

Python Astro: Starting with Python

Author: A. Klotz

Updated : 2022-02-
24

Contents:

- .Variables, inputs, print
- .Alternative sequences
- .Repetitive sequences
- .Lists and Numpy arrays
- .Graphics with Matplotlib
- .Add a package
- .An example of use of an astro package

Level: Beginners

Python prerequisite: Nothing

Astro prerequisite: Nothing

Le langage Python

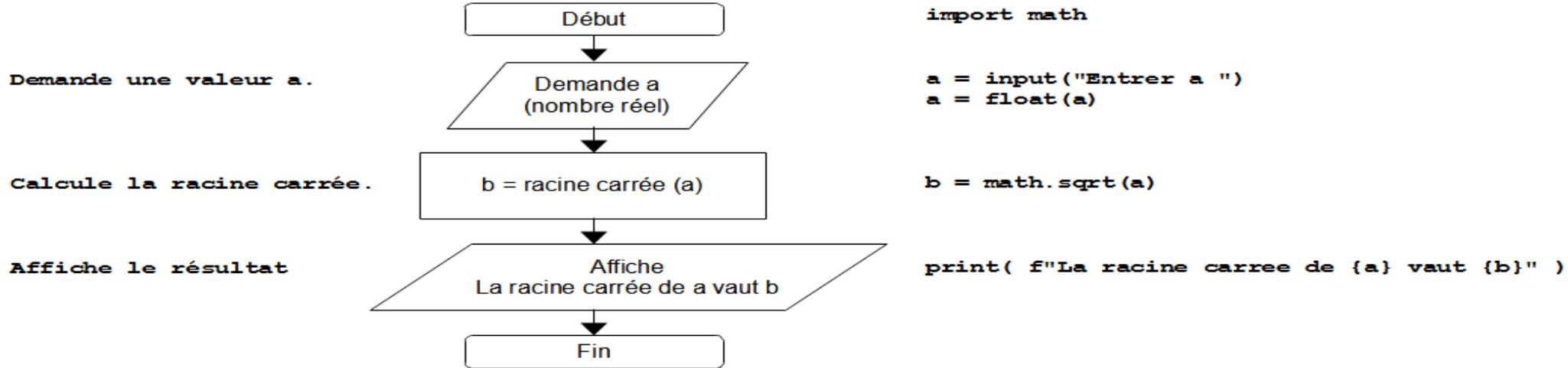
Entrées clavier (= demande)

Sorties console (= affiche)

Les variables

La programmation informatique

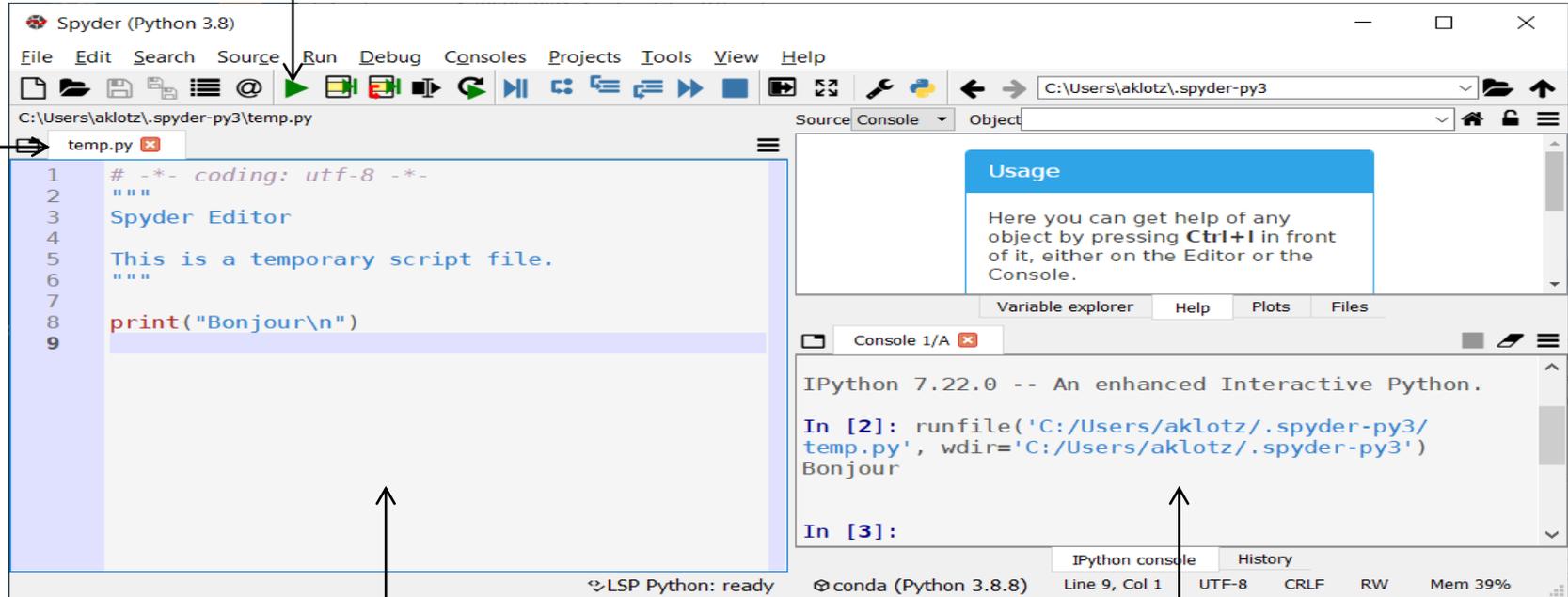
cahier des charges $\xrightarrow{\text{algorithmique}}$ organigramme $\xrightarrow{\text{programmation}}$ langage Python



Ecrire du Python et l'exécuter avec Spyder

Compile et exécute

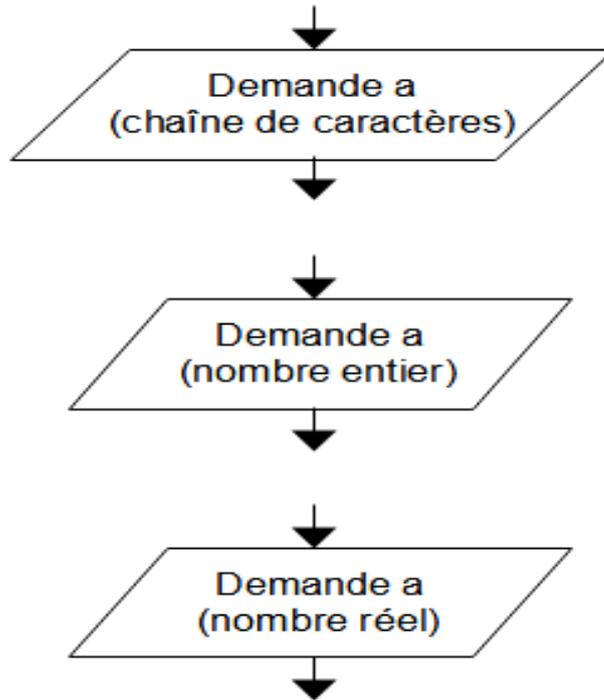
Nom du
fichier
Python



Fenêtre
du code
source

Fenêtre
de la console
d'exécution

Langage Python : Entrées à la console



```
a = input("Entrer a ")
```

```
a = input("Entrer a ")  
a = int(a)
```

```
a = input("Entrer a ")  
a = float(a)
```

Les variables ont un type

int : Nombre entier

float : Nombre décimal

str : Chaîne de caractères

input retourne toujours une chaîne de caractères

On peut convertir le type d'une variable.

Langage Python : Les calculs

Les nombres entier. Les quatre opérations :

1 + 2 * (3 / 4 - 5)

-7.5

Les nombres décimaux à virgule. Le séparateur décimal doit être un point :

6.723 - 5.01

1.713

Notation scientifique avec la lettre e. Par exemple $5 \cdot 10^3$:

5e3

5000.0

Python est aussi une calculatrice

Langage Python : Assignations et calculs

Les symboles de variables (pas de caractères spéciaux) :

longueur = 2

largeur = 3

longueur * largeur

6

Les symboles de variables (distinctions majuscules/minuscules) :

Longueur = 4

largeur = 3

longueur * largeur

6

Les symboles de variables suivent des règles :

Uniquement des lettres, des nombres et _

Distinction majuscules et minuscules

Pas d'accent

Pas de caractère blanc

Pas de caractères spéciaux

Tout en majuscules pour les constantes

longueur_totale3 = 45.67

C = 3e8

Langage Python : Les fonctions mathématiques

| Notation mathématique | Notation Python | Remarques |
|--|-----------------------------------|-------------------------|
| $y=e^x$ | <code>y = math.exp(x)</code> | |
| $y= x $ | <code>y = abs(x)</code> | Pas de math. |
| $y=\sqrt{x}$ | <code>y = math.sqrt(x)</code> | |
| $y=x^z$ | <code>y = math.pow(x, z)</code> | |
| $y=\ln(x)$ | <code>y = math.log(x)</code> | |
| $y=\log(x)$ | <code>y = math.log10(x)</code> | |
| $a=2\cdot\pi$ | <code>a = 2 * math.pi</code> | Pas de parenthèses |
| $a=\arctan(y/x)$ avec $a\in]-\pi/2;+\pi/2[$ | <code>a = math.atan(y/x)</code> | a en radians. Cf. atan2 |
| $a=\arctan(y/x)$ avec $a\in[-\pi;+\pi]$ | <code>a = math.atan2(y, x)</code> | Préférable à atan |
| $a=\arcsin(z)$ avec $a\in[-\pi/2;+\pi/2]$ | <code>a = math.asin(z)</code> | a en radians |
| $a=\arccos(z)$ avec $a\in[0;+\pi]$ | <code>a = math.acos(z)</code> | a en radians |

importer le module des fonctions mathématiques

```
import math
```

abs est une fonction "built-in"

Langage Python : Affichage à la console

Formatage de chaîne de caractères

| Script Python | Résultat affiché dans la console |
|---|----------------------------------|
| <pre>a = 2 print(f"valeur = {a}")</pre> | <pre>valeur = 2</pre> |

| Script Python | Résultat affiché dans la console |
|---|----------------------------------|
| <pre>a = 2 b = 3 print(f"valeurs = {a} et {b}")</pre> | <pre>valeurs = 2 et 3</pre> |

Formatage des colonnes

| Script Python | Résultat affiché dans la console |
|--|---------------------------------------|
| <pre>a = 2 b = 3000 print("123456789") print(f"{a:6}") print(f"{b:6}")</pre> | <pre>123456789 2 3000</pre> |

| Script Python | Résultat affiché dans la console |
|---|---|
| <pre>import math a = math.pi print("123456789 123456789") print(f"{a:10.3f}") print(f"{a:10.8f}")</pre> | <pre>123456789 123456789 3.142 3.14159265</pre> |

| Script Python | Résultat affiché dans la console |
|--|--|
| <pre>print("123456789") temperature = 3.6734 print(f"{temperature:+6.1f}") temperature = -12.078 print(f"{temperature:+6.1f}")</pre> | <pre>123456789 +3.7 -12.1</pre> |

Le langage Python

Programmer les séquences alternatives

Langage Python : Séquence alternative (if, else)

Cahier des charges

Demande une valeur `a`
à entière.

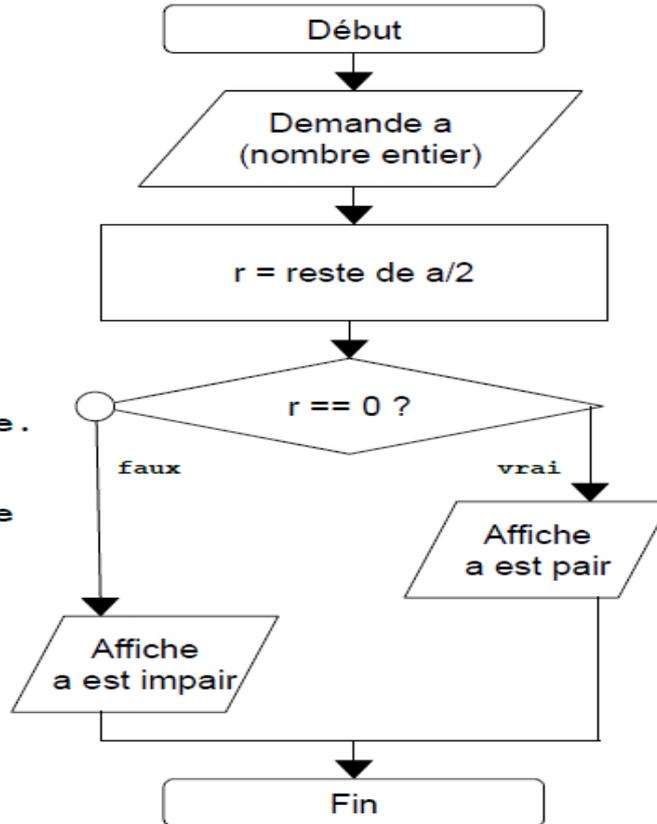
Calcule le reste de la
division euclidienne de
`a` par 2.

Teste la valeur du reste.

Affiche `a` est pair si le
reste vaut 0.

Affiche `a` est impair si
le reste ne vaut pas 0.

Organigramme



Code source (Python)

```
a = input("Entrer a ")  
a = int(a)
```

```
r = a % 2
```

```
if r == 0 :
```

```
    print( f"{a} est pair" )
```

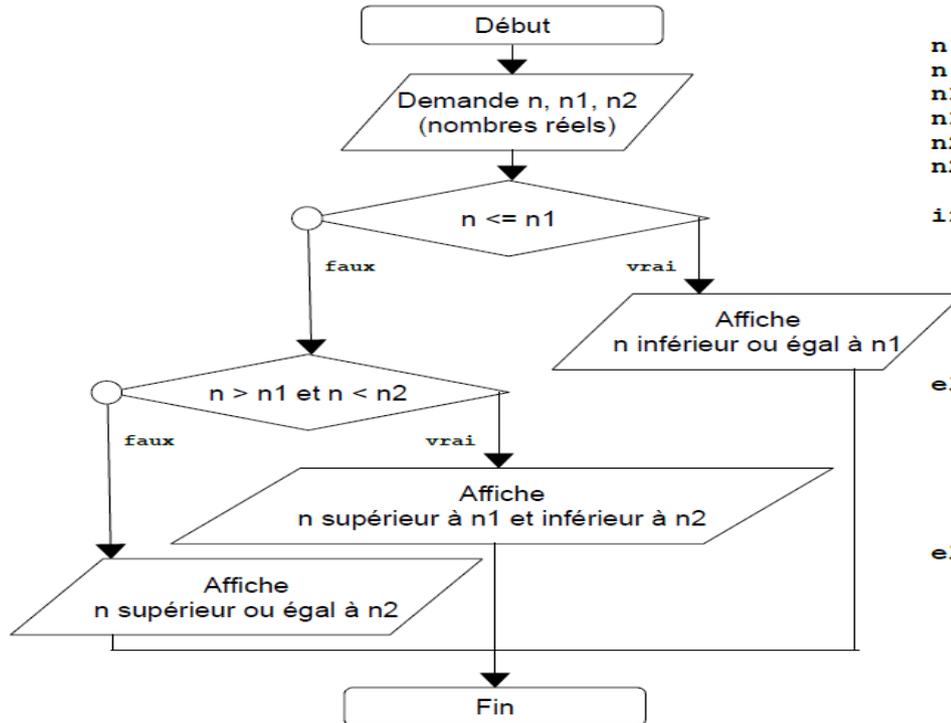
```
else :
```

```
    print( f"{a} est impair" )
```

Langage Python : Séquence alternative (if, elif, else)



Cahier des charges : Au lancement du code exécutable on demande un nombre n et deux autres nombres $n1$ et $n2$. On affiche des commentaires différents en fonction du test $n \leq n1$, $n1 < n < n2$ ou $n \geq n2$.



```
n = input("Entrer n ")
n = float(n)
n1 = input("Entrer n1 ")
n1 = float(n1)
n2 = input("Entrer n2 ")
n2 = float(n2)
```

```
if n <= n1 :
```

```
    print( f"{n} inférieur ou égal à {n1}" )
```

```
elif (n > n1) and (n < n2) :
```

```
    print( f"{n} supérieur à {n1} et inférieur à {n2}" )
```

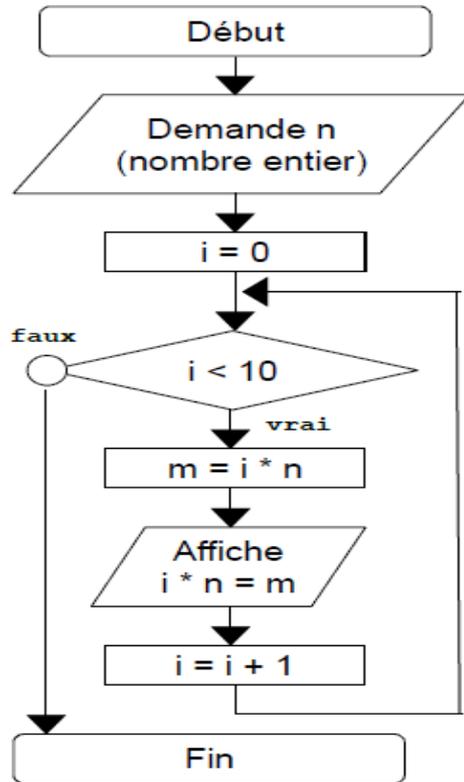
```
else :
```

```
    print( f"{n} supérieur ou égal à {n2}" )
```

Le langage Python

Programmer les séquences répétitives

Langage Python : Séquence répétitive (for et while)



Utilisation d'une boucle for

```
n = input("Entrer n ")  
n = int(n)
```

```
for i in range(0, 11, 1):
```

```
    m = i*n
```

```
    print( f"{i} * {n} = {m}" )
```

Utilisation d'une boucle while

```
n = input("Entrer n ")  
n = int(n)
```

```
i = 0
```

```
while i <= 10:
```

```
    m = i*n
```

```
    print( f"{i} * {n} = {m}" )
```

```
    i = i + 1
```

Langage Python : Séquence répétitive (boucle infinie)

On entre dans une boucle sans savoir quand on va la quitter.

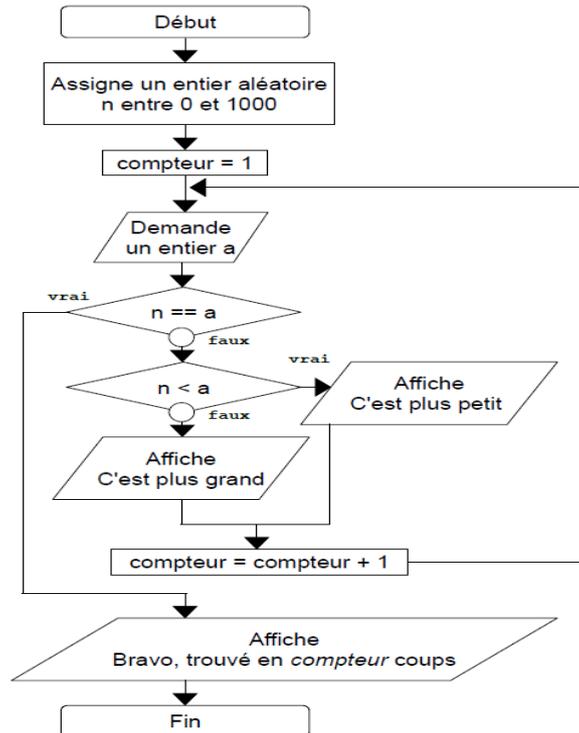
Exemple du jeu du "plus ou moins" :

Au départ, le programme tire au sort un nombre secret entre 0 et 1000.

On demande à l'utilisateur d'entrer des nombres jusqu'à ce qu'il trouve le nombre secret.

A chaque échec, on indique si le nombre secret est plus grand ou plus petit.

Une fois qu'il a trouvé, on indique à l'utilisateur combien de coups il a utilisés.



```
import random

n = random.randrange(0,1001,1)

compteur = 1
while True:
    a = input( "Entrer un nombre " )
    a = int(a)

    if n == a:
        break

    elif n < a:
        print( "C'est plus petit" )

    else:
        print( "C'est plus grand" )

    compteur = compteur + 1

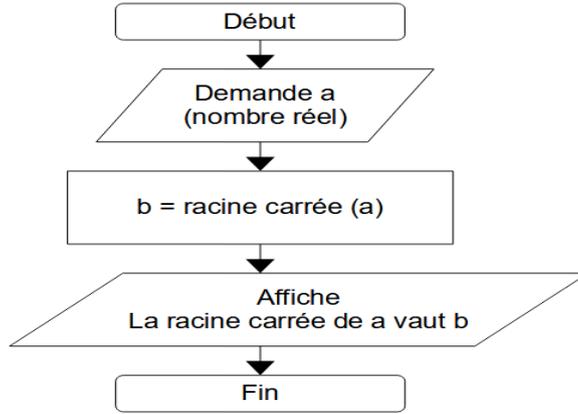
print( f"Bravo, trouvé en {compteur} coups" )
```

Le langage Python

Résumons les syntaxes des
séquences alternatives et répétitives

Les séquences algorithmiques et la **syntaxe Python**

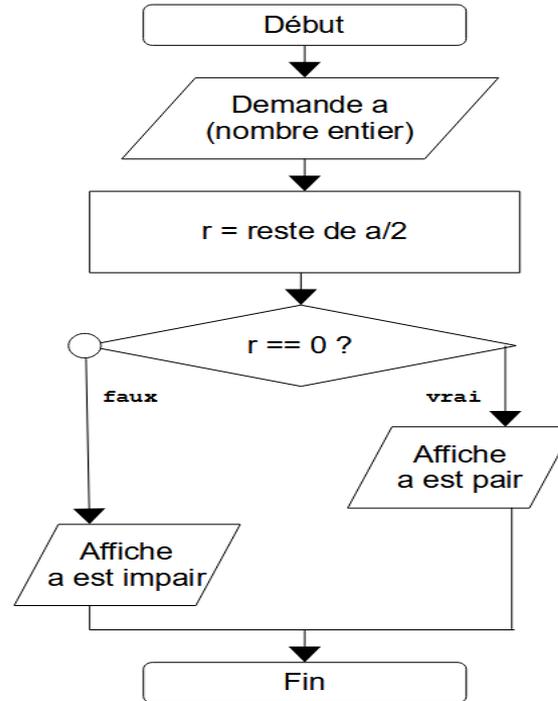
Séquence linéaire



import
input
print

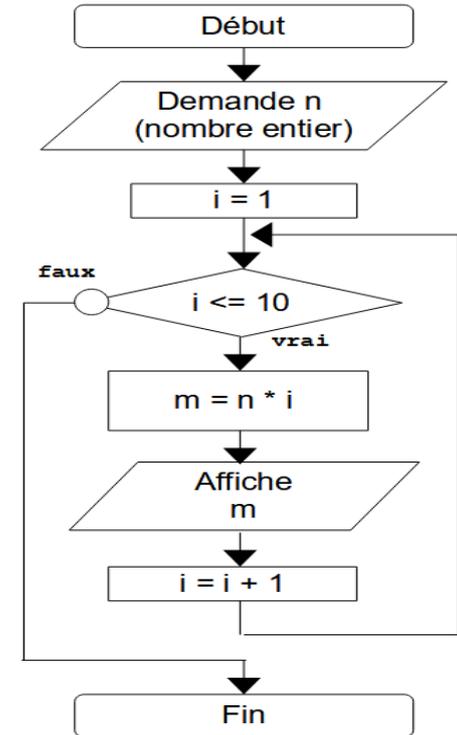
assignments ($a = 12$)
operations ($b = a + 10$)

Séquence alternative



if
elif
else

Séquence répétitive



for, range
while
break

Langage Python : Les indentations (les décalages)

Python est très rigoureux sur l'utilisation des indentations. Source d'erreurs !

```
n = input("Entrer n ")
n = float(n)
n1 = input("Entrer n1 ")
n1 = float(n1)
n2 = input("Entrer n2 ")
n2 = float(n2)

if n <= n1 :
    print( f"{n} inférieur ou égal à {n1}" )
elif (n > n1) and (n < n2) :
    print( f"{n} supérieur à {n1} et inférieur à {n2}" )
else :
    print( f"{n} supérieur ou égal à {n2}" )
```

1 niveaux
2 niveaux

ceci est un bloc

```
# Utilisation d'une boucle for

n = input("Entrer n ")
n = int(n)

for i in range(0, 11, 1):
    m = i*n
    print( f"{i} * {n} = {m}" )
```

1 niveaux
2 niveaux

ceci est un bloc

```
# Utilisation d'une boucle while

n = input("Entrer n ")
n = int(n)

i = 0
while i <= 10:
    m = i*n
    print( f"{i} * {n} = {m}" )
    i = i + 1
```

1 niveaux
2 niveaux

ceci est un bloc

Indenter lorsqu'une ligne se termine par le symbole : (if, elif, else, for, while, ...)
Une indentation est un décalage d'un "tab" ou de 4 espaces.

L'indentation démarre un bloc
La fin d'indentation signale la fin du bloc

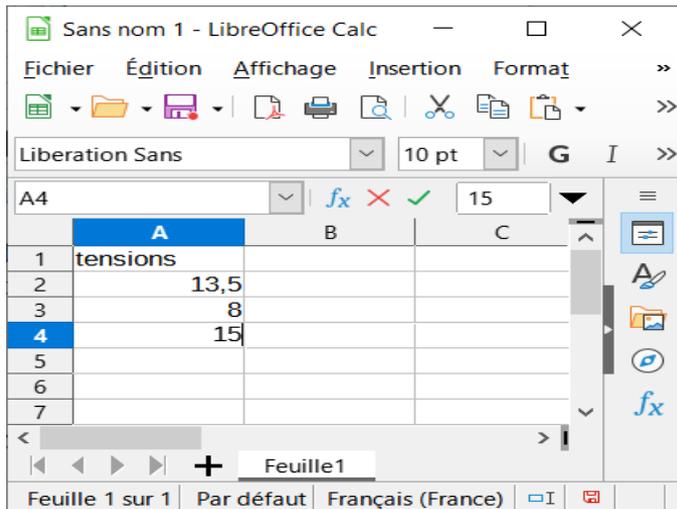
Le langage Python

Les listes

Les array de Numpy

Langage Python : Les listes

TABLEUR



| | A | B | C |
|---|----------|---|---|
| 1 | tensions | | |
| 2 | 13,5 | | |
| 3 | 8 | | |
| 4 | 15 | | |
| 5 | | | |
| 6 | | | |
| 7 | | | |

PYTHON

```
tensions = [13.5, 8, 15]
```

Les **crochets** [].

La **virgule** sert à séparer les éléments de la liste.

Accès à un seul élément de la liste en indiquant son **indice** :

```
tensions[0]
```

```
13.5
```

Les indices démarrent à zéro en Python.

Accès à plusieurs éléments de la liste en indiquant les **indices** début et fin :

```
tensions[0:2]
```

```
[13.5, 8]
```

L'indice de fin est exclu.

Créer une liste vide et **ajouter des éléments** un par un :

```
tensions = [ ]
```

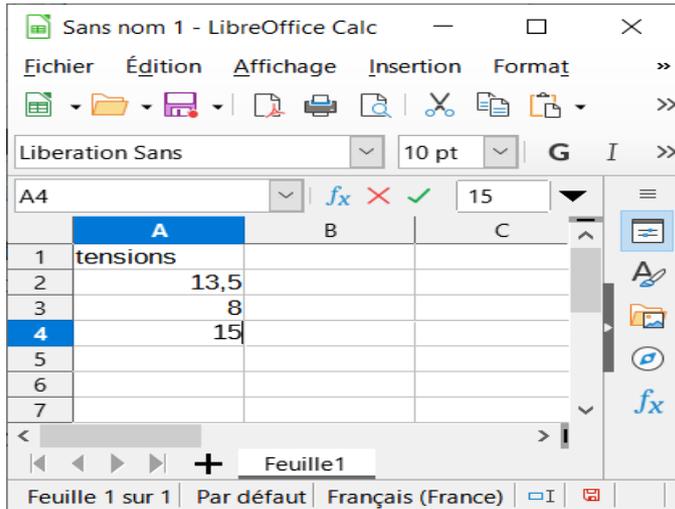
```
tensions.append(13.5)
```

```
tensions.append(8)
```

```
tensions.append(15)
```

Langage Python : Les listes et les séquences répétitives

TABLEUR



The screenshot shows a spreadsheet with the following data:

| | A | B | C |
|---|----------|---|---|
| 1 | tensions | | |
| 2 | 13,5 | | |
| 3 | 8 | | |
| 4 | 15 | | |
| 5 | | | |
| 6 | | | |
| 7 | | | |

PYTHON

Utilisation de append dans une boucle infinie:

```
tensions = [ ]
```

```
while True:
```

```
    tension = input( "Entrer une valeur " )
```

```
    tension = float(tension)
```

```
    tensions.append(tension)
```

```
    reponse = input( "Continuer à entrer une autre valeur ? (o/n)" )
```

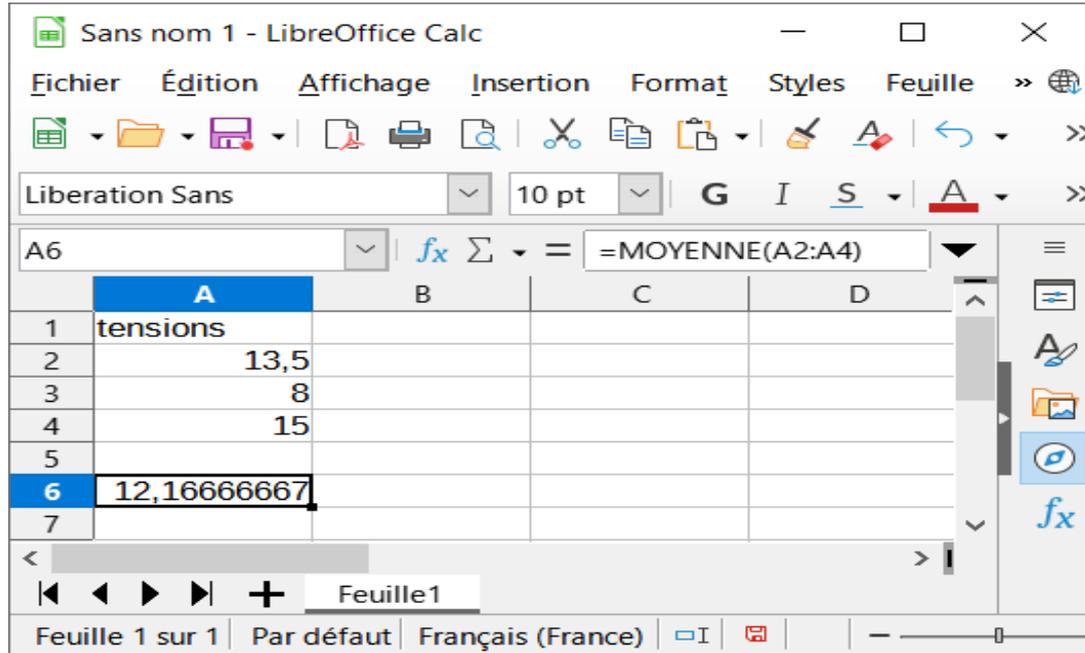
```
    if reponse == "n":
```

```
        break
```

En sortie du programme on a une liste de valeurs entrées au clavier.

Langage Python : Calculs avec les listes (moyenne)

TABLEUR



The screenshot shows the LibreOffice Calc interface. The spreadsheet has the following data:

| | A | B | C | D |
|---|-------------|---|---|---|
| 1 | tensions | | | |
| 2 | 13,5 | | | |
| 3 | 8 | | | |
| 4 | 15 | | | |
| 5 | | | | |
| 6 | 12,16666667 | | | |
| 7 | | | | |

The formula bar shows the formula `=MOYENNE(A2:A4)` for cell A6. The status bar at the bottom indicates 'Feuille 1 sur 1', 'Par défaut', and 'Français (France)'.

PYTHON

1) Créer la liste des valeurs:

```
tensions = [13.5, 8, 15]
```

2) Transformer la liste Python en array de la bibliothèque numpy :

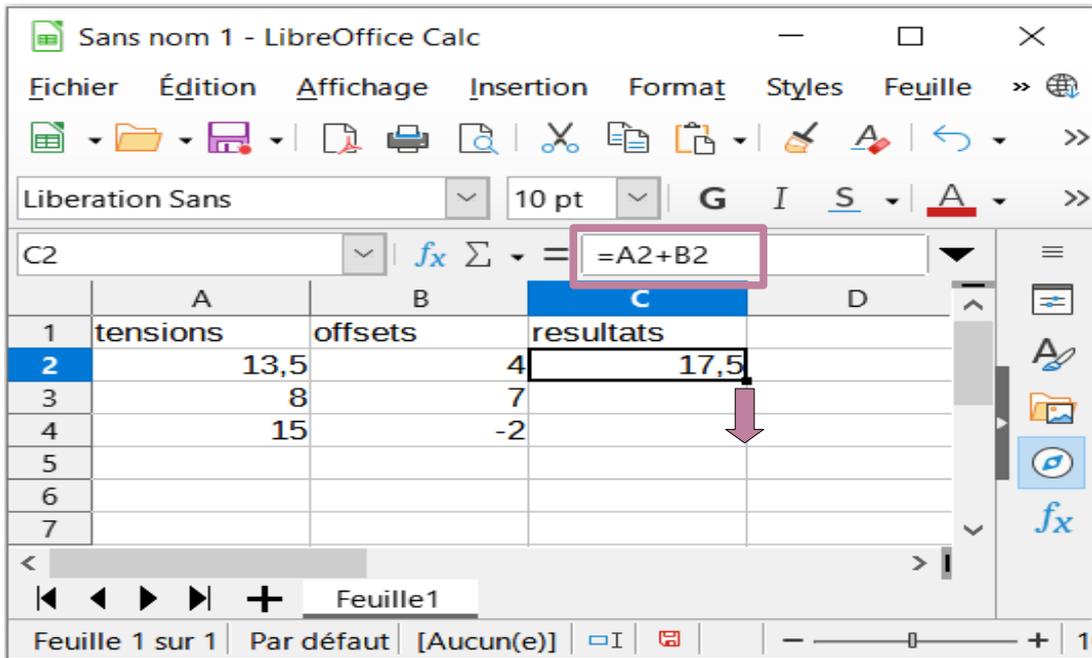
```
import numpy as np  
tensions = np.array(tensions)
```

3) Calculer la moyenne avec l'array

```
np.mean(tensions)  
12.166666666666666
```

Langage Python : Somme de deux listes

TABLEUR



| | A | B | C | D |
|---|----------|---------|-----------|---|
| 1 | tensions | offsets | resultats | |
| 2 | 13,5 | 4 | 17,5 | |
| 3 | 8 | 7 | | |
| 4 | 15 | -2 | | |
| 5 | | | | |
| 6 | | | | |
| 7 | | | | |

PYTHON

1) Créer les deux liste :

```
tensions = [13.5, 8, 15]  
offsets = [4, 7, -2]
```

2) Transformer les listes Python en array de la bibliothèque numpy :

```
import numpy as np  
tensions = np.array(tensions)  
offsets = np.array(offsets)
```

3) Calculer la somme des array :

```
resultats = tensions + offsets
```

4) Afficher les valeurs de array resultats :

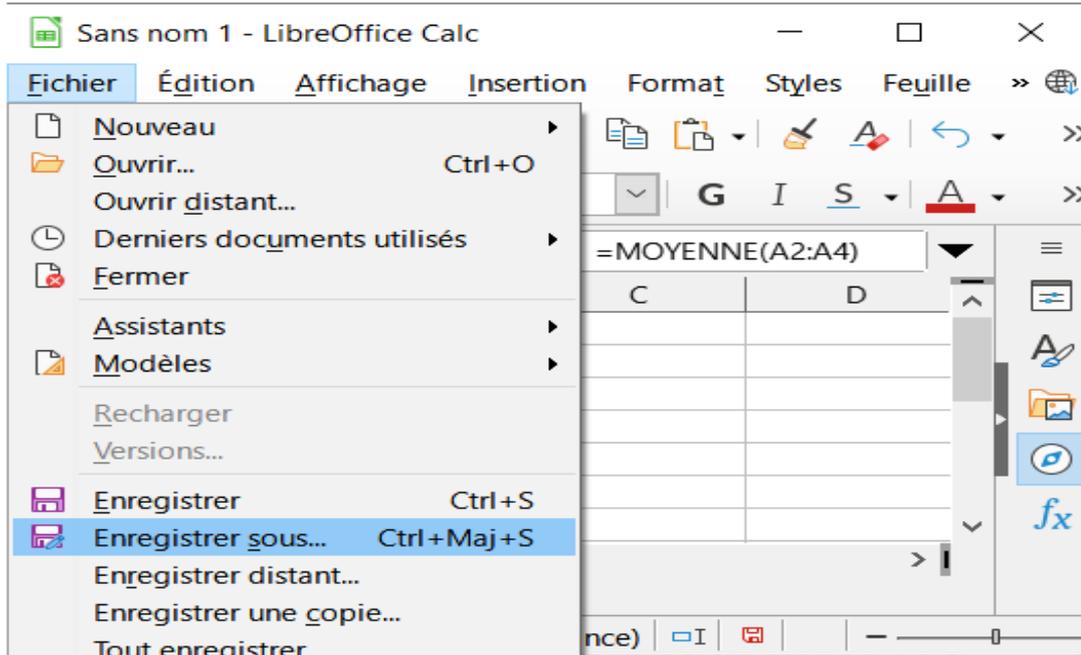
```
resultats  
array([17.5, 15. , 13. ])
```

5) Transformer un array en liste :

```
list(resultats)  
[17.5, 15.0, 13.0]
```

Langage Python : Enregistrer un fichier avec une liste

TABLEUR



PYTHON

1) Créer la liste des valeurs:

```
tensions = [13.5, 8, 15]
```

2) Transformer la liste Python en array de la bibliothèque numpy :

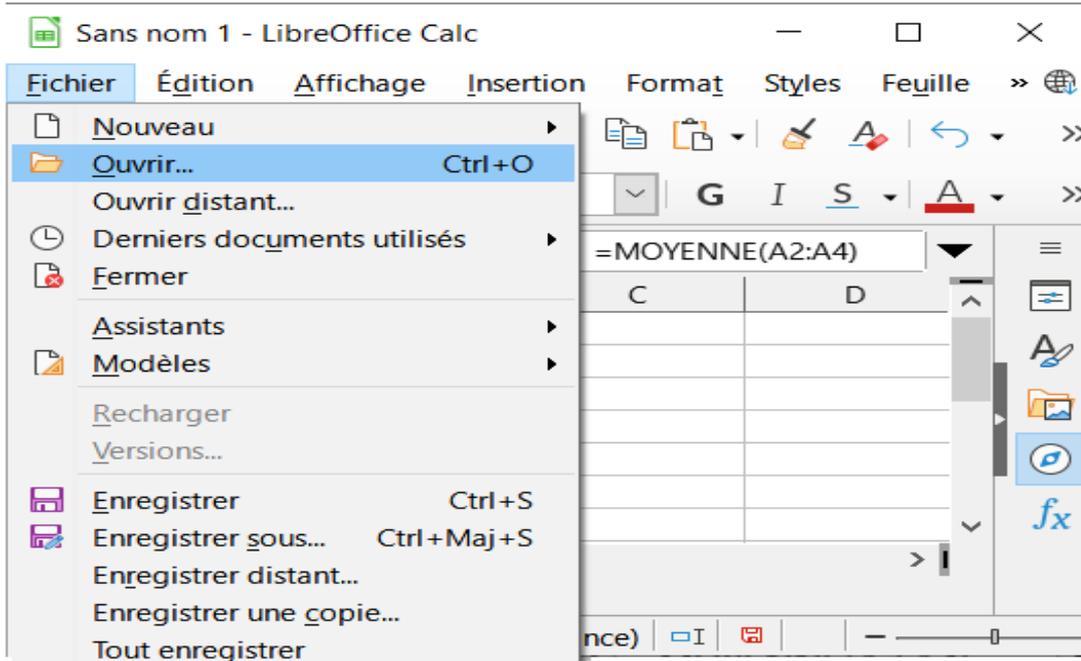
```
import numpy as np  
tensions = np.array(tensions)
```

3) Enregistrer l'array dans un fichier texte

```
np.savetxt( "data.txt" , tensions )
```

Langage Python : Lire un fichier vers une liste

TABLEUR



PYTHON

1) Lire l'array depuis un fichier texte

```
tensions = np.loadtxt( "data.txt" )
```

Langage Python : Générer un array de valeurs régulièrement espacées

Générer un array de tous les nombres entiers entre 0 et 20:

```
xs = np.linspace( 0, 20, 21)
```

```
xs
```

```
array([ 0.,  1.,  2.,  3.,  4.,  5.,  6.,  7.,  8.,  9., 10., 11., 12., 13., 14., 15., 16., 17., 18., 19., 20.])
```

Générer un array de 1000 nombres régulièrement espacés en échelle linéaire entre 0 et 20:

```
xs = np.linspace( 0, 20, 1000)
```

```
xs
```

```
array([ 0.          , 0.02002002, 0.04004004, 0.06006006, 0.08008008,  
       0.1001001 , 0.12012012, 0.14014014, 0.16016016, 0.18018018,  
       ...  
       19.81981982, 19.83983984, 19.85985986, 19.87987988, 19.8998999 ,  
       19.91991992, 19.93993994, 19.95995996, 19.97997998, 20.          ])
```

Générer un array de 9 nombres régulièrement espacés en échelle logarithmique entre 1 et 100:

```
xs = np.logspace( 0, 2, 9)
```

```
xs
```

```
array([ 1.          ,  1.77827941,  3.16227766,  5.62341325,  
       10.          , 17.7827941 , 31.6227766 , 56.23413252,  
       100.          ])
```

Interpréter le 0 pour $10^0 = 1$ et le 2 pour $10^2 = 100$.

L'échelle log est utile pour les calculs de diagrammes de Bode en électronique.

Le langage Python

Les graphiques avec Matplotlib

Langage Python : Affichage graphique avec deux arrays

La librairie Matplotlib permet d'afficher des courbes de données scientifiques facilement.

Premier exemple simple : $y = 4 \cdot e^{-x/5}$

Langage Python : Affichage graphique avec deux arrays

La librairie Matplotlib permet d'afficher des courbes de données scientifiques facilement.

Premier exemple simple : $y = 4 \cdot e^{-x/5}$

Générer un array xs de tous les nombres entiers entre 0 et 10:

```
import numpy as np
xs = np.linspace( 0, 10, 11)
xs
array([ 0.,  1.,  2.,  3.,  4.,  5.,  6.,  7.,  8.,  9., 10.]
```

Langage Python : Affichage graphique avec deux arrays

La librairie Matplotlib permet d'afficher des courbes de données scientifiques facilement.

Premier exemple simple : $y = 4 \cdot e^{-x/5}$

Générer un array xs de tous les nombres entiers entre 0 et 10:

```
import numpy as np
xs = np.linspace( 0, 10, 11)
xs
array([ 0.,  1.,  2.,  3.,  4.,  5.,  6.,  7.,  8.,  9., 10.]
```

Générer un array ys correspond à l'équation :

```
ys = 4 * np.exp(-xs/5)
ys
array([4.          , 3.27492301, 2.68128018, 2.19524654, 1.79731586,
       1.47151776, 1.20477685, 0.98638786, 0.80758607, 0.66119555,
       0.54134113])
```

Langage Python : Affichage graphique avec deux arrays

La librairie Matplotlib permet d'afficher des courbes de données scientifiques facilement.

Premier exemple simple : $y = 4 \cdot e^{-x/5}$

Générer un array xs de tous les nombres entiers entre 0 et 10:

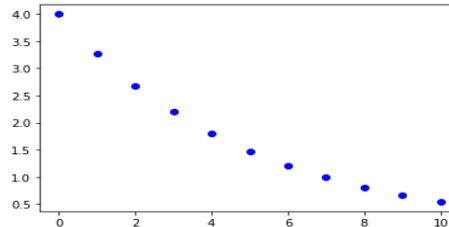
```
import numpy as np
xs = np.linspace( 0, 10, 11)
xs
array([ 0.,  1.,  2.,  3.,  4.,  5.,  6.,  7.,  8.,  9., 10.]
```

Générer un array ys correspond à l'équation :

```
ys = 4 * np.exp(-xs/5)
ys
array([4.          , 3.27492301, 2.68128018, 2.19524654, 1.79731586,
       1.47151776, 1.20477685, 0.98638786, 0.80758607, 0.66119555,
       0.54134113])
```

Afficher le graphique de y en fonction de x:

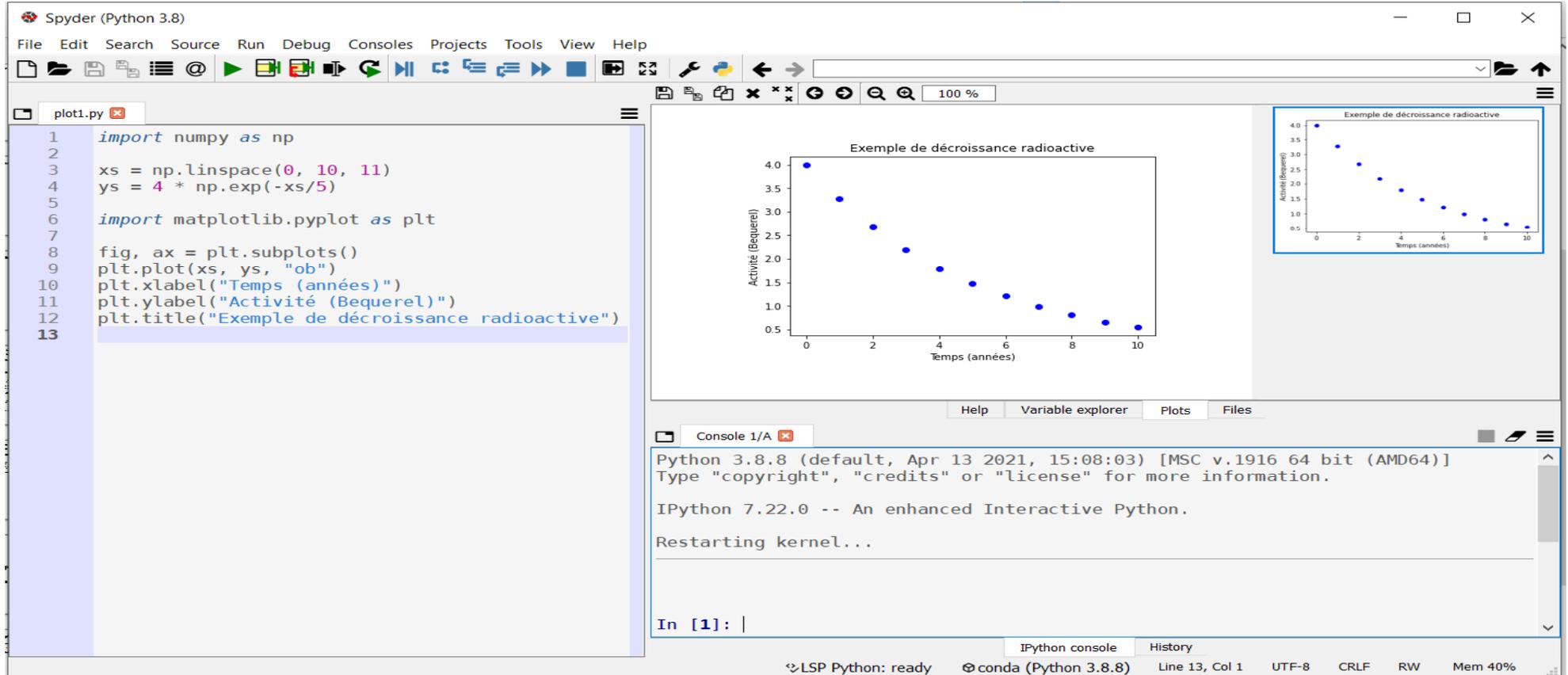
```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
plt.plot(xs, ys, "ob")
```



Langage Python : Affichage graphique avec deux arrays

On montre ici comment ajouter les indications sur les axes et donner un titre au graphique

Améliorations du premier exemple



The screenshot displays the Spyder Python IDE interface. On the left, a code editor shows a Python script named 'plot1.py' with the following code:

```
1 import numpy as np
2
3 xs = np.linspace(0, 10, 11)
4 ys = 4 * np.exp(-xs/5)
5
6 import matplotlib.pyplot as plt
7
8 fig, ax = plt.subplots()
9 plt.plot(xs, ys, "ob")
10 plt.xlabel("Temps (années)")
11 plt.ylabel("Activité (Bequerel)")
12 plt.title("Exemple de décroissance radioactive")
13
```

The main window displays a scatter plot titled "Exemple de décroissance radioactive". The x-axis is labeled "Temps (années)" and ranges from 0 to 10. The y-axis is labeled "Activité (Bequerel)" and ranges from 0.5 to 4.0. The plot shows a series of blue circular markers connected by a thin line, illustrating exponential decay. A smaller version of the same plot is visible in the top right corner of the main window.

Below the plot, the console window shows the following output:

```
Python 3.8.8 (default, Apr 13 2021, 15:08:03) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.22.0 -- An enhanced Interactive Python.

Restarting kernel...

In [1]: |
```

The bottom status bar indicates: LSP Python: ready, conda (Python 3.8.8), Line 13, Col 1, UTF-8, CRLF, RW, Mem 40%.

Langage Python : Affichage graphique avec deux arrays

On montre ici comment calculer une régression linéaire et superposer l'ajustement aux points

Régression linéaire

The image shows the Spyder Python IDE interface. On the left, a code editor displays a Python script for linear regression. The script imports numpy and matplotlib, generates data points, and performs a linear fit. The main window shows a plot titled "Exemple de décroissance radioactive" with a scatter plot of blue dots and a red line representing the linear fit. The x-axis is labeled "Temps (années)" and the y-axis is "Activité (Bequerel)". A console window at the bottom shows the output of the linear fit coefficients: `array([-0.33072312, 3.43739475])`. Two blue arrows point from the text "Pente" and "Ordonnée à l'origine" to the first and second elements of the array, respectively.

```
1 import numpy as np
2
3 xs = np.linspace(0, 10, 11)
4 ys = 4 * np.exp(-xs/5)
5
6 import matplotlib.pyplot as plt
7
8 fig, ax = plt.subplots()
9 plt.plot(xs, ys, "ob")
10 plt.xlabel("Temps (années)")
11 plt.ylabel("Activité (Bequerel)")
12 plt.title("Exemple de décroissance radioactive")
13
14 # Regression lineaire
15 coefs = np.polyfit(xs, ys, 1)
16 yys = np.polyval(coefs, xs)
17 plt.plot(xs, yys, "-r")
18
```

Console output:

```
In [3]: coefs
Out[3]: array([-0.33072312, 3.43739475])
In [4]:
```

Annotations:

- Pente (points to `-0.33072312`)
- Ordonnée à l'origine (points to `3.43739475`)

Le langage Python

Ajouter un package