

## I) Introduction :

### Schématiquement :

Un algorithme est une succession finie d'instructions à effectuer pour aboutir à un certain résultat.

Un algorithme peut être rédigé dans une langue donnée ( le Français pour nous ), décrit à l'aide d'un schéma, transcrit dans un langage informatique, construit physiquement ( machine mécanique )....

Exemple :

Entrer un réel

Multiplier ce réel par 2

Ajouter 5

Elever à la puissance

Afficher le résultat

L'ordre dans lequel apparaissent les instructions est important.

Ainsi dans l'exemple ci-dessus, on peut réécrire l'algorithme en utilisant un schéma

Entrer  $X \rightarrow 2X \rightarrow 2X + 5 \rightarrow (2X + 5)^2 \rightarrow$  Afficher  $(2X + 5)^2$

Avec les Texas Instrument on peut le réécrire

Disp " Entrer X"	La calculatrice affiche la phrase « Entrer X »
Input X	La calculatrice stocke la valeur saisie par l'utilisateur dans la variable X
2*X → A	La calculatrice stocke la valeur la quantité 2X dans la variable A
A+5 → B	La calculatrice stocke la valeur la quantité A+5 dans la variable B ( donc la valeur 2X + 5 dans B )
B <sup>2</sup> → C	La calculatrice stocke la valeur la quantité B <sup>2</sup> dans la variable C ( donc la valeur (2X + 5) <sup>2</sup> dans C )
Disp " La réponse est "	La calculatrice affiche la phrase « La réponse est »
Disp C	La calculatrice affiche la valeur stockée dans C ( c'est-à-dire (2X + 5) <sup>2</sup> )

Cela donne dans la console Texas :

```
PROGRAM:EXEMPLE1
:Disp "ENTRER X"
:Input X
:2*X→A
:A+5→B
:B²→C
:Disp "LA REPONSE EST"
:Disp C
:■
```

On le teste avec  $X = 2$

```
PrgmEXEMPLE1
ENTRER X
?2
LA REPONSE EST
81
.....Fait.
■
```

On va utiliser le langage Python pour programmer.

C'est un langage qui permet de faire du calcul non formel : Il ne faut pas s'attendre à ce qu'un programme « simple » sous Python vous sorte une formule.

Si besoin on peut le télécharger gratuitement à l'adresse <https://www.python.org>

Choisir la version la plus récente.

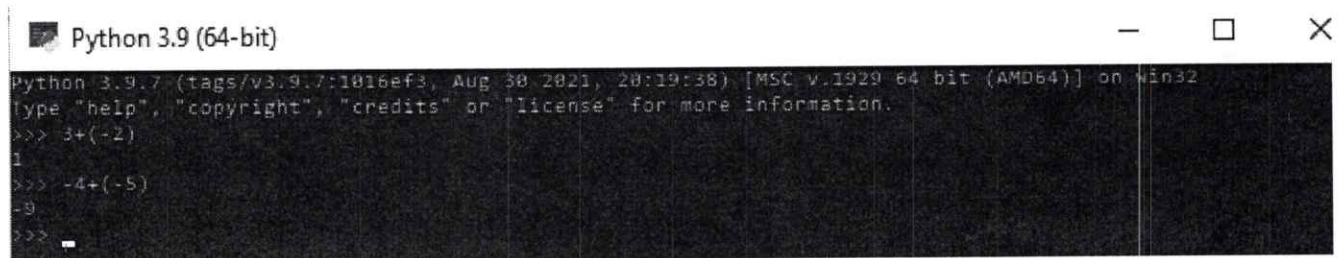
## II) Présentation de Python :

On peut utiliser Python de 2 façons :

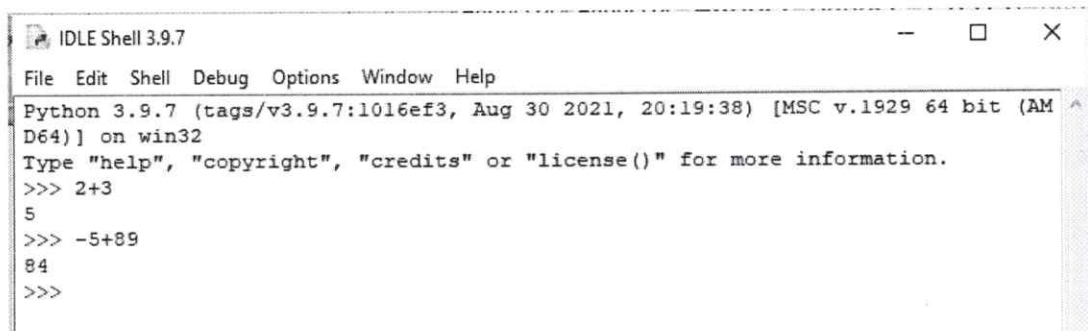
- En mode console. On tape directement les instructions et on les exécute. On travaille avec ce que l'on appelle **Python Shell**. On tape les instructions ligne par ligne ( on voit un symbole `>>>` ). On doit appuyer sur « entrée » à la fin de chaque ligne.

Ce mode n'est pas très pratique si on a besoin d'effectuer plusieurs calculs utilisant le même procédé. Il faudrait réécrire à nouveau chaque ligne, ce qui peut finir par être très long.

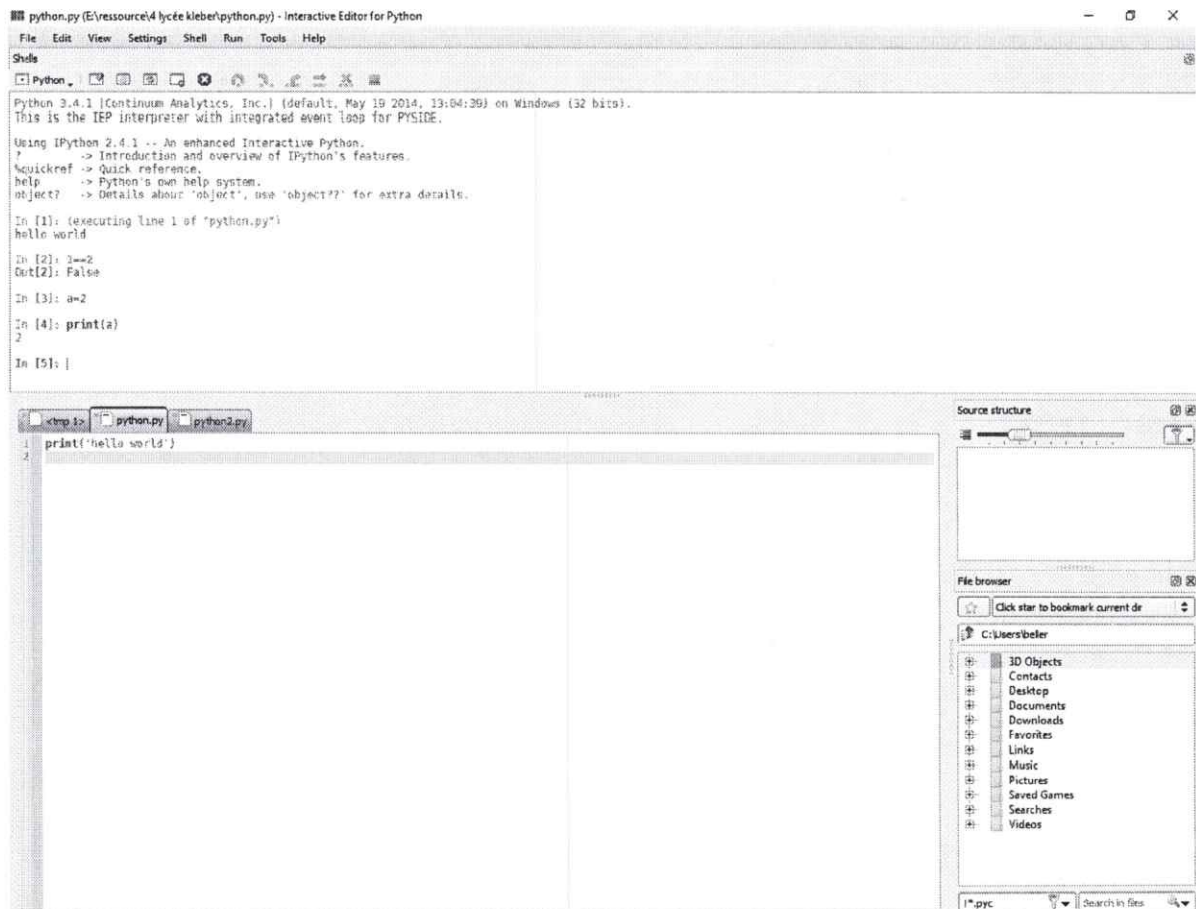
Par exemple : Vous avez ci-dessous 3 présentations possibles de l'affichage Python.



```
Python 3.9 (64-bit)
Python 3.9.7 (tags/v3.9.7:1016ef3, Aug 30 2021, 20:19:38) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 3+(-2)
1
>>> -4+(-5)
-9
>>>
```

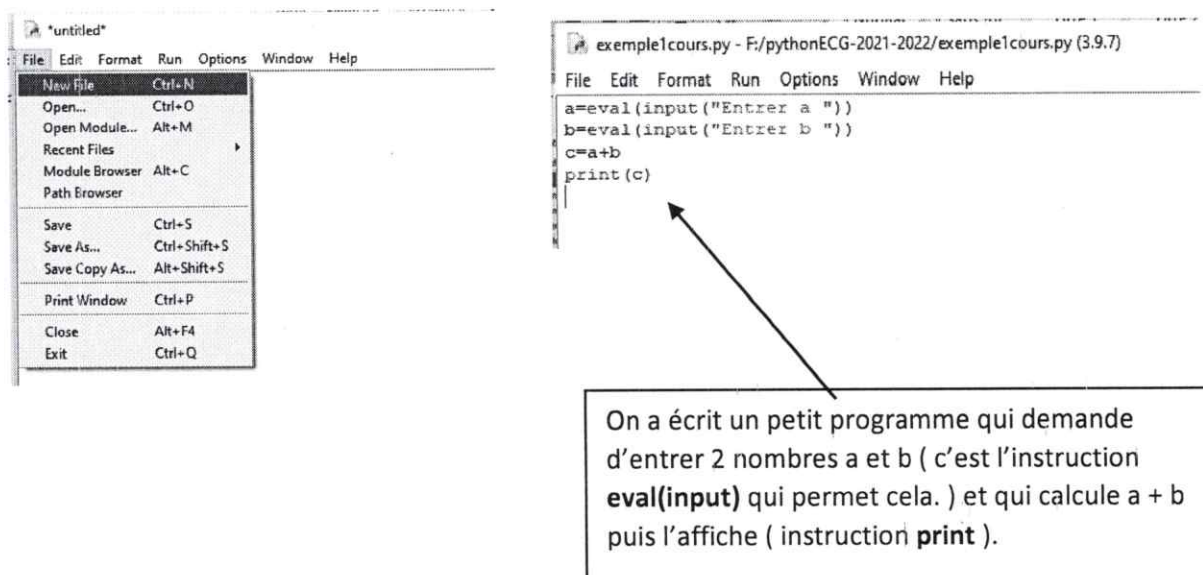


```
IDLE Shell 3.9.7
File Edit Shell Debug Options Window Help
Python 3.9.7 (tags/v3.9.7:1016ef3, Aug 30 2021, 20:19:38) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 2+3
5
>>> -5+89
84
>>>
```



- En mode programmation. On écrit une suite d'instructions qui seront enregistrées. Il suffira alors de lancer le programme ( on dit aussi script ) avec les valeurs souhaitées.  
Pour utiliser ce mode il faut ouvrir la fenêtre Python Shell puis ouvrir l'éditeur de textes « New file ».  
Pour lancer le programme on va sur « run », le programme s'exécute alors dans Python Shell.

Par exemple :



Ecrire et exécuter plusieurs fois ce petit script...

### III) Premiers exemples :

Pour le moment, on travaille dans Python Shell.

Tapez les instructions suivantes et regarder ce qui se passe.

>>> 8.12 *8.12*

>>> 8,12 *(8,12)*

>>> 1.5e3 *1500.0*

>>> 3.45+8 *11.45*

>>> 9.54-4.68 *4.85999 (erreur de base 2)*

>>> 2\*3 *6*

>>> 2\*\*3 *8*

>>> 48/7 *6.85714...*

>>> 48//7 *6*

>>> 3\*\*4 - 6\*7/11 *77.1818*

Tableau à retenir :

Symboles	opérations
+	
-	
*	
**	
/	
//	

#### Exercices 1 :

1) Exécutez les calculs suivants (à l'aide de Python) :

$$\begin{array}{llll}
 125 \times 347 & ; & 4 \times 3,52 + (2,5 - 7,5)3 & ; & \frac{3,25+7,8}{1,26-0,57} & ; & 6 \times \frac{(10,68-2,786)^4}{5^2} \\
 = 43375 & & = -0,91999 & & = 10,01449... & & = 931,965...
 \end{array}$$

2) Calculez le quotient et le reste de la division euclidienne de  $a$  par  $b$  dans les cas suivants :

$$\begin{array}{llll}
 a = 125 & b = 11 & a = 689 & b = 17 & a = -145 & b = 21 & a = -368 & b = -6 \\
 q = 11 & r = 4 & q = 40 & r = 9 & q = -7 & r = 23 & q = 61 & r = -2
 \end{array}$$

En programmation on utilise **des variables**.

Les variables sont de plusieurs types. Pour faire simple on a :



- Les variables numériques (celles sur lesquelles on peut effectuer des calculs usuels comme l'addition, la multiplication, appliquer la fonction exponentielle, appliquer la fonction cosinus...).
- Les chaînes de caractères ce qui correspond à un mot.
- Les variables booléennes. Il s'agit de variables qui prennent la valeur vraie ou faus. Une variable booléenne est généralement le résultat d'un test logique.

On travaillera surtout les 2 premiers types de variables. Il faut voir une variable comme une boîte. Quand on crée une variable on crée une boîte (éventuellement vide). Lorsque vous mettez une nouvelle valeur dans la boîte, la précédente est automatiquement retirée et disparaît.

### Exemple :

On écrit le programme ou algorithme ou script suivant :

```
Python 3.9 (64-bit)
Python 3.9.7 (tags/v3.9.7:1016ef3, Aug 30 2021, 20:19:38) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> A=3
>>> B=5
>>> A=B
>>> print(A)
5
>>> print(B)
5
>>>
```

Ici, on a défini 2 variables que l'on a appelées A et B.

Au début du programme A vaut 3. L'affectation de la valeur 3 à la variable A se fait à l'aide du **signe =**.

De la même façon, B vaut 5.

A la troisième ligne, la variable A prend ce que contient la variable B. A ce stade, A vaut donc 5.

On le vérifie en affichant A et B grâce à l'instruction `print()`.

Lorsque l'on écrit une suite d'instructions, Python les lit toutes du haut vers le bas une et une seule fois (On verra plus tard des exceptions à cette règle qu'il faut avoir en tête pour pouvoir lire un programme).

Cela peut paraître logique ou évident mais il y a des conséquences dans l'exécution du programme.

### Exercice 2 :

Prédire le résultat des instructions suivantes puis les exécuter avec python :

a) $q = r$	b) $r = 1$	c) $a = 2$	d) $a = 2$
$r = 1$	$q = r$	$a = a * 0$	$a = a + 1$
$print(r)$	$print(q)$	$a = a + 1$	$a = a * 0$
<i>erreur</i> <i>non défini</i>		$print(a)$	$print(a)$
	1	1	0

Commentez.

### Exercices 3 :

a) Ecrire un petit « script » qui permet d'échanger les valeurs de  $a$  et  $b$  avec  $a = 2$  et  $b = 5$ . On veut donc qu'à la fin du programme que  $a = 5$  et  $b = 2$ .

$a, b = b, a$

b) Tapez le script suivant

```
x = 10
x = x + 5
print(x)
```

Commentez.

$x = x + 5$  revient à ajouter 5 à  $x$ ,  $\Rightarrow x = 15$   
 $\text{print}(x)$  renvoie 15

c) Tapez le script suivant et dire ce qu'il permet de faire. Justifiez !

```
a = 2
b = 3
```

```
a = a + b  5
b = a - b  2
a = a - b  3
```

} inverser a et b.

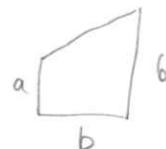
l3: On ajoute b à a  
l4: b prend la valeur de a-b, donc la valeur de a initialement  
l5: On met a à la différence, afin de donner la valeur b.

d) Tapez le script suivant : A traduire en langage python.

```
a = 10
b = 20
h = 5
Aire = 0.5 * (a + b) * h
Afficher Aire
```

```
a = 10
b = 20
h = 5
aire = 0.5 * (a + b) * h
print(aire)
```

Que fait ce script ? Calculer l'aire d'un triangle de côté  $a+b=10+20$  et de hauteur  $h=5$  et l'affiche.



trapeze rectangle

Remarque :

On peut noter que le nom d'une variable peut être un mot. Ici on a défini une variable : Aire.

Cela peut être très utile lorsque l'on veut rédiger un programme et qu'il intervient de nombreuses variables. Le nom de ces variables n'impacte pas le programme mais permet de s'y retrouver à la lecture du programme.

### Remarque importante :

Bien retenir que le symbole `=` sert à affecter « quelque chose » à une variable. Il ne s'agit pas du tout d'une égalité.

Il arrive souvent que l'on doive définir de nombreuses variables dans un script. On peut affecter les valeurs ( ou autres ) sur une seule ligne.

Tapez les lignes suivantes dans Python Shell

```
a, b = 2, 5
print(a) => 2
print(b) => 5
```

Commentez.  $a = (2, 5)[0]$   
 $b = (2, 5)[1]$

### Exercices 4 :

a) Tapez le script suivant et expliquez ce qu'il fait.

```
a, b, c, d, e = 0, 2, 4, 6, 8    a=0; b=2; c=4; d=6; e=8
a, b, c, d, e = a, d, c, b, a    a=a; b=d; c=c; d=b; e=a    donc a=0, b=6, c=4, d=2, e=0
print("a =", a, "b =", b, "c =", c, "d =", d, "e =", e)
```

$\Rightarrow$  "a=0 b=6 c=4 d=2 e=0"

b) Ecrire un script

- qui affecte la valeur 2 à a, 3 à b et -5 à c.
- puis affecte la valeur 2 à b et 3a à c.
- puis affiche les valeurs de a, b et c.

```
a=2
b=3
c=-5
b=2
c=3*a
print(a, b, c)
```

### D'autres instructions :

Tapez les instructions suivantes et dites ce qu'elles font.

```
>>> abs(3.5)    |3,5| = 3,5
>>> abs(-2.7)   |-2,7| = 2,7
>>> int(3.2)    3    (valeur entière)
>>> int(-3.2)   -3
>>> max(2, -3, 6) 6
>>> min(5, 8, 7) 5
>>> round(2.3659, 2) 2,36
>>> round(2.3659, 3) 2,366
```

Tableau à retenir :

Instruction	résultat
<code>abs(x)</code>	$ x $
<code>int(x)</code>	$\text{floor}(x)$
<code>max(a, b, c)</code>	plus grande valeur entre a, b, c
<code>min(a, b, c)</code>	plus petite valeur entre a, b, c
<code>round(x, n)</code>	arrondir x à n près.

Ces fonctions sont utilisables directement mais il existe d'autres fonctions définies par Python ou par des utilisateurs, elles sont stockées dans des bibliothèques.

Pour accéder à ces autres fonctions, il faut les **importer**.

Dans Python Shell, il faut taper la ligne suivante : `from math import *` et la valider.

Dans l'éditeur de programme on tapera la même ligne dans le script et elle sera validée chaque fois qu'on exécutera ce script.

Tapez les instructions suivantes et commentez.

```
>>> from math import *
>>> pi                 $\pi$ 
>>> floor(2.5)        entier inférieur 2
>>> floor(-3.56)      entier inférieur -4
>>> floor(10.34)      entier inférieur 10
>>> sqrt(9)            $\sqrt{9} = 3$ 
>>> sqrt(16)           $\sqrt{16} = 4$ 
>>> sqrt(2)            $\sqrt{2} = 1,414...$ 
```

Tableau à retenir :

Instruction	résultat
<code>pi</code>	renvoie la valeur de $\pi$ (3,14...)
<code>floor(x)</code>	arrondi au int inférieur (partie entière)
<code>sqrt(x)</code>	$\sqrt{x}$

### Exercices 5 :

a) Quelle est la différence entre les instructions `floor()` et `int()`?

Sur les négatifs, `int` enlève la partie décimale (`int(-2.1)  $\Rightarrow$  -2`) alors que `floor` arrondit à l'inférieur précis (`floor(-2.1)  $\Rightarrow$  -3`)

b) Taper les lignes suivantes :

```
>>> from fractions import *
>>> a = Fraction(3,5)
>>> print(a) 3/5       $\Rightarrow \frac{3}{5}$ 
>>> p = a.numerator
>>> print(p) 3
>>> q = a.denominator
>>> print(q) 5
```



Commentez les diverses instructions écrites ci-dessus.

Recommencez avec  $a = \text{Fraction}(123,12)$

41,4 (simplifie la fraction  $\frac{123}{12}$ )

c) Tapez les lignes suivantes :

```
>>> from fractions import *
>>> Fraction(165,35) + Fraction(58,18) 500/63
>>> Fraction(112,14) + Fraction(24,8) 11
>>> Fraction(165,35) * Fraction(52,16) 429/28
>>> Fraction(85,25)/Fraction(104,12) 51/130
```

Vérifiez les réponses données par Python à la main.

$$\frac{165}{35} + \frac{58}{18} = \frac{33}{7} + \frac{29}{9} = \frac{33 \times 9 + 29 \times 7}{7 \times 9} = \frac{500}{63}$$

$$\frac{112}{14} + \frac{24}{8} = 8 + 3 = 11$$

$$\frac{165}{35} \times \frac{52}{16} = \frac{33}{7} \times \frac{13}{4} = \frac{429}{28}$$

$$\frac{85}{25} / \frac{104}{12} = \frac{85}{25} \times \frac{12}{104} = \frac{17}{5} \times \frac{3}{26} = \frac{51}{130}$$

Remarque :

L'instruction  $\text{Fraction}(a,b)$  retourne toujours une fraction irréductible.

c) En utilisant l'instruction  $\text{Fraction}$ , calculer  $\frac{1+\frac{3}{4}}{\frac{5}{2}+\frac{7}{3}}$  et  $\frac{\frac{19}{15}-\frac{3}{7}}{\frac{11}{2}+\frac{8}{35}}$ .  $\frac{21}{58}$   $\frac{7169}{6015}$

Vérifiez à la main les résultats.

$$\frac{1+\frac{3}{4}}{\frac{5}{2}+\frac{7}{3}} = \frac{\frac{7}{4}}{\frac{15+14}{6}} = \frac{7}{4} \times \frac{6}{29} = \frac{7 \times 6}{4 \times 29} = \frac{21}{58}$$

$$\frac{\frac{19}{15}-\frac{3}{7}}{\frac{11}{2}+\frac{8}{35}} = \frac{\frac{133-45}{105}}{\frac{77+16}{70}} = \frac{\frac{88}{105}}{\frac{93}{70}} = \frac{88}{105} \times \frac{70}{93} = \frac{88 \times 70}{105 \times 93} = \frac{7169}{6015}$$

Les tests (valeurs booléennes) :

Tapez les instructions suivantes et noter ce qui se passe.

```
>>> a = 5
>>> b = 8
>>> a == b False
>>> a < b True
>>> a != b True
>>> a >= b False
```

Recommencez avec  $a = 12$  et  $b = 3$  puis  $a = 5$  et  $b = 5$ .

False  
False  
True  
True

True  
False  
False  
True

Tableau à retenir :

opérateurs	significations
$a == b$	Compare la valeur de $a$ et de $b$ , renvoie True si $a == b$ et False si $a \neq b$
$a > b$	si $a$ strictement supérieur à $b$ , renvoie True, sinon False
$a != b$	si $a$ n'est pas égal à $b$ , renvoie True, sinon False
$a <= b$	si $a$ inférieur ou égal à $b$ , renvoie True, sinon False

### Exercices 6 :

a) Tapez les instructions suivantes et dire ce qui se passe.

```
>>> from math import *
```

```
>>> a = sqrt(2)    a =  $\sqrt{2}$ 
```

```
>>> b = sqrt(3)    b =  $\sqrt{3}$ 
```

```
>>> c = 5           c = 5
```

```
>>> a ** 2 == 2     False car erreur d'arrondi
```

```
>>> (sqrt(3)) ** 2 == 3 False car erreur d'arrondi
```

```
>>> a ** 2 + b ** 2 == c True car les erreurs d'arrondi s'annulent
```

```
>>> a > 1.732      False car  $\sqrt{2} < 1,732$ 
```

TP2:

- 1) demander une valeur pour prénom dans le terminal. Renvoie "Bonjour Nicolas, comment vas-tu ?"
- 2) Script 1 ne marche pas car `type(input("")) = str`  
Script 2 marche car `type(eval(input(""))) = int`
- 3) OK
- 4) OK
- 5) OK