

# Initiation à la Programmation

**Jour 1 : Fondamentaux**

**Langage Python**

# Table des matières

1. Langage pour cette formation : Python
2. IDE : PyCharm Community
3. Exécution d'un premier script Python
4. Syntaxe : les premières règles pour coder en Python
5. Exercices récapitulatifs

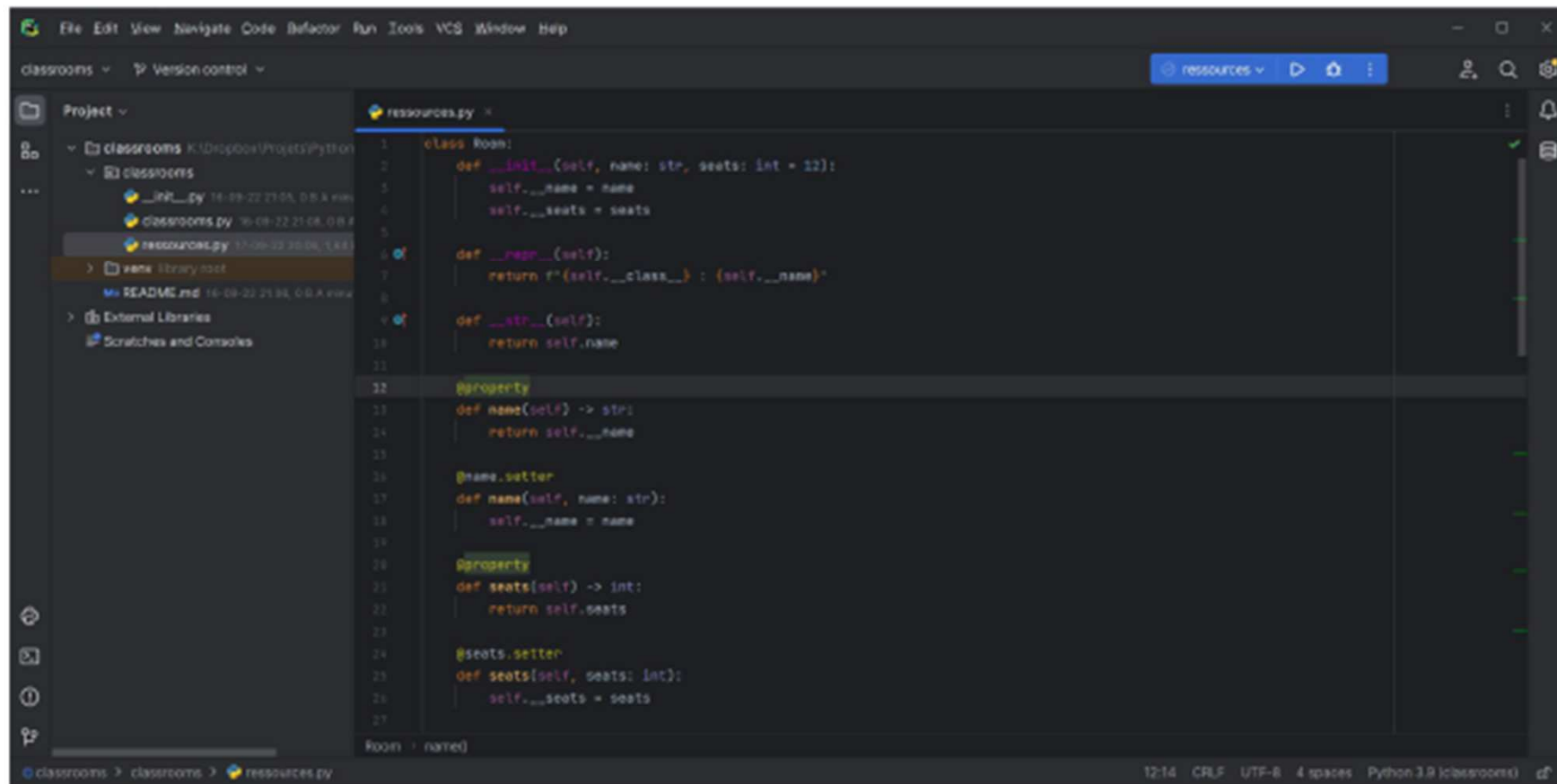
# Langage pour cette formation : Python

- **Généralités de ce langage de programmation**
  1. **Langage répandu & polyvalent:** création de sites web, création de logiciels, création d'applications back-end, machine learning/big data, développement de jeux vidéo...
  2. **Langage de haut-niveau** ⇔ indépendant du matériel
  3. **Langage interprété** : interpréteur traduit en langage machine chaque ligne et l'exécute
  4. **Librairies** : Python possède de nombreuses librairies telles que NumPy, IPython, matplotlib et pandas

# Pycharm Community

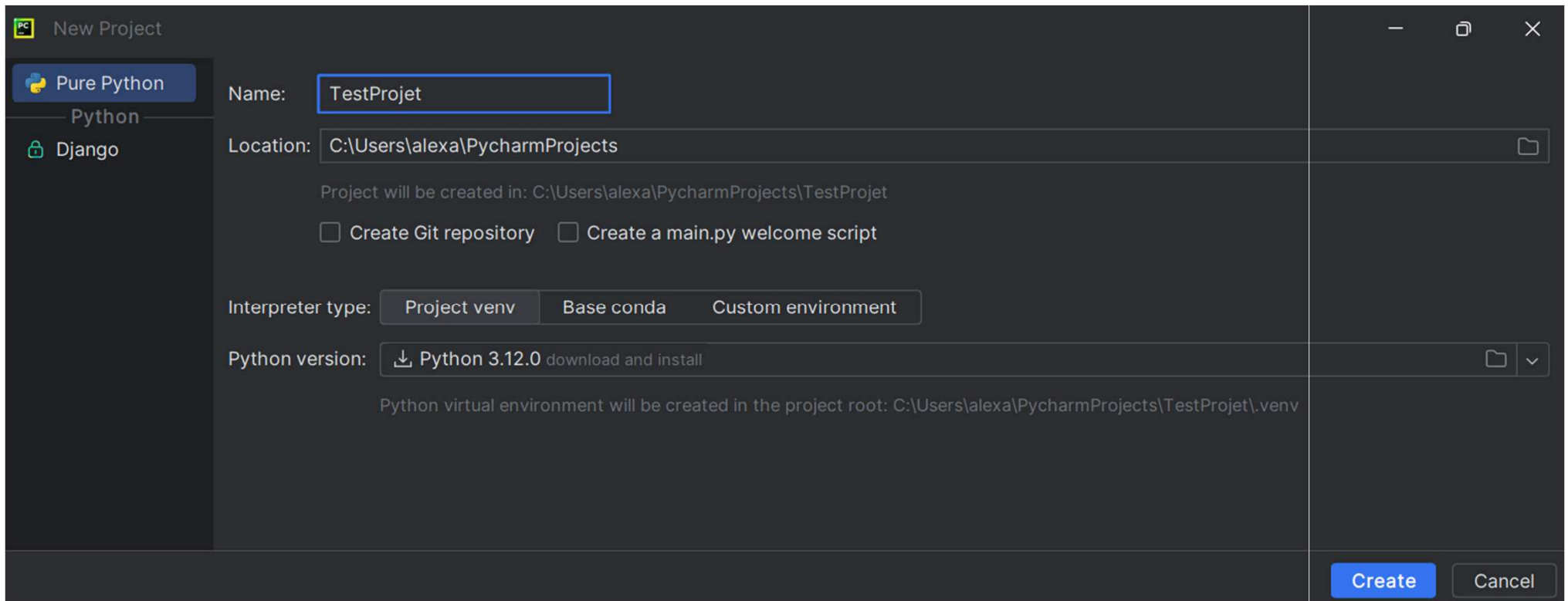


- **Pycharm Community = IDE pour Python**
  - **Objectif : aider à créer ses projets et son code en Python**



# Pycharm Community

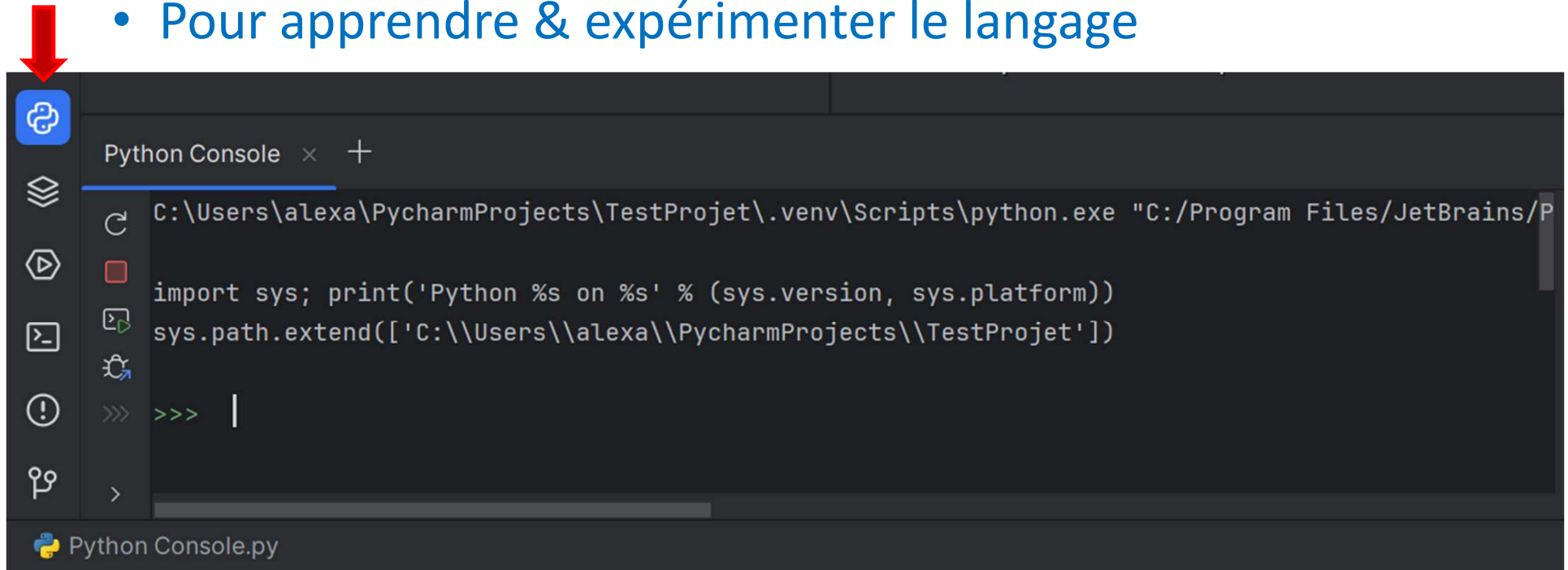
- Tâche N°1 : Création d'un nouveau projet



➔ A vous : ouverture de Pycharm & Création d'un nouveau projet ProjetTest

# Pycharm Community

- Tâche N°2 : Apprendre à utiliser la console
  - On dit aussi que l'on travaille en « mode prompt »
  - Pour apprendre & expérimenter le langage



- A vous : cliquer sur « Console » pour la faire apparaître et lancer une opération mathématique (exemple : 3+5)

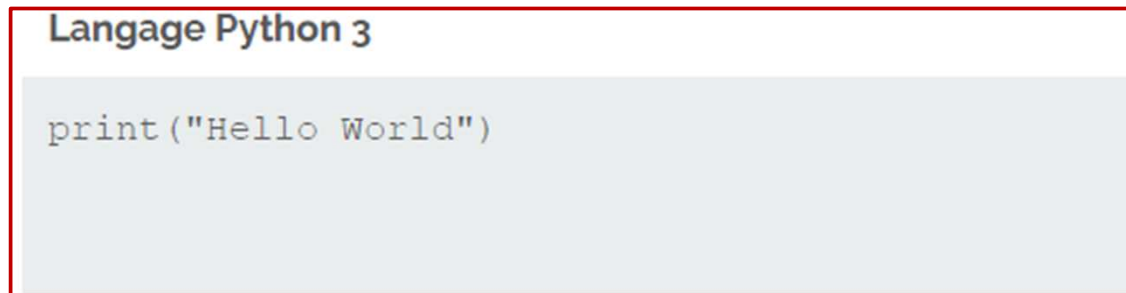
# Pycharm Community

- Tâche N°3 : Création d'un nouveau fichier main.py
  - Les fichiers écrits en langage Python doivent avoir l'**extension** « .py » pour que les logiciels utilisant Python les reconnaissent
  - Ces fichiers sont des fichiers « **texte** » et peuvent très bien être lu avec un éditeur de texte (ou Notepad++ sous Windows)
  - **Script** : Lignes de codes contenues dans un fichier texte

A vous : Création d'un nouveau fichier « main.py » grâce à File > New > Python File

# Exécution d'un premier script

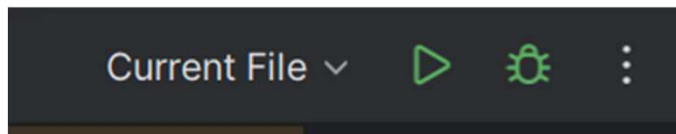
- Tâche 4 : Exécution d'un programme avec PyCharm
  - Création d'un « Hello world » dans votre fichier main.py



```
Langage Python 3

print("Hello World")
```

- Exécution avec PyCharm



- Objectif : lorsque l'on exécute le fichier, les mots « Hello World » s'affichent



# Exécution d'un premier script

- **Tâche 5 : Exécution du Script sans PyCharm Community**
  - On peut très bien se débrouiller sans PyCharm Community
  - **Méthode**
    - Ouvrir votre fichier main.py avec l'éditeur de textes
    - Le modifier : `print('Hello World !!!!!')` et le sauver
    - Ouvrir le terminal (Konsole sous Linux)
    - Lancer l'exécution du script (avec chemin personnel)

```
PS C:\Users\alexa> python C:\Users\alexa\PycharmProjects\TestProjet\.venv\Scripts\main.py
h
```

➔ **PyCharm est un logiciel ergonomique avec autocomplétion mais on peut très bien se débrouiller sans**

# Syntaxe Python

# Syntaxe Python

- **Syntaxe** ⇔ **Ce qui différencie les langages**
  - Permet à **l'interpréteur** de comprendre le langage utilisé
  - **Règle N°1** : Python est sensible à la casse ⇔ Python distingue les minuscules des majuscules
- Remarques : Langages insensibles à la casse
  - Fortan
  - Basic
  - HTML
  - VHDL

➔ devient rare

# Syntaxe Python

## I. Opérations mathématiques

- On peut réaliser des opérations mathématiques en Python en « console » ou dans un fichier.py

- **Opérations principales**

Symbôle	Opération
+	Addition
-	Soustraction
*	Multiplication
/	Division
//	Division entière
%	Reste (Modulo)
**	Exposant

- Utilisation des () également, pour l'ordre des opérations

➔ Illustration en console

# Syntaxe

## I. Opérations mathématiques

- **En console : Que vaut**

1. la division entière de 5 par 2 = ?
2. le reste de la division de 32 par 10 ?
3.  $2^8$  ?
4.  $5.8 + 4$  ?
5.  $18 * 2$
6.  $97 \% 3$

# Syntaxe

## II. Variables & assignations de variables

- **Variable** : entité qui porte un nom unique auquel on associe une valeur et un type
  - **Règle N°2** : Deux variables ne peuvent pas porter le même nom
  - Python est sensible à la casse ⇔ a et A sont deux variables différentes
  - le signe « = » représente une **assignation**
  - Exemple
    - `codePostal = 1000`
    - `codePostal = 1060`
    - `ville = 'Bruxelles'`
    - `Ville = 'Bruxelles'`
- ➔ Illustration en console

# Syntaxe

## II. Variables & assignations de variables

- Mots réservés

False	elif	lambda	None	yield	continue
else	nonLocal	True	exept	is	import
not	and	finally	or	del	while
as	for	pass	assert	with	def
from	raise	break	global	in	try
return	class	if			

- Règles pour choix du nom de variable

1. Le premier caractère doit être une lettre ou un '\_' (underscore)
2. Les caractères '\$' et '?' ne peuvent jamais être utilisés
3. On ne peut pas commencer ou terminer avec une paire de underscore '\_\_'

# Syntaxe

## II. Variables & assignations de variables

- Exercice : Quels sont les noms de variable autorisés en Python dans la liste suivante?

1. var
2. \_\_agce
3. \_lui\$
4. 12345
5. pass
6. Test3

- **Conseil pour le choix des variables**
  - Mots explicites : age, ville
  - Lower Camel Case : ChoixDePlat



# Syntaxe

## III. Types de variables

- **Les plus importants**
  1. nombres entiers (integer)
  2. nombres à virgule flottante (float)
  3. les chaînes de caractères (string)
  4. Les booléens
- **En Python, le type de la variable dépend de la valeur que vous lui assignez**
- **Exemple :**  
`age = 50` → Entier  
`ville = 'Bruxelles'` → String

# Syntaxe

## III. Types de variables

- **En Python, le type est dit « faible »**
  - On peut en cours de programme changer le type de la variable en lui affectant une valeur d'un autre type
  - `age = 50`
  - `age = 'cinquante'`
  - ➔ On a changé le type de la variable de int à string
- **Pourquoi c'est important ?**
  - On ne fait pas la même chose avec un entier, un float ou une chaîne de caractères
- **Utilisation de la fonction `type ( )`**
  - Quand on veut connaître le type d'une variable

# Syntaxe

## IV. Aide au débbug : « Fonction `help ( )` »

- **Usage** : quand on ne sait pas comment utiliser une fonction
  - **Exemple**
    - `help(print)`
    - `help(type)`
- ➔ Apprendre à comprendre l'interpréteur de Python

# Syntaxe

## IV. Aide au débbug : « Commentaires »

- Commentaires : une ligne, une section

### 1. Pour commenter une ligne : #

```
# calcul du delta
```

### 2. Pour commenter tout un paragraphe : 3 x "

```
"""
```

```
Le calcul du delta utilise  
une formule qui est le b au carré  
moins 4 fois le a fois le c
```

```
"""
```

➔ Illustration dans main.py

# Syntaxe

## IV. Aide au débbug : « Commentaires »

- Usages courants des commentaires
  1. Explications / Restructuration en texte
  2. Permet le travail en équipe / Réutilisation de notre code
  3. Débbugger : Pour enlever temporairement une portion de code lorsque l'on débbugge

# Syntaxe

## IV. Aide au débbug : « Fonction print( ) »

- Affichage : Permet d'afficher **une chaîne de caractères**
  - Exemple : `print (Hello)` → Erreur!
  - Exemple : `print ( 'Hello' )` → Hello
- Si l'on veut afficher ' ou " il faut rajouter un backslash \
  - Exemple : `print ( ' \"Hello\" ' )`
- Débbug : pour afficher l'évolution d'une variable : très utile!

```
>>> myString = 'Hello'
>>> print(myString)
Hello
```

- Les 2 syntaxes sont autorisées

```
print("Delta = " , delta)
print("Delta = " + str(delta))
```

# Syntaxe

## V. Concaténation

- La concaténation de deux chaînes de caractères donne une nouvelle chaîne de caractères qui représente le « collage » de la seconde chaîne derrière la première
- Exemple : concaténer “Bonjour” avec “Madame” donne une nouvelle chaîne de caractères “BonjourMadame”
- En Python, pour concaténer des chaînes de caractères, on utilise le « + »

```
>>> myString = 'Hello'  
>>> print (myString + myString)  
HelloHello
```

- Remarque : on ne peut concaténer que des chaînes de caractères

```
>>> "Hello" + 3
```

# Syntaxe

## V. Concaténation

- Il faut d'abord transformer les nombres (float ou int) en chaîne de caractères avec la fonction **str()** avant de pouvoir les concaténer

```
>>> a = 5
>>> myString = 'Hello'
>>> print(myString + str(a))
Hello5
```

- On peut utiliser le « \* » sur des chaînes de caractères

```
>>> print(myString * 4)
HelloHelloHelloHello
```



# Syntaxe

## VI. Fonctions prédéfinies

- Une fonction est une entité
  - qui porte un nom **unique**
  - qui reçoit éventuellement des paramètres, aussi appelés « **arguments** », séparés par des virgules, le tout entre parenthèses
  - qui renvoie éventuellement une valeur
  - qui réalise une opération ou un ensemble d'opérations
- **Exemple : fonction `abs ( )`**
  - accepte un paramètre de type integer ou float et renvoie la valeur absolue du nombre donné en argument

```
>>> abs (-5.2)
```

```
5.2
```

# Les fonctions

## VI. Fonctions prédéfinies

- **Conversion de type**

- **float ( )** : conversion en float → Nombre à virgule
- **int ( )** : conversion en int → Nombre entier
- **str ( )** : conversion en string → Chaîne de caractères

```
>>> b = float(3)
>>> b
3.0
```

# Syntaxe

## VI. Fonctions prédéfinies

- Exemples

Fonction	Résultat
<code>abs(x)</code>	Retourne la valeur absolue de $x$
<code>pow(x, y, z)</code>	Retourne $(x \text{ exposant } y) \% z$
<code>pow(x, y)</code>	Retourne $x \text{ exposant } y$

```
>>> a = abs(-5)
>>> a
5
>>> b = pow(2, 5)
>>> b
32
>>> b = pow(2, 5, 3)
>>> b
2
```

# Syntaxe

## VI. Fonctions prédéfinies

- **Fonction `input ( )`**

```
nom = input("Tapez votre nom")
```

```
print('Bonjour ' + nom)
```

➔ Très utile pour créer un dialogue avec l'utilisateur

# Syntaxe

## VII. Variables booléennes

- Une variable booléenne, ou « booléen » peut avoir deux états possibles : **True** et **False**

Symbôle	Opération
==	Egal à
!=	Différent de
<	Plus petit que
>	Plus grand que
<=	Plus petit ou égal à
>=	Plus grand ou égal à

- Remarque : **a = b** différent de **a == b**

# Syntaxe

## VIII.Conditions if/else

```
if (variable == 10):  
    print ( ``Vrai`` )  
else :  
    print ( ``Faux`` )
```

- **Particularité de Python**
  - L'indentation qui consiste à décaler des portions de code (en utilisant la tabulation) permet de délimiter les blocs en « dépendance » avec d'autres blocs

# Exercices

# Exercices

- **Exercice**
  - Réaliser un programme qui calcule et affiche, à partir de 3 variables (a, b, et c) le delta d'une équation du second degré :  $\text{delta} = b^2 - 4.a.c$ , sauvegarder ce fichier sous le nom calculDelta.py et exécutez le programme

```
a = 1
b = 1
c = 1
delta = (b*b) - (4*a*c)
print(delta)
```



# Exercices

- **Exercice**
  - Réaliser un programme qui, à partir de l'heure de la journée, stockée dans 3 variables heure, minute et seconde, calcule et affiche le nombre total de secondes qui se sont écoulées depuis le début de la journée

# Exercices

- **Exercice**
  - Réaliser un programme qui, à partir d'un nombre quelconque de secondes, stockée dans la variable seconde, transforme ce nombre en son équivalent heure:minute:seconde et l'affiche.
  - Par exemple 3740 secondes correspondent à 1:2:20 (1 heure, 2 minutes et 20 secondes)

# Exercices

- **Exercice**

- Réalisez un logiciel qui calcule vos intérêts bancaires.
- Votre programme demande à l'utilisateur le montant, le taux d'intérêt annuel et le nombre d'années.
- Votre code calcule ensuite les intérêts et affiche le résultat sous forme d'un tableau qui montre l'évolution de votre capital au cours des années.
- Exemple : Montant de départ 100, Intérêt 3%, Durée 10 ans

Départ	100
1	103
2	106,09
3	109,2727
4	112,550881
5	115,92740743
6	119,40522965
7	122,98738654
8	126,67700814
9	130,47731838
10	134,39163793

# Exercices

- **Exercice**
  - Créer un nouveau projet : **ProjetJour1**
  - Créer un nouveau fichier Python : **main.py**
  - Créer un script qui
    - demande notre prénom et notre date de naissance (jour, mois et année) dans l'interface (fonction input)
    - calcule notre âge
    - Calcule le nombre de jours restants avant notre anniversaire
    - affiche (fonction print() ) un texte avec notre prénom, notre âge et le nombre de jours restants avant notre anniversaire

# Vocabulaire

- Code ou Script
- Console ou Prompt
- Librairies
- Sensible à la casse
- Assignation
- Coder ou Implémenter

# Fonctions vues

- `print()`
- `help()`
- `float()`
- `int()`
- `str()`
- `abs()`
- `type()`
- `pow(a, b)`
- `pow(a, b, c)`
- `input()`