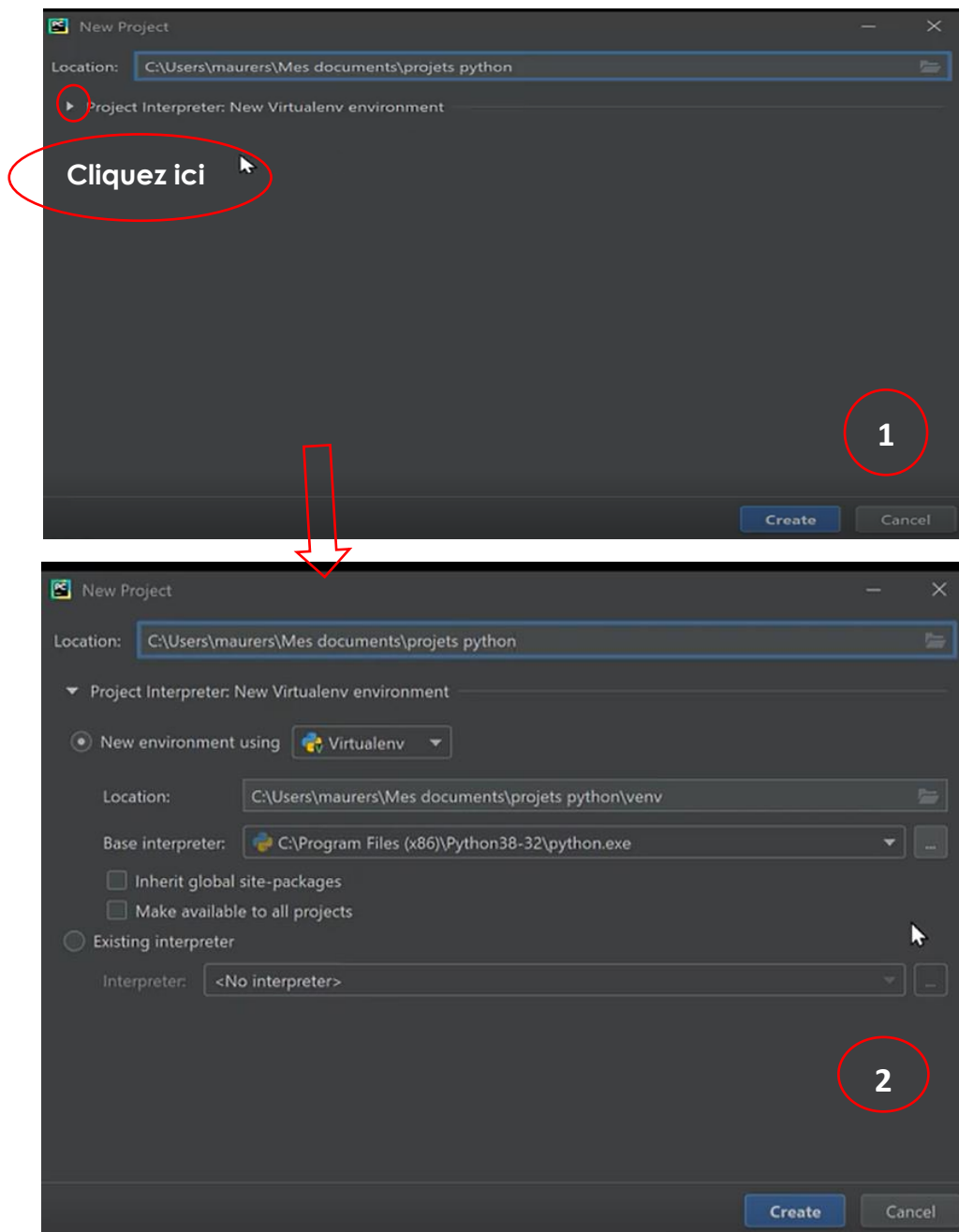
- Ouvrez le logiciel Pycharm

Vous verrez un message de bienvenu si c'est votre première fois d'utiliser le logiciel

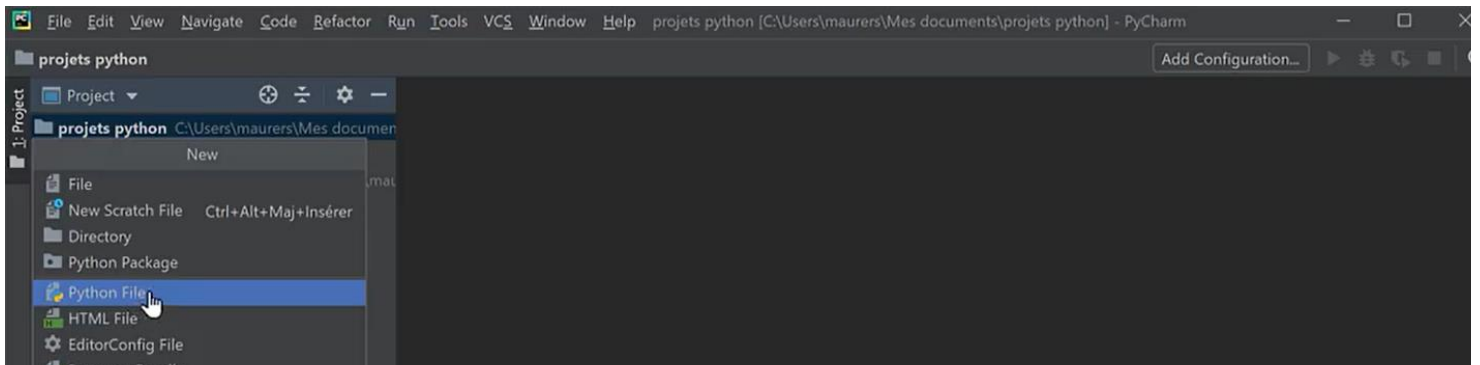
- Cliquer sur « **Create new project** » puis donner un nom au projet
- Choisissez l'emplacement du programme à travers le chemin qui vous sera proposé

Vous pouvez vérifier si votre environnement virtuel (il permet d'exécuter les programmes qu'on va écrire sur Pycharm) est bien installé en cliquant sur le triangle indiquant la liste déroulante.

Si conforme, vous verrez les informations suivantes :

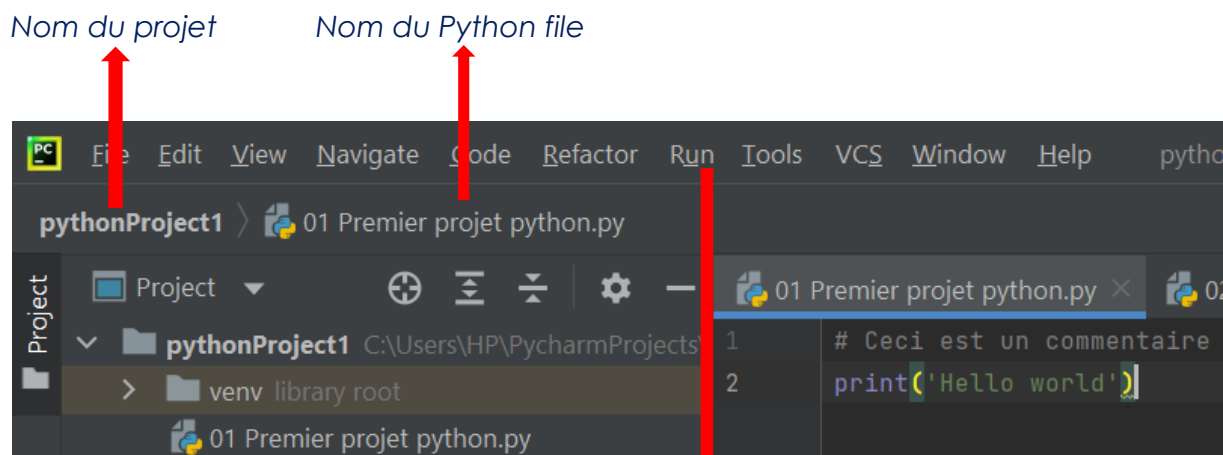


- Après avoir cliqué sur **create**, vous serez dans l'interface. Vous allez maintenant créer un nouveau fichier **Python file** auquel vous donnerai un nom

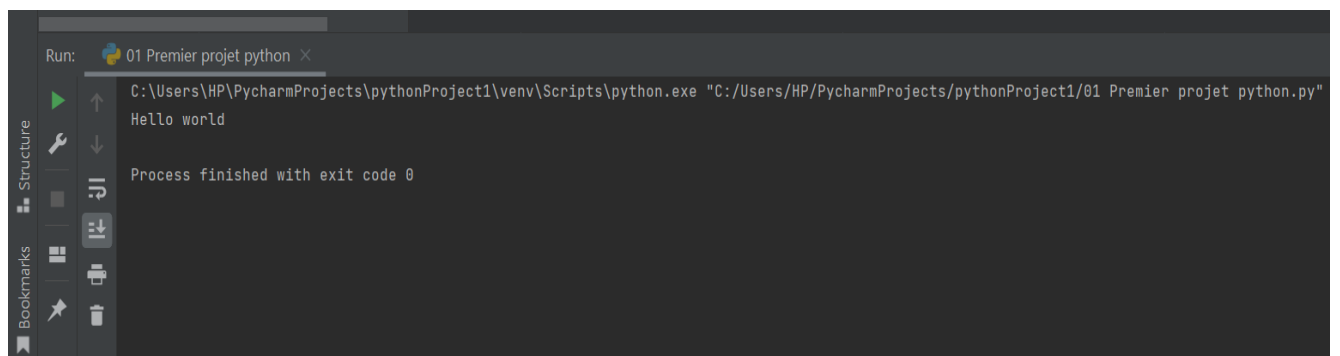


**EXERCICE 1 :** Ecris un programme qui comporte un commentaire de départ expliquant l'objectif de ce premier programme puis qui affiche « **Hello World** »

**Solution :**



Pour compiler le programme, on clique sur « **run** » et vous verrez afficher au bas de votre écran ;



## Chap. 3 : Variables et Opérateurs

### 1. Utilisation d'une variable

Une variable est associée à un espace mémoire. Elle est utilisée pour stocker une valeur qui peut être de différentes type ; soit un nombre, une chaîne de caractère, un booléen, ...

Ce qui est bien avec python, c'est qu'on n'est pas obligé de déclarer explicitement les variables ; C'est python qui va se charger de réserver un espace approprié à la variable en fonction de sa valeur.

- Une variable est identifiée par un nom. Ce nom doit commencer par une lettre et ne peut contenir ni espace, ni caractères spéciaux
- On peut éventuellement commencer par écrire le nom de la variable avec un « Under score » c'est-à-dire « \_ » : Personnellement je n'utilise pas ce genre d'écriture

Pour affecter une valeur à une variable (c'est-à-dire stocker une valeur dans une variable), on utilise le symbole **=** suivi de la valeur.

#### Utilisation d'une variable (instruction Python)

```
# On crée une variable appelée nombre1 et on lui affecte la valeur 5
nombre1 = 5
# On crée une variable appelée nombre2 et on lui affecte la valeur -3,4
nombre2 = -3.4
# On crée une variable appelée somme et on lui affecte la somme nombre1 + nombre2
somme = nombre1 + nombre2
# On crée une variable appelée produit et on lui affecte le produit nombre1 x nombre2
produit = nombre1 * nombre2
# On crée une variable appelée quotient et on lui affecte le quotient nombre1/nombre2
quotient = nombre1 / nombre2
# On crée une variable appelée puissance3 et on lui affecte la valeur nombre1 au cube
puissance3 = nombre1 **3
# On crée une variable appelée phrase et on lui affecte la valeur "Hello World !"
phrase = "Hello World !"
```

### 2. Affichage dans la console

Pour afficher les informations liées au programme, on utilise l'instruction **print**

#### Affichage dans la console (instruction Python)

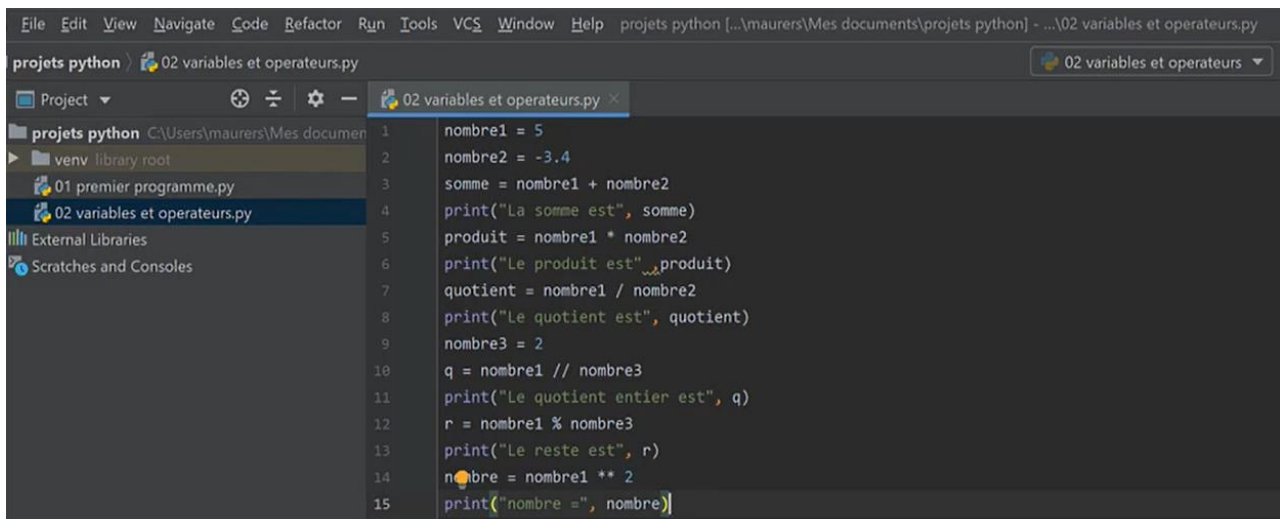
```
# Avec l'instruction print, on peut afficher un message (avec guillemets) dans la console
print("J'aime bien apprendre le langage Python.")
# Avec l'instruction print, on peut afficher le contenu d'une variable (sans guillemet)
# dans la console
phrase = "Hello World !"
print(phrase)
```

**Affichage dans la console (instruction Python)**

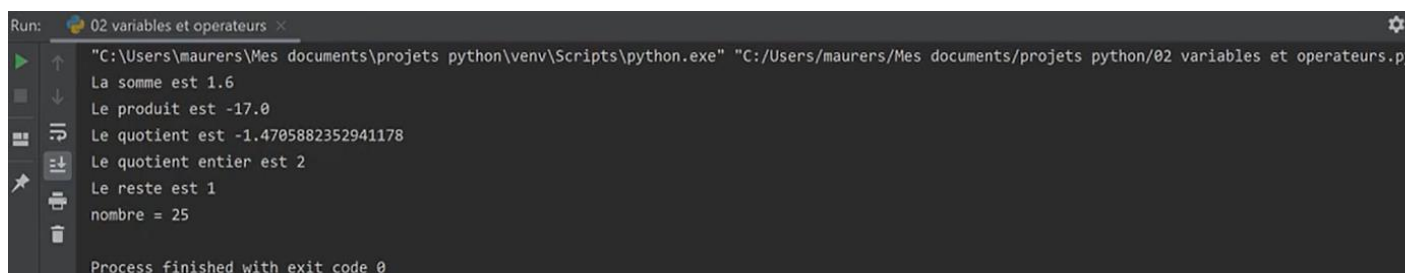
```
# On peut combiner l'affichage de messages et de variables
nom = "Pierre"
age = 16
print(nom, "est âgé de", age, "ans.")
```

**EXERCICE 2 :**

1. Déclarer deux variable nombre1 et nombre2
2. Faites toutes les opérations avec ces nombres puis afficher le résultat à la console.

**Solution :****Programme :**

```
File Edit View Navigate Code Refactor Run Tools VCS Window Help projets python [...]\maurers\Mes documents\projets python] - ...02 variables et operateurs.py
projets python 02 variables et operateurs.py
Project 02 variables et operateurs.py
projets python C:\Users\maurers\Mes documents
venv library root
01 premier programme.py
02 variables et operateurs.py
External Libraries
Scratches and Consoles
1 nombre1 = 5
2 nombre2 = -3.4
3 somme = nombre1 + nombre2
4 print("La somme est", somme)
5 produit = nombre1 * nombre2
6 print("Le produit est", produit)
7 quotient = nombre1 / nombre2
8 print("Le quotient est", quotient)
9 nombre3 = 2
10 q = nombre1 // nombre3
11 print("Le quotient entier est", q)
12 r = nombre1 % nombre3
13 print("Le reste est", r)
14 nombre = nombre1 ** 2
15 print("nombre =", nombre)
```

**Après le run :**

```
Run: 02 variables et operateurs
"C:\Users\maurers\Mes documents\projets python\venv\Scripts\python.exe" "C:/Users/maurers/Mes documents/projets python/02 variables et operateurs.p
La somme est 1.6
Le produit est -17.0
Le quotient est -1.4705882352941178
Le quotient entier est 2
Le reste est 1
nombre = 25
Process finished with exit code 0
```

En python ;

**\** désigne une division décimale : la division normale qu'on connaît.

**//** désigne une division entière c'est-à-dire retourne que la partie entière de la division.

**%** retourne le reste de la division

**\*\*** est utilisé pour faire les calculs de puissance ;

## Chap. 4 : Les entrées au clavier

Dans ce chapitre, nous allons voir les instructions à utiliser pour récupérer des données taper au clavier et les stocker dans des variables pour les utiliser après dans notre programme

### 1. Instruction à utiliser

- ❖ Pour stocker une chaîne de caractère saisie au clavier, on utilise l'instruction **input**.

#### Saisie d'une chaîne de caractères au clavier (instruction Python)

```
# L'instruction input suivi d'un message permet de stocker une chaîne caractères dans une  
# variable  
phrase = input("Entrer votre nom.")
```

**NB :** cette instruction ne permet que de stocker une chaîne de caractère.

La variable **phrase** va contenir une chaîne de caractère grâce au symbole **=** qui désigne une affectation.

- ❖ Pour stocker un nombre saisi au clavier, on utilise l'instruction input puis on fait une conversion dans le type de donnée adapté

#### Saisie d'un nombre au clavier (instruction Python)

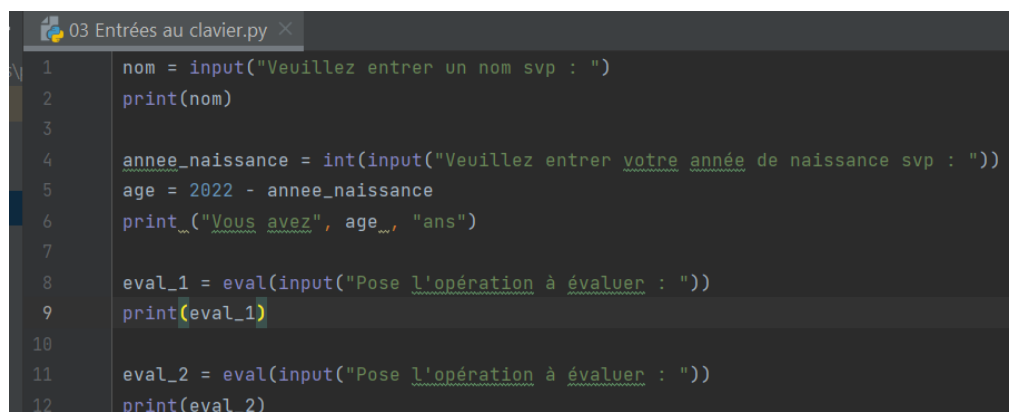
```
# Saisie et conversion d'un entier  
nombre_entier = int(input("Entrer un nombre entier : "))  
# Saisie et conversion d'un décimal  
nombre_decimal = float(input("Entrer un nombre décimal : "))  
# Saisie et conversion d'un nombre sous forme quelconque (exemple : 7/2)  
nombre_decimal = eval(input("Entrer un nombre réel : "))
```

### 2. EXERCICE

1. Ecrire un programme qui affiche votre nom après que vous ayez entré un nom.
2. Ecrire un programme qui calcule votre âge après avoir rentré votre date de naissance.
3. Ecrire un programme qui évalue les opérations  $1+1$  et  $10^2$ .

### SOLUTION :

#### Programme :



```
03 Entrées au clavier.py x
1 nom = input("Veuillez entrer un nom svp : ")
2 print(nom)
3
4 annee_naissance = int(input("Veuillez entrer votre année de naissance svp : "))
5 age = 2022 - annee_naissance
6 print("Vous avez", age, "ans")
7
8 eval_1 = eval(input("Pose l'opération à évaluer : "))
9 print(eval_1)
10
11 eval_2 = eval(input("Pose l'opération à évaluer : "))
12 print(eval_2)
```



## Après le run :

```

03 Entrées au clavier X
C:\Users\HP\PycharmProjects\pythonProject1\venv\Scripts\python.exe "C:/Users/HP/PycharmProjects/pythonProject1/03 Entrées au clavier.py"
Veuillez entrer un nom svp : DHYLAN
DHYLAN
Veuillez entrer votre année de naissance svp : 2001
Vous avez 21 ans
Pose l'opération à évaluer : 1+1
2
Pose l'opération à évaluer : 10**2
100

```

## Chap. 5 : Instructions Conditionnelles

Dans le déroulement d'un algorithme, certaines instructions sont exécutées que si une condition a été satisfaite auparavant. La condition est une comparaison entre deux valeurs du même type.

- ❖ En langage mathématique, les opérateurs de comparaison sont :  $=$ ,  $\neq$ ,  $<$ ,  $>$ ,  $\leq$  et  $\geq$
- ❖ En langage Python, ces opérateurs de comparaison s'écrivent simultanément :  $==$ ,  $!=$ ,  $<$ ,  $>$ ,  $<=$  et  $>=$

### 1. Les instructions conditionnelles

- ❖ **Si... alors...**

En langage Python, l'instruction s'écrit : **if condition :**

Le bloc d'instruction qui suit le **if** sera placé en dessous et décalé d'une tabulation (en général 3 ou 4 caractères).

On dit que les instructions sont **indentées**

### Exemple :

```

Si ... alors ... (instruction Python)

# Saisie d'une chaîne de caractères
mot_de_passe = input("Entrer le mot de passe.")
# Test de la validité du mot de passe qui est sesame
# Si le mot de passe proposé est correct alors le message s'affiche sinon le programme se termine
if mot_de_passe == "sesame":
    print("Merci. Vous pouvez entrer.")

```

**Nb :** le langage Python (comme plusieurs autres langage) est sensible à la casse. Les chaînes de caractère Sesame et sesame sont différentes.

La condition peut comporter plusieurs tests.

Les opérateurs reliant ces tests sont **ET**, **OU**.

- **(condition1) ET (condition2) vraie**, signifie que la *condition1* est vraie et que la *condition2* est vrai

Les deux conditions doivent obligatoirement être vraies pour que le test soit vrai.

- **(condition1) OU (condition2) vraie**, signifie que la *condition1* est vraie ou que la *condition2* est vrai ou que les deux conditions sont vraies.

En Python, ET s'écrit **and** et OU s'écrit **or**.

Exemple :

#### ET, OU (instruction Python)

```
# Tester si un nombre est compris entre 2 valeurs données
nombre = float(input("Entrer un nombre."))
# On teste si le nombre est compris entre 1 et 2
if nombre >=1 and nombre<=2:
    print("Le nombre appartient à [1;2].")
# On teste si le nombre appartient à ]-infini;1[ U ]2;+infini[
if nombre <1 or nombre>2:
    print("Le nombre appartient à ]-infini;1[ U ]2;+infini[.")
```

❖ Si... alors... sinon

En langage python, l'instruction s'écrit : **if condition:**

*instructions*

**else :**

En reprenant l'exemple précédent, on peut ajouter *instructions*

#### Si ... alors ... sinon (instruction Python)

```
# Saisie d'une chaîne de caractères
mot_de_passe = input("Entrer le mot de passe.")
# Test de la validité du mot de passe qui est sesame
# Si le mot de passe proposé est correct alors le message s'affiche sinon le programme
# se termine
if mot_de_passe == "sesame":
    print("Merci. Vous pouvez entrer.")
else:
    print("Désolé. Le mot de passe proposé est incorrect.")
```

❖ **Branchement conditionnel**

En langage python, l'instruction s'écrit : **if condition :**

*instructions*

**elif condition :**

*instructions*

...

**else :**

*instructions*

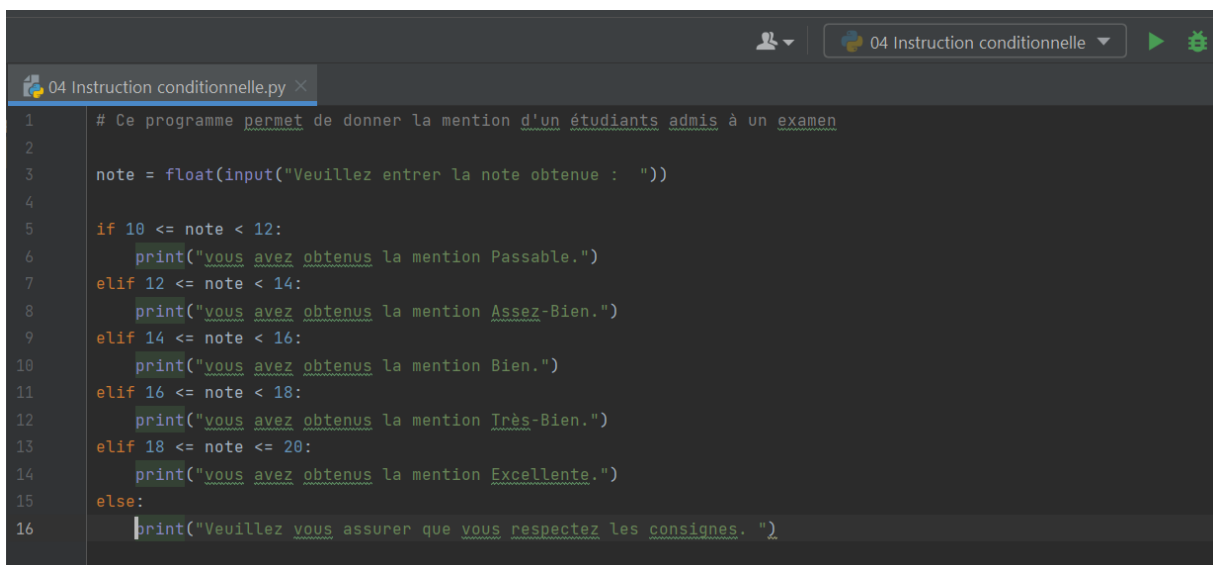
Le **else** est facultatif, mais il permet de traiter tous les cas possibles.

Exemple :**Branchement conditionnel (instruction Python)**

```
# On stocke l'âge de l'utilisateur dans la variable age
age = int(input("Entrer votre age."))
# Branchement conditionnel en fonction de l'âge donné
if age < 3:
    print("Vous êtes un bébé.")
elif age >= 3 and age < 12:
    print("Vous êtes un enfant.")
elif age >= 12 and age < 18:
    print("Vous êtes un adolescent.")
else:
    print("Vous êtes un adulte.")
```

**2. EXERCICE**

Ecrit un programme qui demande d'entrer une note supérieure à 10 et donne par la suite la mention correspondante à cette note.



```
04 Instruction conditionnelle.py
1 # Ce programme permet de donner la mention d'un étudiants admis à un examen
2
3 note = float(input("Veuillez entrer la note obtenue : "))
4
5 if 10 <= note < 12:
6     print("vous avez obtenus la mention Passable.")
7 elif 12 <= note < 14:
8     print("vous avez obtenus la mention Assez-Bien.")
9 elif 14 <= note < 16:
10    print("vous avez obtenus la mention Bien.")
11 elif 16 <= note < 18:
12    print("vous avez obtenus la mention Très-Bien.")
13 elif 18 <= note <= 20:
14    print("vous avez obtenus la mention Excellente.")
15 else:
16    print("Veuillez vous assurer que vous respectez les consignes. ")
```

## Chap. 6 : Instructions Itératives

Les instructions itératives sont aussi appelées les boucles.

Pour cette initiation, nous allons étudier la boucle **for** et la boucle **while**.

### 1. La boucle for

Pour ... allant de ... à ... faire ; ainsi peut être décrite la boucle for.  
Ce type de boucle est appelé boucle bornée.

En effet, le nombre d'itération (répétitions) est connu à l'avance. On utilise donc un compteur qui, une fois atteint une certaine valeur, arrête d'exécution de la boucle.

A chaque passage dans la boucle, le compteur sera incrémenté (avancé de +1) ou décrémenté (soustraire de -1) selon le cas.

En langage python, l'instruction (la syntaxe) s'écrit :

**for ... in range (debut, fin, pas) :** *debut, fin et pas sont des entiers relatifs (cohérents)*  
*instructions*

ou bien

**for ... in range (debut, fin) :** *Ici le pas est de 1*  
*instructions*

ou bien

**for ... in range (fin) :** *Ici debut=0 et pas = 1*  
*instructions*

Le bloc d'instruction qui suit sera indenté d'une tabulation.

Exemple :

**Pour ... allant de ... à ... faire (instruction Python)**

```
# Calcul de 2 à la puissance 3
# Le compteur est incrémenté de 1 à chaque passage dans la boucle
a = 1
for compteur in range (3):
    a = a * 2
print("La valeur de 2 à la puissance 3 est :",a)

# Calcul de la somme des carrés des nombres pairs inférieurs à 10
somme_carres = 0
for i in range(0,10,2):
    somme_carres = somme_carres + i*i
print("La somme des carrés des nombres pairs inférieurs à 10 est :",somme_carres)
```

## 2. La boucle while

Tant que... faire ; ainsi peut être décrite la boucle while.

Avec ce type de boucle, on répète l'exécution tant qu'une condition préalable définie est satisfaite. Le test s'effectue en début de boucle, à chaque passage. Ce type de boucle est appelée **boucle non bornée** ou **boucle conditionnelle**. En effet, le nombre d'**itération** (répétitions) n'est pas connu à l'avance.

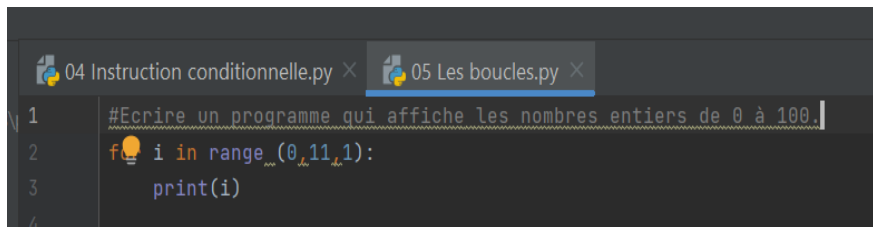
Exemple :

### Tant que ... faire (instruction Python)

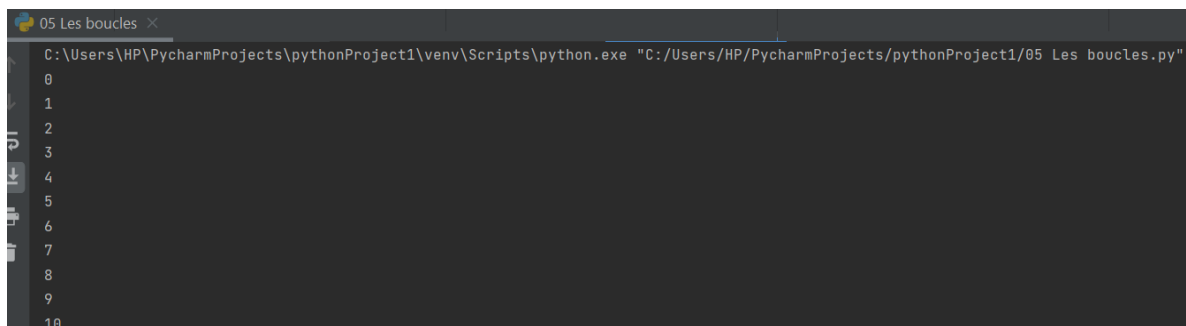
```
# L'utilisateur doit impérativement taper un nombre supérieur ou égal à zéro
x = -1
while x <= 0:
    x = float(input("Entrer un nombre positif ou nul :"))
print("La valeur de x est", x)
```

## EXERCICE

1. Ecrire un programme qui affiche les nombres entiers de 0 à 10 par pas



```
04 Instruction conditionnelle.py × 05 Les boucles.py ×
1 #Ecrire un programme qui affiche les nombres entiers de 0 à 100.
2 for i in range(0,11,1):
3     print(i)
4
```



```
05 Les boucles ×
C:\Users\HP\PycharmProjects\pythonProject1\venv\Scripts\python.exe "C:/Users/HP/PycharmProjects/pythonProject1/05 Les boucles.py"
0
1
2
3
4
5
6
7
8
9
10
```

## Chap. 7 : Mon premier jeu avec le langage Python

Ce chapitre portant le nom de jeu va nous permettre d'utiliser toutes les notions vues dans cette formation.

### 1. Nom du jeu

Le jeu va s'appeler le nombre juste selon Dhyland.

### 2. But du jeu

Le but du jeu est de trouver un nombre tiré au hasard entre 1 et 100 en ayant au plus 10 tentatives.

A chaque proposition du joueur, l'animateur (le programme) répond par

- C'est plus
- C'est moins
- Bravo ! Vous avez trouvé le juste nombre.

Si après 10 tentatives le joueur n'a pas trouvé la bonne réponse, la programme met fin au jeu en annonçant que la partie est perdue.

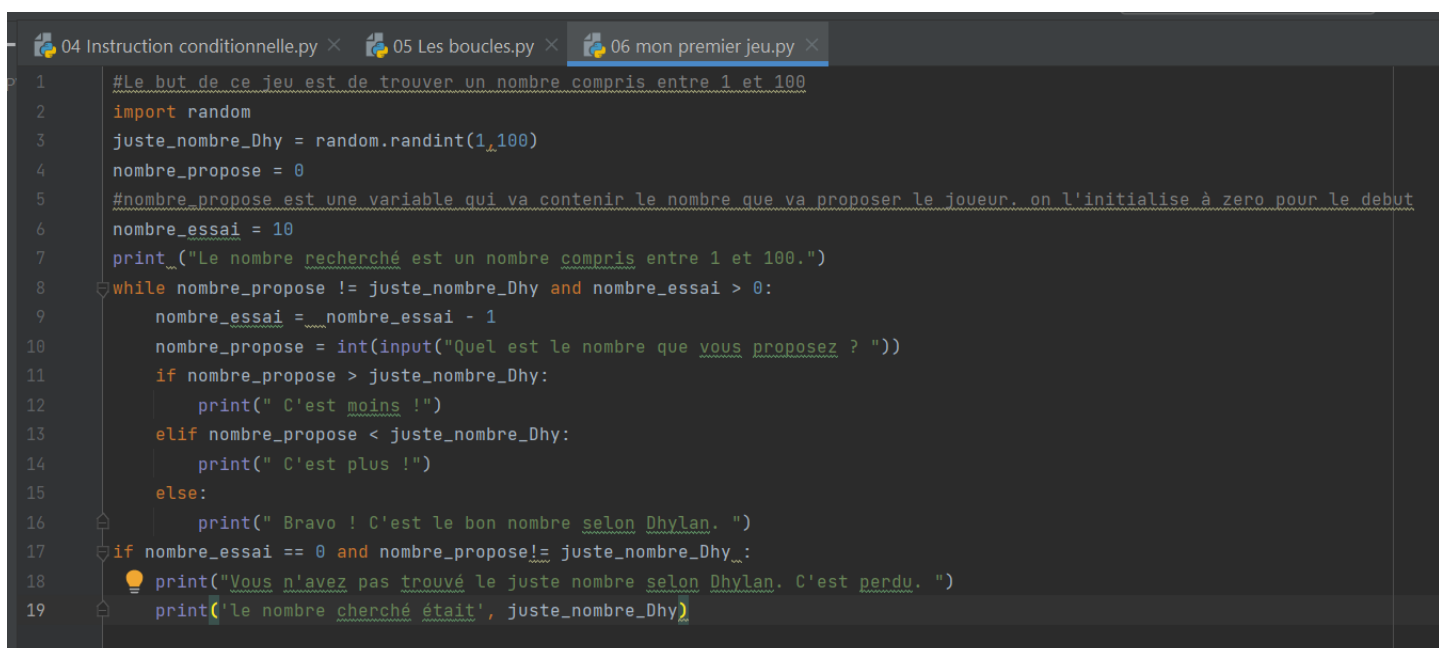
### 3. Indications

Pour programmer ce jeu, on a besoin d'une fonction qui va tirer au hasard un nombre entre 1 et 100.

Cette fonction n'a pas encore été étudié donc il est nécessaire de la préciser.

Le programme doit commencer par : **import random**

`juste_nombre_dhy = random.randint(1,100)`



```
04 Instruction conditionnelle.py x 05 Les boucles.py x 06 mon premier jeu.py x
1 #Le but de ce jeu est de trouver un nombre compris entre 1 et 100
2 import random
3 juste_nombre_Dhy = random.randint(1,100)
4 nombre_propose = 0
5 #nombre_propose est une variable qui va contenir le nombre que va proposer le joueur. on l'initialise à zero pour le debut
6 nombre_essai = 10
7 print("Le nombre recherché est un nombre compris entre 1 et 100.")
8 while nombre_propose != juste_nombre_Dhy and nombre_essai > 0:
9     nombre_essai = nombre_essai - 1
10    nombre_propose = int(input("Quel est le nombre que vous proposez ? "))
11    if nombre_propose > juste_nombre_Dhy:
12        print(" C'est moins !")
13    elif nombre_propose < juste_nombre_Dhy:
14        print(" C'est plus !")
15    else:
16        print(" Bravo ! C'est le bon nombre selon Dhyland. ")
17 if nombre_essai == 0 and nombre_propose != juste_nombre_Dhy:
18     print("Vous n'avez pas trouvé le juste nombre selon Dhyland. C'est perdu. ")
19 print("le nombre cherché était", juste_nombre_Dhy)
```