

Chapitre 1 : PREMIERS PAS EN PYTHON

1- Python et son origine

- Langage de programmation généraliste
- Développé en 1988 par **Guido Von Rossum** à Amsterdam
- Grand fan de la série télé britannique « **Monthy Python's Flying circus** », d'où le nom du langage
- Très utilisé autant dans l'industrie que dans le milieu scolaire
- Syntaxe facile à apprendre
- Libre de droits (« open source »), portable (multi plateformes) et gratuit



Vocabulaire à connaître

Langage interprété: le script ou code source (texte du programme) est retranscrit en langage machine

Langage compilé: le script est traduit une fois pour toutes par un programme annexe,

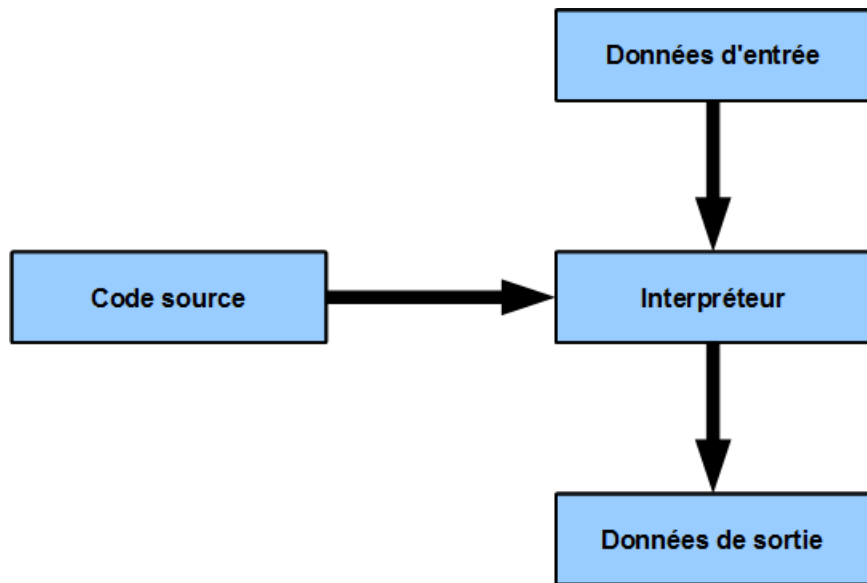
Quelques différences :

langage interprété : modification du programme accessible à tous, nécessité de l'interpréteur sur l'ordinateur pour exécuter le programme source,

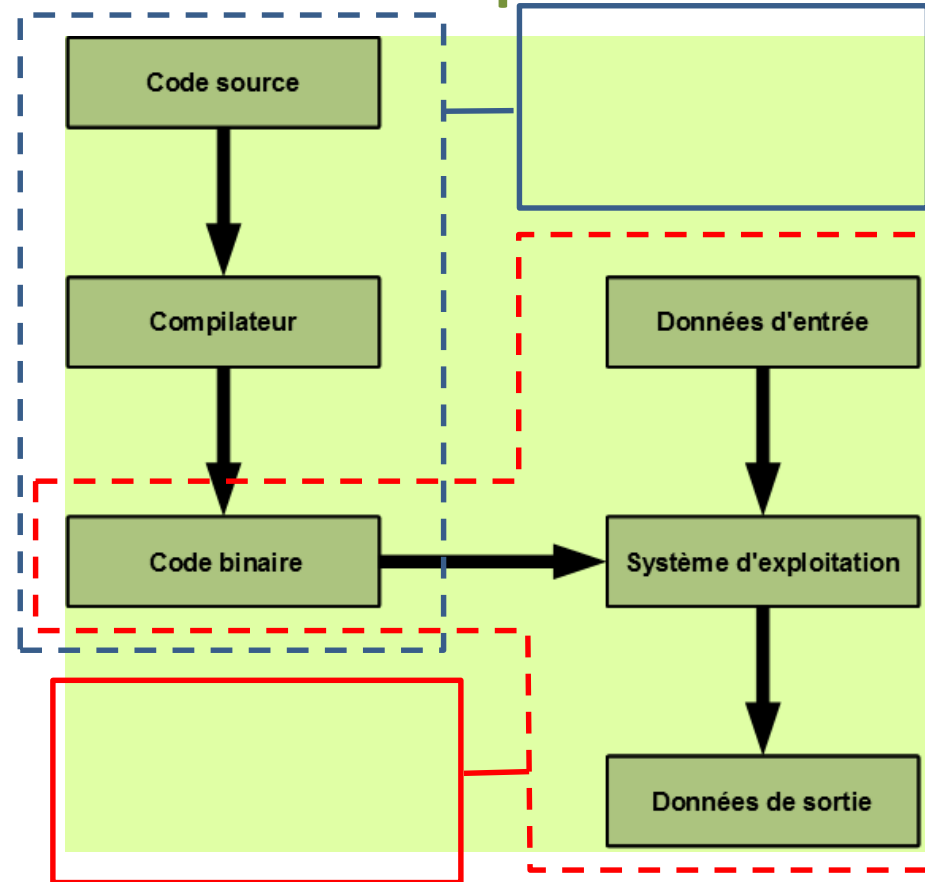
langage compilé : programme plus rapide à l'exécution, sécurité du code source.



Interprété



Compilé



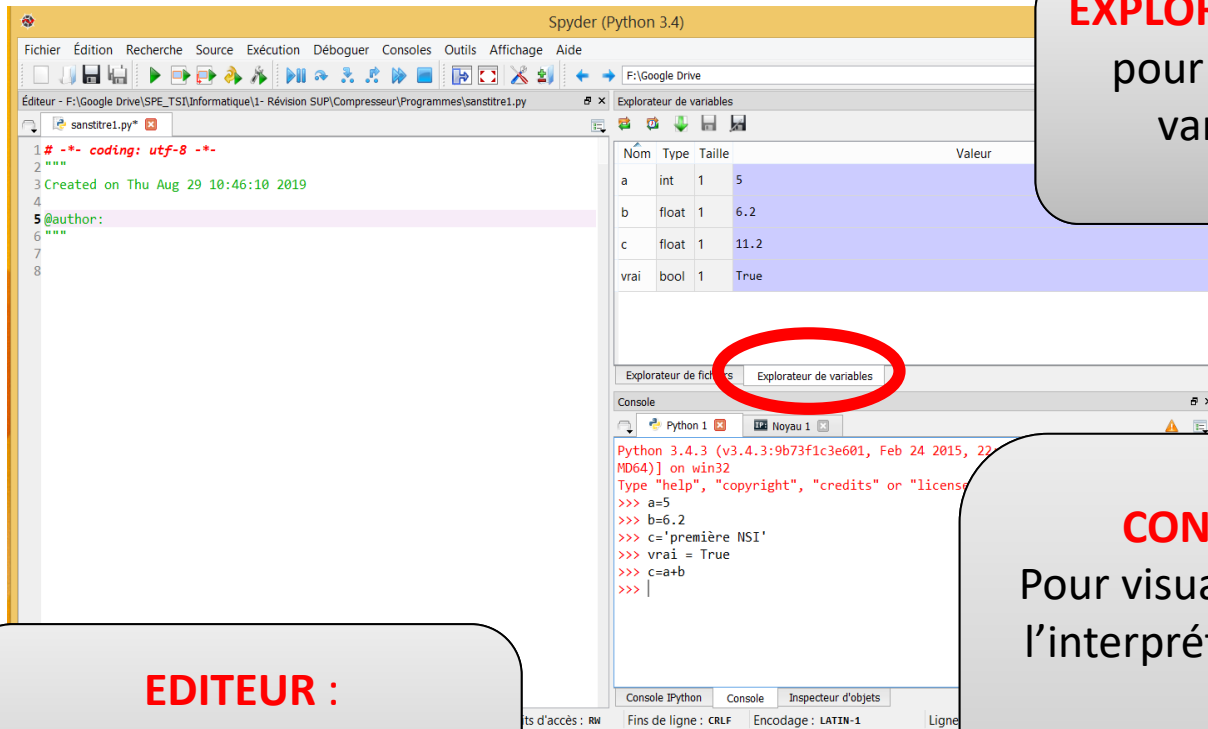
Dans un langage interprété, le même code source pourra fonctionner directement sur tout ordinateur, quel que soit son système d'exploitation : il est

Avec un langage compilé, il faudra tout recompiler à chaque changement de système d'exploitation (windows, macOS, android, linux ...) avant de pouvoir utiliser le programme.

2- Environnement de travail

⇒ **IDLE** - Integrated **D**evelopment and **L**earning **E**nvironment

Fenêtres de notre IDLE : SPYDER



EXPLORATEUR DE VARIABLES :

pour suivre les valeurs des variables en phase de « débogage »

EDITEUR :

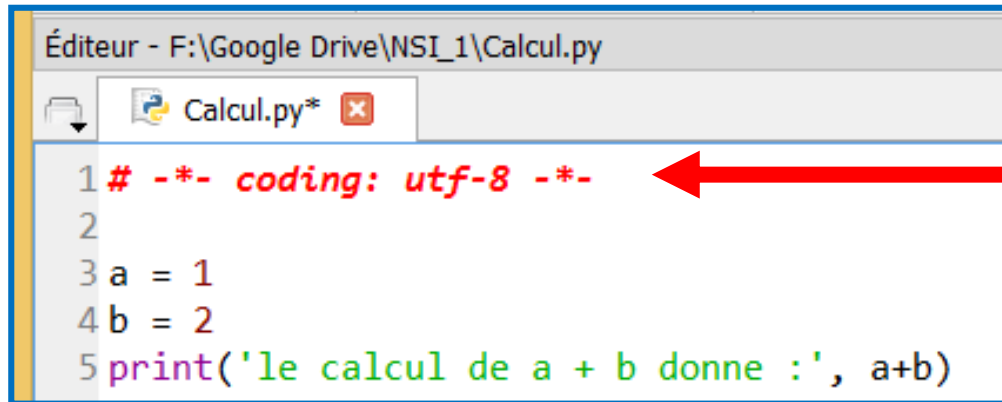
pour écrire des programmes et les enregistrer

CONSOLE ou SHELL :

Pour visualiser et interagir avec l'interprétation du programme de l'éditeur.

Pour saisir et tester des commandes simples et visualiser les résultats.

Saisie du programme dans l'éditeur



```
1 # -*- coding: utf-8 -*-
2
3 a = 1
4 b = 2
5 print('le calcul de a + b donne :', a+b)
```

elle permet à l'interpréteur de décoder le texte du fichier de votre code python.

Remarques :

- les numéros de ligne 1,2...5
- aucun espace ne doit être placé

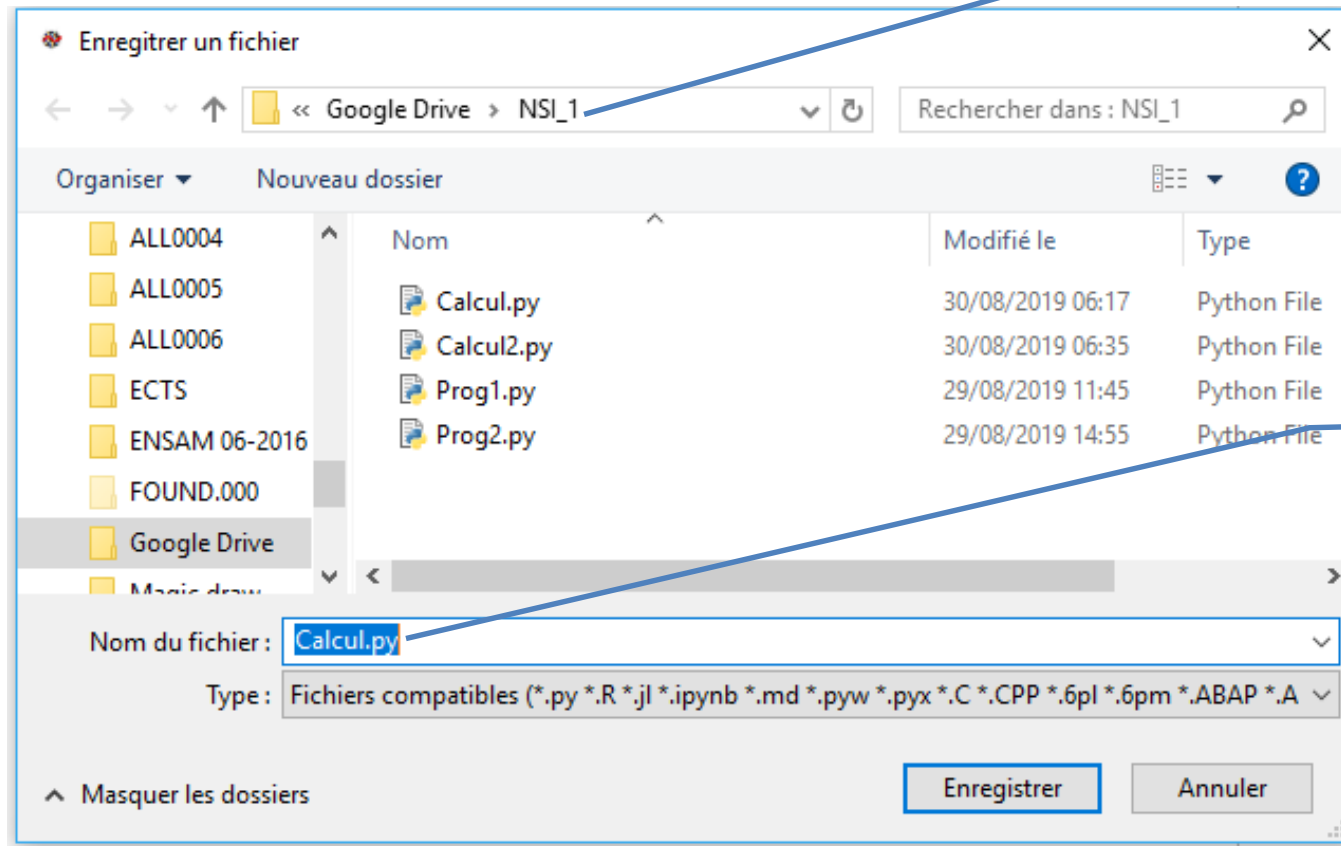
print(attributs) est une commande de python. Elle permet de visualiser, dans la console d'exécution,

Dans le programme ci-dessus:

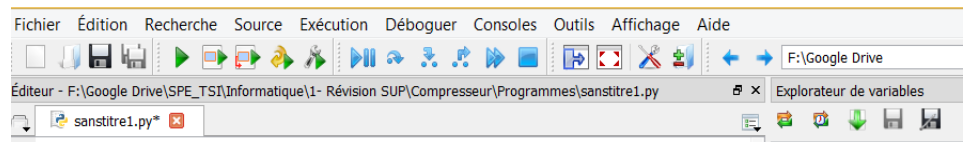
- La phrase **'le calcul de a + b donne :'** est le 1^{er} attribut. C'est une
- L'expression **a+b** constitue le 2^{ème} attribut. C'est une

Les différents attributs de la commande **print()** doivent être

Sauvegarde du programme : on l'enregistre dans un fichier avec le nom « Calcul.py » par exemple: ⇨

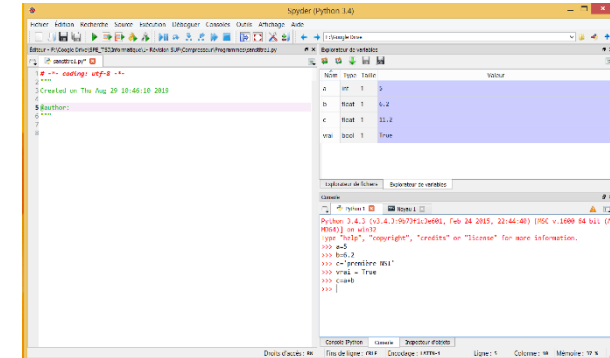
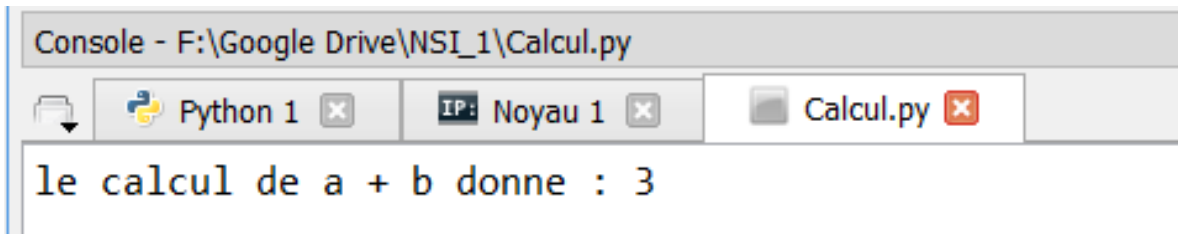


Exécution du programme

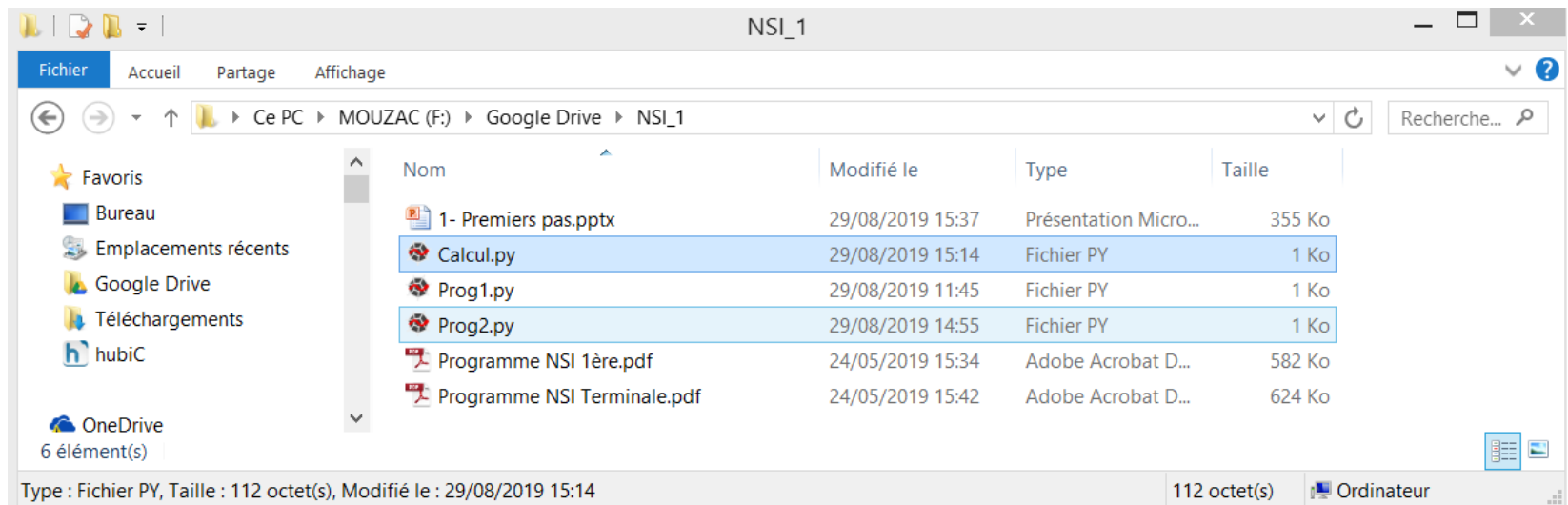


ou la touche **F5** pour exécuter tout le fichier,

⇒ Le résultat de l'exécution apparaît dans la



Vérification de la présence du fichier dans l'explorateur de fichiers



3- Notions de base

Utilisation des commentaires

Tout ce qui suit un **#** sur **une** ligne n'est pas exécuté. L'interpréteur Python le considère comme

Toutes les lignes encadrées par les triple guillemets, simple ou double,

```
# symbole dièse : une ligne en commentaire

""" guillemets double
des lignes de commentaires
des lignes de commentaires
"""

''' guillemets simple
des lignes de commentaires
des lignes de commentaires
'''
```

Déclaration de variables et affichage:

Code python et commentaires

```
a=1 # La variable a reçoit la valeur 1 : a <-- 1
b=2 # La variable b reçoit la valeur 2 : b <-- 2

print('le calcul de a+b donne :',a+b)
```

Résultat : Affichage dans la console pendant l'interprétation

```
In [5]: runfile('D:/TAFF_2022_2023/COURS/Spé NSI 1/Chapitre 1 Premiers pas/Cours/Chp_1.py',
wdir='D:/TAFF_2022_2023/COURS/Spé NSI 1/Chapitre 1 Premiers pas/Cours')
le calcul de a+b donne : 3
```

Vocabulaire à connaître

On parle de **déclaration de variable** lorsque celle-ci

Affectations simples, multiples et permutations de variables

Exemple de script

Résultats dans la console

```
Calcul3.py* [X]  
1 a = 1      # simple affectation  
2 b = 1      # simple affectation  
3 print('1er calcul de a-b donne :', a-b)
```



```
5 a,b = 3,4   # double affectation  
6 print('2ème calcul de a-b donne :', a-b)  
7
```



```
8 a,b = b,a    # permutation de a et b  
9 print('3ème calcul de a-b donne :', a-b)
```



Premiers calculs

Opérations arithmétiques :

+ **addition,**

- **soustraction,**

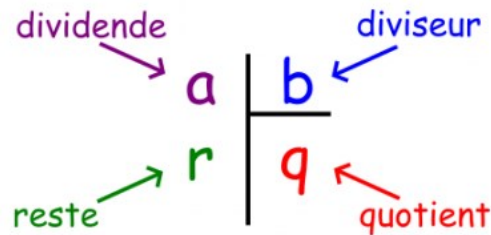
* **multiplication,**

/ **division,**

// **quotient d'une division euclidienne,**

% **reste d'une division euclidienne,**

La division Euclidienne est parfois aussi appelée



Remarque : Les parenthèses s'emploient comme sur une calculatrice.

**** élévation à la puissance :**

Le calcul d'une puissance est

4- Compléments sur les variables en python

```
a=1 # La variable a reçoit la valeur 1 : a <-- 1  
b=2 # La variable b reçoit la valeur 2 : b <-- 2  
  
print('le calcul de a+b donne :',a+b)
```

Vocabulaire à connaître

Affectation de variable: attribution d'une donnée

La donnée est stockée **dans la mémoire vive de l'ordinateur** lors de l'affectation.

Dans l'exemple ci-dessus, **on a affecté la valeur entière 1 à la variable a.**

Attention: le nom de la variable doit **impérativement** être placé

En effet, en informatique, $a = 1$ est

Remarque pour la suite du document :

```
>>> print(a)
```

← Ligne de code du programme

```
10
```

← Résultat de l'affichage dans la console de la ligne précédente

Types de variable:

```
>>> type(a)
<class 'int'>
```

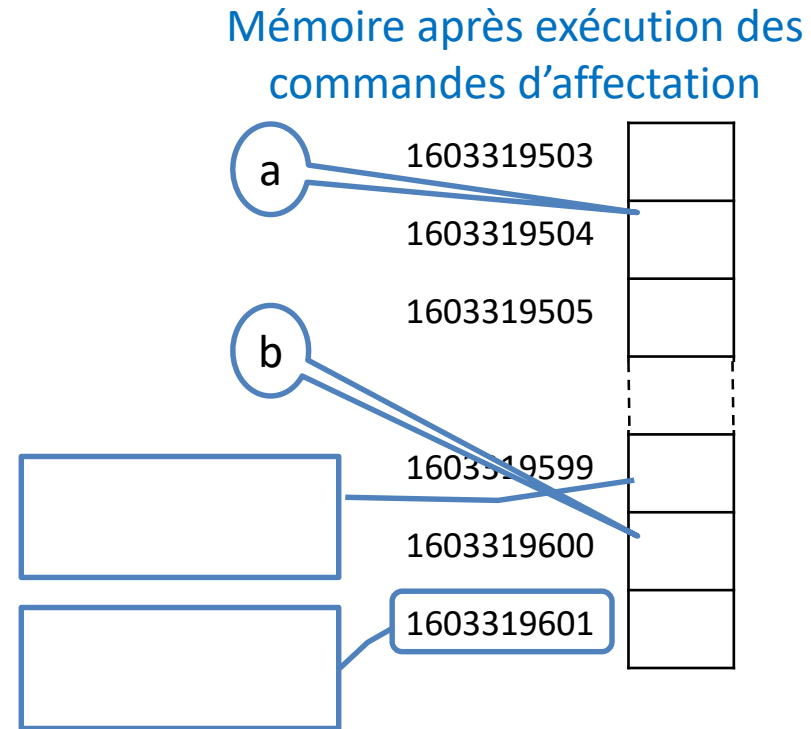
← Entier (**integer**)

```
>>> type(b)
<class 'int'>
```

```
>>> type(f)
<class 'float'>
```

← Réel (nombre à virgule flottante : **floating point**)

Emplacement mémoire:

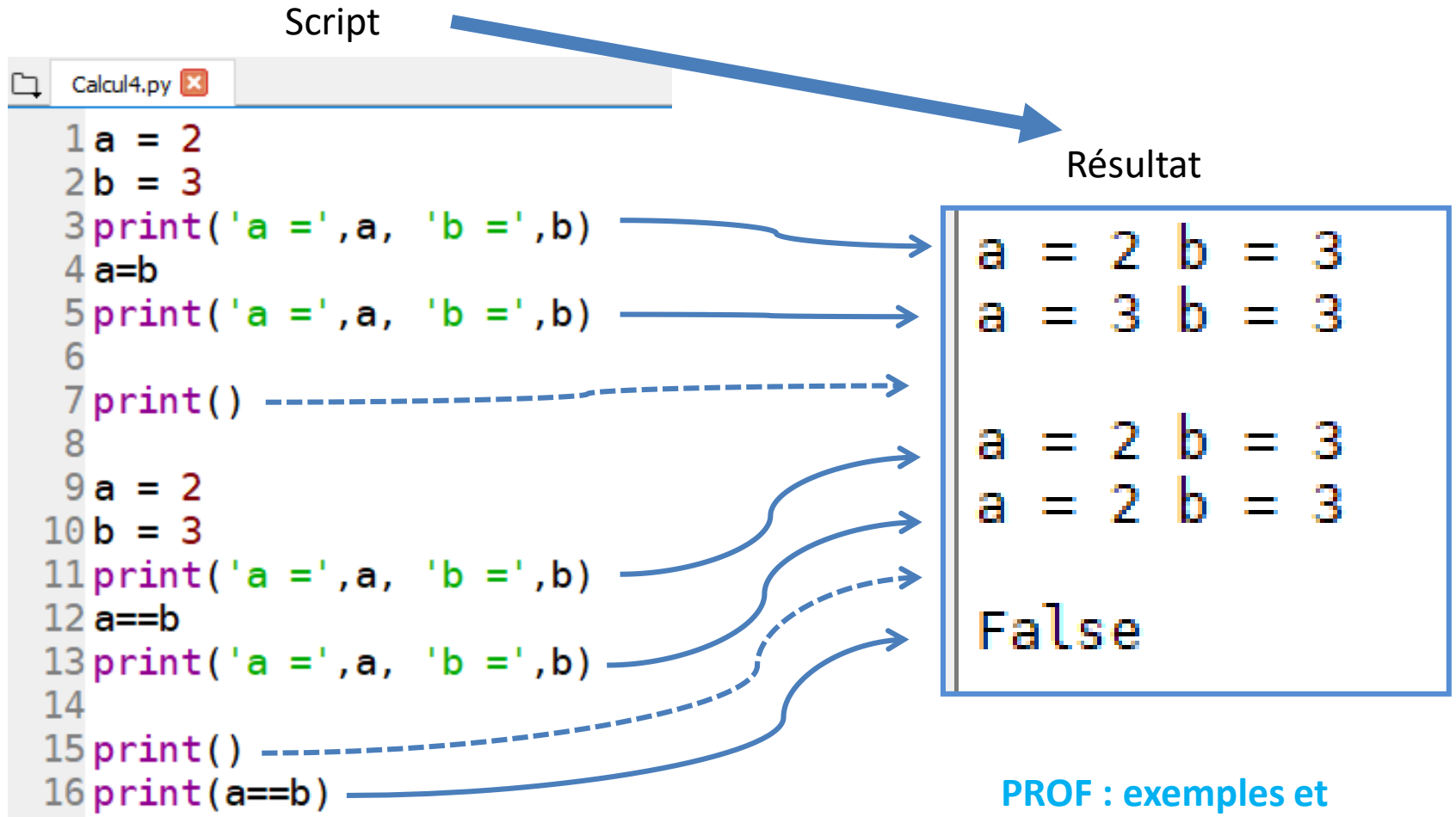


↓

```
>>> id(a) # id(x) retourne l'emplacement mémoire de x
1603319504
>>> id(b)
1603319600
>>> id(z)
NameError: name 'z' is not defined
>>> Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
```

Attention à ne pas confondre l'affectation `=` avec le test d'égalité `==`

Exemple:



PROF : exemples et
compléments N° 7

Ecriture scientifique

Exemple d'affectation d'un nombre réel en notation scientifique dans la console:



```
>>> k=1.5e-2
>>> print(k)
0.015
>>>
>>> j=1.5*10**-2
>>> print(j)
0.015
```

Rq.: la virgule se note

Modification du contenu d'une variable

Pour multiplier par 2 le contenu de la variable a:

```
>>> a=5
>>> a=2*a # a est multiplié par 2
>>> print(a)
10
```

ou

```
>>> a=4
>>> a*=2 # a est multiplié par 2
>>> print(a)
8
```

Pour ajouter 2 au contenu de la variable a:

```
>>> print(a)
10
>>> a=a+2 # on ajoute 2 à a
>>> print(a)
12
```

ou

```
>>> print(a)
8
>>> a+=2 # on ajoute 2 à a
>>> print(a)
10
```

Idem pour les soustractions et divisions.

6- Affectation externe et transtypage

Lors de l'exécution d'un programme, il est possible de demander à l'utilisateur de saisir une donnée au clavier.

Exemple A:

```
>>> nom = input('Nom de ton lycée ? ')
Nom de ton lycée ? Cabanis
>>> print (nom)
Cabanis
>>> type(nom)
<class 'str'>
```

Affichage écran du programme

Entrée clavier de l'utilisateur

☛ La fonction **input()** renvoie

Explorateur de variables:

Explorateur de variables				
Nom	Type	Taille	Valeur	
nom	str	1	Cabanis	

Exemple B:

```
>>> age = input('Quel est on âge ? ')
Quel est on âge ? 18
>>> type(age)
<class 'str'>
>>> age = int(age)
>>> type(age)
<class 'int'>
```

Entrée clavier de l'utilisateur

Transtypage : la variable

Affichage écran du programme

Autre possibilité:

```
>>> age = int(input('Quel est on âge ? '))
Quel est on âge ? 18
>>> type(age)
<class 'int'>
```

La chaine de caractères saisie au clavier est

Exemple C:

```
>>> taille = input('Quelle est ta taille en m ? ')
Quelle est ta taille en m ? 1.75
>>> type(taille)
<class 'str'>
>>> taille=float(taille)
>>> type(taille)
<class 'float'>
```

Transtypage : la variable

Autre possibilité:

```
>>> taille = float(input('Quelle est ta taille en m ? '))
Quelle est ta taille en m ? 1.75
>>> type(taille)
<class 'float'>
```

La chaine de caractères saisie au clavier est

Bonne pratique en informatique: commenter les programmes

Exercice 1

1- Ecrire un programme commenté qui:

- déclare et affecte la variable **nb1** avec la valeur 125
- déclare et affecte la variable **nb2** avec la valeur 1,25
- déclare et affecte la variable **nb3** avec la valeur 12
- calcule la somme **nb1+nb2** et place le résultat dans la variable **nb4**
- calcule le quotient de la division euclidienne de **nb1 par nb3** et place le résultat dans la variable **nb5**
- calcule le reste de la division euclidienne de **nb1 par nb3** et place le résultat dans la variable **nb6**
- affiche sur une seule ligne le nom et la valeur des variables **nb1, nb2 et nb3**
- affiche la valeur de **nb4**, avec un message explicatif
(exemple d'affichage: *la somme 125 + 1,25 vaut 126,25*)
- affiche le résultat de **nb1^{nb3}** , avec un message explicatif
- affiche sur 2 lignes séparées les valeurs de **nb5 et nb6**, avec un message explicatif
- affiche sur une seule ligne les types de **nb4 et nb5**, avec un message explicatif
- affiche sur une seule ligne les adresses mémoire de **nb5 et nb6**, avec un message explicatif

2- Sauvegarder le programme avec le nom **Ex3.py**, puis tester son fonctionnement.

Exercice 2

L'accélération de la pesanteur **g** dépend de l'altitude **h** à laquelle on se situe selon la relation suivante:

$$\mathbf{g} = \frac{G * m_{Terre}}{(R_{Terre} + h)^2}$$

Avec : **R_{Terre} = 6371 km**, **G = 6,67x10⁻¹¹ N.m².kg⁻²**, **m_{Terre} = 6,0x10²⁴kg**

G est la constante universelle de gravitation

- 1- Ecrire **dans l'éditeur** un script python qui permet d'évaluer la constante **g** à une altitude de **1m** et d'afficher sa valeur de manière conviviale (c'est-à-dire avec un peu de texte, puis le résultat suivi de son unité SI).
- 2- Commenter, sauvegarder avec le nom **Ex4.py** et valider le fonctionnement.

5- Autre type de base: les booléens

Un booléen est une variable qui ne peut prendre que 2 états distincts:

- **True (vrai)** que l'on associe
- **False (faux)** que l'on associe

Intérêt: **tester si une expression logique**

Exemple dans
la console:

```
>>> a,b,c = 4,2,1
>>> a<b
False
>>> b>c
True
```

```
>>> d = (a<b) # parenthèses non obligatoires
>>> print(d)
False
>>> type(d)
<class 'bool'>
```

⇒ Les booléens sont

Instructions de test arithmétique et logique en python:

⇒ **Exercice 5** : affecter les valeurs 4, 2, 1 aux variables a, b, c en faisant **une affectation multiple**, puis écrire le programme pour compléter les tableaux suivants :

Instruction	Signification	Résultat
a == b	a est égal à b ?	
a > b	a est strictement supérieur à b ?	
a < b	a est strictement inférieur à b ?	
a >= b	a est supérieur ou égal à b ?	
a <= b	a est inférieur ou égal à b ?	
a != b	a est différent de b ?	

Instruction	Signification	Résultat
$a < b$ and $b > c$	a est inférieur à b ET b est supérieur à c ?	
$a < b$ or $b > c$	a est inférieur à b OU b est supérieur à c ?	
not(True)	Le contraire de True ?	
not(False)	Le contraire de False ?	

Vocabulaire à connaître

Les mots **and**, **or**, **xor** et **not** sont appelés

XOR provient de

Exercice 5 (suite)

2- Ecrire et tester un programme pour compléter les tables suivantes:

Table de l'expression
booléenne a and b

a	b	a and b
0	0	
0	1	
1	0	
1	1	

Table de l'expression
booléenne a or b

a	b	a or b
0	0	
0	1	
1	0	
1	1	

Table de l'expression
booléenne a xor b

a	b	a xor b
0	0	
0	1	
1	0	
1	1	

3- Ecrire un programme qui utilise les opérateurs **and**, **or** et **xor** et qui affiche les tables ci-dessus. Rq.: la chaîne '\t' produit une tabulation sur la ligne en cours.

Exercice 5 (suite)

4- Avec la console de spyder, rechercher les résultats des expressions suivantes:

Expression	Résultat
1 and 0 or 0	
1 or 0 and 0	

5- Justifier le fait que les résultats sont différents.

Exercice 6 : Modifier le script de l'exercice 4 pour que l'utilisateur puisse saisir l'altitude **h** à laquelle il désire obtenir la valeur de l'accélération **g** de la pesanteur.

Exercice 7 : Ecrire un programme qui, à partir de la affiche la valeur d'un angle en radian alors qu'elle est saisie par l'utilisateur en degré.

Exercice 8 : Ecrire un programme qui demande un nombre à l'utilisateur, puis élève ce nombre au cube et enfin affiche le résultat dans une phrase (Par exemple, si l'utilisateur saisit le nombre 4, le programme devra afficher la phrase suivante: « Le cube de 4 vaut 64 »).

Exercice 9 : Ecrire un programme qui demande à un élève ses 4 dernières notes obtenues en NSI et qui affiche la moyenne correspondante.

Exercice 10 : Ecrire un script qui permet à l'utilisateur de saisir le montant de ses économies actuelles. Sachant qu'il dépense chaque semaine 10 % de ce qu'il lui reste, compléter le script de telle sorte qu'il affiche le montant des économies dans 5 semaines.