

---

# UE 1.2 : Informatique

## Nombres ou comment représenter le réel

Karine BRIFAULT et Tony FEVRIER

# Séance 2

---

## ☐ **1ères notions**

- **Algorithme**
- **Programme, Fonction**
- **Variable**
- **Paramètre**

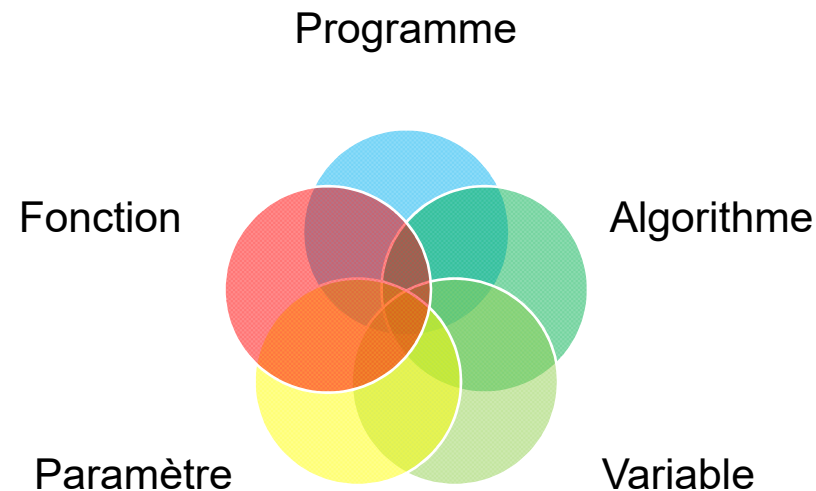
## ☐ **Ecrire et interpréter un programme**

## ☐ **La notion d'algorithme**

# Quelques notions... avant de programmer

---

## ❑ Avant de démarrer la programmation : quelques définitions



# Algorithme

---

## ❑ Algorithme

– **Suite finie d'opérations** permettant de **résoudre un problème**.

- ❑ Les algorithmes existaient avant l'informatique mais elle apporte un moyen pratique de programmer un algorithme et de l'exécuter automatiquement.

### Exemples :

1. Algorithme pour déterminer le plus grand de 2 nombres X et Y ou  $\max(X, Y)$  est :

Si  $X \geq Y$  alors  $\max(X, Y) = X$   
sinon  $\max(X, Y) = Y$

2. Pseudo-code de la **fonction** factorielle (n)

$r = 1$   
**Pour** i de 1 jusqu'à n  
     $r = r * i$   
**Fin pour**  
**Retourner** r

# Programme

---

## ❑ Programme

- Un **programme** est une **séquence d'instructions**

**Comme la séquence d'instructions pour faire un montage en Lego**

## ❑ Ces **instructions**, écrites dans un **langage de programmation**, sont des **ordres**, donnés à l'ordinateur afin qu'il exécute des actions.

- de base :
  - addition, multiplication,...
- ou plus complexes :
  - comparer des valeurs,
  - afficher des résultats,...

# Fonction (1)

---

## ❑ Fonction

- Une **Fonction** est aussi une **séquence d'instructions définie pour accomplir une tâche précise**

**Comme les fonctions de base du montage en Lego**

Programme

Fonction 2

Fonction 1

Fonction 4

Fonction 1

Fonction 2

Fonction 3

Fonction 4

# Fonction (2)

---

## □ Fonction

- Une **Fonction** est aussi une **séquence d'instructions définie pour accomplir une tâche précise**
- La fonction est à l'extérieur du programme principal
- Exemple :

Fonction « factorielle » qui comprend la séquence de code nécessaire au calcul d'une factorielle

`r = 1`

`Pour i de 1 jusqu'à n`

`r = r*i`

`Fin pour`

`Retourner r`

# Variable

## ❑ Variable

- Une variable est un **nom** qui renvoie à une **position de mémoire** (une boîte, un conteneur) dont le **contenu** peut prendre **successivement différentes valeurs** pendant l'exécution d'un programme.

- Exemple :

Fonction « factorielle »

$r = 1$

Pour  $i$  de 1 jusqu'à  $n$

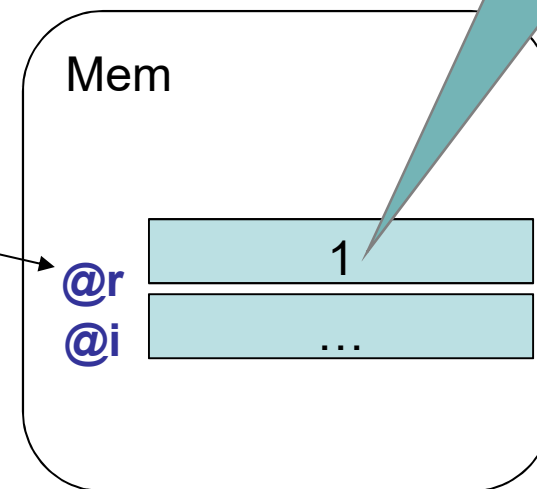
$r = r * i$

Fin pour

Retourner  $r$

Variable  $r$  qui contient 1

Valeur de  $r$

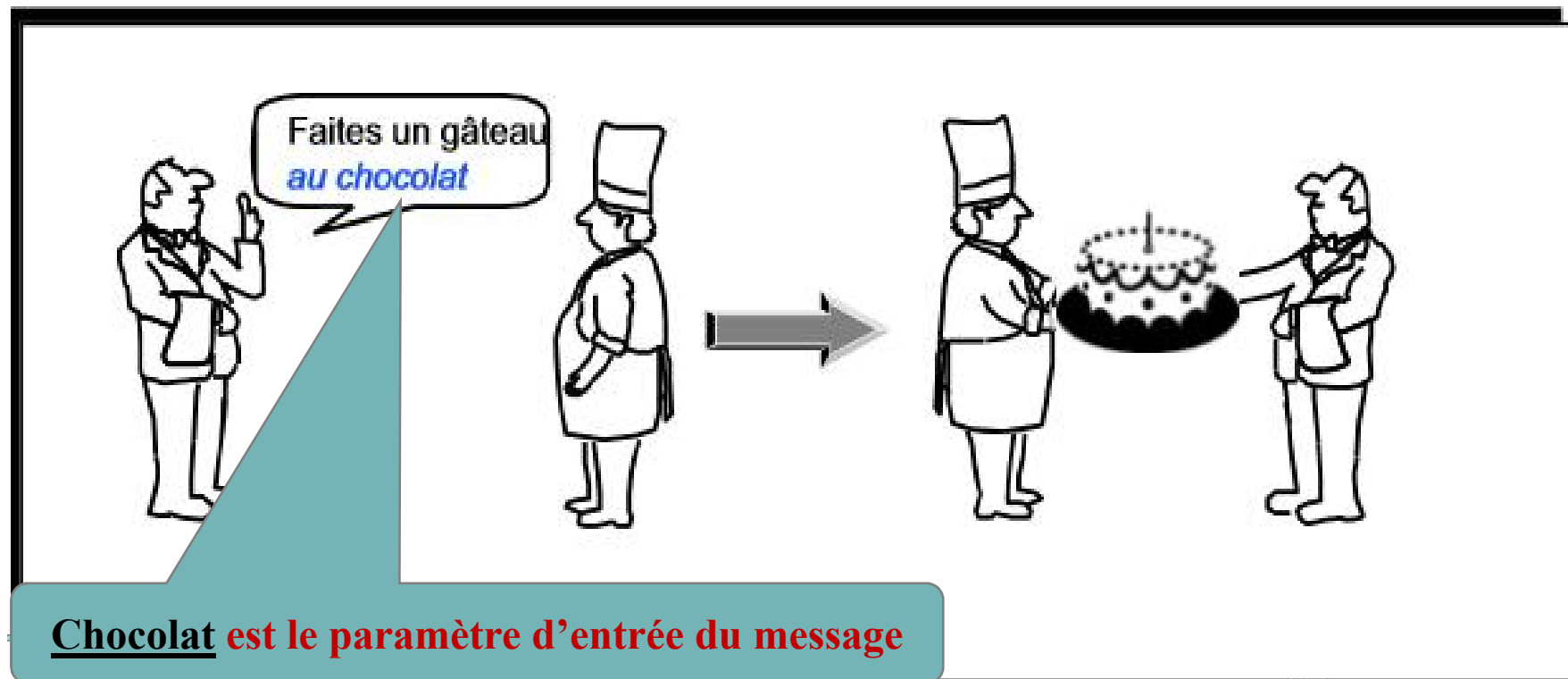




# Paramètre

## ❑ Paramètre:

- Un **paramètre** est au sens large un élément d'information à prendre en compte pour prendre une décision ou pour **effectuer un calcul**. On parle aussi d'*argument* dans une fonction.



# Séance 2

---

## □ 1ères notions

### □ *Ecrire et interpréter un programme*

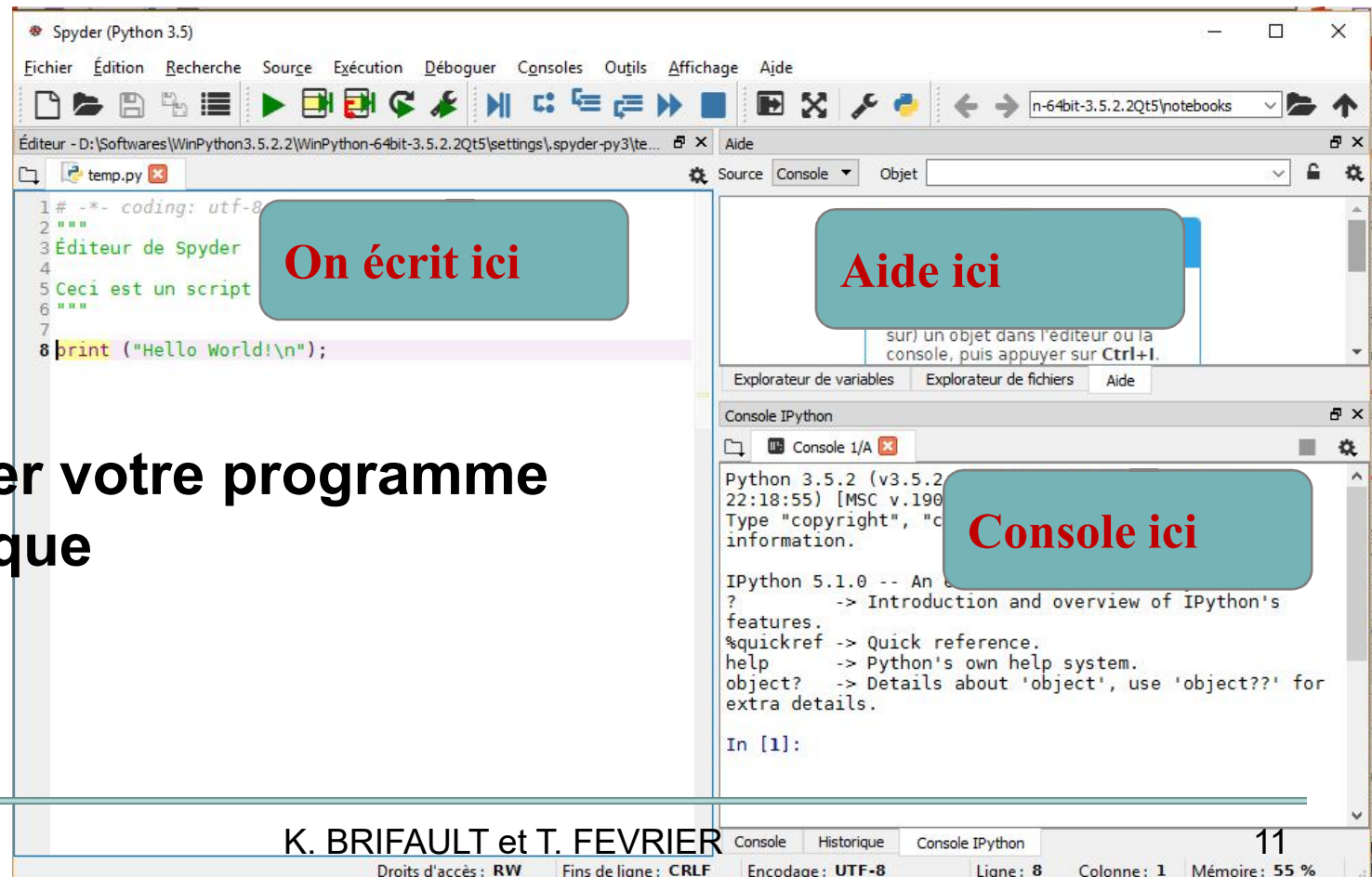
- Utilisation de l'IDE SPYDER
- Ecrire et lancer son 1<sup>er</sup> programme
- La fonction print

## □ La notion d'algorithme

# Comment écrire et interpréter un programme ?

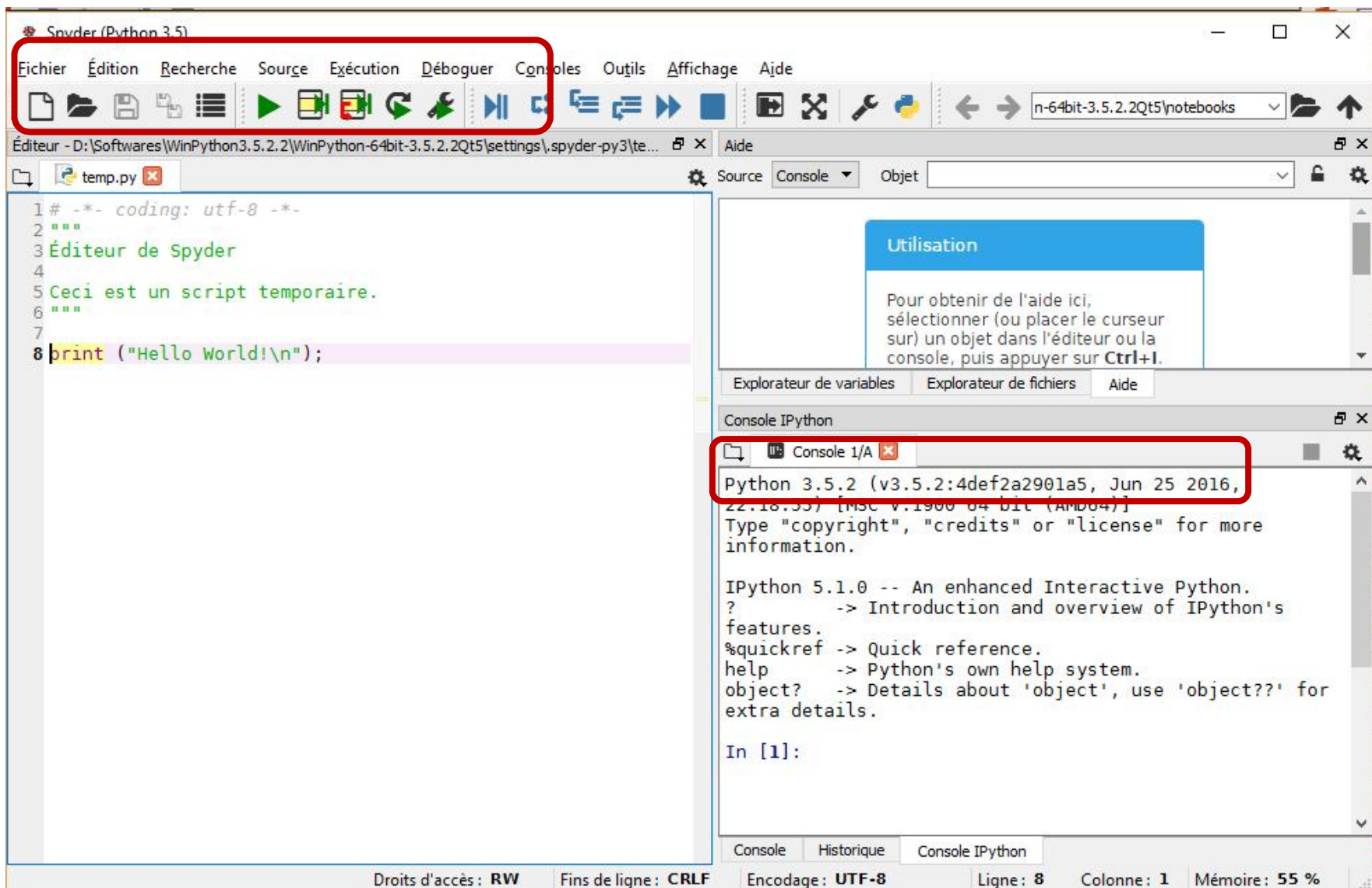
## ❑ Écrire son programme au moyen d'un éditeur de texte ou d'un IDE

- gedit
- spyder



## ❑ Enregistrer votre programme sur le disque

# L'IDE : SPYDER



# Un programme simple

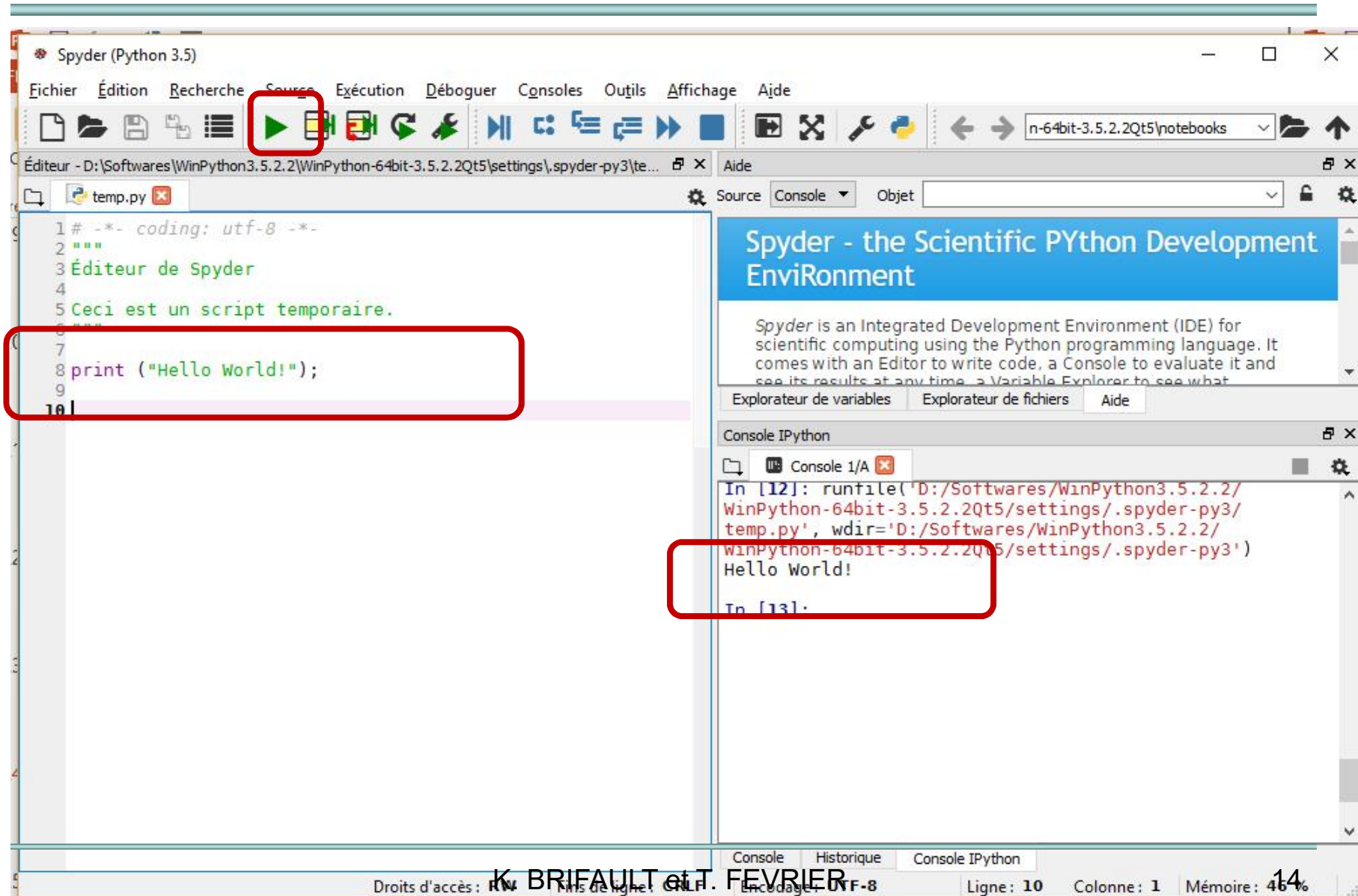
---

- ❑ Lancez-vous avec un programme simple :

```
print("Hello World !");
```

- ❑ Quelques questions :
  - Comment exécutez le programme ?
  - Que fait le programme ?
  - Quel est le rôle de **print** ?

# Correction : un programme simple





# Console : les erreurs classiques (1)

The screenshot shows the Spyder Python IDE interface. The main editor window displays a file named `temp.py` with the following code:

```
1 # -*- coding: utf-8 -*-
2 """
3 Éditeur de Spyder
4
5 Ceci est un script temporaire.
6 """
7
8 print ("Hello World!")
9
10
```

Annotations on the image:

- A red box highlights the file name `temp.py` in the toolbar, with a callout: **Nom du fichier**.
- A red box highlights line 10, which is empty, with a callout: **Ligne 10 : rien donc l'erreur est à la fin de l'écriture du programme**.
- A red box highlights the error message in the IPython console: **SyntaxError: unexpected EOF while parsing**, with a callout: **L'erreur relevée : claire :D pour un ordinateur !**.

The IPython console output shows the execution of the script:

```
WinPython-64bit-3.5.2.2Qt5/settings/.spyder-py3/
temp.py', wdir='D:/Softwares/WinPython3.5.2.2/
WinPython-64bit-3.5.2.2Qt5/settings/.spyder-py3')
Hello World!

In [14]: runfile('D:/Softwares/WinPython3.5.2.2/
WinPython-64bit-3.5.2.2Qt5/settings/.spyder-py3/
temp.py', wdir='D:/Softwares/WinPython3.5.2.2/
WinPython-64bit-3.5.2.2Qt5/settings/.spyder-py3')
File "D:/Softwares/WinPython3.5.2.2/
WinPython-64bit-3.5.2.2Qt5/settings/.spyder-py3/
temp.py", line 10
    ^
SyntaxError: unexpected EOF while parsing

In [15]:
```

The status bar at the bottom indicates: `Droits d'accès : RWK. BRIEAULT et T. FEVRIER`, `Ligne : 8`, `Colonne : 22`, `Mémoire : 4615`.

# Console : les erreurs classiques (2)

---

- Format classique d'une erreur :

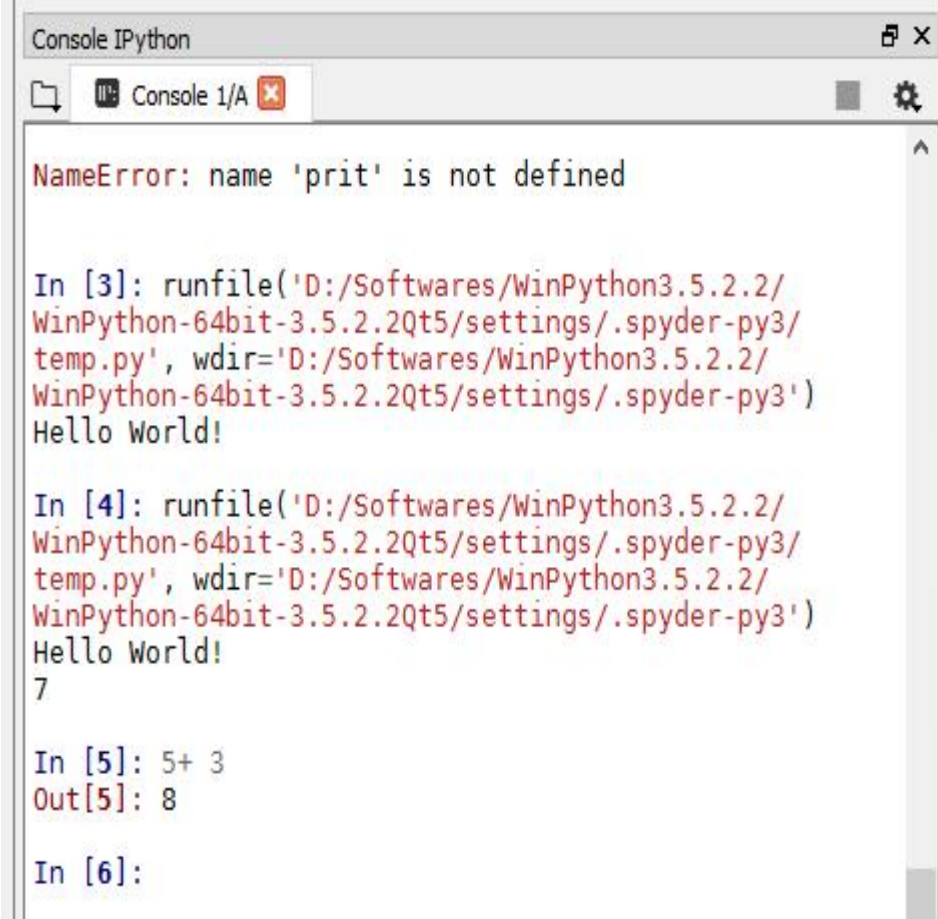
`temp.py, line 10`  
  
**Fichier concerné**      **Ligne dans le fichier**      **Description du problème**  
`Syntax error : unexpected EOF while parsing`

- Unexpected EOF while parsing
  - EOF : end of file. La fin du fichier est incorrecte : dans l'exemple, il manque « ) ».
- `name 'prit' is not defined`
  - Il y a une erreur de syntaxe ( $\Leftrightarrow$  faute d'orthographe)



# Console sous Spyder : Utilisation simple

- ❑ Faire le calcul  $5 + 3$  dans la console.
  - Cela revient à faire du python dans n'importe quel terminal
  - Les espaces sont optionnels
    - $5+3 \Rightarrow 8$
    - $5 + 3 \Rightarrow 8$
    - $5+3 \Rightarrow 8$



```
Console IPython
Console 1/A

NameError: name 'prit' is not defined

In [3]: runfile('D:/Softwares/WinPython3.5.2.2/
WinPython-64bit-3.5.2.2Qt5/settings/.spyder-py3/
temp.py', wdir='D:/Softwares/WinPython3.5.2.2/
WinPython-64bit-3.5.2.2Qt5/settings/.spyder-py3')
Hello World!

In [4]: runfile('D:/Softwares/WinPython3.5.2.2/
WinPython-64bit-3.5.2.2Qt5/settings/.spyder-py3/
temp.py', wdir='D:/Softwares/WinPython3.5.2.2/
WinPython-64bit-3.5.2.2Qt5/settings/.spyder-py3')
Hello World!
7

In [5]: 5+ 3
Out[5]: 8

In [6]:
```

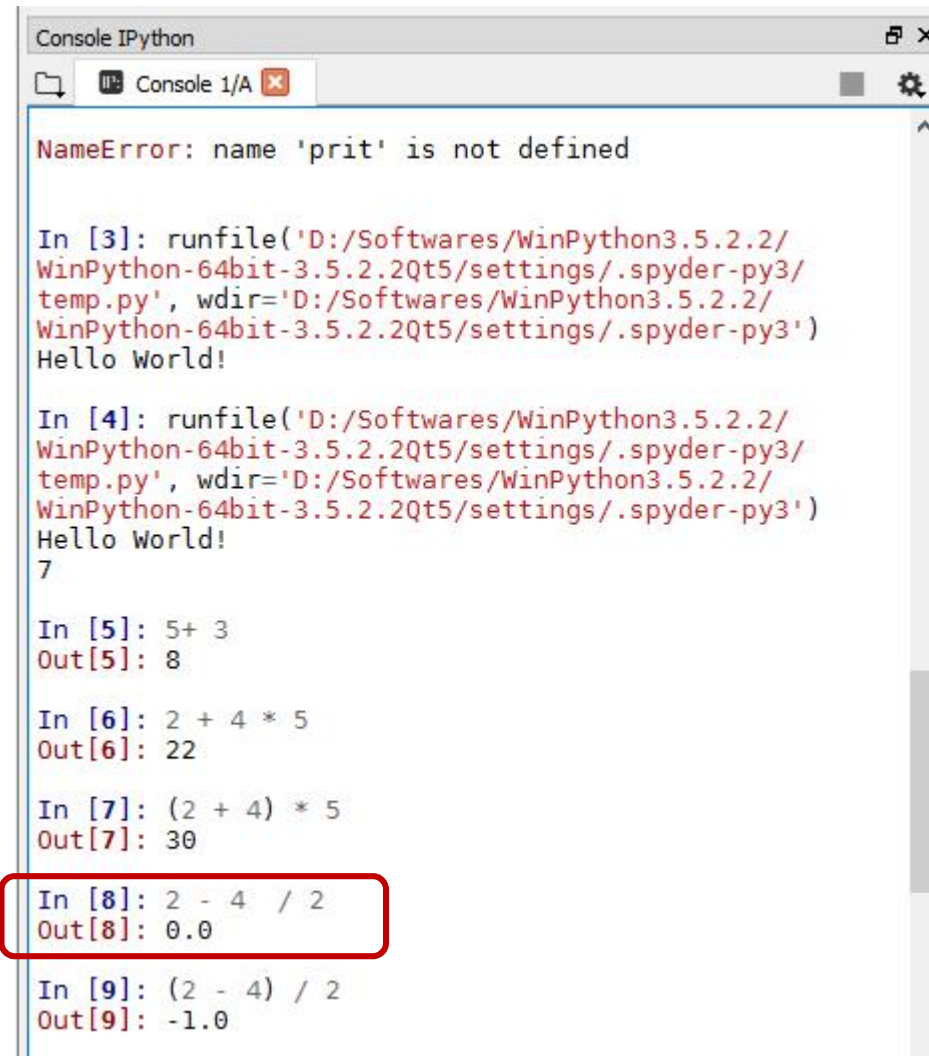
# Console sous Spyder : les priorités

❑ Les opérateurs ont des **priorités entre eux** :

- \* et / ont la priorité sur + et -.

❑ La hiérarchie des opérateurs Mathématique est respectée en Informatique.

On note qu'en python 5.2.2, la **conversion est automatique et Implicite**.



```
Console IPython
Console 1/A

NameError: name 'prit' is not defined

In [3]: runfile('D:/Softwares/WinPython3.5.2.2/WinPython-64bit-3.5.2.2Qt5/settings/.spyder-py3/temp.py', wdir='D:/Softwares/WinPython3.5.2.2/WinPython-64bit-3.5.2.2Qt5/settings/.spyder-py3')
Hello World!

In [4]: runfile('D:/Softwares/WinPython3.5.2.2/WinPython-64bit-3.5.2.2Qt5/settings/.spyder-py3/temp.py', wdir='D:/Softwares/WinPython3.5.2.2/WinPython-64bit-3.5.2.2Qt5/settings/.spyder-py3')
Hello World!
7

In [5]: 5+ 3
Out[5]: 8

In [6]: 2 + 4 * 5
Out[6]: 22

In [7]: (2 + 4) * 5
Out[7]: 30

In [8]: 2 - 4 / 2
Out[8]: 0.0

In [9]: (2 - 4) / 2
Out[9]: -1.0
```

# Les noms de variables

---

## ❑ Les noms de variables doivent obéir à quelques règles simples :

- Un **nom de variable est une séquence de lettres** (a ® z , A ® Z) et de **chiffres** (0 ® 9), qui doit toujours commencer par une lettre.
- Seules les **lettres ordinaires sont autorisées**. Les lettres accentuées, les cédilles, les espaces, les caractères spéciaux tels que \$, #, @, etc. sont interdits, à l'exception du caractère \_ (souligné).
- La **casse est significative** (les caractères majuscules et minuscules sont distingués).

### Exemples

uneVariable <> unevariable <> une\_Variable

# Les mots réservés

---

❑ **Interdit d'utiliser** ces mots pour le nom de variables, car ils sont utilisés par le langage python lui-même.

and

assert

break

del

elif

else

for

from

global

is

lambda

not

raise

return

try

class

continue

def

except

exec

finally

if

import

in

or

pass

print

while

yield

# Les variables et leurs types

---

## ❑ Nombre entier

**Déclaration de i et affectation à 10**

- `i = 10;`
- Afficher i (dans la console)

## ❑ Caractère alphanumérique

- `c = 'A';`
- Afficher c

**Les types sont IMPLICITES !!!**

## ❑ Type réel

- `x = 10.456;`
- Afficher x

## ❑ Afficher en utilisant la fonction `print`

# Correction : les variables et leurs types

## ❑ Nombre entier

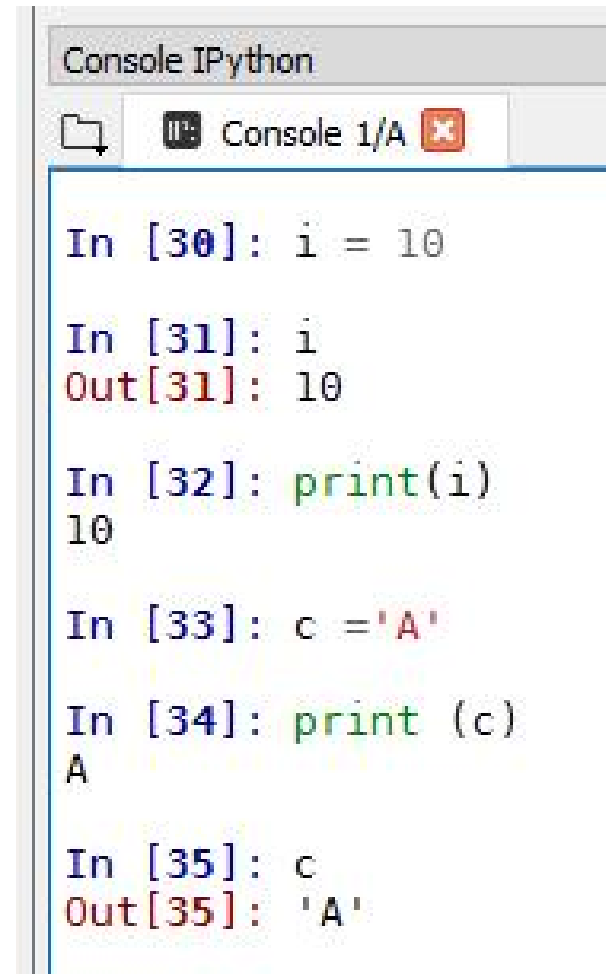
- `i = 10;`
- Afficher `i` (dans la console)

## ❑ Caractère alphanumérique

- `c = 'A';`
- Afficher `c`

## ❑ Type réel

- `x = 10.456;`
- Afficher `x`



```
Console IPython
Console 1/A

In [30]: i = 10

In [31]: i
Out[31]: 10

In [32]: print(i)
10

In [33]: c = 'A'

In [34]: print(c)
A

In [35]: c
Out[35]: 'A'
```

## ❑ Afficher en utilisant la fonction `print`

# Les variables et affectations parallèles

## ❑ Affectation par valeur identique

- `i = j = 10;`
- Afficher `i` et `j`

## ❑ Affectation par valeur différente

- `a, b = 1.2, 4;`
- Afficher `a` et `b`

Console IPython



Console 1/A



```
In [37]: i = j = 10
```

```
In [38]: i
```

```
Out[38]: 10
```

```
In [39]: j
```

```
Out[39]: 10
```

```
In [40]: a, b = 1.2, 4
```

```
In [41]: a
```

```
Out[41]: 1.2
```

```
In [42]: b
```

```
Out[42]: 4
```

# Exercices : Fahrenheit

---

☐ Ecrivez un programme de conversion de degrés Celsius en degrés Fahrenheit.

☐ On rappelle que :

$$F = 9/5 C + 32$$

☐ Avez-vous obtenu le résultat attendu ? Pourquoi ?

$$22^{\circ}\text{C} \Leftrightarrow 71.6^{\circ}\text{F}$$



# Correction : Fahrenheit

---

- ❑ Ecrivez un programme de conversion de degrés Celsius en degrés Fahrenheit.

- ❑ On rappelle que :

$$F = 9/5 C + 32$$

```
5 Ceci est un script temporaire.  
6 """  
7  
8 C = 22  
9 F = (9 / 5) * C + 32  
10 print("%d°C <=> %.2f°F" % (C,F))
```

```
In [6]: runfile('D:/Softwares/W.  
wdir='D:/Softwares/WinPython3.5  
22°C <=> 71.60°F
```

# Exercices : Savoir programmer (1)

---

1. On se donne un carré de côté 2. Ecrire un programme stockant dans une variable  $p$ , le périmètre du carré et dans  $a$  l'aire du carré.
2. On se donne deux points  $(1, 2)$ ,  $(3, 0)$ . Ecrire en python une séquence d'instructions donnant le coefficient directeur  $a$  de la droite passant par ces deux points et l'ordonnée à l'origine  $b$ . Le programme définira deux variables nommées  $a$  et  $b$ .
3. On se donne un triangle rectangle dont les côtés adjacents à l'angle droit sont de longueur 1 et 2. Ecrire un programme calculant la longueur de l'hypoténuse et la stockant dans une variable nommée  $h$ .

# Exercices : Savoir programmer (2)

---

Composons des fonctions : on se donne deux fonctions

$$f : x \mapsto \frac{1}{x} \text{ et } g : x \mapsto x^2.$$

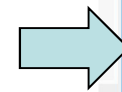
- Ecrire un programme calculant la valeur de  $g(2)$  et la stockant d'une variable notée  $a$  puis calculant la valeur de  $f(g(2))$  et la stockant dans une variable  $b$ .

# Correction : Savoir programmer (1)

---

1. On se donne un carré de côté 2. Ecrire un programme stockant dans une variable p, le périmètre du carré et dans a l'aire du carré.

```
/
8 c = 2
9
10 " perimetre du carre "
11 p = 4 * c
12 print("perimetre = %d m" %(p))
13
14 " aire du carre "
15 a = c * c
16 print("aire = %d m^2" %(a))
```



```
In [11]: runfile('D:
wdir='D:/Softwares/w
perimetre = 8 m
aire = 4 m^2
```

# Correction : Savoir programmer (2)

On se donne deux points (1, 2), (3, 0). Ecrire en python une séquence d'instructions donnant le coefficient directeur  $a$  de la droite passant par ces deux points et l'ordonnée à l'origine  $b$ . Le programme définira deux variables nommées  $a$  et  $b$ .

```
8 "point A"
9 x_a = 1
0 y_a = 2
1
2 "point B"
3 x_b = 3
4 y_b = 0
5
6 "coefficient directeur"
7 a = (y_a - y_b) / (x_a - x_b)
8 print("a = %.1f" %(a))
9
0 "la valeur b pour la droite de type y = a*x + b"
1 b = y_a - (a * x_a)
2 print("b = %.1f" % (b))
```



```
In [18]:
wdir='D:/
a = -1.0
b = 3.0
```

# Correction : Savoir programmer (3)

On se donne un triangle rectangle dont les côtés adjacents à l'angle droit sont de longueur 1 et 2. Ecrire un programme calculant la longueur de l'hypoténuse et la stockant dans une variable nommée h.

```
5
6 "Pythagore"
7 import math
8
9 AB = 1
10 AC = 2
11 h = math.sqrt (AB**2 + AC**2)
12 print("hypothénuse = %.2f"% h)
```

OU

```
6 "Pythagore"
7 from math import sqrt
8
9 AB = 1
10 AC = 2
11 h = sqrt (AB**2 + AC**2)
12 print("hypothénuse = %.2f"% h)
```



```
In [24]: runfile('E:/0%
FORMATIONS/VILLEBON/PY
hypothénuse = 2.24
```

# Correction : Savoir programmer (4)

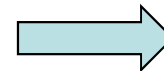
---

Composons des fonctions : on se donne deux fonctions

$$f : x \mapsto \frac{1}{x} \text{ et } g : x \mapsto x^2.$$

- ❑ Ecrire un programme calculant la valeur de  $g(2)$  et la stockant d'une variable notée  $a$  puis calculant la valeur de  $f(g(2))$  et la stockant dans une variable  $b$ .

```
7
8 "g(x) -> x**2"
9 x=2
10 a = x**2
11 b = 1 / a
12 print ("a=%.2f, b=%.2f" % (a,b))
```



```
In [26]: runfile
02 - Travail/07
a=4.00, b=0.25
```

# Exercices : Savoir debugger

---

Taper les programmes suivants, lire l'erreur, la comprendre et proposer une correction

1.  $a = b$

$b = 2$

$b = b * a$

2.  $a, b, c = 1, 2, 3$

$a = (((a + b) * c + 2 * b) * (a - b) + 2$



# Correction : Savoir debugger

Taper les programmes suivants, lire l'erreur, la comprendre et proposer une correction

**b n'est pas défini, donc a ne peut pas être défini**

1.  $a = b$

$b = 2$

$b = b * a$

**a n'est pas défini, donc b ne peut pas être redéfini**



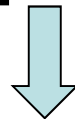
**$b = 2$**

**$a = b$**

**$b = b * a$**

2.  $a, b, c = 1, 2, 3$

$a = (((a + b) * c + 2 * b) * (a - b) + 2$



**Manque une parenthèse**

**$a = (((a + b) * c + 2 * b) * (a - b) + 2)$**

# Exercices : Savoir lire un programme

---

Donner (sans utiliser l'ordinateur) pour chacun des jeux d'instructions suivantes les valeurs des variables a,b. Vous noterez toutes les évolutions des valeurs de ces variables au cours du programme.

1. `a = 1`  
`b = 2`  
`a = a + 2`  
`b = b * a`

2. `a = 2`  
`b = 2`  
`a = a**2 - 2`  
`b = b / a`

3. `b = 1,`  
`a = 20`  
`a = a + a % 6`  
`a = a**2 - 2`

# Correction : Savoir lire un programme

---

Donner (sans utiliser l'ordinateur) pour chacun des jeux d'instructions suivantes les valeurs des variables a,b. Vous noterez toutes les évolutions des valeurs de ces variables au cours du programme.

1. a = 1  
b = 2  
a = a + 2  
b = b \* a



**a = 3**  
**b = 6**

2. a = 2  
b = 2  
a = a\*\*2 - 2  
b = b / a



**a = 2**  
**b = 1**

3. b = 1,  
a = 20  
a = a + a % 6  
a = a\*\*2 - 2



**a = 482**  
**b = 1**

# Boucle while

---

❑ L'instruction '**while**' permet d'exécuter plusieurs fois la même série d'instructions : **c'est une boucle !**

❑ Syntaxe de la boucle '**while**':

---

```
1 i = 1
2 while i <= 5:
3     print(i)
4     i = i + 1
5 print('Fini !')
```

---

Sortie du programme:

---

```
1
2
3
4
5
Fini !
```

# Boucle for

---

❑ L'instruction '**for**' permet d'exécuter plusieurs fois la même série d'instructions : **c'est une boucle !**

❑ Syntaxe de la boucle '**for**':

---

```
for i in range (1,6):  
    print (i)  
print("Fini !")
```

---

Sortie du programme:

```
1  
2  
3  
4  
5  
Fini !
```

## Exercice : boucle

---

- ❑ Ecrire un programme qui calcule la somme des 30 premiers entiers

$$1 + 2 + 3 + \cdots + 30$$

$$\sum_{i=1}^{30} i$$

# Correction : boucle

---

- ❑ Ecrire un programme qui calcule la somme des 30 premiers entiers

$$1 + 2 + 3 + \dots + 30$$

$$\sum_{i=1}^{30} i$$

```
1 # -*- coding: utf-8 -*-
2 """
3 Éditeur de Spyder
4
5 Ceci est un script temporaire.
6 """
7
8 sum = 0
9 for i in range (1,31):
10     sum = sum + i
11 print ("sum(%d) = %d" % (i, sum))
12
```

# Séance 2

---

□ 1ères notions

□ Ecrire et interpréter un programme

□ *La notion d'algorithme*