Université Batna 2
Faculté de Technologie
Département de Génie Industriel

TP:1

# Initiation à Python



H.zermane@univ-batna2.dz
N.zerari@univ-batna2.dz

# 1. Introduction

La programmation est l'art de commander à un ordinateur de faire ce qu'on souhaite.

Python compte parmi les langages de programmation qui offre cette possibilité.

# 2. Caractéristiques du langage

- Python est un langage de programmation :
  - Portable.
  - Libre.
  - Gratuit.

- Python permet une approche modulaire et orientée objet de la programmation.

## Installer Anaconda

## Lancer Anaconda

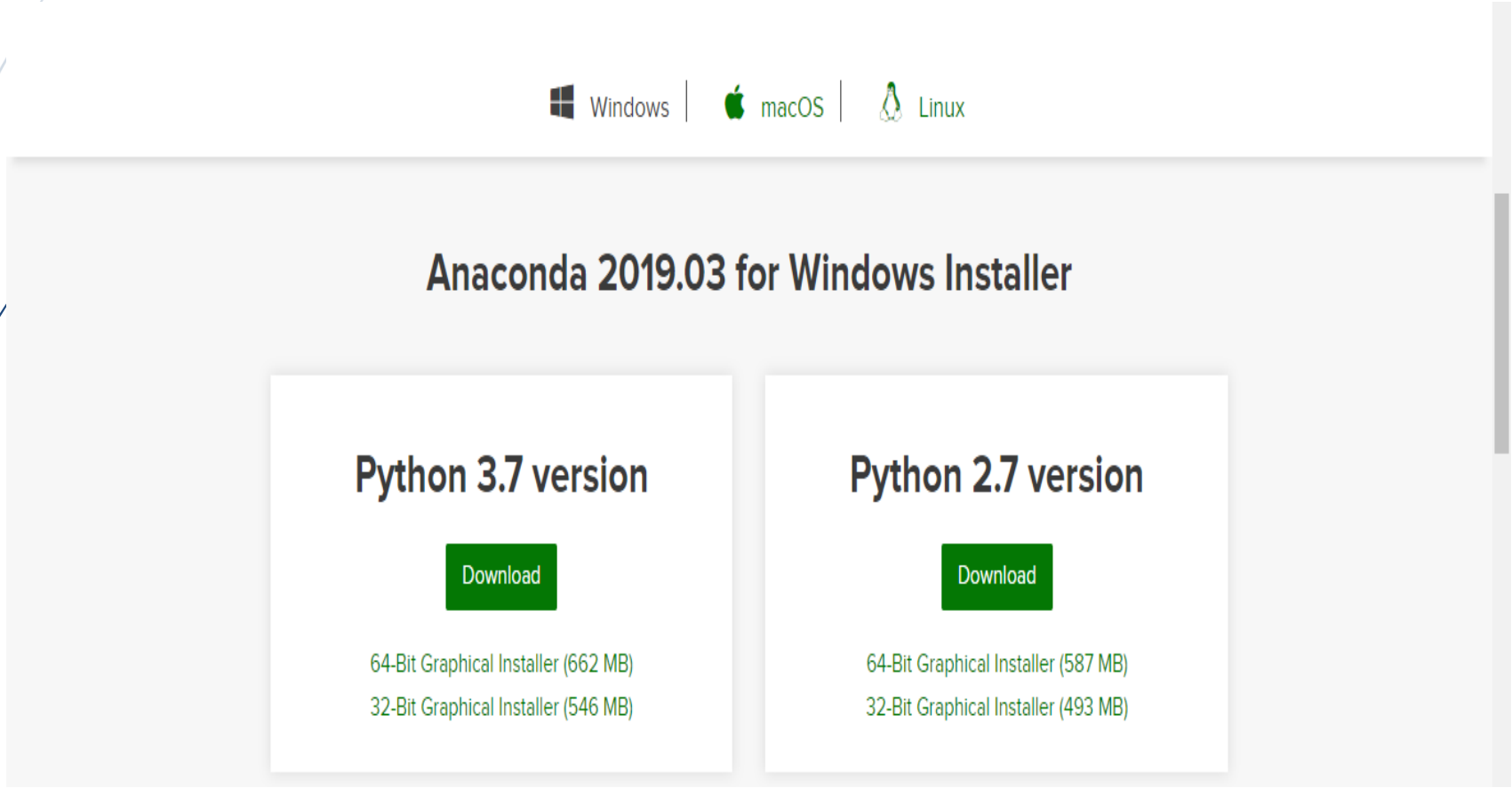
# Anaconda

Anaconda est une plateforme logicielle libre comprenant l'environnement Python complet et de très nombreuses librairies (numpy, Seaborn, …).

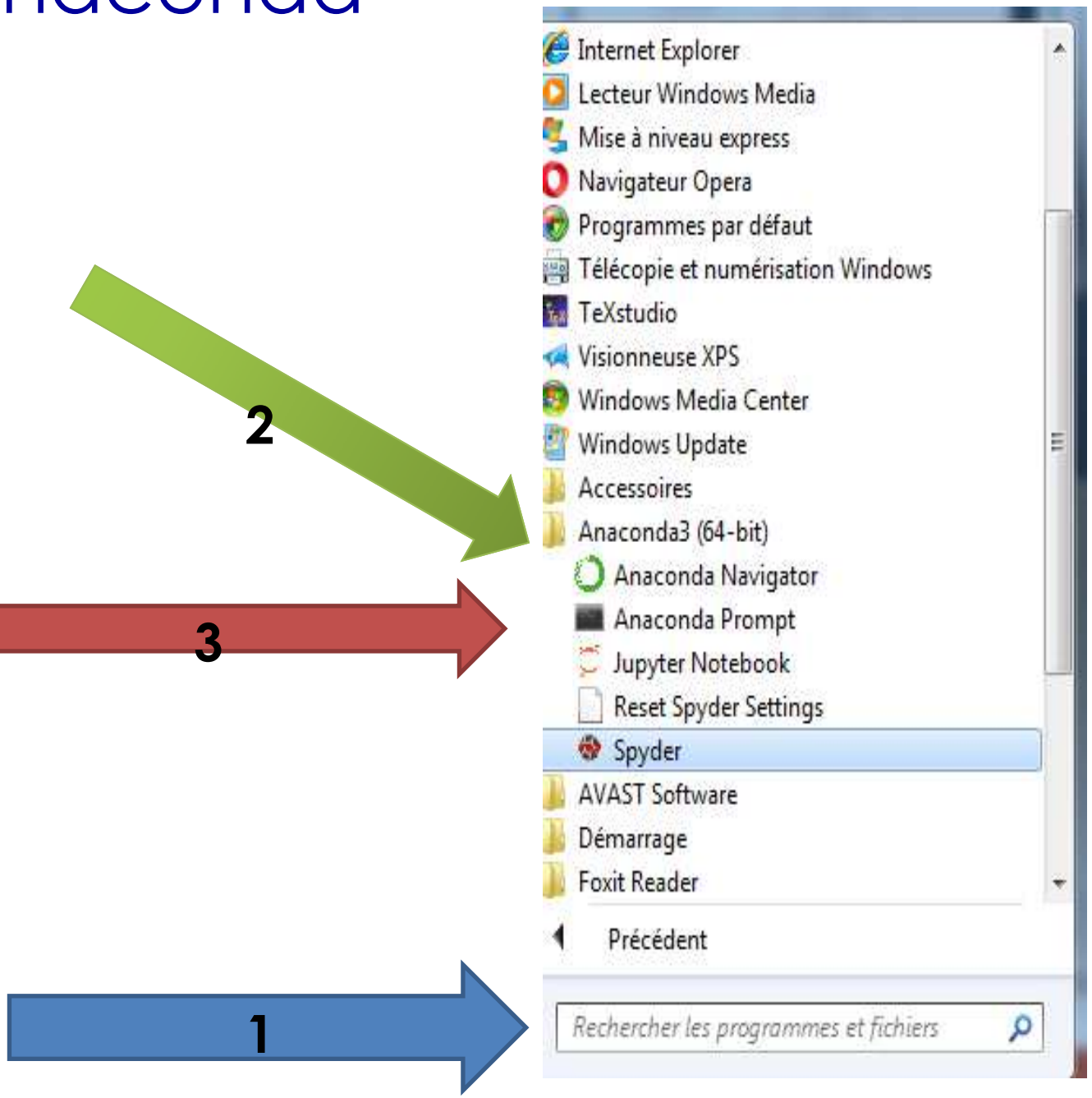
Cette plateforme est appliquée au développement d'applications dédiées à l'apprentissage automatique sous différent systèmes d'exploitation telle que Unix, Windows.

# Installer Anaconda

**www.anaconda.com**



# Lancer anaconda



# Utilisation de Python

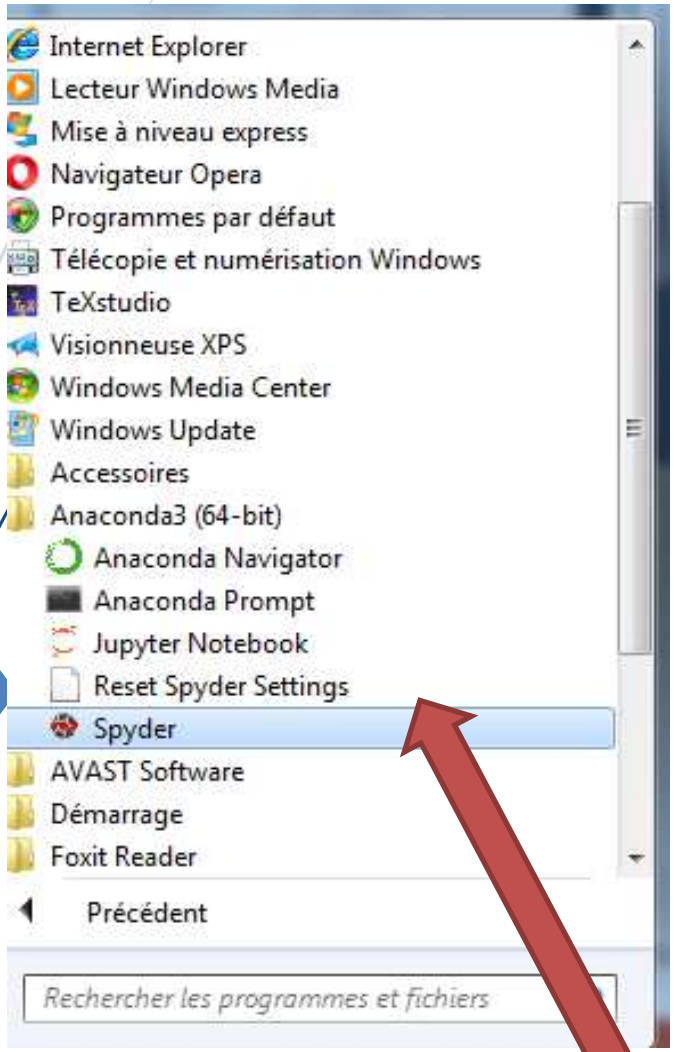
Pour utiliser Python, on fait appel à

**Spyder :** Environement de développement scientifique

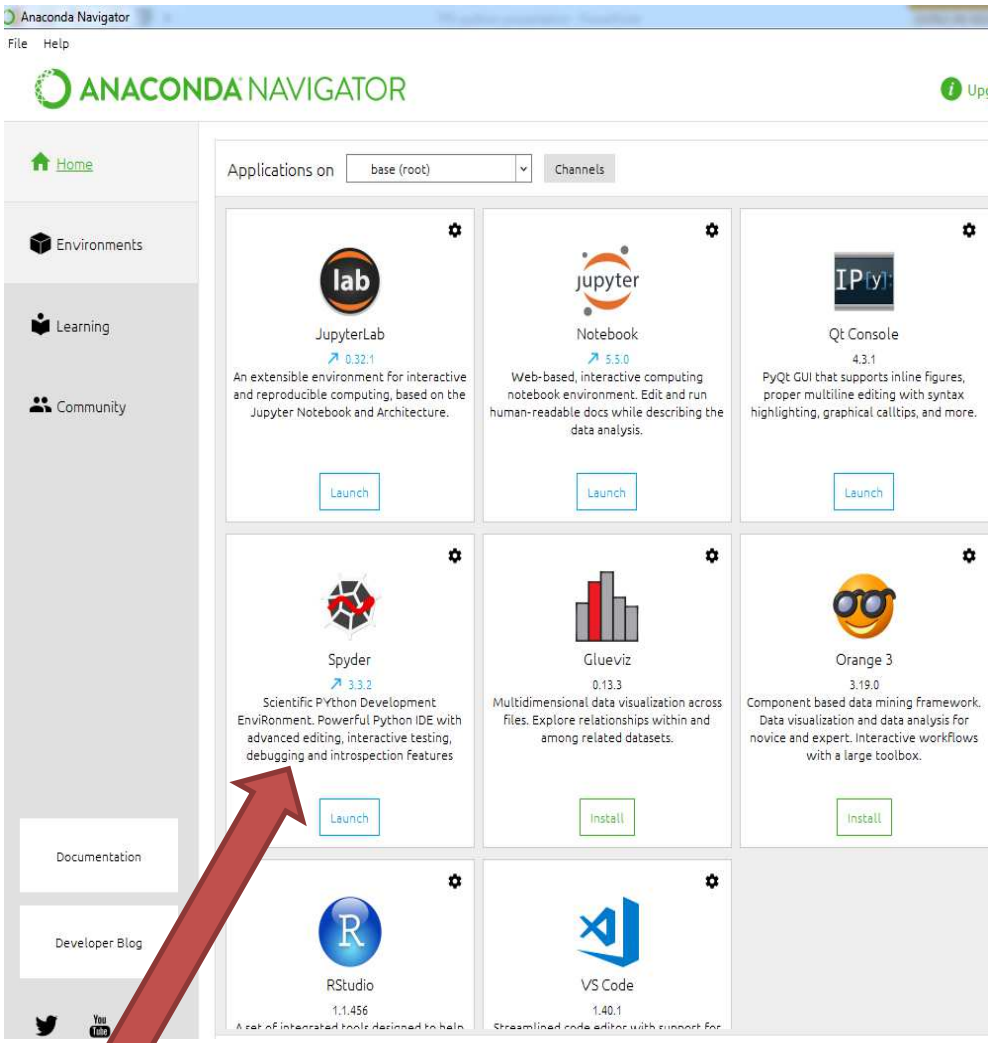
Lancer Spyder

# Lancer Spyder

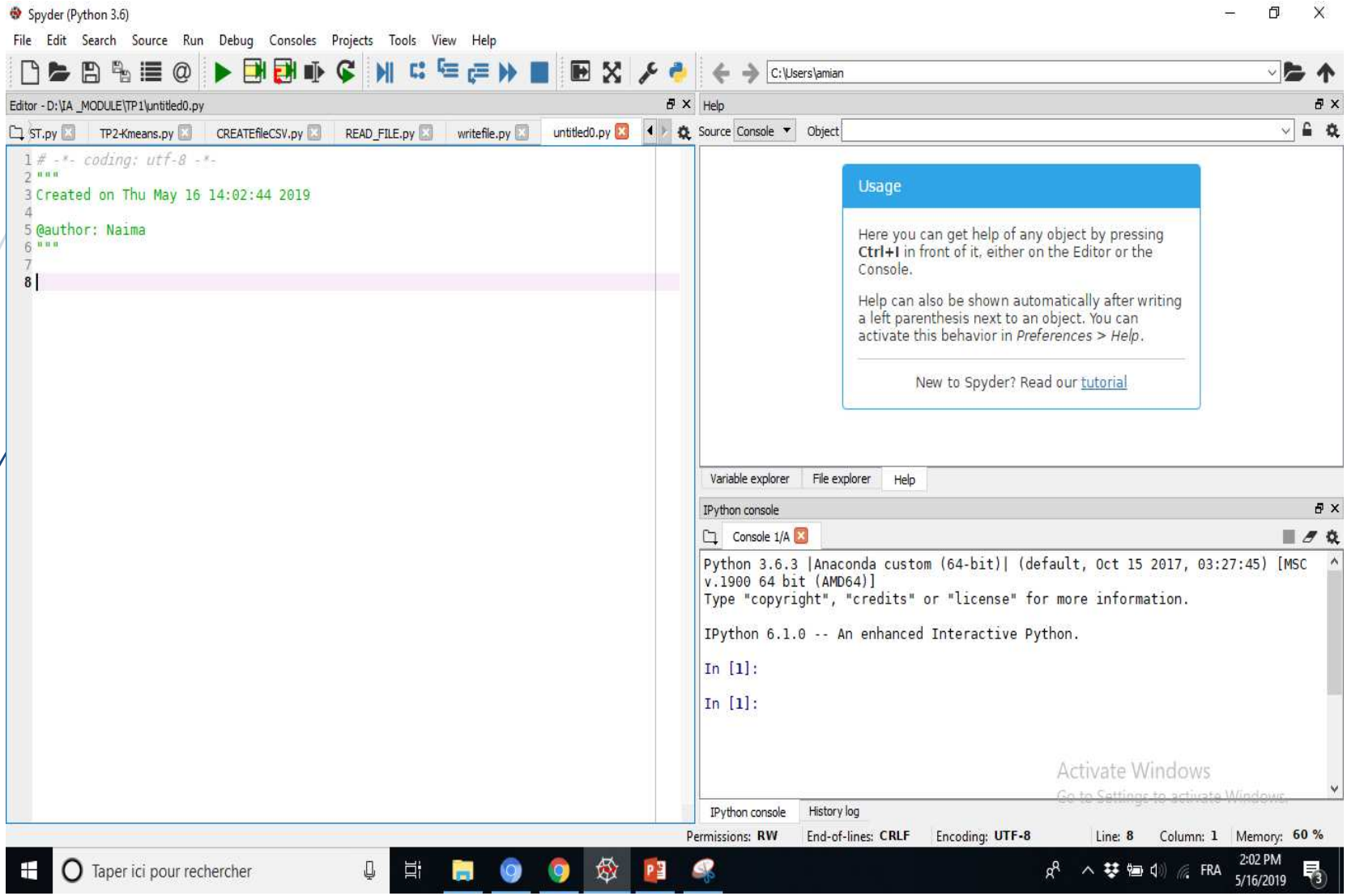
**(1)**




**(2)**



# SPYDER :
# Environement de développement scientifique



# 3. Mode d'Utilisation de Python

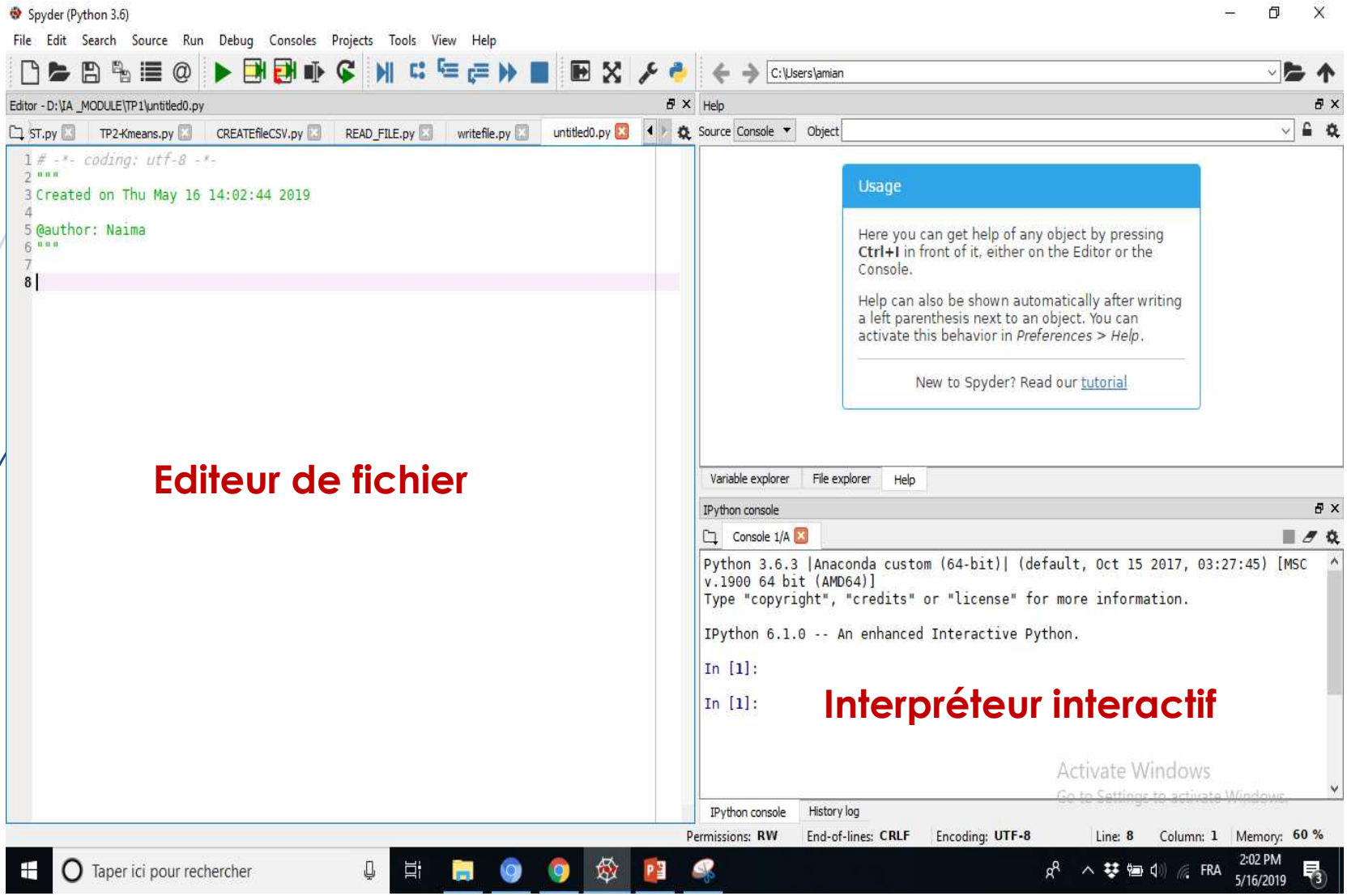
Python peut être utilisé en deux modes:

**a. Mode interactif**, i.e., dialoguer avec Python directement en utilisant l'invite de commande.

(sans les sauvegarder au préalable dans un fichier).

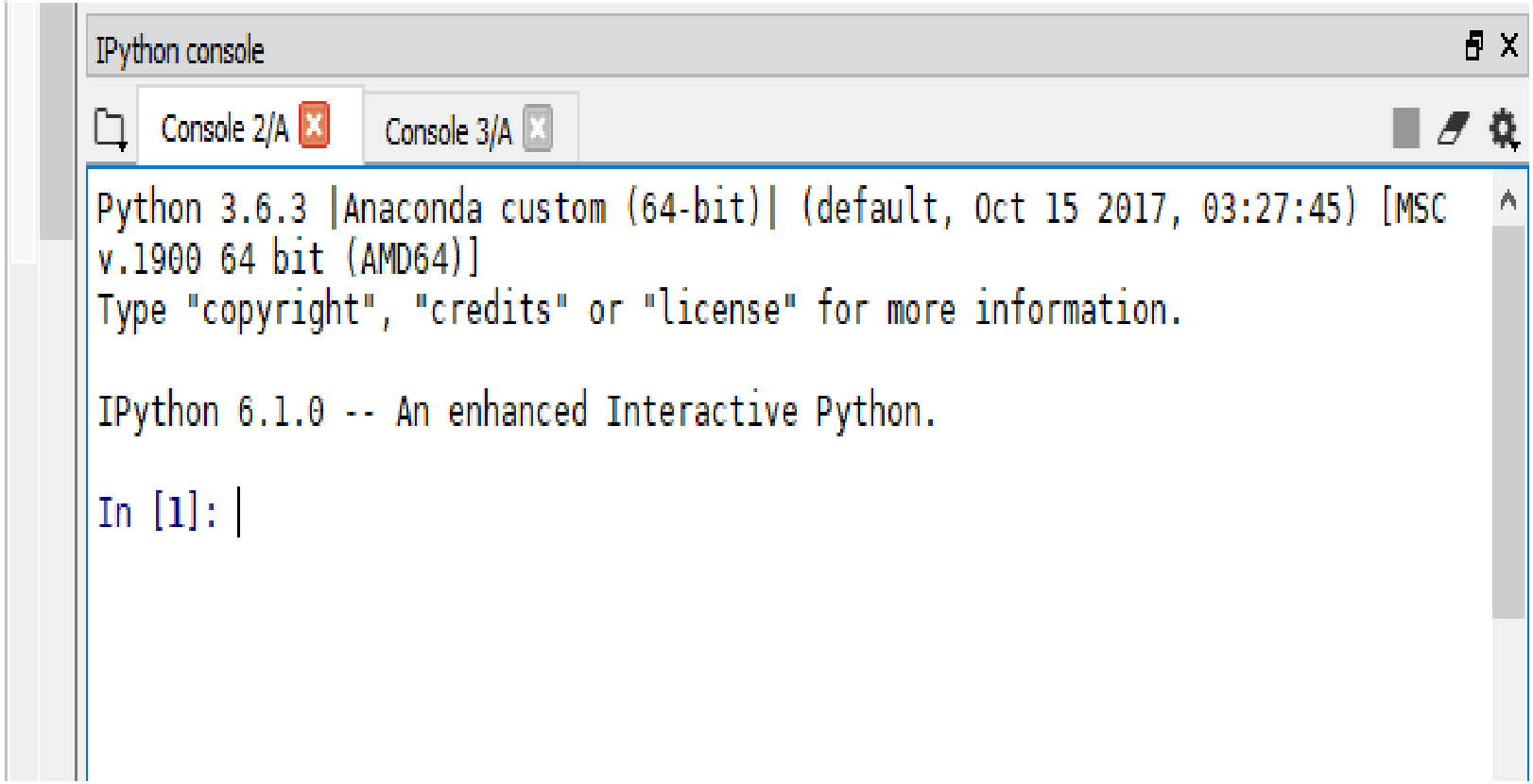
**b. Mode script:** écrire un programme (code) et le sauvegarder sur disque puis l'interpréter.

# SPYDER :
# Environement de développement scientifique



# a. Mode interactif

- L'interpréteur interactif permet d'écrire et d'exécuter du code, de faire des tests rapides.



# Exemples de Calcul

```
In [1]: 8575+965
Out[1]: 9540

In [2]: 58*63
Out[2]: 3654

In [3]: 254/96+50-33
Out[3]: 19.645833333333336
```

# 4. Noms de variables

Les noms de variables doivent respecter quelques règles simples :

Un nom de variable est une séquence de lettres (a → z , A → Z) et de chiffres (0 → 9).

- Un nom de variable doit commencer par une lettre.
- Le caractère "_" est **autorisé**.
- Les lettres accentuées, les cédilles, les espaces, les caractères spéciaux tels que $, #, @, etc. sont **interdits**.
- La casse est significative (les caractères majuscules et minuscules sont distingués).

# 5. Opérations de base

1. Affectation
2. Entrée des données
3. Sortie des résultats

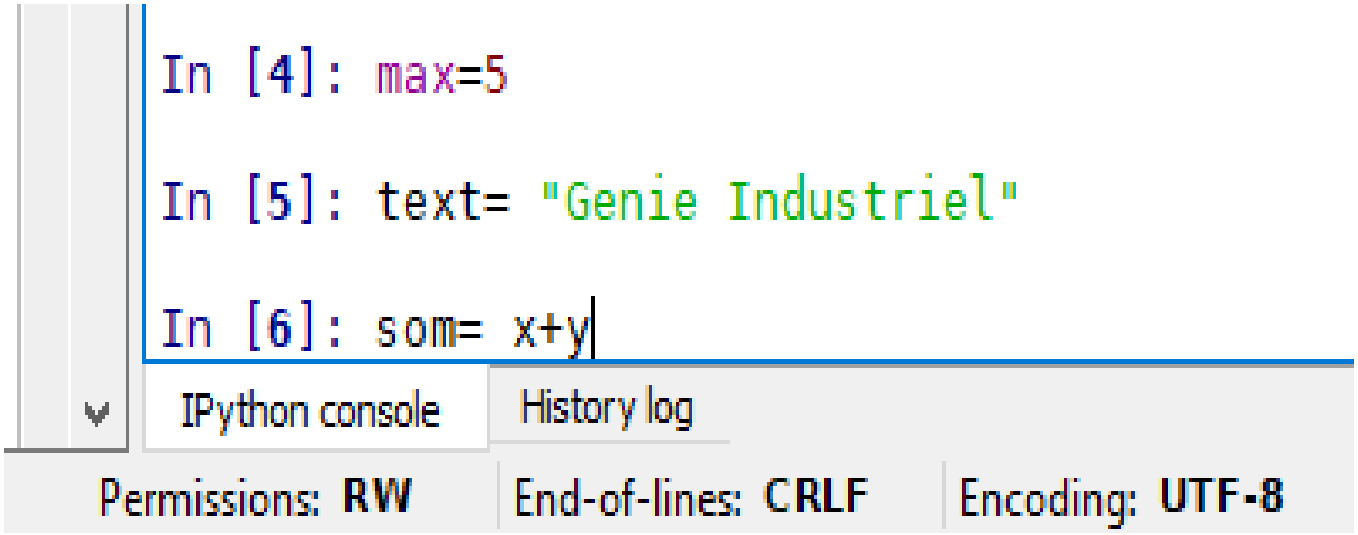
# 5. Opérations de base

## 1. Affectation

Syntaxe :

Nom de variable = expression

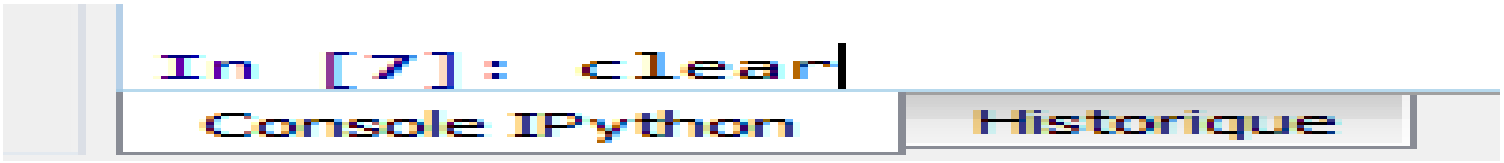
**Exemples:**

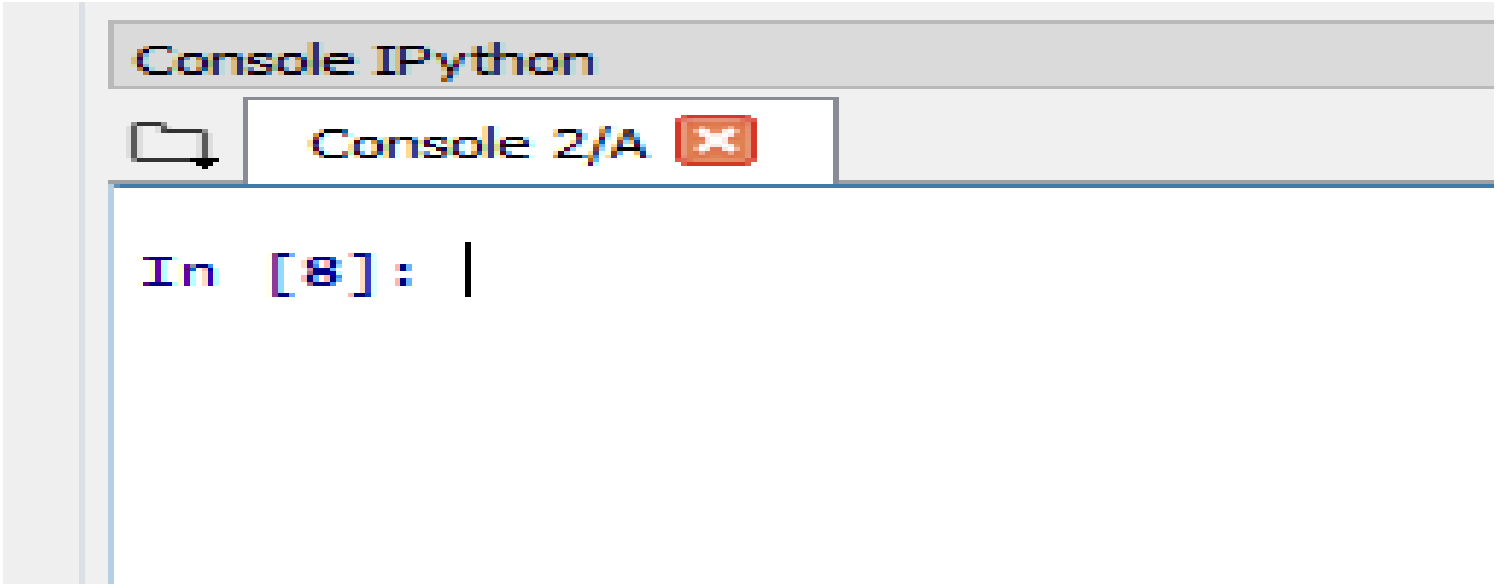



**clear**

# 5. Effacer l'écran

**clear**



# 5. Opérations de base

## 2. Entrée des données

**La fonction input()**

L'utilisateur est invité à entrer des caractères au clavier et à terminer avec <Enter>.

Le résultat de cette fonction est de type ***"chaine de caractères"***.

**Syntaxe:**

**nom_variable = input()**

**nom_variable = input("Msg … : ")**

# 5. Opérations de base

## 2. Entrée des données: chaine de caractères

**Exemple1**

```
In [48]: nom= input("Donnez votre Nom : ")

Donnez votre Nom : GI

In [49]:
```

**Afficher le résultat**

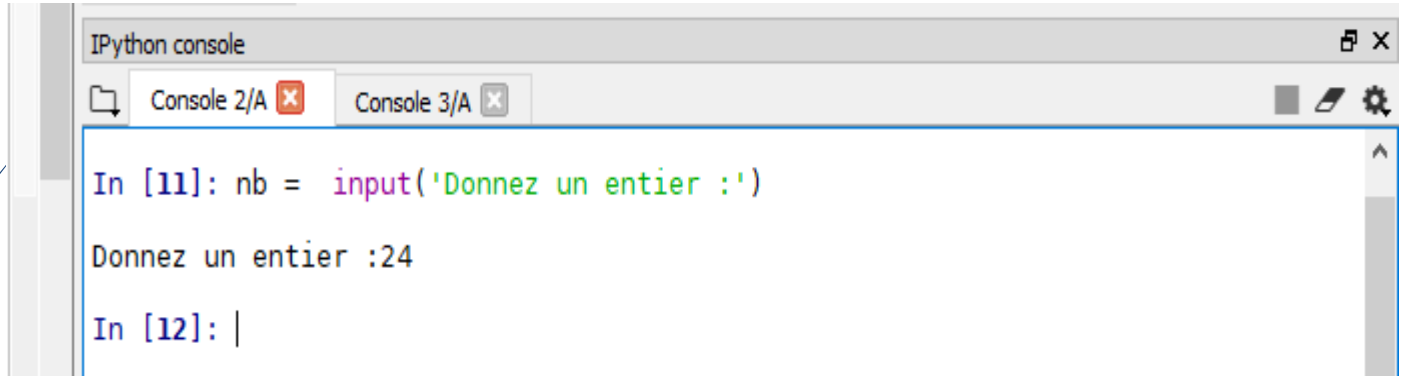
```
In [49]: nom
Out[49]: 'GI'

In [50]:
```

**Taper le nom de la variable**

# 5. Opérations de base

## 2. Entrée des données : entier/réel

**Exemple2**




**Afficher le résultat**

# 5. Opérations de base

## 2. Entrée des données : entier/réel

**Exemple3**

- Lire un 2eme entie.

  - Calculer la somme(nb, nb2)

  - Calculer la multiplication (nb,nb2)

# 5. Opérations de base

**Afficher le résultat**

```
In [13]: nb2 =  input('Donnez un 2eme entier :')

Donnez un 2eme entier :4

In [14]: som=nb+nb2

In [15]: som
Out[15]: '244'
```

```
In [16]: m=nb*nb2
Traceback (most recent call last):

  File "<ipython-input-16-752dc4a30b59>", line 1, in <module>
    m=nb*nb2

TypeError: can't multiply sequence by non-int of type 'str'
```

**Nécessite une conversion de type**

**Chaine → entier/réel**

# 5. Opérations de base

**Les fonctions : int & float**

```
In [18]: som= int(nb)+int(nb2)

In [19]: m= int(nb)*int(nb2)

In [20]: som
Out[20]: 28

In [21]: m
Out[21]: 96
```

```
In [22]: m1= float(nb)*float(nb2)

In [23]: m1
Out[23]: 96.0
```

# b. Mode script

Pour garder le code écrit et l'exécuter à chaque fois, il est nécessaire d'utiliser le concept de script.

1. Ecrire le code en utilisant un éditeur de texte
2. Exécuter le code.
3. Obtenir les résultats

**L'extension d'un fichier de code Python est « .py »**

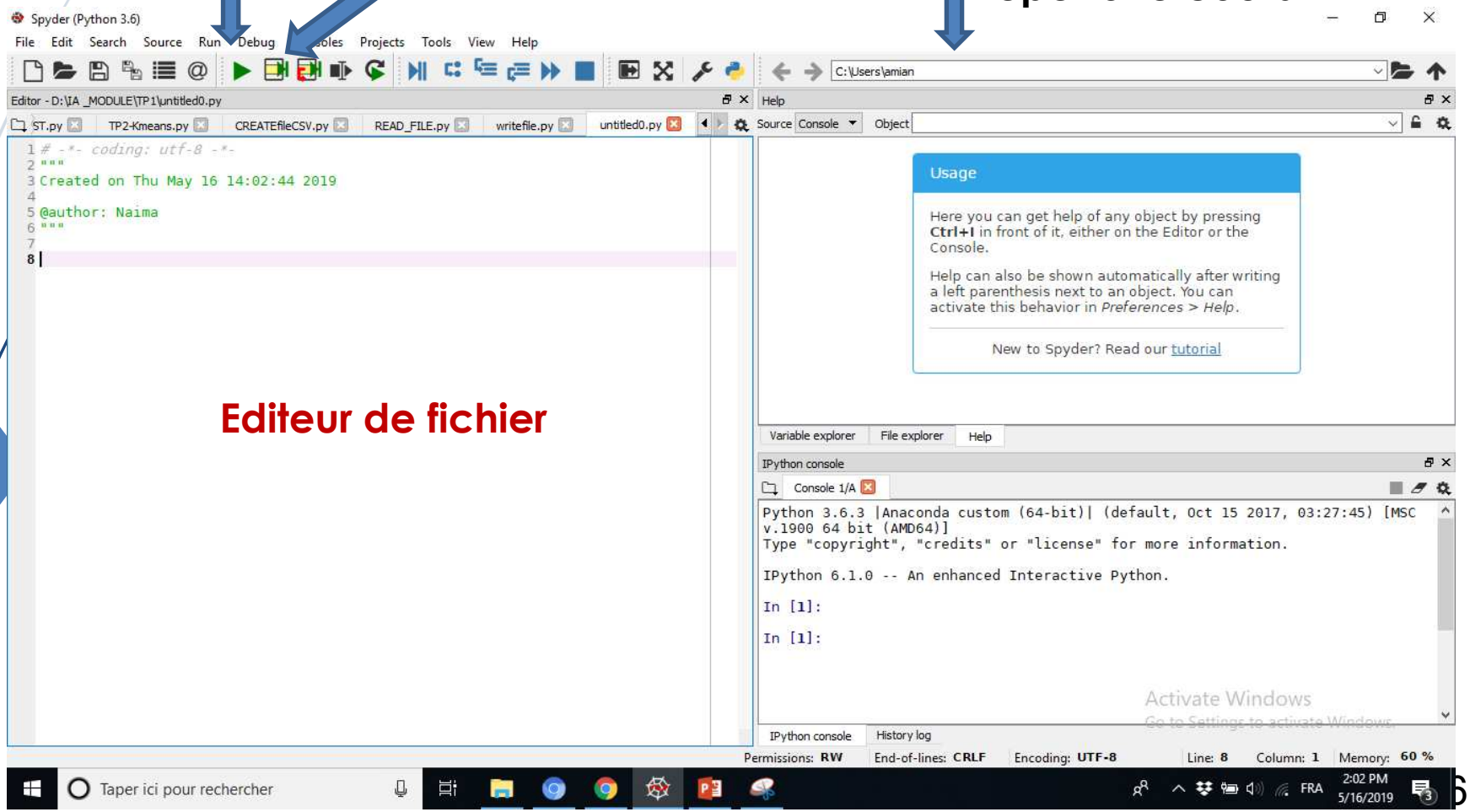
# b. Mode script

## SPYDER : Environement de développement scientifique

**Exécution totale**

**Exécution partielle**

**Répertoire courant**




**Editeur de fichier**

# 5. Opérations de base

## 3. Sortie des résultats (Affichage de message)

Syntaxe

**print(“msg”)**

**print(“msg”)**

**print (“msg”, variable)**

```
 8 nb =  input('Donnez un entier :')
 9 print('Le nombre lu = ', int(nb))
10 print('Le double du nombre = ',int(nb)*2,'\n')
11
12 module = "***Module Intelligence Artificielle d''ingénierie***"
13 print(module)
14
```

# Exercices:

1. Assignez les valeurs respectives 30, 60, 90 à trois variables x, y, z.
2. Effectuez l'opération (x * y )/z.
3. Lisez votre nom & prénom.
4. Affichez votre nom & Prénom

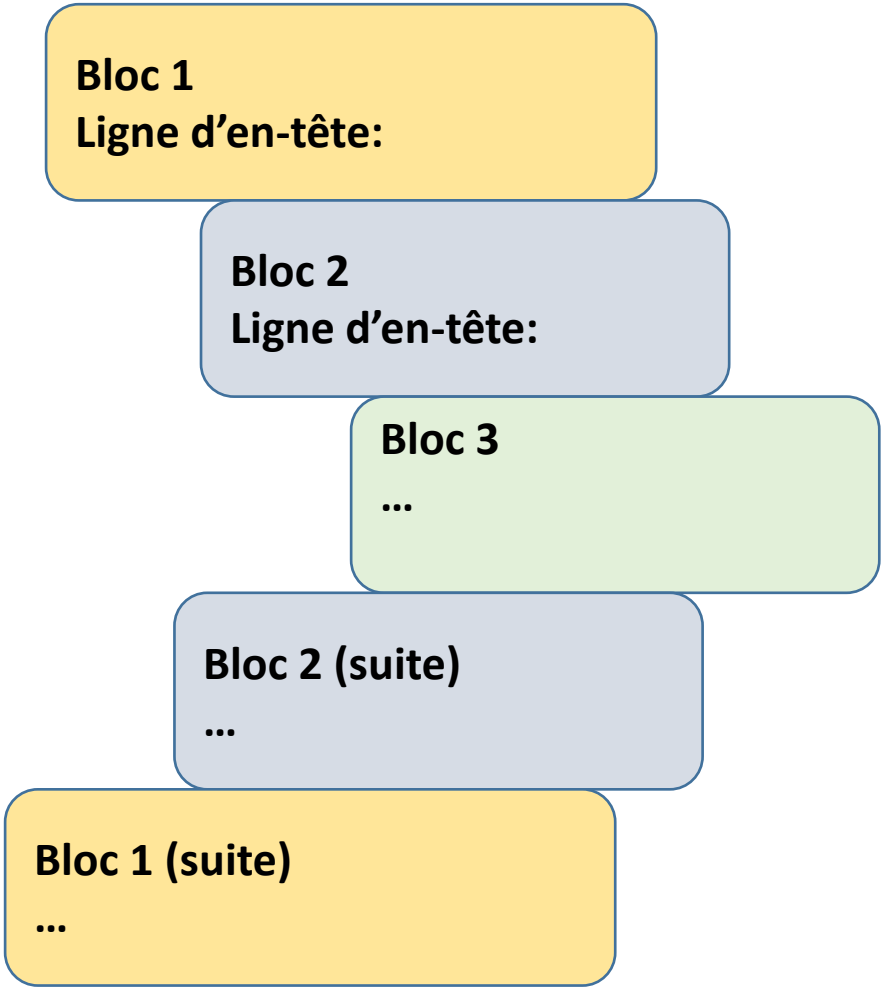
# 6. Les instructions de contrôle

a. Instructions conditionnelles

b. Instructions répétitives

# 6. Les instructions de contrôle

## Indentation générale



# 6. Les instructions de contrôle

## a. Les instructions conditionnelles

**Syntaxe**

```
if condition:
        Bloc d'instructions
```

Exemple :

```
if x > 0:
        print (" Nombre Positif " )
```

# 6. Les instructions de contrôle

## a. Les instructions conditionnelles

**Syntaxe**

```
if condition:
        Block d'instructions1
else:
        Block d'instructions2
```

Exemple :

```
if (a > 0):
    print(" Nombre positif")
else:
    print(" Nombre négatif ")
```

# 6. Les instructions de contrôle

## a. Les instructions conditionnelles

**Syntaxe**

```
if condition:
        Block d'instructions1
elif condition2:
        Block d'instructions2
else:
        Block d'instructions3
```

# 6. Les instructions de contrôle

## a. **Les instructions conditionnelles**

Exemple : afficher le signe d'un entier (positif/négatif/null).

```
x=input(" Donnez la valeur de x: ")
if int(x) > 0:
        print ("  Le nombre est positif ")
elif int(x)<0:
        print (" Le nombre est négatif ")
else:
        print (" Nombre est nul ")
```

# 6. Les instructions de contrôle

## a. Les instructions conditionnelles

- ***Opérateurs de comparaison***

| Signification | Symbole Mathématique | Symbole Python |
|---|---|---|
| Inférieur | < | < |
| Supérieur | > | > |
| Inférieur ou égal | ≤ | <= |
| Supérieur ou égal | ≥ | >= |
| Egal | = | == |
| Différent | ≠ | != |
| Reste de la division | | % |

# Exercice

Ecrire un script qui permet de saisir un nombre puis déterminer s'il est pair ou impair.

# Corrigé

```
nbre=input(" Entrez un entier : ")

if int(nbre)%2 == 0:
        print ("  Le nombre est pair ")
else:
        print ("  Le nombre est impair ")
```

# Corrigé

```
b1= int(input(" Entrez la borne inférieure [b1: "))
b2= int(input(" Entrez la borne supérieure  b2]: "))

valeur= int(input(" Entrez la valeur à vérifier: "))

if b1<=valeur<=b2:
        print (valeur, " appartenant à l'intervalle donné ")
else:
        print (valeur, "  ,n'appartenant pas à l'intervalle
donné ")
```

# 6. Les instructions de contrôle

b. Les instructions répétitives

Le concept boucle permet de répéter une certaine opération autant de fois que nécessaire.

- **while**
- **for**

# 6. Les instructions de contrôle

## b. **Les instructions répétitives**

- L'instruction while

**Syntaxe:**

```
while condition:
        Bloc d'instructions
```

# 6. Les instructions de contrôle

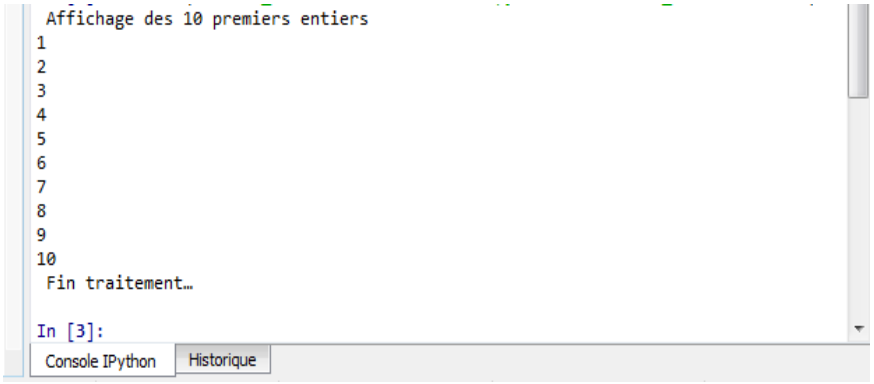
## b. **Les instructions répétitives**

**Exemple:**

Afficher les 10 premiers entiers.

```
print(" Affichage des 10 premiers entiers ")
i = 1 # variable compteur utilisée par la boucle
while i <= 10:
    print(i)
    i += 1      # Incrémenter i de 1 (i=i+1)
print( " Fin traitement…")
```

Résultat



# 6. Les instructions de contrôle

## b. **Les instructions répétitives**

**Exemple:**

Création de la table de multiplication d'un entier.

```
nb1= input("Donnez un entier : ")
nb2= int(nb1)
print(" Création de la table de multiplication de: " ,nb1)
i = 1 # variable compteur utilisée par la boucle
while i <= 10:
    print(i, "*", nb2, "=", (i * nb2))
    i += 1 #   Incrémenter i de 1 (i=i+1)
print( " Fin traitement…")
```

# 6. Les instructions de contrôle

## b. **Les instructions répétitives**

- **L'instruction for**

  Syntaxe:

  **for** elt in sequence**:**
  Bloc d'instructions

**« *elt* »** est une variable créé par **« *for* ».**
Elle n'est pas instancier par le programmeur. Elle prend successivement chacune des valeurs figurant dans la séquence parcourue.

# 6. Les instructions de contrôle

## b. **Les instructions répétitives**

- L'instruction for

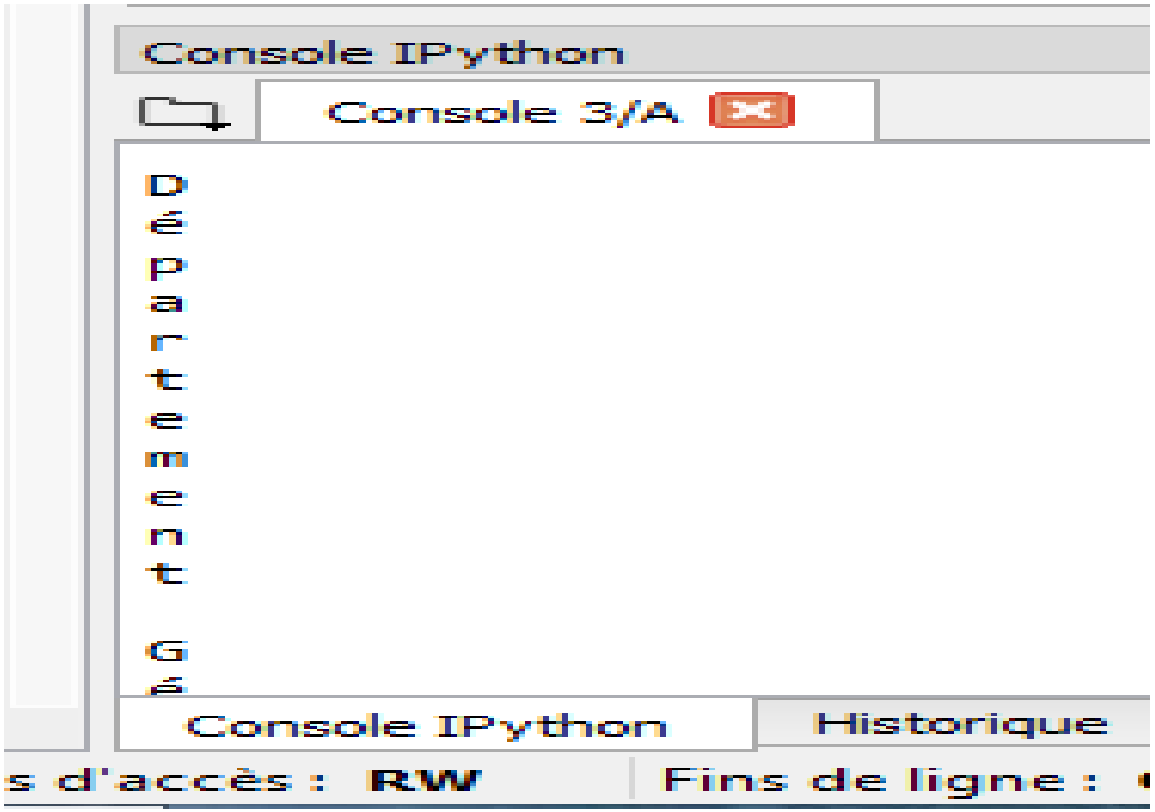
Exemple

```
dpt = "Département Génie Industriel"
for lettre in dpt:
    print(lettre)

print(" Fin traitement…")
```

# 6. Les instructions de contrôle

## b. **Les instructions répétitives**

Résultat



# 6. Les instructions de contrôle

## b. **Les instructions répétitives**

L'instruction **range()** : génère une liste de nombres.

Syntaxe :

range([debut], arrêt [, pas])

Exemple

```
# Un seul paramètre
for i in range(5):
        print(i)
# Deux paramètres
for i in range(3, 6):
        print(i)
 # Trois paramètres
for i in range(4, 10, 2):
        print(i)
```

**Résultat =** 0 1 2 3 4

**Résultat =** 3 4 5

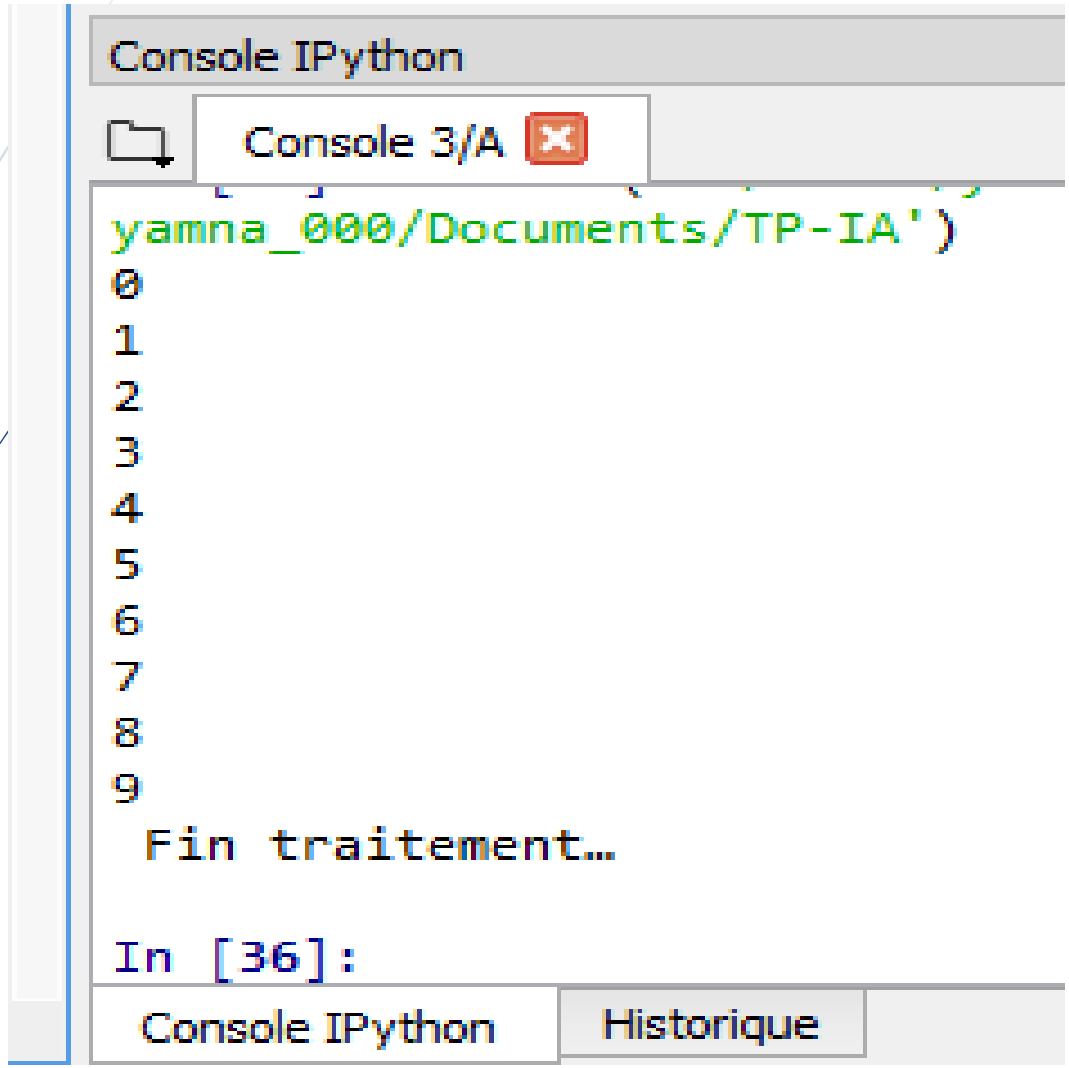
**Résultat =** 4 6 8

# Exercice

Afficher les 10 premiers chiffres.

```
liste = 10
for chiffre in range(liste):
    print(chiffre)

Print " Fin traitement…"
```

# Corrigé

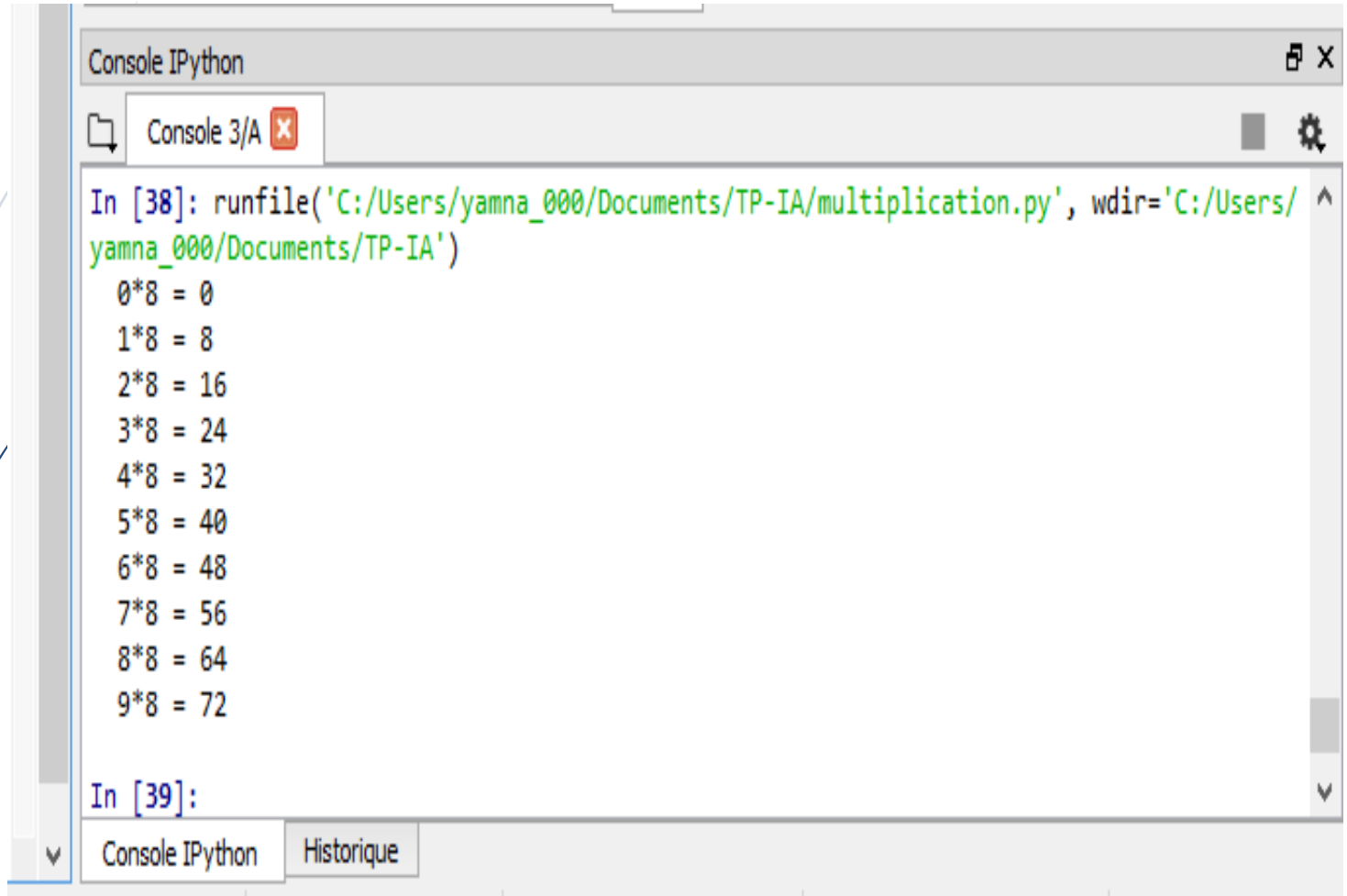

Console IPython
Console 3/A
yamna_000/Documents/TP-IA')
0
1
2
3
4
5
6
7
8
9
Fin traitement…
In [36]:
Console IPython
Historique

# Exercice

Écrire un programme qui affiche les 10 premiers multiples de 8

```
for i in range(10):
        print("  "+str(i)+"*8 = "+str(i*8))
```

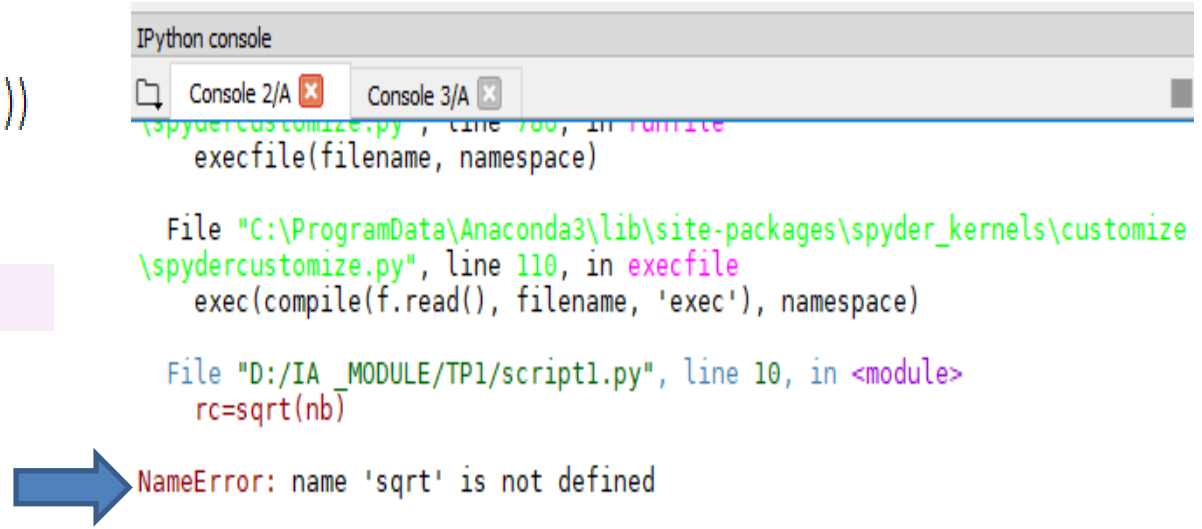
# Corrigé


Console IPython
Console 3/A
In [38]: runfile('C:/Users/yamna_000/Documents/TP-IA/multiplication.py', wdir='C:/Users/yamna_000/Documents/TP-IA')
0*8 = 0
1*8 = 8
2*8 = 16
3*8 = 24
4*8 = 32
5*8 = 40
6*8 = 48
7*8 = 56
8*8 = 64
9*8 = 72
In [39]:
Console IPython
Historique

# Exercice

Calculer la racine carré d'un entier avec la fonction **sqrt()**.

```
10 nb= int(input("Donnez un entier  : "))
11 rc= sqrt(nb)
12 print("Racine Carrée =",rc)
13
```




## Importer les librairies

# 7. Importation des librairies

Python offre de très nombreuses librairies de fonctions pré-définies pour réaliser des tâches.

L'importation de la librairie se fait à travers l'instruction

**import** nom_librairie

Pour importer une fonction précise de la librairie

**from** nom_librairie **import** nom_fonction

# Corrigé

```
 7
 8 import math
 9 nb =  int(input('Donnez un entier :'))
10 print('Le nombre = ', nb)
11 rc=math.sqrt(nb)
12 print('La racine carré = ', rc)
```

**Résultat**

```
In [10]: runfile('D:/IA _MODULE/M2-MI/TP2/RACINE.py')

Donnez un entier  : 10000
Racine Carrée = 100.0

In [11]:
```

# 7. Importation des librairies

Quelques fonctions de la librairie math

| Python | mathématique |
|---|---|
| math.pi | π |
| math.factorial(n) | n! |
| math.fabs(x) | \|x\| |
| math.exp(x) | $e^x$ |
| math.log(x) | Ln(x) |
| math.log10(x) | Log10(x) |
| math.sqrt(x) | √x |
| math.cos(x) | cos(x) |
| math.sin(x) | sin(x) |

# Exercice

1. Calculer le factoriel de x
2. Calculer l'exponentiel de x

# Corrigé

```
import math
nb=  int(input("Donnez un entier  : "))
fact= math.factorial(nb)
print("Factoriel =",fact)
```

**Résultat**

```
In [5]: runfile('D:/IA _MODULE/M2-MI/TP2/FACT.py')

Donnez un entier  : 9
Factoriel = 362880
```

# 7. Importantes des librairies

**NumPy**
permet d'effectuer des calculs numériques avec Python. Elle introduit une gestion facilitée des tableaux de nombres.

```
import numpy as np
```


**Matplotlib**
destinée à tracer et visualiser des données sous formes de graphiques.

```
from matplotlib import pyplot
```

# 7. Importation des librairies

**Scikit-learn**
destinée à l'apprentissage automatique.
Elle comporte divers algorithmes tels que machine à vecteurs de support, k-voisins, RNA, Kmeans,…

**from sklearn import…**

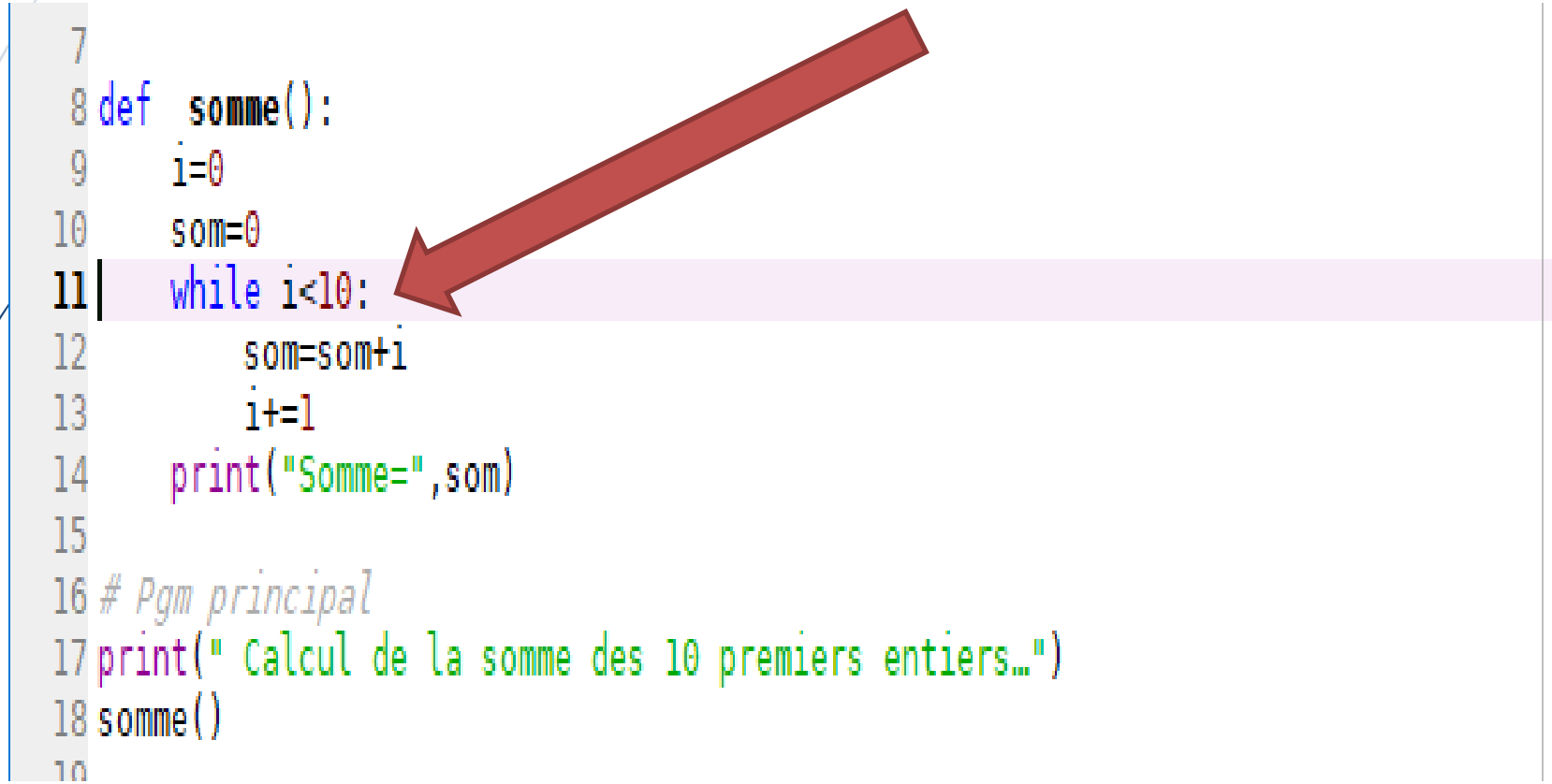
# 8. Définition des fonctions


Une fonction transforme généralement une entrée en sortie

```
def  nom_fonction(liste de paramètres):
        …
        bloc d'instructions de la fonction
        …
```

# 8. Définition des fonctions

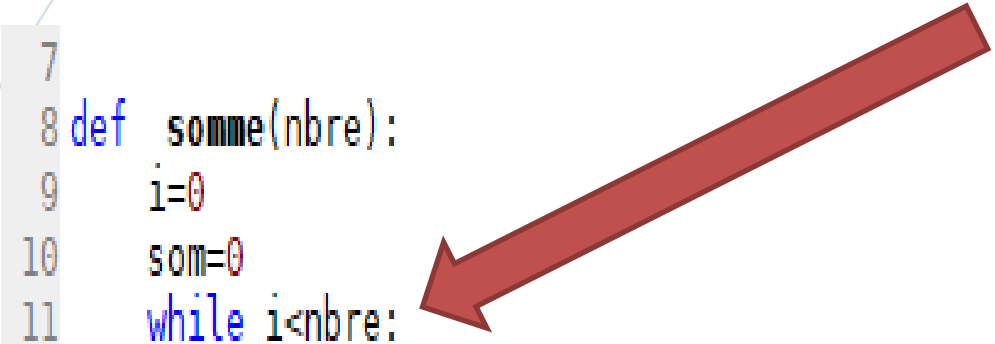
- **Fonction sans paramètres**



```
7
8  def  somme():
9      i=0
10     som=0
11     while i<10:
12         som=som+i
13         i+=1
14     print("Somme=",som)
15
16 # Pgm principal
17 print(" Calcul de la somme des 10 premiers entiers…")
18 somme()
```

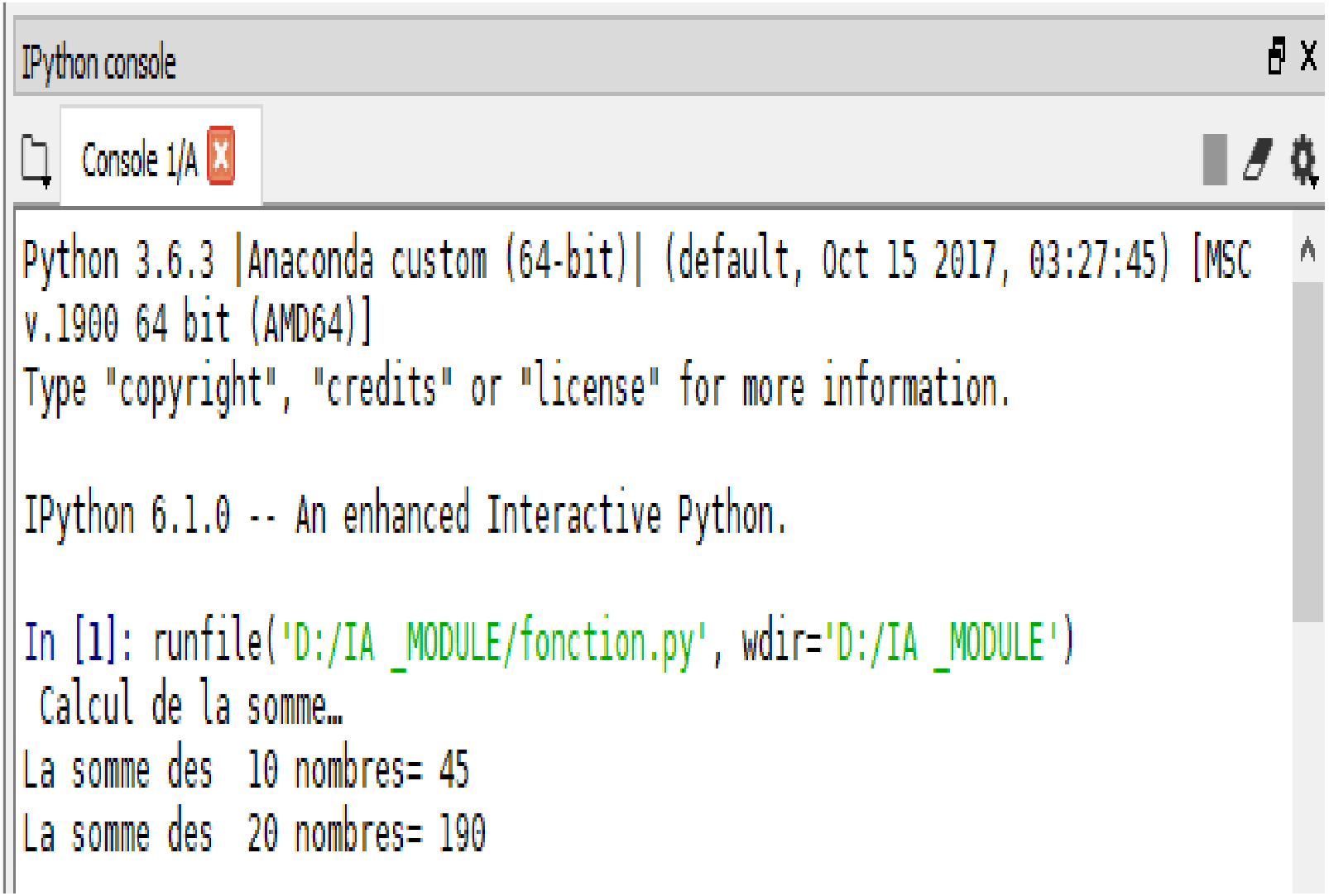
# 8. Définition des fonctions

- **Fonction avec paramètres**



```
7
8  def  somme(nbre):
9      i=0
10     som=0
11     while i<nbre:
12         som=som+i
13         i+=1
14     print("La somme des ",nbre, "nombres=",som)
15
16 # Pgm principal
17 print(" Calcul de la somme…")
18 somme(10)
19 somme(20)
```

# Résultat


IPython console
Console 1/A
Python 3.6.3 |Anaconda custom (64-bit)| (default, Oct 15 2017, 03:27:45) [MSC
v.1900 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.
IPython 6.1.0 -- An enhanced Interactive Python.
In [1]: runfile('D:/IA _MODULE/fonction.py', wdir='D:/IA _MODULE')
Calcul de la somme…
La somme des 10 nombres= 45
La somme des 20 nombres= 190

# 8. Définition des fonctions

**Fonction avec retour de résultat**

Une « vraie » fonction (au sens strict) doit renvoyer un résultat à la fin de ses traitements.

La dernière ligne de la définition de la fonction doit être l'instruction **return** (return : définit ce que doit être la valeur renvoyée par la fonction).

Syntaxe :

**return resultat**

# « *resultat* » peut être une variable ou une expression

# 8. Définition des fonctions

**Exemple**

```
def  nom_fonction(liste de paramètres):
        …
        bloc d'instructions
        …
return
```

**Syntaxe de l'appel de la fonction:**

**Variable_result = nom_fonction(liste_parametres)**

# 8. Définition des fonctions

La fonction s'arrête à la ligne « return » qui indique quelle sortie la fonction doit produire.

Une fois, la fonction créée, il est possible une infinité de fois.

# 8. Définition des fonctions

**Exemple**

```
7
8 #Définition de la fonction
9 def somme(nb):
10     s=0
11     for i in range(0,(nb+1)):
12         s=s+i
13     return s
14
15 #programme principal
16 valeur = int(input("Calcul de la somme : Tapez une valeur:  "))
17 som= somme(valeur)
18 print('La somme =', som)
```

*Dans un programme simple:*

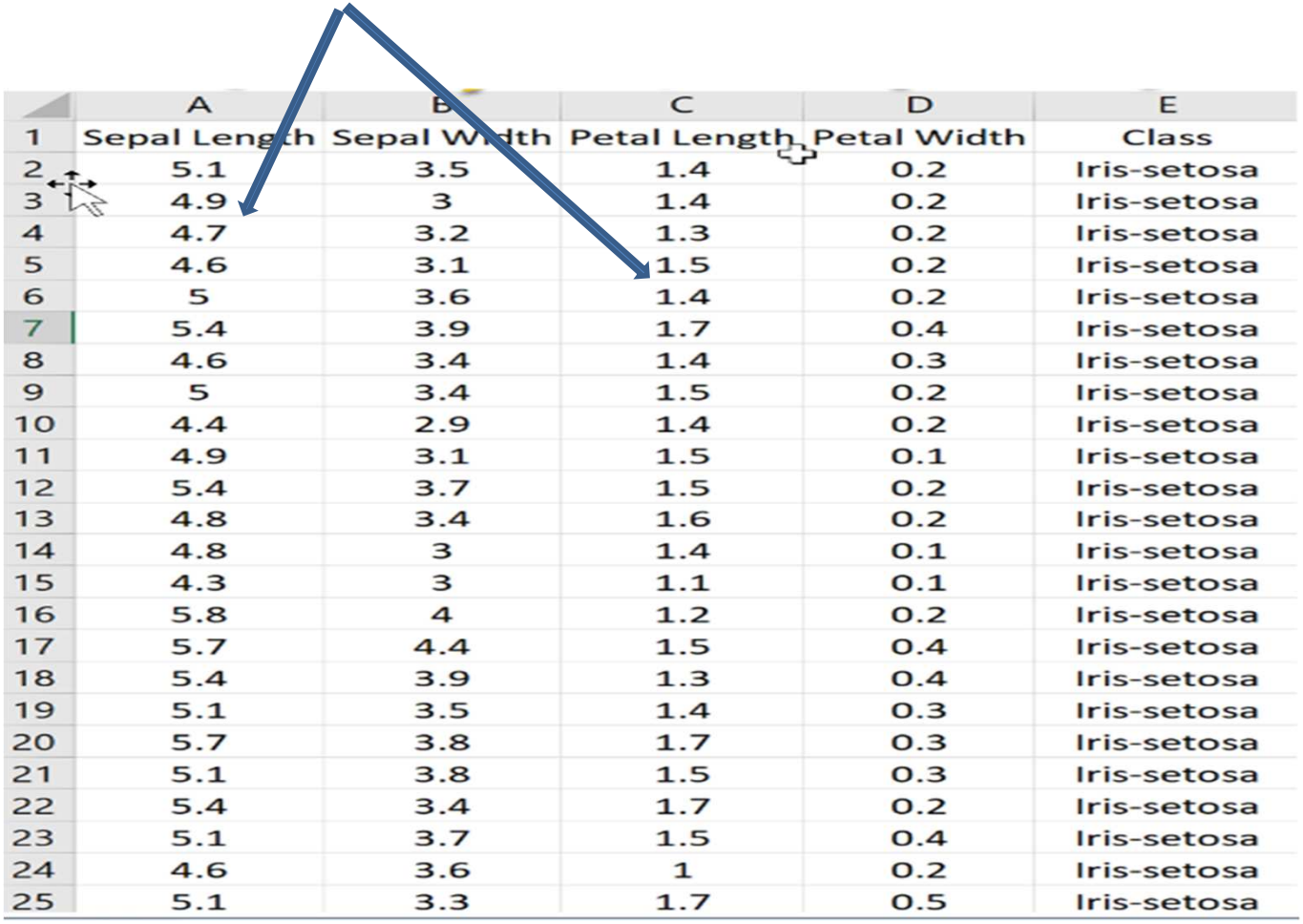
- *Inclure ces données dans le corps du programme lui-même (par exemple dans une liste).*
- *Ce procéder devient tout à fait inadéquat lorsque l'on souhaite traiter une quantité d'informations plus importante.*


***Manipulation des dataset***

# 9. Manipulation du dataset (1)

## Dataset (Jeu de données)

Ensemble de données (ou valeurs).



| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | Sepal Length | Sepal Width | Petal Length | Petal Width | Class |
| 2 | 5.1 | 3.5 | 1.4 | 0.2 | Iris-setosa |
| 3 | 4.9 | 3 | 1.4 | 0.2 | Iris-setosa |
| 4 | 4.7 | 3.2 | 1.3 | 0.2 | Iris-setosa |
| 5 | 4.6 | 3.1 | 1.5 | 0.2 | Iris-setosa |
| 6 | 5 | 3.6 | 1.4 | 0.2 | Iris-setosa |
| 7 | 5.4 | 3.9 | 1.7 | 0.4 | Iris-setosa |
| 8 | 4.6 | 3.4 | 1.4 | 0.3 | Iris-setosa |
| 9 | 5 | 3.4 | 1.5 | 0.2 | Iris-setosa |
| 10 | 4.4 | 2.9 | 1.4 | 0.2 | Iris-setosa |
| 11 | 4.9 | 3.1 | 1.5 | 0.1 | Iris-setosa |
| 12 | 5.4 | 3.7 | 1.5 | 0.2 | Iris-setosa |
| 13 | 4.8 | 3.4 | 1.6 | 0.2 | Iris-setosa |
| 14 | 4.8 | 3 | 1.4 | 0.1 | Iris-setosa |
| 15 | 4.3 | 3 | 1.1 | 0.1 | Iris-setosa |
| 16 | 5.8 | 4 | 1.2 | 0.2 | Iris-setosa |
| 17 | 5.7 | 4.4 | 1.5 | 0.4 | Iris-setosa |
| 18 | 5.4 | 3.9 | 1.3 | 0.4 | Iris-setosa |
| 19 | 5.1 | 3.5 | 1.4 | 0.3 | Iris-setosa |
| 20 | 5.7 | 3.8 | 1.7 | 0.3 | Iris-setosa |
| 21 | 5.1 | 3.8 | 1.5 | 0.3 | Iris-setosa |
| 22 | 5.4 | 3.4 | 1.7 | 0.2 | Iris-setosa |
| 23 | 5.1 | 3.7 | 1.5 | 0.4 | Iris-setosa |
| 24 | 4.6 | 3.6 | 1 | 0.2 | Iris-setosa |
| 25 | 5.1 | 3.3 | 1.7 | 0.5 | Iris-setosa |

# 9. Manipulation du dataset (2)

## Dataset

- chaque valeur est associée **à :**
  - une variable
  - une observation.

➢ **Une variable** décrit l'ensemble des valeurs décrivant la même variable.

➢ **Une observation** contient l'ensemble des valeurs décrivant les variables de l'une unité traitée.

# 9. Manipulation du dataset (3)

- Un dataset peut avoir une structure tabulaire, tel que:

  le fichier au format CSV.

- Un fichier CSV est un fichier texte. Chaque ligne du texte correspond à une ligne du tableau et les virgules correspondent aux séparations entre les colonnes.

| Représentation tabulaire | Fichier au format .csv |
|---|---|

| | A | B | C | D |
|---|---|---|---|---|
| 1 | Sepal Length | Sepal Width | Petal Length | Petal Width |
| 2 | 5.1 | 3.5 | 1.4 | 0.2 |
| 3 | 4.9 | 3 | 1.4 | 0.2 |
| 4 | 4.7 | 3.2 | 1.3 | 0.2 |
| 5 | 4.6 | 3.1 | 1.5 | 0.2 |
| 6 | 5 | 3.6 | 1.4 | 0.2 |
| 7 | 5.4 | 3.9 | 1.7 | 0.4 |
| 8 | 4.6 | 3.4 | 1.4 | 0.3 |
| 9 | 5 | 3.4 | 1.5 | 0.2 |
| 10 | 4.4 | 2.9 | 1.4 | 0.2 |
| 11 | 4.9 | 3.1 | 1.5 | 0.1 |
| 12 | 5.4 | 3.7 | 1.5 | 0.2 |
| 13 | 4.8 | 3.4 | 1.6 | 0.2 |
| 14 | 4.8 | 3 | 1.4 | 0.1 |
| 15 | 4.3 | 3 | 1.1 | 0.1 |
| 16 | 5.8 | 4 | 1.2 | 0.2 |
| 17 | 5.7 | 4.4 | 1.5 | 0.4 |
| 18 | 5.4 | 3.9 | 1.3 | 0.4 |
| 19 | 5.1 | 3.5 | 1.4 | 0.3 |
| 20 | 5.7 | 3.8 | 1.7 | 0.3 |
| 21 | 5.1 | 3.8 | 1.5 | 0.3 |
| 22 | 5.4 | 3.4 | 1.7 | 0.2 |
| 23 | 5.1 | 3.7 | 1.5 | 0.4 |
| 24 | 4.6 | 3.6 | 1 | 0.2 |
| 25 | 5.1 | 3.3 | 1.7 | 0.5 |

```
Sepal length, Sepal Width, Petal length, Petal Width
5.1,3.5,1.4,0.2
4.9,3,1.4,0.2
4.7,3.2,1,3,0,2
.
.
.
.
```

# 9. Manipulation du dataset (3)

- Comment gérer le dataset?

.

**Librairie Pandas**

.

# 9. Manipulation du dataset (4)

## Librairie Pandas

Pandas est dérivé du terme (PANel DAta), un terme économétrique désignant des ensembles de données comprenant des observations sur plusieurs périodes pour les mêmes individus.

Pandas est un package Python fournissant des structures de données rapides, flexibles conçues pour rendre le travail avec des données relationnelles.

Ce package est un outil d'analyse et de manipulation de données pratique dans le monde réel en Python.

# 9. Manipulation du dataset (5)

- Domaines d'utilisation

Finance, Statistique, Sciences sociales et dans de nombreux domaines de l'Ingénierie.

- Structures de données

Les deux principales structures de données des pandas sont :

| Nom de la structure | dimension | Description |
|---|---|---|
| pandas.Series | 1 | Vecteur de données homogènes labellisées |
| pandas.DataFrame | 2 | Tableau structuré de colonnes homogènes |

# 9. Manipulation du dataset (6)

**Exemple:**

| | Modules |
|---|---|
| 0 | Math |
| 1 | Informatique |
| 2 | Electronique |

**Series**

| | Modules | Coefficients |
|---|---|---|
| 0 | Math | 5 |
| 1 | Informatique | 4 |
| 2 | Electronique | 3 |

**DataFrame**

# 9. Manipulation du dataset (7)

- Utilisation du package

```
import pandas as pd
```

- Importer les données en utilisant CSV

Pour importer les données d'un fichier .csv:

la fonction : read_csv() de la librairie pandas.

```
pd.read_csv ("filename.csv")
```

# 9. Manipulation du dataset (8)

**Exemples :**

pd.read_csv("/GI/IA/mydata/ disciplines.csv")

data = pd.read_csv('disciplines.csv',delimiter=';' )

| | Modules | Coefficients | Total |
|---|---|---|---|
| 0 | Math | 5 | 12 |
| 1 | Informatique | 4 | 10 |
| 2 | Electronique | 3 | 11 |

Contenu du fichier 'disciplines.csv'

# 9. Manipulation du dataset (9)

- Division du DataFrame

Supprimer des lignes ou des colonnes en spécifiant les noms d'index ou de colonnes.

  - Utilisation de .drop
  - Utilisation de .loc et .iloc

# 9. Manipulation du dataset (10)

❖Sélection des données par **drop**

DataFrame.drop( labels=None, axis=0)

labels : Index ou colonne à supprimer.
axis : Valeur par defaut =0
0 : ‘index’ : « ligne »
1 : ‘columns’

# 9. Manipulation du dataset (11)

## Exemple1 :

```
X = data.drop('Total', axis=1)
print(X)
```

Avant .drop

| | Modules | Coefficients | Total |
|---|---|---|---|
| 0 | Math | 5 | 12 |
| 1 | Informatique | 4 | 10 |
| 2 | Electronique | 3 | 11 |

Après .drop

| | Modules | Coefficients |
|---|---|---|
| 0 | Math | 5 |
| 1 | Informatique | 4 |
| 2 | Electronique | 3 |

# 9. Manipulation du dataset (12)

Exemple2 :

X = data.drop('Modules',axis=1)

| | Coefficients | Total |
|---|---|---|
| 0 | 5 | 12 |
| 1 | 4 | 10 |
| 2 | 3 | 11 |

Exemple3 :

X = data.drop(2,axis=0)

| | Modules | Coefficients | Total |
|---|---|---|---|
| 0 | Math | 5 | 12 |
| 1 | Informatique | 4 | 10 |

# 9. Manipulation du dataset (13)

**Script en Python**

```
import pandas as pd
data = pd.read_csv('disciplines.csv',delimiter=';' )
print(data)
# data is divided into attributes and labels
X = data.drop('Total', axis=1)
print(X)
y = data['Total']
print(y)
```

# 9. Manipulation du dataset (14)

## ❖ Sélection les données sur les DataFrames

.loc et .iloc, permettent d'effectuer des opérations de sélection de données sur les DataFrames.

a) .loc est basé sur une étiquette

Permet de spécifier des lignes et des colonnes en fonction de leurs étiquettes de lignes et de colonnes.

b) .iloc est basé sur un index entier.

Permet de spécifier les lignes et les colonnes à l'aide de leur index.

# 9. Manipulation du dataset (15)

- Sélection les données sur les DataFrames

**Lignes**

```
data.iloc[0] # première ligne du  data frame
data.iloc[1] # deuxième ligne du  data frame
data.iloc[-1] # dernière ligne du  data frame
```

**Colonnes**

```
data.iloc[:,0] #  Première colonne du data frame
data.iloc[:,1] #  Deuxième colonne du data frame
data.iloc[:,-1] #  Dernière colonne du data frame
```

# 9. Manipulation du dataset (16)

- Sélection les données sur les DataFrames

```
# Multiple lignes & colonnes
data.iloc[0:5] # les 5 premières lignes du  dataframe
data.iloc[:, 0:2] # les deux premières colonnes avec
                  # toutes les lignes
data.iloc[[0,3,6,24], [0,5,6]] # 1ere, 4eme, 7eme, 25eme ligne
                               # + 1ere 6eme 7eme colonnes,
```

# 9. Manipulation du dataset (16)

## Exemple

Afficher la première et dernière ligne avec toutes les colonnes

```
X= data.iloc[[0,2],:]
```

Afficher la première et dernière ligne avec la 1ere et 3eme colonnes

```
X= data.iloc[[0,2],0:2]
```

# 9. Manipulation du dataset (18)

## Exemple

## Sélection des données par .iloc

```
import pandas as pd
data = pd.read_csv('disciplines.csv',delimiter=';')

X= data.iloc[:,0:2]
y= data.iloc[:, -1]
print("Les données : \n")
print(X)
print("Dernière Colonne : \n")
print(y)
```

```
Les données :

        Modules  Coefficients
0          Math             5
1  Informatique             4
2  Electronique             3


Dernière Colonne :

0    12
1    10
2    11
```

# 9. Manipulation du dataset (19)

## Exemple

## Sélection des données par .loc

```
import pandas as pd
data = pd.read_csv('disciplines.csv',delimiter=';')

X1= data.loc[:,['Modules','Coefficients']]
y1= data.loc[:, 'Total']

print("Les données : \n")
print(X1)
print("Dernière Colonne : \n")
print(y1)
```

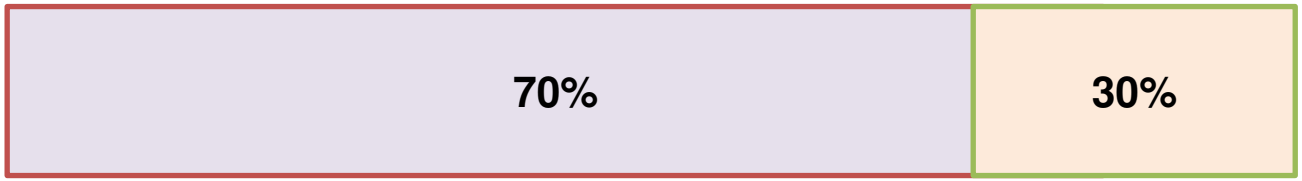
```
Les données :

        Modules  Coefficients
0          Math             5
1  Informatique             4
2  Electronique             3
Dernière Colonne:

0    12
1    10
2    11
```

# 10. Division du dataset (1)

## Division du dataset en un ensemble d'apprentissage et un ensemble de test

- Ensemble d'apprentissage : sous-ensemble destiné à l'apprentissage d'un modèle.

- Ensemble de test : sous-ensemble destiné à l'évaluation du modèle.



# 10. Division du dataset (2)

**Scikit-learn** est une bibliothèque libre Python destinée à l'apprentissage automatique.

- Types d'apprentissage gérés par sklearn:
supervisé, non supervisé, par renforcement, par transfert

- Les algorithmes

  • Les algorithmes : régression linéaire (Linear Regression ),
  • arbre de décision (decision Tree),
  • SVM (machines à vecteur de support),
  • classification naïve bayésienne (Naive Bayes),
  • KNN (Plus proches voisins),
  • Réseaux de neuronnes,
  • ...

# 10. Division du dataset (2)

**train_test_split**

Diviser le dataset aléatoirement en deux sous ensembles : train et test.

## Exemple

from **sklearn.model_selection** import **train_test_split**

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33)

Où
X : ensemble des données
y : les classes

# 10. Division du dataset (2)

Incanter Dataset

| Sepal.Length | Sepal.Width | Petal.Length | Petal.Width | Species |
|---|---|---|---|---|
| 5.1 | 3.5 | 1.4 | 0.2 | setosa |
| 4.9 | 3.0 | 1.4 | 0.2 | setosa |
| 4.7 | 3.2 | 1.3 | 0.2 | setosa |
| 4.6 | 3.1 | 1.5 | 0.2 | setosa |
| 5.0 | 3.6 | 1.4 | 0.2 | setosa |
| 5.4 | 3.9 | 1.7 | 0.4 | setosa |
| 4.6 | 3.4 | 1.4 | 0.3 | setosa |
| 5.0 | 3.4 | 1.5 | 0.2 | setosa |
| 4.4 | 2.9 | 1.4 | 0.2 | setosa |
| 4.9 | 3.1 | 1.5 | 0.1 | setosa |
| 5.4 | 3.7 | 1.5 | 0.2 | setosa |
| 4.8 | 3.4 | 1.6 | 0.2 | setosa |
| 4.8 | 3.0 | 1.4 | 0.1 | setosa |
| 4.3 | 3.0 | 1.1 | 0.1 | setosa |
| 5.8 | 4.0 | 1.2 | 0.2 | setosa |
| 5.7 | 4.4 | 1.5 | 0.4 | setosa |
| 5.4 | 3.9 | 1.3 | 0.4 | setosa |
| 5.1 | 3.5 | 1.4 | 0.3 | setosa |
| 5.7 | 3.8 | 1.7 | 0.3 | setosa |
| 5.1 | 3.8 | 1.5 | 0.3 | setosa |
| 5.4 | 3.4 | 1.7 | 0.2 | setosa |
| 5.1 | 3.7 | 1.5 | 0.4 | setosa |

train_test_split

xtrain | ytrain | xtest | ytest

# 10. Division du dataset (2)

## Exemple

```
import pandas as pd
data = pd.read_csv('disciplines.csv',delimiter=';')
print(data)

X= data.iloc[:,0:2]
y= classes = data.iloc[:, -1]
print("X =\n", X, "\n")
print("y =\n", y, "\n")

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33)
print("X_train =", X_train)
print("y_train =",  y_train)

print("\n")
print("X_test =", X_test)
print("y_test =",  y_test)
```

# 10. Fondamentaux

**Apprentissage (training)**

Modéliser la relation qui existe entre les entrées (instances) et les sorties (classes) de sorte que la probabilité de mauvaise prédiction soit minimale.

Création d'un modèle.

Processus consistant à déterminer les paramètres idéaux d'un modèle.

# 10. Fondamentaux

## Classification

La classification est la tâche consistant à :

- attribuer une classe à une donnée qu'on veut classer.

**Classe (class)**

Un des ensembles de valeurs cibles énumérées pour une étiquette.

Espace des classes est **discret et fini**

# 10. Fondamentaux

## Types de classification

1. Classification binaire
2. Classification multinomiale

### Classification binaire (binary classification)

Type de tâche de classification qui prédit l'une des deux classes mutuellement exclusives.

Exemple, un modèle de ML qui classe les e-mails en tant que "spam" ou "non-spam" est un classifieur binaire.

# 10. Fondamentaux

**Classification à classes multiples : multinomiale (multi-class classification)**

Problèmes de classification qui distingue plus de deux classes.

**Exemple**

26 classes pour classer les caractères de l'alphabet.

Un modèle ML qui les classe serait donc à classes multiples.

**Prédiction (prediction)**

Résultat d'un modèle auquel un exemple est fourni en entrée.

# Etapes de l'apprentissage automatique

## 1. Chargement les données

Importer des bibliothèques

**Import** nom de la librairie

## 2. Chargement le jeu de données

Data =**read_csv**()

## 3. Séparation des classes des données (X ,y)

X = les données

y = les classes

## 4. Visualisation des données

print(data) pour voir la structure des données (optionnelle)

## 5.Partitionnement du jeu de données

La fonction ***train_test_split*** permet de diviser le jeu de données en 2 ensembles : les données d'apprentissage et les données de test.

## 6. Création du classifieur

Il est possible d'utiliser plusieurs algorithmes d'apprentissage automatique.

clf = spécifier nom de l'algorithme utilisé.

## 7. Apprentissage sur les données d'apprentissage

clf.**fit**(données d'apprentissage, leurs classe)

## 8. Prédiction

result = clf.**predic**t(nouvelles données)

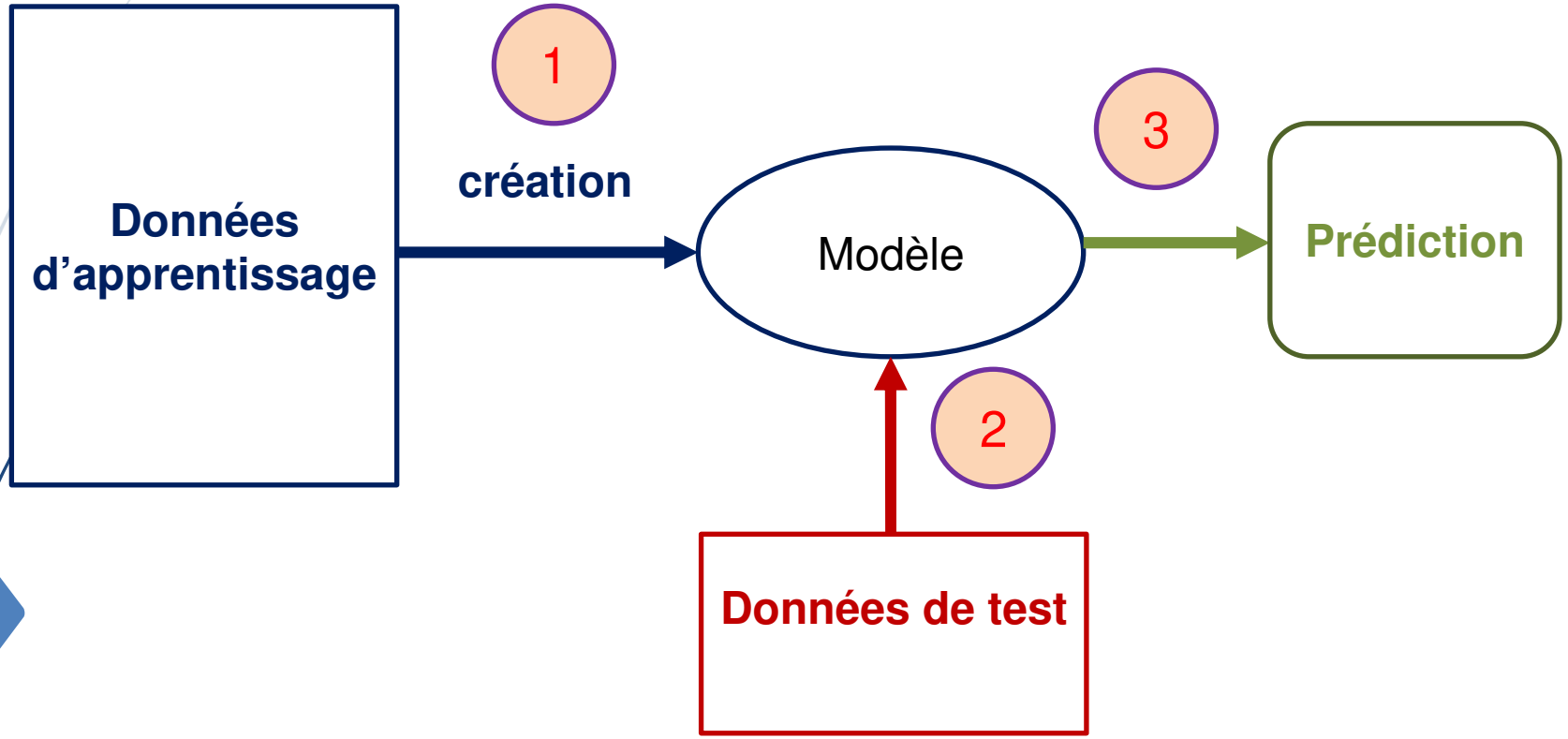
## 9.Calcul du pourcentage d'erreur

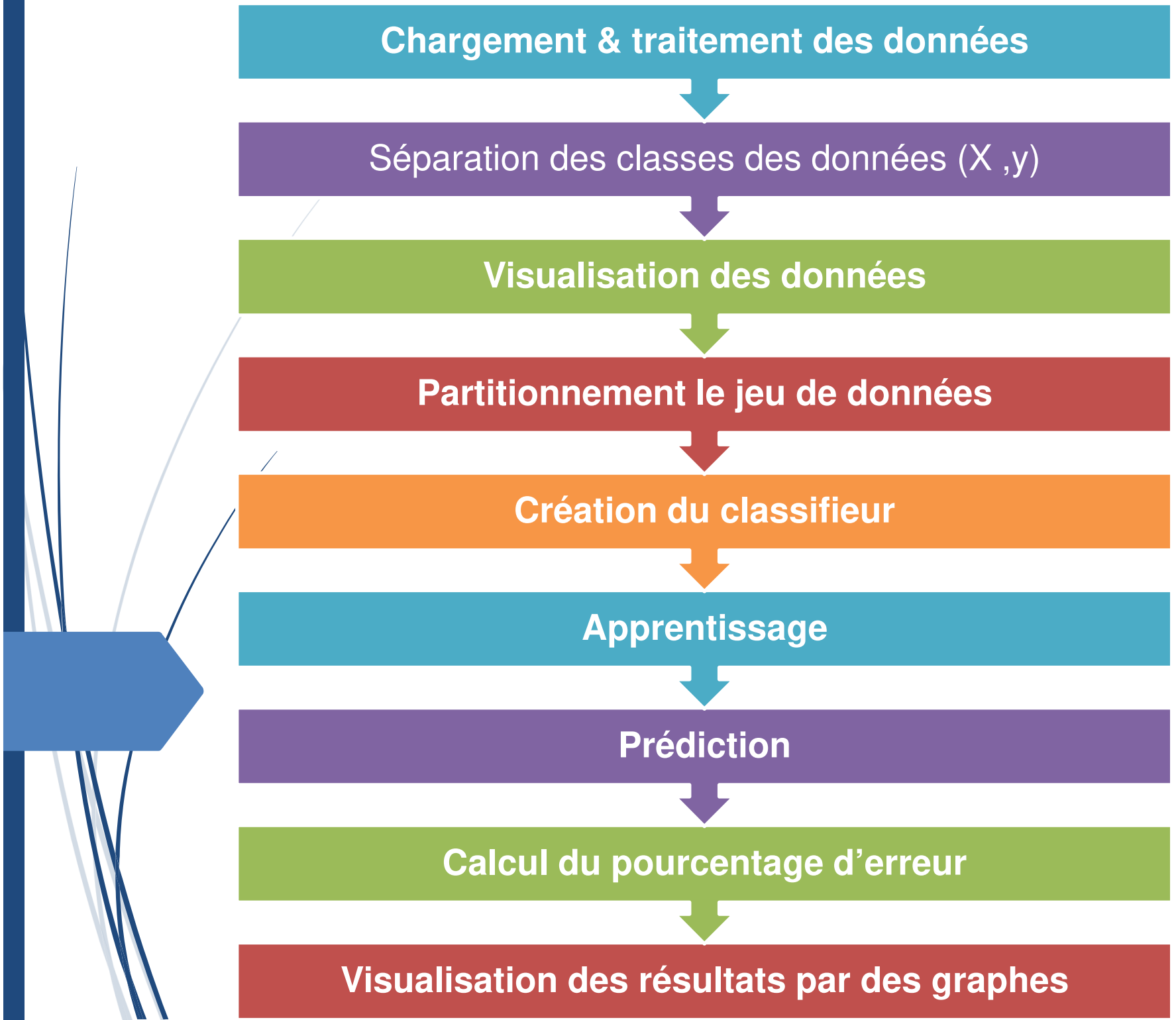
Comparer le résultat réel avec le résultat prédit en utilisant les métriques de sklearn.

- from sklearn.metrics import accuracy_score
  accuracy_score(les classes réelles , les classes prédites)

## 10. Visualisation des résultats par des graphes

# Résumé


Données d'apprentissage
1
création
Modèle
3
Prédiction
2
Données de test


Chargement & traitement des données
Séparation des classes des données (X ,y)
Visualisation des données
Partitionnement le jeu de données
Création du classifieur
Apprentissage
Prédiction
Calcul du pourcentage d'erreur
Visualisation des résultats par des graphes