

Initiation au langage PYTHON via l'Environnement de Développement Intégré « Thonny »

Support à destination d'élèves de terminale de filières générales.

Sommaire

1 - Sources inspirantes.....	2
2 - Introduction.....	2
3 - Présentation de l'IDE Thonny.....	3
4 - Afficher des données dans le shell et parler à python.....	8
5 - Stocker des données dans des variables.....	11
6 - Calculer.....	14
7 - Les séries de données – une liste pour vos mesures.....	17
8 - Le test booléen – avec des si.....	21
9 - Boucle bornée - les tâches répétitives mais ayant une fin.....	27
10 - Boucle non bornée – les tâches répétitives dont je ne connais pas le terme.....	35
11 - Les fonctions – pour faire comme en maths.....	39
12 - Présentation des bibliothèques – bienvenue dans l'Alexandrie du XXIème siècle.....	49
13 - Import de données d'un fichier csv.....	52
14 - Utilisation de fonctions en maths et calculs.....	60
15 - Visualisation des données avec matplotlib.....	64
16 - Faire de la physique avec scypi.....	79

* Certaines remarques furent inspirées de HAL 9000 ou CARL 500



1 - Sources inspirantes

- [Cours de Python pour les enseignants de Physique Chimie \(PDF de 1.9 Mo\) - Stage lycée en Physique Chimie pour l'académie de Poitiers](#)
- [Adaptation libre de "Apprendre à programmer avec Python" de G. Swinnen](#)
- [Documentation Python en ligne](#)

2 - Introduction

La programmation permet de créer ses propres outils grâce à un succession d'instructions. Pour communiquer ces instructions à l'ordinateur il existe de nombreux langages et Python est le langage enseigné en France de la seconde à l'université en passant par les classes prépa.

Pour l'utiliser concrètement, il vous faudra un environnement de développement (IDE). Vous disposez des possibilités suivantes :

- IDE en local (ex Thonny)

Comme nous utilisons au lycée l'IDE Thonny pour le SNT en classe de seconde, nous verrons comment télécharger et installer cet environnement. Et comme nous utiliserons des bibliothèques spécifiques, nous verrons sous Thonny comment ajouter celles-ci.

- IDE en ligne (ex : <https://repl.it/>)
- Appli sur votre téléphone (ex : Pydroid 3)

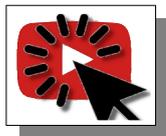
Au regard de la protection des données personnelles, l'Appli génère des annonces, et l'environnement en ligne pourra poser problème dans le cadre de l'utilisation des bibliothèques d'ou mon conseil d'installer en local un environnement de développement.

3 - Présentation de l'IDE Thonny

3.1 - Téléchargement et installation

Suivre le lien puis télécharger et installer la version Windows ou Mac suivant vos besoins. Pour Linux vous avez la procédure d'installation du paquet en survolant le lien.

<https://thonny.org/>



[Tuto Vidéo](#)

Rq :

Lors de l'installation de Thonny, vous installer python ! Donc vous n'avez besoin de rien d'autre... sauf de quelques bibliothèques supplémentaire. C'est la rubrique du dessous;

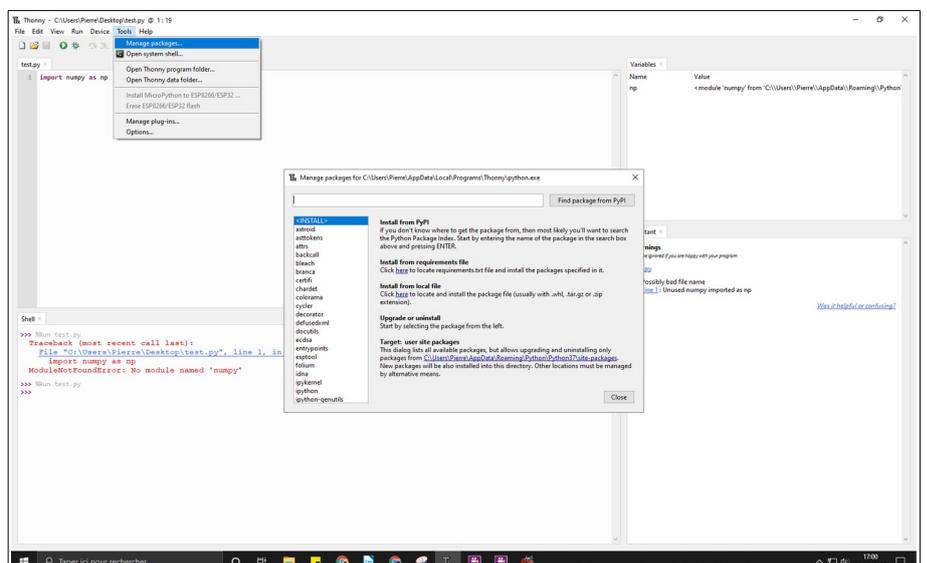
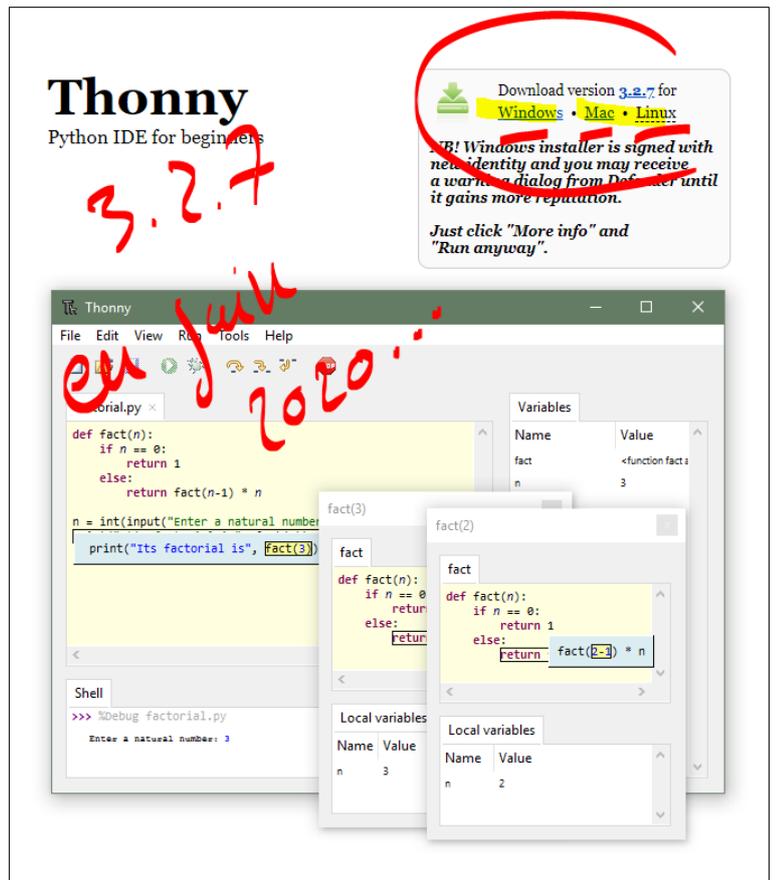
3.2 - Ajout des bibliothèques

Une des grandes forces du langage Python réside dans le nombre important de bibliothèques logicielles externes disponibles. Une bibliothèque est un ensemble de fonctions. Celles-ci sont regroupées et mises à disposition afin de pouvoir être utilisées sans avoir à les réécrire. Elles permettent de faire:

- du calcul numérique,
- du graphisme,
- de la programmation internet ou réseau,
- du formatage de texte,
- de la génération de documents...

3.2.1 - Procédure

Menu Tools / Manage packages...



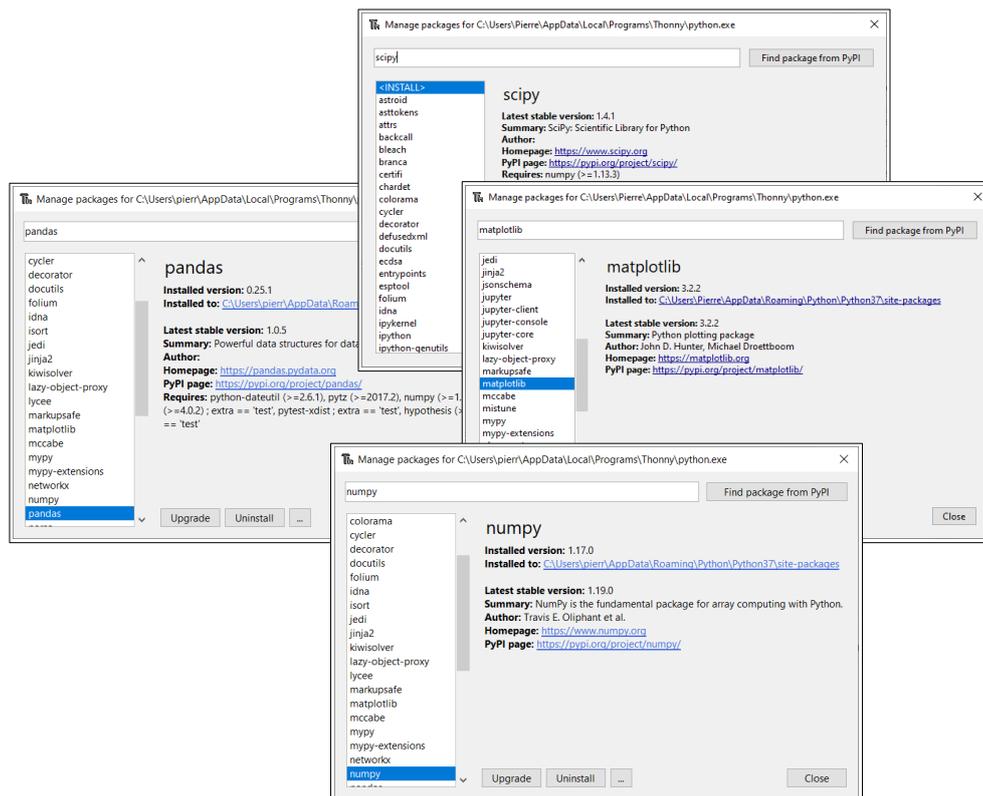


Tuto Vidéo

3.2.2 - Liste des bibliothèques à installer

Voici la liste des bibliothèques qui seront nécessaires à ce support de cours. Vous devez vérifier leurs présences et/ou leurs mises à jour via le gestionnaire d'installation.

- matplotlib
- scipy
- numpy
- pandas



3.3 - Vos notes

3.4 - Présentation de l'environnement de développement intégré – EDI ou IDE en anglais

3.4.1 - Présentation en vidéo



[Tuto vidéo](#)

- Présentation de l'éditeur
- Visualisation de l'exécution du programme dans le Shell
- Présentation du mode debug et visualisation de l'évolution des déclarations et des valeurs des variables

3.4.2 - Les fenêtres de l'environnement

Via le menu View / vous pouvez ajouter à l'éditeur de code les fenêtres Shell / Variables et pourquoi pas l'Assistant.

The screenshot shows the Thonny IDE interface with the following components and annotations:

- Code Editor:** Contains Python code for finding divisors. A red circle '1' is drawn around the code.
- Shell:** Shows the execution output: "Saisir le premier nombre entier: 87" and "Les diviseurs de 87 sont [1, 3, 29]". A red circle '2' is drawn around the output.
- Variables:** A table showing the current state of variables: a=1.0, aini=87, diviseurs=[1, 3, 29], i=29. A red circle '3' is drawn around the table.
- Assistant:** Shows a message: "Possibly bad file name". A red circle '3' is drawn around the Assistant window.
- View Menu:** A red arrow points to the 'View' menu, which is open, showing options like Assistant, Exception, Files, Heap, Help, Notes, Object inspector, Outline, Program tree, Shell, Stack, Variables, Program arguments, and Plotter. Red circles '2' and '3' are drawn around the 'Shell' and 'Variables' options respectively.

Script du dessous à copier / coller dans la zone 1 puis exécuter le programme. Au premier lancement vous devez enregistrer votre script (l'extension d'un programme python est py) « nom de votre fichier.py »

Attention, vous devez double-cliquer sur la zone de texte grise puis sélectionner toutes les lignes de texte avant de faire votre copier/coller.

```
# Etude des Diviseurs d'un nombre quelconque

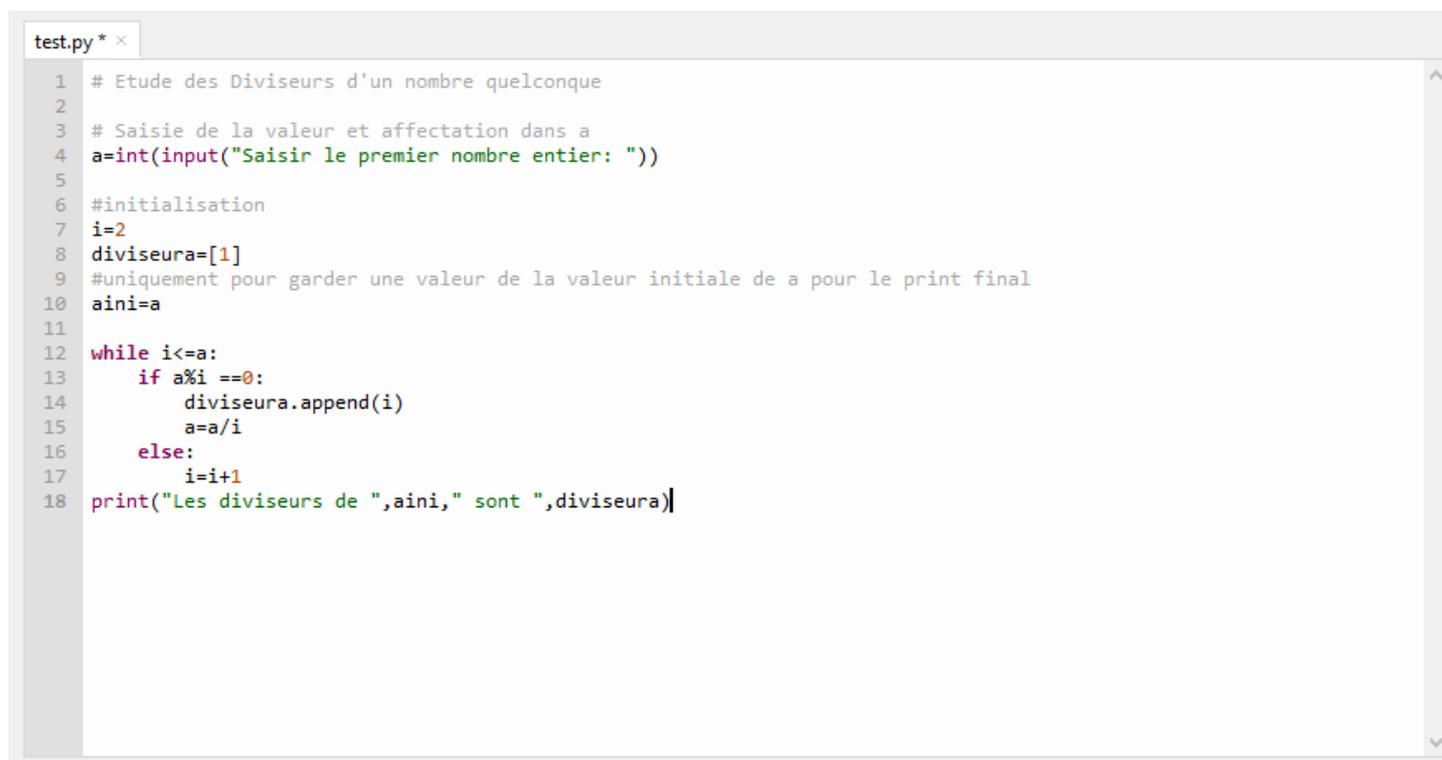
# Saisie de la valeur et affectation dans a
a=int(input("Saisir le premier nombre entier: "))

#initialisation
i=2
diviseurs=[1]
#uniquement pour garder une valeur de la valeur initiale de a pour le print final
aini=a

while i<=a:
    if a%i ==0:
        diviseurs.append(i)
        a=a/i
    else:
        i=i+1
print("Les diviseurs de ",aini," sont ",diviseurs)
```

3.4.2.1 - L'éditeur de code – Zone 1

C'est là que l'on travaille. L'éditeur dispose d'une reconnaissance syntaxique qui colore les commandes. Les lignes de code sont numérotées. Dans l'éditeur se trouve l'âme de votre programme.



```
test.py * x
1 # Etude des Diviseurs d'un nombre quelconque
2
3 # Saisie de la valeur et affectation dans a
4 a=int(input("Saisir le premier nombre entier: "))
5
6 #initialisation
7 i=2
8 diviseurs=[1]
9 #uniquement pour garder une valeur de la valeur initiale de a pour le print final
10 aini=a
11
12 while i<=a:
13     if a%i ==0:
14         diviseurs.append(i)
15         a=a/i
16     else:
17         i=i+1
18 print("Les diviseurs de ",aini," sont ",diviseurs)
```

3.4.2.2 - Le shell – Zone 2

Le shell représente la partie visible du programme (l'enveloppe, la coquille). Dans le shell votre programme s'incarne ;) . Noter que l'on peut directement travailler dans le shell !

Clic droit / Clear pour effacer les données du shell

```
Shell x
Python 3.7.7 (bundled)
>>> %Run essai.py

Saisir le premier nombre entier: 125447862215975669887469
Les diviseurs de 125447862215975669887469 sont [1, 12569, 13682, 14336, 20441, 622333]
>>> |
```

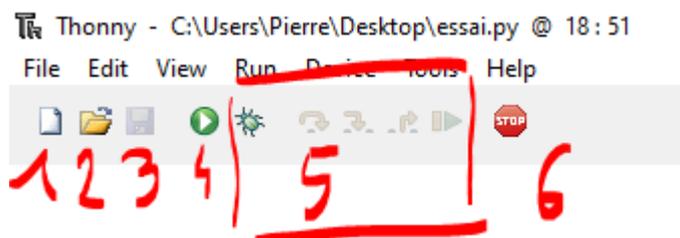
3.4.2.3 - La fenêtre des variables – Zone 3

Fenêtre où vous verrez évoluer vos variables. Très utile pour comprendre ce que l'âme de votre programme essaye de faire ;)

Name	Value
a	4.0
aini	125447862215975669887469
diviseursa	[1, 12569, 13682, 14336, 20441, 622333]
i	622333

3.4.3 - L'exécution d'un programme

1. Nouveau programme extension .py
2. Ouvrir un programme
3. Enregistrer un programme
4. Exécuter un programme
5. Mode debug super pratique ! L'essayer c'est l'adopter
6. Arrêt du script et purge des variables



3.5 - Vos notes

4 - Afficher des données dans le shell et parler à python

Pour afficher du texte dans le shell, on utilise la commande print avec des (). Le contenu à afficher est à mettre entre les ().

Le texte doit être entre ". Pour afficher plusieurs blocs de texte, il faut les séparer par une ,

4.1 - Commandes

Commande	Remarque
print()	"imprime" = affiche le contenu dans les ()
input()	Commande permettant de demander à l'utilisateur de saisir une valeur et l'affecter à une variable ex : a=input("saisir la valeur de a=") !!! a sera de type string avec un input !!!
"""	Le texte doit se trouver entre " "
,	Pour séparer plusieurs éléments
#	Commente une ligne
\n	Pour générer un saut de ligne
\"	Pour afficher des caractères spéciaux comme "

4.2 - Exercices

4.2.1 - Exo – print



Dans le **shell** saisir puis valider :

- print("bonjour")
- print("bonjour à", "tous")
- print("bonjour à tous)
- print("a", "b", "c", sep="-")

Notez en rouge le message d'erreur lors de l'exécution des commandes. L'important est de repérer le n° de la ligne contenant l'erreur puis la position de celle-ci ^

```
Shell x
Python 3.7.7 (bundled)
>>> print("bonjour")
bonjour
>>> print("bonjour à", "tous")
bonjour à tous
>>> print("bonjour à tous)
File "<pyshell>", line 1
    print("bonjour à tous)
    ^
SyntaxError: EOL while scanning string literal
>>> print("a", "b", "c", sep="-")
a-b-c
>>> |
```

4.2.2 - Exo – print et input



Dans l'éditeur saisir le code puis l'exécuter :

```
# ceci est un commentaire
print("Les commentaires sont là pour \"expliquer\" la logique de votre code")
print("car chacun aura son approche")
print()
print("Ceci est une chaîne plutôt longue\n contenant plusieurs lignes \
... de texte (Ceci fonctionne\n de la même façon en C/C++.\n\
... Notez que les blancs en début\n de ligne sont significatifs.\n")
```

Je comprends :

- print() => saut de ligne
- \ => saut le ligne dans l'éditeur
- \n => saut de ligne dans le shell

Dans l'éditeur saisir le code puis l'exécuter :

```
# ce qui suit est un dialogue
prenom=input("Saisir votre prénom: ")
print("*****")
print("Bonjour HAL")
print("Bonjour ",prenom, ". Que puis-je faire pour vous?")
```

Je comprends :

- `prenom=input("Saisir votre prénom: ")`
 - `input("Saisir votre prénom: ")` => Affiche le texte "Saisir votre prénom: "
 - et attend votre réponse. Quand la réponse est saisie puis validée (Touche Entrée)
 - affectation de votre saisie à la variable `prenom`
- Exécution des lignes de code suivantes...

4.3 - Vos notes

Rq : Maintenant HAL peut nous parler... et en tant qu'utilisateur je peux lui répondre...

5 - Stocker des données dans des variables

Une variable permet de stocker des informations (nombres, texte, ...). **Attention**, certains noms pour les variables seront interdits car correspondants à des commandes – par exemple `complex`

Le séparateur de décimal est le `.` (point)

Le nom des variables est sans espace (utiliser `_` si besoin) et est sensible à la casse donc `Variable=` est différent de `variable=`.

La longueur de la variable importe peu, donc vous pouvez être explicite. Pour l'énergie cinétique entre `Ec` et `E_cinetique` je pense que je préfère la seconde.

Le signe `=` est à lire comme ceci " affecte la valeur à " et se remplace dans un algorithme par `←`

Python devine le type de variable en fonction de vos saisies, donc si `a=10.2` `a` est une variable numérique de type décimale mais si `a="10.2"` alors `a` est une chaîne de caractère de type texte !

5.1 - Commandes

Commande	Remarque
<code>a=20</code>	Affecte la valeur numérique entière 20 à la variable <code>a</code>
<code>a,b=10,13.5</code>	Affecte les valeurs 10 et 13,5 respectivement aux variable <code>a</code> et <code>b</code>
<code>print(type(a))</code>	La commande <code>type()</code> renvoie le type de la variable. <code>str</code> = texte / <code>int</code> = entier / <code>float</code> = décimale / <code>bool</code> = Vrai ou Faux / <code>list</code> pour list / <code>complex</code> pour nombre complexe (attention <code>a=2+1j</code> avec <code>j</code> notion de l'imaginaire)
<code>a=1-2j</code> <code>b=4+1j</code> <code>a+b</code>	Notation complexe et somme
<code>a="aie aie aie"</code> <code>b="ouille ouille ouille"</code> <code>c=a+b</code>	"Somme" de chaînes de caractères

5.2 - Exercices

5.2.1 - Exo – variables et affichage



Dans l'éditeur copier/coller ou saisir :

```
# affectation des valeurs
# val numerique
a,b=1,2.1
# val complexe
compl1=2+1j
compl2=-1-5j
# val texte
texte1="aie aie aie"
texte2=" ouille ouille ouille" #piège ;)

#operation
c=a+b
som_compl=compl1+compl2
som_texte=texte1+"\n"+Texte2

print(type(a),type(b),type(c))
print(type(som_compl),"Somme=",som_compl)
print(som_texte)
```

Je comprends :

- que le piège est grossier;
- que le type de c est float car lors de la somme de a et b il ne peut faire autrement à cause de b
- que le i étant souvent utilisé dans les scripts, l'imaginaire est j et qu'il faut déclarer 1j
- que python peut faire des calculs avec du texte !!
- que jeter un œil dans la fenêtre variable est souvent utile
- que som_texte contient 3 chaînes de caractère dont une "\n" qui sera interprétée comme saut de ligne

```
Thonny - C:\Users\pierr\Downloads\test.py @ 9:39
File Edit View Run Device Tools Help

test.py x
1 # affectation des valeurs
2 # val numerique
3 a,b=1,2.1
4 # val complexe
5 compl1=2+1j
6 compl2=-1-5j
7 # val texte
8 texte1="aie aie aie"
9 Texte2=" ouille ouille ouille" #piège |.)
10
11 #operation
12 c=a+b
13 som_compl=compl1+compl2
14 som_texte=texte1+"\n"+Texte2
15
16 print(type(a),type(b),type(c))
17 print(type(som_compl),"Somme=",som_compl)
18 print(som_texte)

Variables x
Name Value
Texte2 'ouille ouille ouille'
a 1
b 2.1
c 3.1
compl1 (2+1j)
compl2 (-1-5j)
som_compl (1-4j)
som_texte 'aie aie aie\n ouille ouille ouille'
texte1 'aie aie aie'

Assistant x
Warnings
May be ignored if you are happy with your program.
test.py
Possibly bad file name
Was it helpful or confusing?

Shell x
>>> %Run test.py
<class 'int'> <class 'float'> <class 'float'>
<class 'complex'> Somme= (1-4j)
aie aie aie
ouille ouille ouille
>>>
```

5.2.2 - Exo – variables et pile mémoire



Je veux obtenir dans le shell une pyramide d'étoile comme ceci →

Et mon programme doit commencer par cela →

```
test.py x
1 s=""
2 print(s)
3 s=s+"*"
```

```
>>> %Run test.py
*
**
***
****
*****
>>> |
```

Coller votre proposition dans la zone texte du dessous !

Votre script que vous collez ici avec une extrême fierté

5.3 - Vos notes

Rq : Maintenant, HAL peut se souvenir...

J'insiste sur le `s=s+"*"` qui DOIT se lire de la droite vers la gauche ← comme ceci :

- je prends la valeur de `s` et je lui ajoute le texte `*` (donc si `s` contient déjà une `*` cela en fait `**`)
- j'affecte ce nouveau résultat (donc `**`) dans la variable `s` (et donc la valeur de `s` change et correspond à `**`)

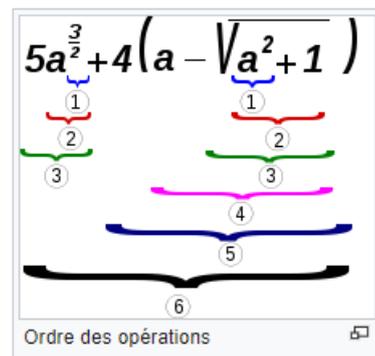
6 - Calculer

Python peut faire des calculs de la même façon qu'un tableur mais vous ne pourrez pas additionner des valeurs numériques avec des valeurs de type texte

La gestion des priorités se fait grâce aux () et non les []

Vous ne devez pas omettre les symboles car $2d$ n'est pas $2*d$

Par curiosité si la variable s contient la chaîne de caractère "*" alors vous pouvez lui demander de calculer `grand_S=4*s` et alors `grand_S` aura la valeur '*****'



6.1 - Opérateurs et symboles

Symbole	Opération
+	Addition
-	Soustraction
*	Multiplication
**	Puissance. Exemple $a=2^{**}3$ équivaut à affecter 2 à la puissance 3 donc $2*2*2=8$ à la variable a
/	Division. Exemple $a=13/4$ affecte la valeur décimale 3,25 à la variable a
//	Division entière. Exemple $a=13//4$ affecte la partie entière donc 3 à la variable a car $13=4*3+1$
%	Modulo ou "Reste de la division entière". Exemple $a=13\%4$ affecte le reste donc 1 à la variable a car $13=4*3+1$

6.2 - Exercices

6.2.1 - Exo - calculs et affectation des valeurs

6.2.1.1 - Énoncé



Dans l'éditeur, saisir un script permettant d'afficher les valeurs successives de $S_n = S_{n-1}^{S_{n-1}}$ pour n allant de 0 à 3. (lire S_n est égale à S_{n-1} à la puissance S_{n-1}) sachant que :

$S_0 = 2$ puis $S_1 = S_0^{S_0}$ puis $S_2 = S_1^{S_1}$ puis $S_3 = S_2^{S_2}$!

Coller votre proposition dans la zone texte du dessous !

Votre script que vous collez ici avec une extrême fierté

6.2.1.2 - Un exemple de correction

```
test.py *
1 # valeur initiale
2 s=2
3 # 2 lignes strategique pour la recopie
4 print("S0= ",s)
5 s=s**s
6 # fin de la copie
7 print("S1= ",s)
8 s=s**s
9 print("S2= ",s)
10 s=s**s
11 print("S3= ",s)
12
```

```
>>> %Run test.py
S0= 2
S1= 4
S2= 16
S3= 32317006071311007300714876688669951960444102669715484032130345427524655138867890893197201411522913463688717960921898019494119559150490921
095098152386448283120430877367300996091750197803986821067960576393840675882767922186426197561418980943384761704705816489520363050428875758915
41045300407523591239303852191433396483424200849747865645694945617603532432205807780585933102416270846031415025859236427711471294360371946
185735759835115230164590440369761323287231227125684710820209725157101724931323469678542580656697935045997268352988638215525166389437335543602
135433229604645318478604952148193555853611059596230656
>>>
```

Variables

Name	Value
s	323170060713110073007148766886699519604441026697154840321303454275

Assistant

Warnings

May be ignored if you are happy with your program.

test.py

Possibly bad file name

[Was it helpful or confusing?](#)

Rq :

- Noter la puissance de calcul de Python avec les entiers, je n’essaye même pas de lire S_3 !
- Noter qu’il n’en est pas de même avec les valeurs décimales, [pour les curieux voir ici](#)

6.2.2 - Exo – calculs et affectation des valeurs

6.2.2.1 - Énoncé



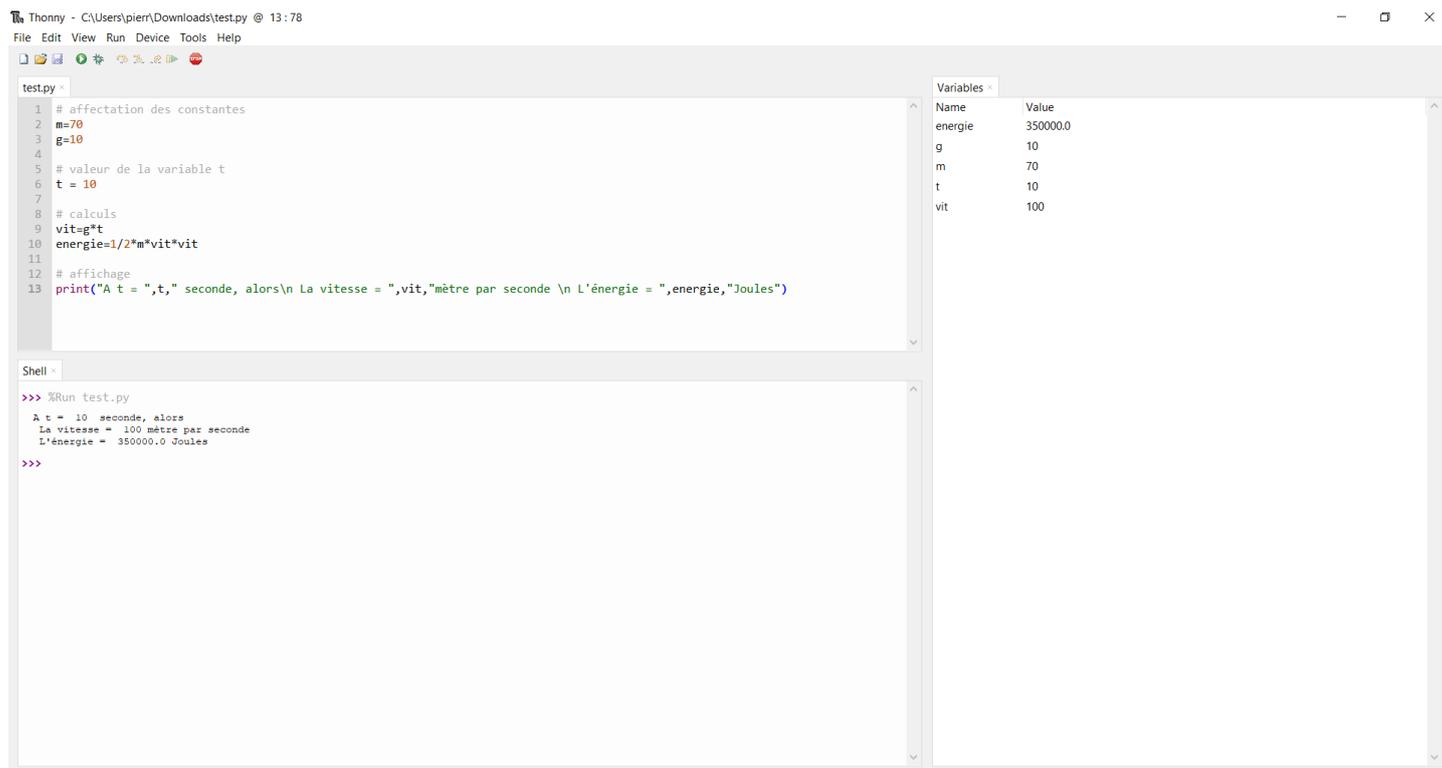
Soit les relations $Energie\ cinétique = \frac{1}{2} \cdot masse \cdot vitesse^2$ et $Vitesse = cte\ de\ gravitation \cdot temps$
 $E_c = \frac{1}{2} \cdot m \cdot V^2$ $V = g \cdot t$

On donne $m = 70\ kg$
 $g = 10\ m \cdot s^{-2}$

Dans l'éditeur, écrire un programme permettant d'afficher pour $t = 10$ s les valeurs de la vitesse et de l'énergie cinétique. Vous commenterez votre programme.

Votre script que vous collez ici avec une extrême fierté

6.2.2.2 - Un exemple de correction



The screenshot shows the Thonny Python IDE interface. The main editor window displays a Python script named 'test.py' with the following code:

```
1 # affectation des constantes
2 m=70
3 g=10
4
5 # valeur de la variable t
6 t = 10
7
8 # calculs
9 vit=g*t
10 energie=1/2*m*vit**2
11
12 # affichage
13 print("A t = ",t," seconde, alors\n La vitesse = ",vit,"mètre par seconde \n L'énergie = ",energie,"Joules")
```

The Shell window shows the execution output:

```
>>> %Run test.py
A t = 10 seconde, alors
La vitesse = 100 mètre par seconde
L'énergie = 350000.0 Joules
>>>
```

On the right side, the Variables window displays the current state of the program's variables:

Name	Value
energie	350000.0
g	10
m	70
t	10
vit	100

6.3 - Vos notes

Rq : Je commence à avoir peur !!! HAL peut se souvenir de très grands nombres.
Mais si je calcule S_4 alors je constate que Dieu existe;)

Au fait, vous avez fait probablement plus court dans votre second programme : $energie=1/2*m*vit**2$

7 - Les séries de données – une liste pour vos mesures

Tant en physique qu'en mathématique, et à l'aire du BigData, les données sont légions... Alors pour pouvoir s'amuser avec il faut pouvoir les enregistrer ! Que dis-je, les enregistrer et les indexer !! C'est là qu'interviennent les listes.

La liste est une variable pouvant contenir plusieurs variables. Elle a le bon goût d'être itérable, donc modifiable, triable, évolutive...

Une liste peut aussi contenir des listes...

Une liste peut mémoriser votre liste de course et mélanger carottes (des nombres) avec serviettes (des chaînes de caractères)

Les données d'une liste sont séparées par des ,

Les éléments d'une liste s'appellent grâce à leurs indices qui sont uniques - un élément par indice

Remarque :

Python peut enregistrer des données dans des variables, des listes, des matrices (array) et des dictionnaires... La commande `type(nom_de_votre_variable)` vous donne retourne le type en question : int, float, str, array, list, dict....

7.1 - Commandes et symboles

Commande ou Symbole	Opération
[]	Symbole associé aux listes
[1,2,58,"AA"]	Valeurs de la liste 1 puis 2 puis 58 puis AA (str) Attention, cette liste n'ayant pas de nom elle ne fonctionne pas dans un script !
nom_de_ma_liste=[]	Liste vide [] de nom "nom_de_ma_liste"
a=[] a.append(25)	a est une liste vide La méthode append ajoute 25 à la liste a
x=[0,1,2,3] y=3*x	Concatène 3 fois la liste x dans la liste y donc y contient [0,1,2,3,0,1,2,3,0,1,2,3]
print(x[2])	Affiche la 3ème valeur de la liste x (si la liste n'a pas 3 valeurs alors message d'erreur index out of range)
print(x[:5])	Affiche les valeurs de x de l'indice 0 à l'indice 4 ! donc les 5 premières valeurs

Étude d'une liste de valeurs

Numéro de la mesure	1ère valeur	2ème	3...	4...	5...	6ème et dernière valeur
Indice	0	1	2	3	4	5
Valeurs de la série x (Abscisse)	0,5	3	4,5	8,5	12	25

Pour Python, ces 6 valeurs peuvent être mémorisées dans la liste suivante :

```
valeur_x=[ 0.5 , 3 , 4.5 , 8.5 , 12 , 25 ]
```

- N'oubliez pas que le séparateur de décimale est le `.`
- Les données sont séparées par une `,`
- La première valeur **0,5** s'appelle avec `valeur_x[0]` (0 étant son indice)
- `valeur_x[2]` correspond à **4,5** (si vous pensiez obtenir 3 alors relire;)
- Le contenu complet de la liste s'appelle simplement avec `valeur_x`

7.2 - Exercices

7.2.1 - Exo – liste et indice

7.2.1.1 - Énoncé



Dans l'éditeur :

- créer une liste de nom "vitesse" et contenant les valeurs suivantes :

n° de la valeur	1	2	3	4	5
Vitesse	0,0	0,25	0,51	1,02	2,42

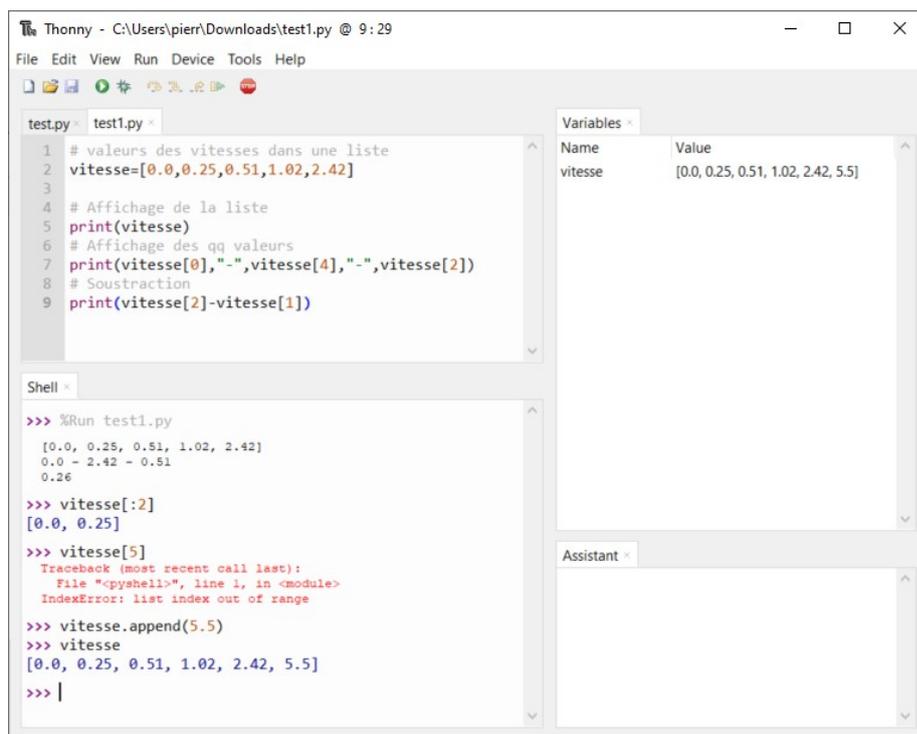
- avec un `print()`, afficher le contenu de la liste vitesse
- avec un `print()` faire apparaître la première valeur, la dernière valeur et celle du milieu (la 3ème). Vos valeurs seront séparées par `"-"`
- avec un `print()` faire apparaître la différence de vitesse entre 0,51 et 0,25 en appelant les valeurs de la liste vitesse ad hoc

Dans le shell :

- tester ceci et comprenez ce qui se passe dans l'affichage `vitesse[:2]`
- tester ceci et comprenez ce qui se passe dans l'affichage `vitesse[5]`
- ajouter 5,5 à la liste avec la méthode `append`
- afficher la liste vitesse

7.2.1.2 - Un exemple de correction

- Noter que dans le shell le print peut être omis
- vitesse[:2] affiche les 2 premières valeurs (indice 0 et 1)
- vitesse[5] n'existe pas car les indices utilisés dans cette liste sont :
0,1,2,3,4



The screenshot shows the Thonny Python IDE with a file named 'test1.py'. The code in the editor is as follows:

```
1 # valeurs des vitesses dans une liste
2 vitesse=[0.0,0.25,0.51,1.02,2.42]
3
4 # Affichage de la liste
5 print(vitesse)
6 # Affichage des qq valeurs
7 print(vitesse[0],"-",vitesse[4],"-",vitesse[2])
8 # Soustraction
9 print(vitesse[2]-vitesse[1])
```

The Shell window shows the output of running the script:

```
>>> %Run test1.py
[0.0, 0.25, 0.51, 1.02, 2.42]
0.0 - 2.42 - 0.51
0.26

>>> vitesse[:2]
[0.0, 0.25]

>>> vitesse[5]
Traceback (most recent call last):
  File "<pyshell>", line 1, in <module>
IndexError: list index out of range

>>> vitesse.append(5.5)
>>> vitesse
[0.0, 0.25, 0.51, 1.02, 2.42, 5.5]
>>> |
```

The Variables window shows the variable 'vitesse' with the value [0.0, 0.25, 0.51, 1.02, 2.42, 5.5].

7.2.2 - Exo – liste, sous-liste et append

7.2.2.1 - Énoncé



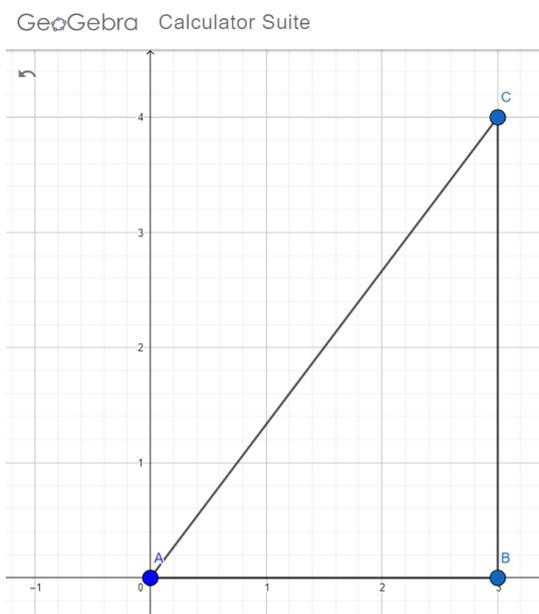
Soit le joli triangle suivant rectangle en B →

Dans l'éditeur :

copier/coller le code suivant :

```
# Liste des points du joli triangle
# la liste point contient 3 sous listes
point=[
    ["A",0,0],
    ["B",3,0],
    ["C",3,4],
]

print(point)
print(point[0])
print(point[1][0])
```



Tester le script puis comprendre :

- La liste point contient 3 éléments et chaque élément est une liste (donc sous-liste) contenant 3 valeurs :
 - le nom du point i

- la valeur en abscisse du point i
- la valeur en ordonnée du point i
- La "profondeur" de la liste point est de 2 niveaux donc je peux appeler l'élément d'une sous-liste de liste avec [indice la liste][indice de la sous-liste].



Dans l'éditeur :

- modifier le script du dessus pour ajouter dans la liste point un nouvel élément - le point D - qui aura les coordonnées du milieu de [A,C].

Vous adapterez les relations suivantes

$$\begin{cases} x_D = \frac{(x_C - x_A)}{2} \\ y_D = \frac{(y_C - y_A)}{2} \end{cases}$$

- afficher la liste point modifiée

7.2.2.2 - Un exemple de correction

- J'ai décomposé mon raisonnement en calculant xd et yd lignes 14 et 15 => le code comme cela est plus clair...
- Il est possible de faire autrement évidemment !!!

7.3 - Vos notes

Rq : Bon, et bien HAL semble pouvoir mettre en liste l'humanité entière. Notez qu'il est presque en capacité d'étudier les suites car identifier un terme par U_n en maths ou $U[n]$ en info... c'est totalement adaptable !

```
test1.py
1 # Liste des points du joli triangle
2 # la liste point contient 3 sous listes
3 point=[
4     ["A",0,0],
5     ["B",3,0],
6     ["C",3,4],
7 ]
8
9 print(point)
10 print(point[0])
11 print(point[1][0])
12
13 # Calcul du point D
14 xd=(point[2][1]-point[0][1])/2
15 yd=(point[2][2]-point[0][2])/2
16
17 # Ajout de l'element calculé dans la liste
18 point.append(["D",xd,yd])
19
20 # Affichage de la liste point
21 print(point)
```

Shell

```
>>> %Run test.py
[["A", 0, 0], ["B", 3, 0], ["C", 3, 4]]
["A", 0, 0]
B
[["A", 0, 0], ["B", 3, 0], ["C", 3, 4], ["D", 1.5, 2.0]]
>>> ]
```

Variables

Name	Value
point	[["A", 0, 0], ["B", 3, 0], ["C", 3, 4], ["D", 1.5, 2.0]]
xd	1.5
yd	2.0

Assistant

Warnings

test.py

Possibly bad file name

Was it helpful or confusing?

8 - Le test booléen – avec des si...

Dans certains cas, il faut faire des choix, et pour faire des choix il faut faire des tests. La question commence toujours par si puis après vous avez les choix suivants :

- La condition if (“si”)
- La condition if...else (“si...sinon”)
- La condition if...elif...else (“si...sinon si... sinon”)

Vous connaissez l’expression avec des si on mettrait Paris en bouteille ! Je ne sais pas pour Paris mais il est sûr qu’avec des si on peut faire des scripts sympa.

8.1 - Commandes

Commande ou Symbole	Opération
if	Si ... ce qui suit
else	Sinon ... ce qui suit
:	: fin de ligne puis indentation
indentation	4 espaces
Opérateur de comparaison <ul style="list-style-type: none">• < strictement inférieur• > strictement supérieur• <= inférieur ou égal• >= supérieur ou égal• == égal• != différent	Comme = se traduit pas "affecter xxx à la variable..." il est normal que le concept d'égalité s'écrive différemment => ==

8.2 - Exercices

8.2.1 - Exo – Test avec if, si vrai... sinon...

8.2.1.1 - Énoncé



Dans l'éditeur :

copier / coller le script suivant

```
# Vitesse du point a en m/s
Va = 3
# Vitesse du point b dans une autre unité
Vb = 6
# Que fait la ligne 6?
Vb = Vb*1000/3600

# Ce qui suit est un test
if Va > Vb :
    # Si Va est > à Vb alors faire toutes les instructions qui sont indentées(décalées) ci-dessous
    print("L'objet a est plus rapide que l'objet b ")
    Vmax = Va
else :
    # Sinon, faire toutes les instructions qui sont indentées ci-dessous
    print("L'objet b est plus rapide que l'objet a ")
    Vmax = Vb

# conclusion
print("Cette vitesse maximale est de ", Vmax, " m/s")
```

Remarques :

- Vous voyez bien les : ?
- Lignes 10 à 12 = bloc traité si la condition est vrai
- Ligne 14 à 16 = bloc traité si la condition est fausse
- Ligne 18 sans indentation donc toujours traitée, retour au code principal

```
essai.py x
1 # Vitesse du point a en m/s
2 Va = 3
3 # Vitesse du point b dans une autre unité
4 Vb = 6
5 # Que fait la ligne 6?
6 Vb = Vb*1000/3600
7
8 # Ce qui suit est un test
9 if Va > Vb :
10 # Si Va est > à Vb alors faire toutes les instructions qui sont indentées(
11 print("L'objet a est plus rapide que l'objet b ")
12 Vmax = Va
13 else :
14 # Sinon, faire toutes les instructions qui sont indentées ci-dessous
15 print("L'objet b est plus rapide que l'objet a ")
16 Vmax = Vb
17
18 # conclusion
19 print("Cette vitesse maximale est de ", Vmax, " m/s")
```

Dans l'éditeur MAIS en mode

Débug :

(la bestiole - touche 1)

Exécutez via la touche "Step into"
(F7 – touche 2) instruction par
instruction le programme (clic-clic-
clic) et comprenez...



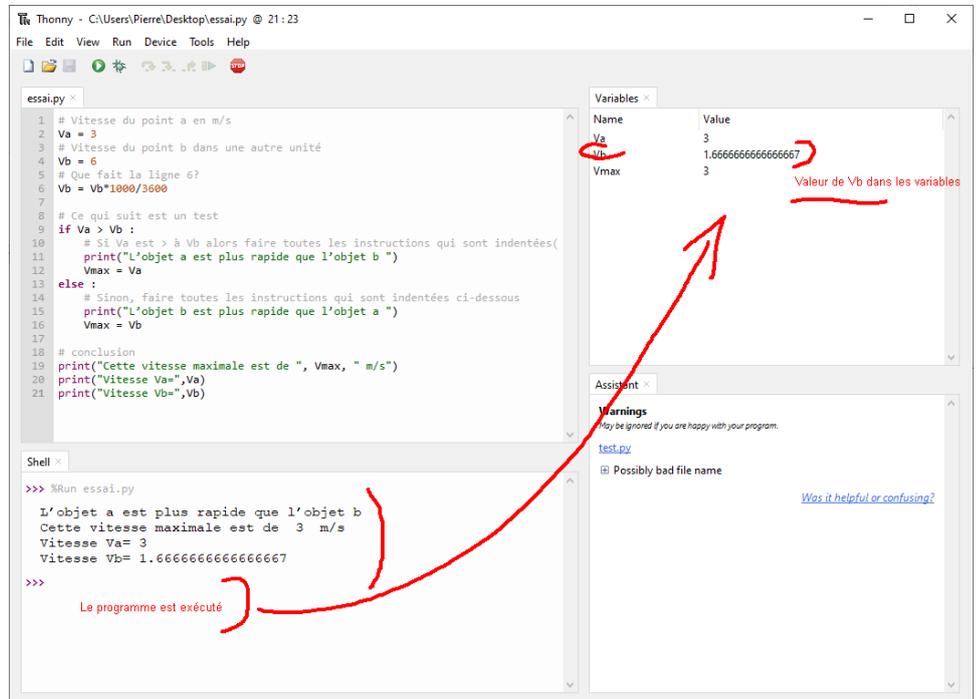
Dans l'éditeur :

- modifier le script pour afficher les valeurs de Va et Vb finales !

8.2.1.2 - Un exemple de correction

Rq :

- A la fin de l'exécution du programme, la valeur Vb = 6 n'existe plus dans les variables car...



8.2.2 - Exo – Plusieurs tests if / elif

8.2.2.1 - Énoncé



Étude des solutions des fonctions du second degré. Pour la théorie voir ici si besoin :
<https://www.maths-et-tiques.fr/telech/Secondegre2ESL.pdf>

En dessous nous étudions :

$$f(x) = 2x^2 + 2x - 12$$

donc
(nous en déduisons que)

$$a = 2,$$
$$b = 2 \text{ et}$$
$$c = -12$$

Attention, vous avez besoin de la **racine carrée** dans cet exercice donc dans l'état actuel de nos connaissances je rappelle que $\sqrt[2]{4} = 4^{0.5} = 2$ donc pour le calcul de la racine nous utiliserons la **puissance 0,5** !

Attention ceci n'est pas un script mais un algorithme

Mon premier algorithme

Début du programme

Données

Saisir les valeurs de a, b, c (paramètres)

Calculs

Calculer delta = $b^2 - 4ac$

Test(s)

Si delta < 0 alors :

 signe ← "négatif"

 Afficher "Pas de solution dans IR"

sinon :

 Si delta = 0 alors :

 signe ← "nul"

 Calculer $x_0 = -b/2a$

 Afficher x_0

 sinon :

 signe ← "positif"

 Calculer x_1 qui prend la valeur $(-b - \text{racine}(\text{delta}))/2a$

 Calculer x_2 qui prend la valeur $(-b + \text{racine}(\text{delta}))/2a$

 Afficher x_1 et x_2

 Fin si

Fin si

Affichage finale

Afficher le texte Delta est égal à +delta

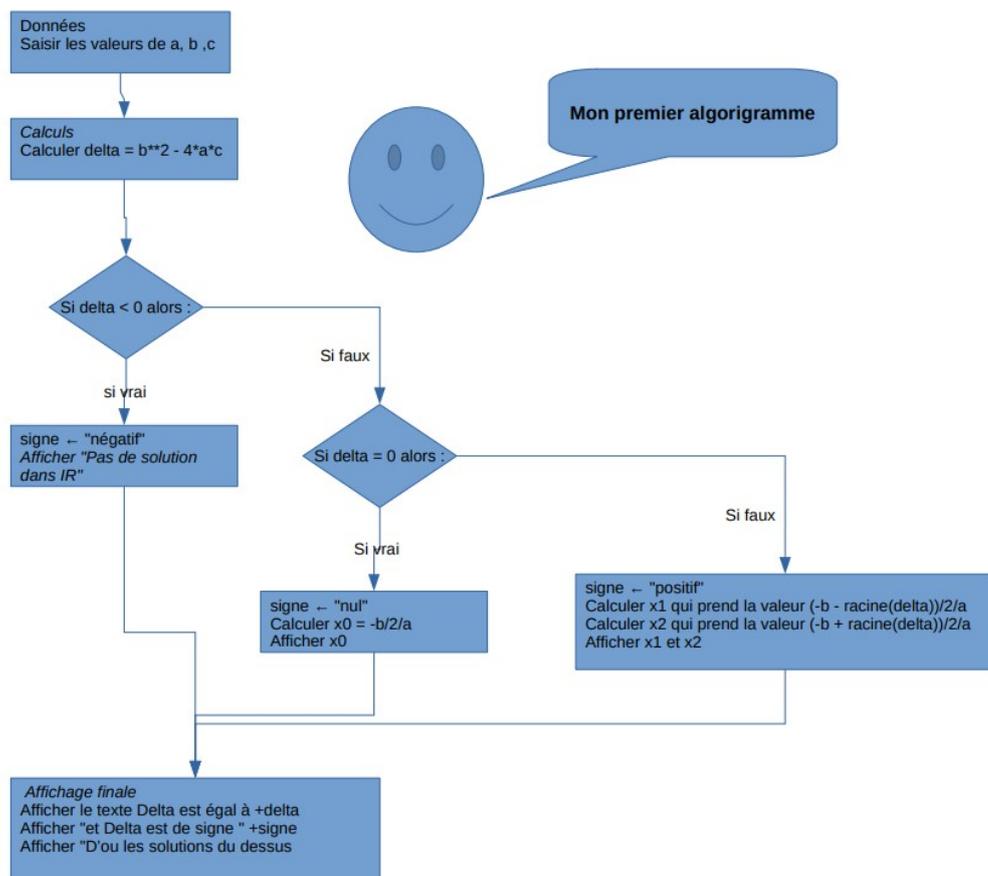
Afficher "et Delta est de signe " +signe

Afficher "D'ou les solutions du dessus

1er bloc
conditionnel

2ème bloc

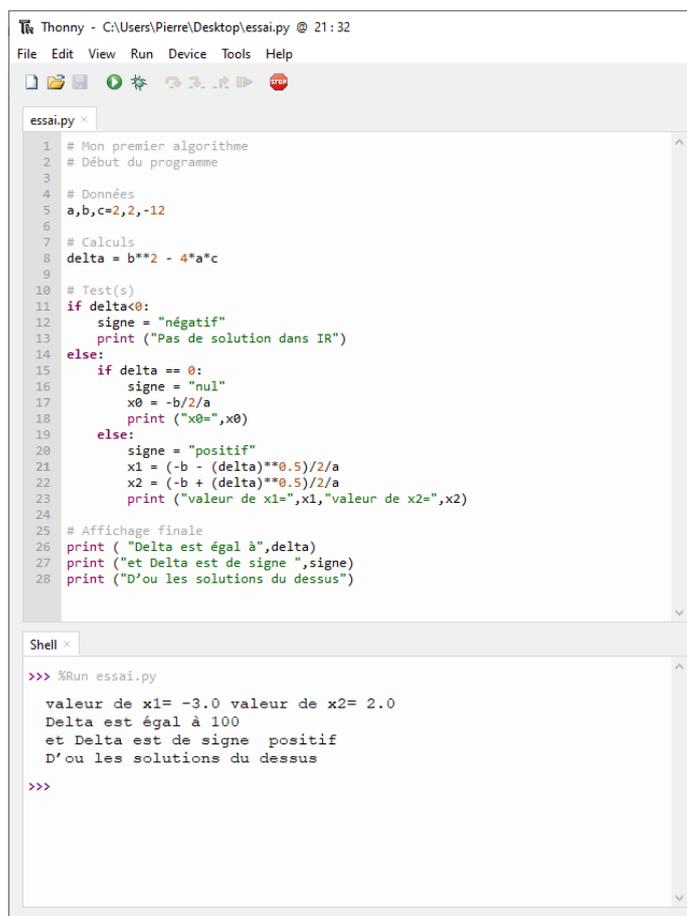
Attention ceci n'est pas un script mais un algorithme



Dans l'éditeur :

- Écrire le script correspondant en utilisant 2 fonctions IF (la seconde est imbriquée dans la première)
- Écrire un autre script avec 1 fonction IF et ELIF

8.2.2.2 - Deux exemples de corrections

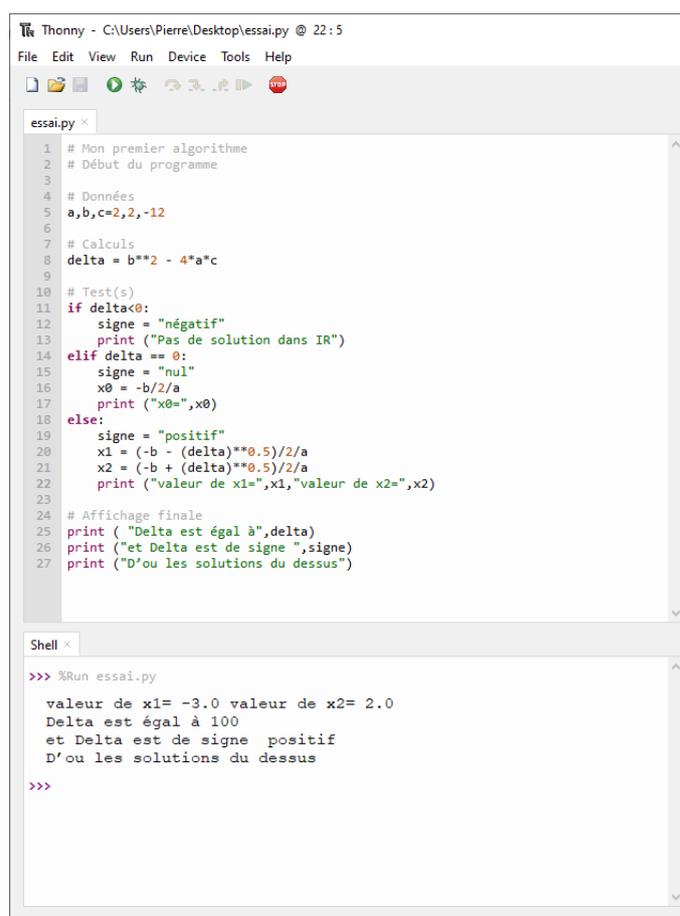


```
Thonny - C:\Users\Pierre\Desktop\essai.py @ 21:32
File Edit View Run Device Tools Help

essai.py x
1 # Mon premier algorithme
2 # Début du programme
3
4 # Données
5 a,b,c=2,2,-12
6
7 # Calculs
8 delta = b**2 - 4*a*c
9
10 # Test(s)
11 if delta<0:
12     signe = "négatif"
13     print ("Pas de solution dans IR")
14 else:
15     if delta == 0:
16         signe = "nul"
17         x0 = -b/2/a
18         print ("x0=",x0)
19     else:
20         signe = "positif"
21         x1 = (-b - (delta)**0.5)/2/a
22         x2 = (-b + (delta)**0.5)/2/a
23         print ("valeur de x1=",x1,"valeur de x2=",x2)
24
25 # Affichage finale
26 print ("Delta est égal à",delta)
27 print ("et Delta est de signe ",signe)
28 print ("D'où les solutions du dessus")

Shell x
>>> %Run essai.py
valeur de x1= -3.0 valeur de x2= 2.0
Delta est égal à 100
et Delta est de signe positif
D'où les solutions du dessus
>>>
```

Figure 1: 2 fonctions {SI + SINON} imbriquées



```
Thonny - C:\Users\Pierre\Desktop\essai.py @ 22:5
File Edit View Run Device Tools Help

essai.py x
1 # Mon premier algorithme
2 # Début du programme
3
4 # Données
5 a,b,c=2,2,-12
6
7 # Calculs
8 delta = b**2 - 4*a*c
9
10 # Test(s)
11 if delta<0:
12     signe = "négatif"
13     print ("Pas de solution dans IR")
14 elif delta == 0:
15     signe = "nul"
16     x0 = -b/2/a
17     print ("x0=",x0)
18 else:
19     signe = "positif"
20     x1 = (-b - (delta)**0.5)/2/a
21     x2 = (-b + (delta)**0.5)/2/a
22     print ("valeur de x1=",x1,"valeur de x2=",x2)
23
24 # Affichage finale
25 print ("Delta est égal à",delta)
26 print ("et Delta est de signe ",signe)
27 print ("D'où les solutions du dessus")

Shell x
>>> %Run essai.py
valeur de x1= -3.0 valeur de x2= 2.0
Delta est égal à 100
et Delta est de signe positif
D'où les solutions du dessus
>>>
```

Figure 2: Fonction {SI + SINON-SI + SINON}

8.3 - Vos notes

Rq :

Pour info j'ai copier / coller l'algorithme dans Thonny puis modifier les commandes et les indentations.

Après j'ai lancé le programme avec Run et j'ai constaté que mon programme contenait une erreur ligne 11. J'ai corrigé l'erreur puis relancé et découverte d'une autre erreur... Je me suis fait avoir 4 fois;)).

L'intérêt de la programmation, c'est aussi ça : La démarche essai – erreur ! [Plus d'info](#)

Pour en finir avec HAL : maintenant il peut prendre des décisions. Dans le film "[2001 l'odyssée de l'espace](#)" elles furent fatales !

9 - Boucle bornée - les tâches répétitives mais ayant une fin

A partir d'ici on va réellement commencer à s'amuser car avec les boucles, un programme peut répéter très rapidement une série de tâche TRÈS rapidement – moi ça me laisse pantois !

La boucle `for ... in ...` : est dite bornée car elle sa fin est connue et programmée => d'où la notion de boucle bornée.

```
for i in range(10):  
    print(i,"au carré donne ",i**2)  
    print(i,"au cube donne ",i**3)  
print("Au dessus les carrés et les cubes pour i allant de 0 à 9 !")
```

9.1 - Commandes

Commande ou Symbole	Opération
<code>for</code>	Pour ...
<code>in</code>	... Allant
<code>:</code>	Début du bloc après les :
indentation	4 espaces
<code>len()</code> <code>range()</code> ou <code>range(start, stop, step)</code>	Compte les éléments d'une liste Renvoi une série de valeurs entières

9.2 - Exercices

9.2.1 - Exo – for et suite

9.2.1.1 - Énoncé



L'idée ici, dans l'état de nos connaissances, est de s'amuser à étudier les suites arithmétiques et géométriques en utilisant les relations de récurrences. Pour la théorie voir ici :

<https://www.maths-et-tiques.fr/telech/SuitesAG.pdf>

On donne :

Suite arithmétique :

Premier terme $A_0=2$

Raison $r=3$

Relation de recurrence $A_n = A_{n-1} + r$

Suite géométrique :

Premier terme $G_0=2$

Raison $q=3$

Relation de recurrence $G_n = G_{n-1} * q$

Notre objectif est d'étudier les **50** premiers termes de ces deux suites et d'utiliser notre niveau actuel de connaissance en python

Dans l'éditeur :

copier/coller le programme du dessous puis exécuter le programme.

```
# Suite arithmétique et géométrique
# Premiers termes
A=2
G=2
# Raisons
r=2
q=2

# Boucle pour répéter les tâches
for i in range(50):
    print(i,"-",A,"-",G)
    A=A+r
    G=G*q

# Affichage final
print(i)
print("Valeur finale de S. arithmétique=",A)
print("Valeur finale de S. géométrique=",G)
```

```
essai.py x
1 # Suite arithmétique et géométrique
2 # Premiers termes
3 A=2
4 G=2
5 # Raisons
6 r=2
7 q=2
8
9 # Boucle pour répéter les tâches
10 for i in range(50):
11     print(i,"-",A,"-",G)
12     A=A+r
13     G=G*q
14 |
15 # Affichage final
16 print(i)
17 print("Valeur finale de S. arithmétique=",A)
18 print("Valeur finale de S. géométrique=",G)
```

Mon shell me donne →

Comprends les 4 dernières lignes !

- Le 49, c'est la dernière valeur de i qui est parti de 0 pour attendre 50-1
- J'attaque la ligne 10 avec cette dernière valeur
- Le print de la ligne 11 correspond bien alors à 49 - 100 - 1125899906842624
- Je traite la ligne 12 et 13 - je calcule les valeurs **suivantes** de A et G. Puis je sors de la boucle et traite les lignes 16-17-18 => donc les 3 dernières lignes du Shell sont logiques !!!

49 (c'est bien ma dernière valeur de i)

Valeur finale de S. arithmétique= 102 (ici c'est bien la dernière valeur calculée)

Valeur finale de S. géométrique= 2251799813685248 (idem)

```
Shell x
33 - 68 - 17179869184
34 - 70 - 34359738368
35 - 72 - 68719476736
36 - 74 - 137438953472
37 - 76 - 274877906944
38 - 78 - 549755813888
39 - 80 - 1099511627776
40 - 82 - 2199023255552
41 - 84 - 4398046511104
42 - 86 - 8796093022208
43 - 88 - 17592186044416
44 - 90 - 35184372088832
45 - 92 - 70368744177664
46 - 94 - 140737488355328
47 - 96 - 281474976710656
48 - 98 - 562949953421312
49 - 100 - 1125899906842624
49 16
Valeur finale de S. arithmétique= 102 17
Valeur finale de S. géométrique= 2251799813685248 18
>>>
```

n° des lignes dans l'éditeur

Dans l'éditeur :

- modifier ce programme pour que tous les termes des suites A et G soient dans des listes !

Pour vous aider j'ai dans l'idée d'initialiser 2 listes ListeA=[] et ListeB=[] puis avec la méthode append() j'ajoute mes éléments...

Votre script que vous collez ici avec une extrême fierté

9.2.1.2 - Un exemple de correction

The screenshot shows the Thonny Python IDE with a file named 'essai.py'. The code in the editor is as follows:

```
1 # Suite arithmétique et géométrique
2 # Premiers termes
3 A=2
4 G=2
5 # Raisons
6 r=2
7 q=2
8
9 # Listes contenant les premiers termes
10 ListeA=[A]
11 ListeG=[G]
12
13 # Boucle pour répéter les tâches
14 for i in range(50):
15     print(i,"-",A,"-",G)
16     A=A+r
17     G=G*q
18     ListeA.append(A)
19     ListeG.append(G)
20
21 # Affichage final
22 print(i)
23 print("Valeur finale de S. arithmétique=",A)
24 print("Valeur finale de S. géométrique=",G)
```

Handwritten red annotations include arrows pointing to lines 10-11 and 14-19, and a large note: "Boucle 50 fois !!".

The Variables panel on the right shows the following state:

Name	Value
A	102
G	2251799813685248
ListeA	[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50]
ListeG	[2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384, 32768, 65536, 131072, 262144, 524288, 1048576, 2097152, 4194304, 8388608, 16777216, 33554432, 67108864, 134217728, 268435456, 536870912, 1073741824, 2147483648, 4294967296, 8589934592, 17179869184, 34359738368, 68719476736, 137438953472, 274877906944, 549755813888, 1099511627776, 2199023255552, 4398046511104, 8796093022208, 17592186044416, 35184372088832, 70368744177664, 140737488355328, 281474976710656, 562949953421312, 1125899906842624, 2251799813685248]
i	49
q	2
r	2

The Assistant panel shows a warning: "Possibly bad file name".

The Shell panel shows the output of the program:

```
38 - 78 - 549755813888
39 - 80 - 1099511627776
40 - 82 - 2199023255552
41 - 84 - 4398046511104
42 - 86 - 8796093022208
43 - 88 - 17592186044416
44 - 90 - 35184372088832
45 - 92 - 70368744177664
46 - 94 - 140737488355328
47 - 96 - 281474976710656
48 - 98 - 562949953421312
49 - 100 - 1125899906842624
49
Valeur finale de S. arithmétique= 102
Valeur finale de S. géométrique= 2251799813685248
>>>
```

Rq :

Oui je sais... ListeA et ListeG contiennent 1 terme en trop;)

9.2.2 - Exo – for et somme de série

9.2.2.1 - Énoncé



Un prof de maths nous donne :

Soit pour tout entier $n \geq 1$, la suite (u_n) définie par :

$$u_n = \sum_{k=1}^n \frac{1}{k^2}$$

On a montré que cette suite était convergente.

L'objectif de cet exercice est de valider les affirmations de la remarque historique.

Remarque historique

En mathématiques, le problème de Bâle (ou problème de Mengoli) est un problème qui consiste à demander la valeur de la somme de la série des inverses de carrés des entiers (non nuls).

Le problème a été résolu par le génial mathématicien suisse Leonhard Euler (1707-1783) qui parvient à démontrer que cette somme tend vers $\frac{\pi^2}{6}$. Il en donna la première démonstration rigoureuse en 1741 mais annonce en 1735 la découverte de la somme exacte.

$$1 + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \frac{1}{5^2} + \dots = \sum_{k=1}^{+\infty} \frac{1}{k^2} = \frac{\pi^2}{6} \quad 150\,000$$

Pour obtenir 4 décimales exactes, il faut additionner plus de ~~15 000~~ termes de la somme. Avec 1000 termes, on n'obtient que 2 décimales et la fraction irréductible comporte déjà plus de 800 chiffres. Cela reste rêveur quand on pense qu'Euler a calculé 20 décimales exactes (mais avec des méthodes d'accélération de convergence).

Dans l'éditeur :

- écrivez un script permettant de vérifier que cette somme nécessite non pas 15 000* mais **150 000** termes pour atteindre une précision exacte à 4 décimale. Montrez moi que vous êtes brillants;)

Je donne $\frac{\pi^2}{6} = 1,6449340668\dots$

* Une erreur historique ? Je penche plutôt pour un coquille dans l'impression

Mes pistes : une boucle, 150000 tours, une valeur que j'écrase...

Votre script que vous collez ici avec une extrême fierté (même s'il est court, car il est vôtre ! - PT)

9.2.2.2 - Un exemple de correction

```
Thonny - C:\Users\Pierre\Desktop\essai.py @ 5:13
File Edit View Run Device Tools Help

essai.py x
1 # Problème de Bâle
2 bale=0
3 for i in range(150000):
4     bale=1/(i+1)**2+bale
5     print(i,"-",bale)

Shell x
>>> %Run essai.py
149999 - 1.6449274002037997
>>>

Variables x
Name Value
bale 1.6449274002037997
i 149999

Assistant x
Warnings
May be ignored if you are happy with your program.
test.py
```

Rq :

- Ligne 4 j'ai besoin de bale pour calculer bale !!! donc je l'initialise ligne 2
- Je vous déconseille de mettre le print de la ligne 5 dans la boucle, car afficher 150000 valeurs ça prend du temps ! Ceci dit testez quand même, vous mourrez moins bêtes;)
- Mon script répond bien à la question car si $i=149999$ donc ma boucle à tournée 150000 fois et mes 4 premières décimales 1,6449... correspondent bien à $\frac{\pi^2}{6}=1,6449340668...$
- Vous avez là un script non commenté... vous voyez que les commentaires sont utiles!;

9.2.3 - Exo – for, listes, gestion des indices et append

9.2.3.1 - Énoncé



Étude cinétique d'un objet :

Vous travaillez à l'observatoire du [Pic du midi](#) dans le cadre de votre thèse post-doc en tant qu'[astronome](#). Le 26 décembre 2030, au télescope, vous découvrez dans l'espace proche de Jupiter un parallépipède rectangle de proportion 1:4:9 (les 3 premiers nombres entiers au carré) semblant suivre une trajectoire rectiligne. Cela vous intrigue !

Vous mesurez alors les positions de son centre de gravité (arbitrairement vous placez l'origine des x à $t=0$) et vous obtenez les mesures suivantes :

t en s	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
x en m	0	0,05	0,2	0,45	0,8	1,25	1,8	2,45	3,2	4,05	5

Comme il se doit, vous rendez compte à votre responsable de recherche, de l'ancienne école, qui ne maîtrise pas de langage de programmation. Alors celui-ci vous demande :

- "Dites donc, à l'occasion, vous ne pourriez pas me faire une petite étude cinétique de ce truc ?"

Après un instant de réflexion, il ajoute :

- "En restant dans le cadre de la mécanique Newtonienne, hein ! On va rester simple !"

Vous souriez et répondez :

-"OK, je me lance ! Il me faut juste retrouver mes cours de lycée. Je me souviens avoir fait un exercice de ce genre en python en 2020, vous savez, la première année covid !

Cela vous irait ?"

Votre responsable retire ses lunettes pour les nettoyer à l'aide d'un bout de tee-shirt dépassant du short. Sans vous répondre, les yeux dans le vague, il fait demi-tour, suivant ses tongs.

Vous interprétez son départ comme un accord, convaincu d'être à l'aube d'une découverte fondamentale. En pénétrant dans votre modeste bureau – songeur – vous trouvez quand même étrange ces tongs en plein hiver à 2800m d'altitude...

Bon, c'est pas tout, mais il faut s'y mettre :

On donne :

$$\left| \begin{array}{l} \text{Vitesse moyenne} \\ v[i] = \frac{x[i+1] - x[i-1]}{t[i+1] - t[i-1]} \end{array} \right| \left| \begin{array}{l} \text{Accélération moyenne} \\ a[i] = \frac{v[i+1] - v[i-1]}{t[i+1] - t[i-1]} \end{array} \right|$$

Et on rappelle :

indice i	0	1	2	3	4	5	6	7	8	9	10
t en s	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
x en m	0	0,05	0,2	0,45	0,8	1,25	1,8	2,45	3,2	4,05	5

Dans l'éditeur :

- copier/coller le script du dessous afin de vous éviter des erreurs de saisie des valeurs. Vous pouvez le tester pour comprendre les len() si besoin.

```
# Etude cinétique d'un objet
# t pour le temps en s
# x(t) pour les mesures en m
# ex pour t=0.3s alors x(0.2)=0.45m
t=[0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1]
x=[0,0.05,0.2,0.45,0.8,1.25,1.8,2.45,3.2,4.05,5]

# nombre de valeurs dans les séries
nt=len(t)
nx=len(x)
```

- compléter le script pour obtenir les valeurs manquantes du tableau du dessous. Les cellules noires resteront sans valeurs ! (En vous grattant la tête vous avez la réponse!). Ces valeurs seront enregistrées respectivement dans une liste de nom v pour la vitesse et a pour l'accélération. Vous afficherez les valeurs de v et de a !

indice i	0	1	2	3	4	5	6	7	8	9	10
t en s	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
x en m	0	0,05	0,2	0,45	0,8	1,25	1,8	2,45	3,2	4,05	5
v en m/s											
a en m/s ²											

- pour vous aider à ne pas tricher, ma piste:
 - comprendre que pour v je calcule 11-2=9 valeurs et pas 11 et que pour a je calcule 9-2=7 valeurs et pas 9 => je dois probablement faire 2 boucles
 - création de listes vides
 - méthode v.append(delta_x/delta_t) et idem après adaptation pour a
 - n'étant pas sûr de mon coup, j'ai glissé un print(i) pour vérifier ma logique dans les boucles

9.2.3.2 - Un exemple de correction

Rq :

- Je dois comprendre la ligne 18 pour les vitesses
- Je dois aussi comprendre la ligne 23 ;-)

Le surlendemain, vous retournez voir votre responsable, chaussé de nouvelles tongs – achetés la veille – le modèle avec des marguerites en plastique – vous trouvez ce modèle vintage mais vous êtes nostalgiques de ces fleurs d'un autre temps.

Vous lui dites :

- "Ça y est, j'ai trouvé, c'est fantastique, c'est le [premier contact](#), regardez !"

Vous lui montrez les valeurs des vitesses et accélérations la main tremblante car vous avez un peu froid aux pieds.

```

1 # Etude cinétique d'un objet
2 # t pour le temps en s
3 # x(t) pour les mesures en m
4 # ex pour t=0.3s alors x(0.2)=0.45m
5 t=[0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1]
6 x=[0,0.05,0.2,0.45,0.8,1.25,1.8,2.45,3.2,4.05,5]
7
8 # nombre de valeurs dans les séries
9 nt=nt-1 valeurs
10 nt=len(t)
11 nx=len(x)
12
13 # creation d'une liste vide pour vitesse et acceleration
14 v=[]
15 a=[]
16
17 # boucle pour v allant de 1 à 10 (pour les cases noires)
18 for i in range(1,nt-1):
19     # print(i) print pour comprendre si vous décommentez
20     v.append((x[i+1]-x[i-1])/(t[i+1]-t[i-1]))
21
22 # autre boucle pour a car ...
23 for i in range(1,len(v)-1):
24     # print(i) print pour comprendre
25     a.append((v[i+1]-v[i-1])/(t[i+1]-t[i-1]))
26
27 # Affichage
28 print("vitesse",v)
29 print()
30 print("acceleration",a)
    
```

Variables

Name	Value
a	[10.0000000000000002, 9.999999999999999]
i	7
nt	11
nx	11
t	[0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]
v	[1.0, 2.0000000000000004, 3.0000000000000004, 4.0, 5.000000000000001, 6.000000000000001, 7.000000000000001, 8.000000000000001, 9.000000000000001, 10.000000000000001, 11.000000000000001]
x	[0, 0.05, 0.2, 0.45, 0.8, 1.25, 1.8, 2.45, 3.2, 4.05, 5]

Warnings

test.py
Possibly bad file name

Shell

```

>>> %Run essai.py
vitesse [1.0, 2.0000000000000004, 3.0000000000000004, 4.0, 5.000000000000001, 6.000000000000001, 7.000000000000001, 8.000000000000001, 9.000000000000001, 10.000000000000001, 11.000000000000001]
acceleration [10.000000000000002, 9.999999999999999, 9.999999999999999, 10.000000000000002, 10.000000000000012, 9.999999999999999, 9.999999999999966, 10.000000000000014]
    
```

Votre responsable reste songeur, son regard pointant sur votre gros orteil gauche – ou est-ce la marguerite qui l'intrigue ?

- "Mais regardez ! Ce monolithe est uniformément accéléré, il se déplace rectilignement, près de Jupiter ! C'est Newtonement impossible !!! Cette objet est vivant ! Ou alors piloté !

9.2.4 - Exo – for, liste et calculs

9.2.4.1 - Énoncé



Dans l'éditeur :

- copier/colle le code suivant

```
# A commenter
masse=[1.2,2,3.5,7.8,9.8,15.4,18,25]

# A commenter
poids=[]

# A commenter
for m in masse:
    poids.append(9.81*m) # A commenter

# A commenter
print(poids)
```

- après exécution, commentez le !

Rq : Avant de commenter ou de téléphoner au professeur Moustache (et pour mourir moins bête), vous devriez peut-être glisser un `print(m)` dans la boucle juste au dessus du `poids.append(;`). Je dis ça, je dis rien

9.3 - Vos notes

Bientôt les vacances, d'ou les tongs ;-))))))

Vous pouvez inventer la suite;)

10 - Boucle non bornée – les tâches répétitives dont je ne connais pas le terme

Dans la vie courante on ne connaît pas toujours le terme d'une tâche répétitive. Par exemple en EPS vous pouvez vous dire :

```
Faire 10 tours de piste :  
    compter mes foulées  
Total de mes foulées.
```

Dessin 1: Boucle bornée IF

Mais vous pourriez vouloir étudier votre endurance. Il serait alors plus stratégique de penser ceci :

```
Energie initiale  
Tant que energie sup à 0 :  
    compter mes foulées  
    energie = energie - Delta  
Total de mes foulées.  
Total des tours de piste
```

Dessin 2: Boucle non bornée WHILE

Le principe de la boucle non bornée est le même que celui de la boucle bornée ! Une petite remarque cependant, avec un FOR tôt ou tard ça s'arrête ! Avec un WHILE ce n'est pas certain...



Le bouton STOP est parfois utile !

10.1 - Commandes

Commande ou Symbole	Opération
while	Tant que ...
in	... Allant
:	Début du bloc après les :
indentation	4 espaces

10.2 - Exercices

10.2.1 - Exo – while

10.2.1.1 - Énoncé



Je ne sais pas vous, mais moi j'ai un goût d'inachevé depuis l'exercice 9.2.2.1

Je rappelle le problème de Bâle :

$$1 + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \frac{1}{5^2} + \dots = \sum_{k=1}^{+\infty} \frac{1}{k^2} = \frac{\pi^2}{6} \quad \text{avec} \quad \frac{\pi^2}{6} = 1,6449340668\dots$$

Dans l'éditeur :

- écrire un programme permettant de savoir pour quelle valeur de k cette somme atteint une précision à 4 décimale – donc dépasse tout juste 1,6449 ?

Votre script que vous collez ici avec une extrême fierté

10.2.1.2 - Un exemple de correction

The screenshot shows the Thonny IDE interface. The main editor displays a Python script named 'essai.py' with the following code:

```
1 # valeurs initiales
2 bale=0
3 borne = 1.6449
4 i=1
5
6 # tant que la somme bale est inf à la borne
7 while bale < borne:
8     bale=1/i**2+bale # j'ajoute à bale le terme suivant
9     i=i+1
10
11 # affichage
12 print(i)
```

The Shell window shows the command `>>> %Run essai.py` and the output `29355`. The Variables panel on the right shows the current state of the program:

Name	Value
bale	1.6449000005210916
borne	1.6449
i	29355

Handwritten red annotations include a smiley face and "yes!!" next to the variable 'i' in the Variables panel, and a red arrow pointing from the 'print(i)' statement to the output '29355' in the Shell.

Rq :

- Noter mon manque de rigueur, il est question de k dans le sujet de maths et moi je le nomme i par habitude !
- Bon, ce n'est pas les 15 000 termes annoncés dans l'exo, mais on est loin des 150 000 ! Il **semblerait** que **29355** termes suffisent ! **mais j'ai un doute !?**

10.2.2 - Exo – while et listes

10.2.2.1 - Énoncé



Dans l'éditeur :

- copier/coller ce script puis l'exécuter !

```
# valeurs initiales
bale=0
borne = 1.6449
i=1

hist=[]

# tant que la somme bale est inf à la borne
while bale < borne:
    bale=1/i**2+bale # j'ajoute à bale le terme suivant
    i=i+1
    hist.append(bale)

# affichage
print (len(hist)) # nombre de terme dans ma liste "historique des valeurs"
print (hist[len(hist)-2]) # les indice d'une liste = 0 ---> nbre de terme -1 donc ici avt derniere valeur
print (hist[len(hist)-1]) # meme logique donc derniere valeur
print (hist[len(hist)]) # meme logique donc problème
```

The screenshot shows a Python IDE window titled 'Thonny - C:\Users\Pierre\Desktop\essai.py © 5:1'. The code editor contains the script from the previous block. The Shell window shows the output of the script:

```
>>> %Run essai.py
29354
1.64489999993605375
1.64490000005210916
Traceback (most recent call last):
  File "C:\Users\Pierre\Desktop\essai.py", line 18, in <module>
    print (hist[len(hist)]) # meme logique donc problème
IndexError: list index out of range
>>>
```

Handwritten red annotations include:

- Mathematical series: $\frac{1}{1^2} + \frac{1}{2^2} + \frac{1}{3^2} + \dots$
- 'OK !!' with an arrow pointing to the list in the Variables window.
- 'dernier terme !! donc le 29555ème' with an arrow pointing to the last value in the Shell output.
- 'Re 7es' with a smiley face.

The Variables window shows the state of the program:

Name	Value
bale	1.644900005210916
borne	1.6449
hist	[1.0, 1.25, 1.3611111111111112, 1.4236111111111112, 1.4636111111111112, 1.4913888888888889, 1.511797...
i	29355

The Assistant window shows an error message: 'IndexError: list index out of range' at 'essai.py, line 18'. The Assistant also provides suggestions: 'Let Thonny developers know' and 'Search the web'. The Warnings window shows a warning: 'Possibly bad file name'.

- expliquez le code,
- puis conclure en levant mon doute;)

10.3 - Vos notes

Si besoin...

11 - Les fonctions – pour faire comme en maths

Lorsqu'une tâche doit être réalisée plusieurs fois par un programme avec seulement des paramètres différents, on peut l'isoler au sein d'une fonction. Cela permet de simplifier son code et de le rendre plus lisible. Cette approche est également intéressante si la personne qui définit la fonction est différente de celle qui l'utilise.

- Python propose des fonctions intégrées comme `print()` par exemple.
- Python propose aussi des bibliothèques regroupant des fonctions par thématique, nous verrons cela plus tard.
- Et Python vous permet aussi de créer vos fonctions...

Un programme ne contenant qu'une fonction ne fait rien, il attend probablement Godot (remarque uniquement pour les lecteurs avertis;)

11.1 - Commandes

Commande ou Symbole	Opération
<code>def</code>	Abréviation de define en EN = définition en FR
<code>return</code>	Retourne le résultat de la fonction (si pas de paramètres alors pas de return)
<code>:</code>	comme d'hab
indentation	idem
<code>input()</code>	Demande de saisie – la valeur saisie sera de type texte
<code>int()</code>	Transforme le type d'une variable en valeur numérique entière
<code>float()</code>	Transforme le type d'une variable en décimale

11.2 - Exercices

11.2.1 - Exo – def fonction à une variable

11.2.1.1 - Énoncé



Soit la fonction définie pour tout $x \in \mathbb{R} - \{-3\}$ d'équation $f(x) = \frac{2x^4 - \frac{5}{7}}{x+3}$

```
# definition de la fonction f de variable x
def f(x):
    val=(2*x**4-5/7)/(x+3)
    return val

# pour comprendre affichage de f(0) = -5/7/3
print(f(0))
```

Dans l'éditeur :

- compléter ce script pour calculer et mémoriser dans une liste que vous nommerez point tous les couples de points $(x, f(x))$ pour x allant de -10 à 10.
Évidemment il faudra glisser un si pour éviter la valeur $x=-3$!

Bonne chance... Pour vous aider ma piste : Je pense avoir besoin de point[], d'une boucle for, de si et d'append...

```
# Votre premier script sans aide aucune
```

11.2.1.2 - Un exemple de construction par essai – erreur (mais sans les erreurs;)

Étape 1 :

Obtenir les x dans le bon intervalle

```
1 # definition de la fonction f de variable x
2 def f(x):
3     val=(2*x**4-5/7)/(x+3)
4     return val
5
6 # boucle pour avoir des x de -10 à 10
7 for x in range (-10,11):
8     print(x)
```

Étape 2 :

Éliminer le x que je ne dois pas calculer sous peine de message d'erreur

A ce stade, je pense que c'est gagné ! Ne me reste plus qu'à appliquer la méthode .append sous le else :

```
1 # definition de la fonction f de variable x
2 def f(x):
3     val=(2*x**4-5/7)/(x+3)
4     return val
5
6 # boucle pour avoir des x de -10 à 10
7 for x in range (-10,11):
8     # le si pour supprimer la valeur interdite
9     if x==-3:
10        x+1
11    else:
12        print(x)
```

Etape 3 :

Ligne 7 : Création d'un liste vide

Ligne 15 : Ajout des valeurs

Ligne 16 : C'est pour afficher les couples

Ligne 18 : C'est pour afficher la liste

```
1 # definition de la fonction f de variable x
2 def f(x):
3     val=(2*x**4-5/7)/(x+3)
4     return val
5
6 # déclaration de ma liste vide pour mémo les points
7 point=[]
8
9 # boucle pour avoir des x de -10 à 10
10 for x in range (-10,11):
11     # le si pour supprimer la valeur interdite
12     if x==-3:
13        x+1
14     else:
15        point.append([x,f(x)])
16        print(x,"-",f(x))
17
18 print(point)
```

Et j'obtiens cela

```
Shell x
>>> %Run essai.py
-10 - -2857.0408163265306
-9 - -2186.8809523809523
-8 - -1638.257142857143
-7 - -1200.3214285714287
-6 - -863.7619047619047
-5 - -624.6428571428571
-4 - -511.2857142857143
-2 - 31.285714285714285
-1 - 0.6428571428571428
0 - -0.2380952380952381
1 - 0.3214285714285714
2 - 6.257142857142857
3 - 26.88095238095238
4 - 73.04081632653062
5 - 156.16071428571428
6 - 287.92063492063494
7 - 480.1285714285715
8 - 744.6623376623377
9 - 1093.4404761904761
10 - 1538.4065934065934
[[-10, -2857.0408163265306], [-9, -2186.8809523809523], [-8, -1638.257
142857143], [-7, -1200.3214285714287], [-6, -863.7619047619047], [-5,
-624.6428571428571], [-4, -511.2857142857143], [-2, 31.285714285714285
], [-1, 0.6428571428571428], [0, -0.2380952380952381], [1, 0.321428571
4285714], [2, 6.257142857142857], [3, 26.88095238095238], [4, 73.04081
632653062], [5, 156.16071428571428], [6, 287.92063492063494], [7, 480.
1285714285715], [8, 744.6623376623377], [9, 1093.4404761904761], [10,
1538.4065934065934]]
>>>
```

11.2.2 - Exo – def fonction avec plusieurs variables, input() et int()

11.2.2.1 - Énoncé



Une fonction polynôme de d°2 est toujours du type $f(x) = a * x^2 + b * x + c$
avec $a \neq 0$ et $x \in \mathbb{R}$

Dans l'éditeur :

- dans l'état actuel de nos connaissances nous sommes capables de créer un programme me permettant de :
 - définir un intervalle $[x_{min}, x_{max}]$ quelconque et de calculer en fonction du nombre de points désirés le pas entre chaque x par la relation $pas = \frac{(x_{max} - x_{min})}{(N_{nbre\ de\ points} - 1)}$
 - définir f avec les variables x,a,b,c pour envisager tous les cas de l'univers
 - calculer en fonction du pas les couples de points $[x, f(x)]$ et d'en garder la trace dans une liste

Votre script

11.2.2.2 - Un exemple de construction par essai – erreur

Étape 1 :

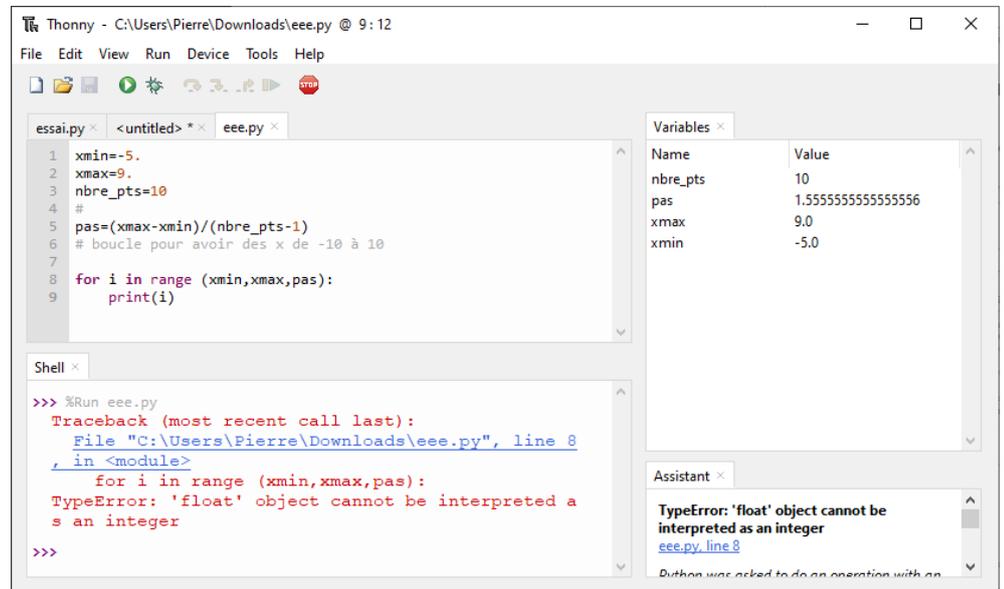
Créer une boucle affichant les valeurs des x désirées. Je déclare des valeurs arbitraires pour xmin, xmax, nbre de points

Essai n°1 :

Utilisation de la commande
range(min, max, pas)

=>

Ne fonctionne pas ligne 8 car i est de type "int" et la valeur de mon pas est "float"



The screenshot shows the Thonny IDE interface. The main editor displays the following Python code:

```
1 xmin=-5.  
2 xmax=9.  
3 nbre_pts=10  
4 #  
5 pas=(xmax-xmin)/(nbre_pts-1)  
6 # boucle pour avoir des x de -10 à 10  
7  
8 for i in range (xmin,xmax,pas):  
9     print(i)
```

The Shell window shows the execution output:

```
>>> %Run eee.py  
Traceback (most recent call last):  
  File "C:\Users\Pierre\Downloads\eee.py", line 8  
    in <module>  
      for i in range (xmin,xmax,pas):  
TypeError: 'float' object cannot be interpreted as an integer  
>>>
```

The Variables window shows the current state of variables:

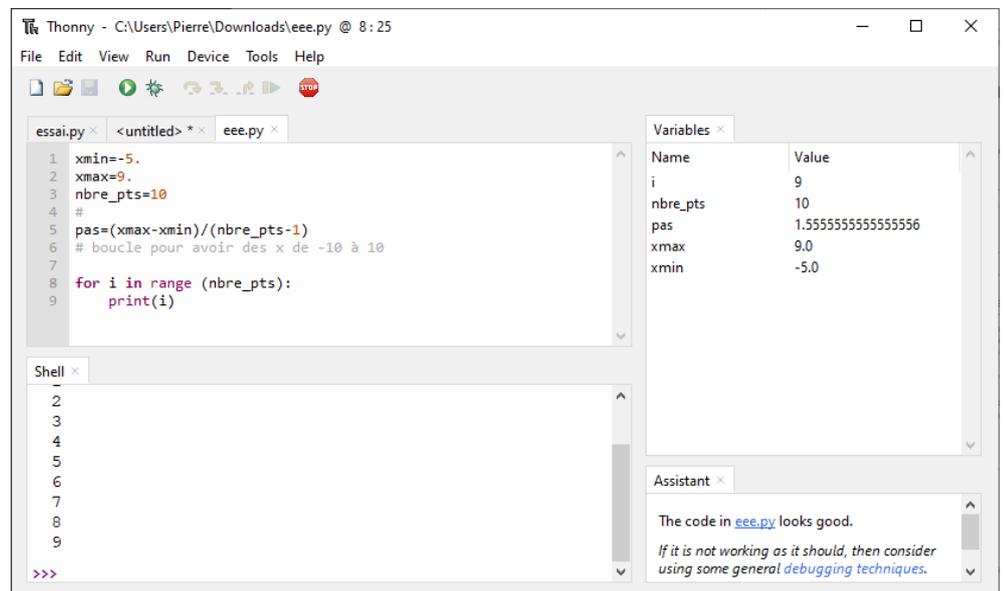
Name	Value
nbre_pts	10
pas	1.5555555555555556
xmax	9.0
xmin	-5.0

The Assistant window displays the error message: "TypeError: 'float' object cannot be interpreted as an integer eee.py, line 8".

Piste n°1 :

Utiliser le nombre de points dans la boucle car lui sera toujours entier !

=> Ok, j'ai des i de 0 à 9 donc j'ai bien 10 valeurs donc 10 points



The screenshot shows the Thonny IDE interface with the corrected Python code:

```
1 xmin=-5.  
2 xmax=9.  
3 nbre_pts=10  
4 #  
5 pas=(xmax-xmin)/(nbre_pts-1)  
6 # boucle pour avoir des x de -10 à 10  
7  
8 for i in range (nbre_pts):  
9     print(i)
```

The Shell window shows the execution output:

```
>>>  
2  
3  
4  
5  
6  
7  
8  
9  
>>>
```

The Variables window shows the current state of variables:

Name	Value
i	9
nbre_pts	10
pas	1.5555555555555556
xmax	9.0
xmin	-5.0

The Assistant window displays a message: "The code in eee.py looks good. If it is not working as it should, then consider using some general debugging techniques."

Essai n°2 :

Calculer tous les x, donc dans la boucle

```
Thonny - C:\Users\Pierre\Downloads\eee.py @ 12:17
File Edit View Run Device Tools Help

essai.py x <untitled> * x eee.py x
1 xmin=-5.
2 xmax=9.
3 nbre_pts=10
4 # calcule de l'intervalle entre chaque x
5 pas=(xmax-xmin)/(nbre_pts-1)
6
7 # initialisation
8 x=xmin
9
10 # boucle pour avoir des x de -10 à 10
11 for i in range (nbre_pts):
12     print(i,"-",x)
13     x=x+pas

Shell x
5 - 2.0000000000000000
6 - 4.3333333333333333
7 - 5.8888888888888888
8 - 7.4444444444444444
9 - 9.0
>>>
```

Variables x

Name	Value
i	9
nbre_pts	10
pas	1.5555555555555555
x	10.555555555555555
xmax	9.0
xmin	-5.0

Assistant x

The code in eee.py looks good.
If it is not working as it should, then consider using some general debugging techniques.

Essai n°3 :

Définir la fonction avec ses paramètres afin de pouvoir ajouter dans ma liste les valeurs de f(x)

=> Erreur indiquée ligne 21!

Et oui, une programme il n'y a pas plus rigoureux !

```
Thonny - C:\Users\Pierre\Downloads\eee.py @ 21:26
File Edit View Run Device Tools Help

essai.py x <untitled> * x eee.py x
1 # definition de la fonction f de variable x,a,b,c
2 def f(x,a,b,c):
3     f=a*x**2+b*x+c
4     return f
5
6 # paramètres
7 a=2
8 b=8
9 c=1
10 xmin=-5.
11 xmax=9.
12 nbre_pts=10
13 # calcule de l'intervalle entre chaque x
14 pas=(xmax-xmin)/(nbre_pts-1)
15
16 # initialisation
17 x=xmin
18
19 # boucle pour avoir des x de -10 à 10
20 for i in range (nbre_pts):
21     print(i,"-",x,"-",f(x))
22     x=x+pas

Shell x
>>> %Run eee.py
Traceback (most recent call last):
  File "C:\Users\Pierre\Downloads\eee.py", line 21, in <module>
    print(i,"-",x,"-",f(x))
TypeError: f() missing 3 required positional arguments: 'a', 'b', and 'c'
>>>
```

Variables x

Name	Value
a	2
b	8
c	1
f	<function f at 0x039127Cf>
i	0
nbre_pts	10
pas	1.5555555555555555
x	-5.0
xmax	9.0
xmin	-5.0

Assistant x

TypeError: f() missing 3 required positional arguments: 'a', 'b', and 'c'
[eee.py, line 21](#)

Python was asked to do an operation with an object which doesn't support it.

- Did you expect another type?
- Maybe you forgot some details about this operation?

Warnings
May help you find the cause of the error.
[eee.py](#)

Erreur corrigée ligne 21

```
Thonny - C:\Users\Pierre\Downloads\eee.py @ 21:26
File Edit View Run Device Tools Help

essai.py x <untitled> * x eee.py x
17 x=xmin
18
19 # boucle pour avoir des x de -10 à 10
20 for i in range (nbre_pts):
21     print(i,"-",x,"-",f(x,a,b,c))
22     x=x+pas

Shell x
>>> %Run eee.py
0 - -5.0 - 11.0
1 - -3.4444444444444444 - -2.827160493827158
2 - -1.8888888888888889 - -6.9753086419753085
3 - -0.3333333333333335 - -1.4444444444444445
4 - 1.2222222222222222 - 13.76543209876543
5 - 2.7777777777777777 - 38.654320987654316
6 - 4.333333333333333 - 73.2222222222221
7 - 5.888888888888888 - 117.46913580246911
8 - 7.444444444444444 - 171.39506172839504
9 - 9.0 - 235.0
>>>
```

Assistant x

Warnings
May be ignored if you are happy with your program.
[eee.py](#)

- Line 2: Redefining name 'x' from outer scope (line 17)
- Line 2: Redefining name 'a' from outer scope (line 7)
- Line 2: Redefining name 'b' from outer scope (line 8)
- Line 2: Redefining name 'c' from outer scope (line 9)
- Line 3: Redefining name 'f' from outer scope (line 2)

[Was it helpful or confusing?](#)

Essai n°4 :

Créer la liste vide `point=[]` puis avec `append` ajouter les couples de points. J'en profiterai grâce à la commande `input` de proposer la saisie des paramètres.

Erreur sournoise, mais ici super importante

Si dans le shell vous taper :
`type(nbre_pts)`

alors vous aurez `<class 'str'>`

Il faut juste forcer ligne 16 le type **numérique** pour la variable `nbre_pts...`

Evidement, toutes mes variables numériques via `input` auront le même problème !!!!

Donc :

a,b,c doivent être de type **float()**
xmin, xmax de type **float()**

MAIS

nbre_pts de type **int()**

```
Thonny - C:\Users\Pierre\Desktop\essai.py @ 16:73
File Edit View Run Device Tools Help

essai.py x <untitled> * x eee.py x

4     return f
5
6     # ititialisation liste vide
7     point=[]
8     # paramètres
9     print("Saisir les valeurs a,b,c")
10    a=input("Valeur de a=")
11    b=input("Valeur de b=")
12    c=input("Valeur de c=")
13    print("Saisir les valeurs Xmin et Xmax")
14    xmin=input("Valeur de Xmin=")
15    xmax=input("Valeur de Xmax=")
16    nbre_pts=input("Combien de couple de valeurs voulez-vous? \n nbr")
17    # calcule de l'intervale entre chaque x
18    pas=(xmax-xmin)/(nbre_pts-1)
19    # initialisation
20    x=xmin
21    # boucle pour avoir les couples
22    for i in range (nbre_pts):
23        point.append([x,f(x,a,b,c)])
24        x=x+pas
25

Shell x
Saisir les valeurs a,b,c
Valeur de a=2
Valeur de b=2
Valeur de c=2
Saisir les valeurs Xmin et Xmax
Valeur de Xmin=-5
Valeur de Xmax=10
Combien de couple de valeurs voulez-vous?
nbre_pts=50
Traceback (most recent call last):
  File "C:\Users\Pierre\Desktop\essai.py", line 18, in <module>
    pas=(xmax-xmin)/(nbre_pts-1)
TypeError: unsupported operand type(s) for -: 'str' and 'str'

>>>
```

Essai final :

YES !

```
# definition de la fonction f de variable x,a,b,c
def f(x,a,b,c):
    f=a*x**2+b*x+c
    return f

# initialisation liste vide
point=[]
# paramètres
print("Saisir les valeurs a,b,c")
a=float(input("Valeur de a="))
b=float(input("Valeur de b="))
c=float(input("Valeur de c="))
print("Saisir les valeurs Xmin et Xmax")
xmin=float(input("Valeur de Xmin="))
xmax=float(input("Valeur de Xmax="))
nbre_pts=int(input("Combien de couple de valeurs voulez-vous? \n nbre_pts="))
# calcul de l'intervale entre chaque x
pas=(xmax-xmin)/(nbre_pts-1)
# initialisation
x=xmin
# boucle pour avoir les couples
for i in range (nbre_pts):
    point.append([x,f(x,a,b,c)])
    x=x+pas
```

The screenshot shows a Python IDE with three main panels:

- Code Editor:** Contains the Python code from the previous block. Red annotations include:
 - A box around lines 10-12 with the note "input → float".
 - A box around line 16 with the note "input → int".
 - A box around line 16 with the note "cohérent".
 - A box around line 10 with the note "102 point (-25; 1760,5)".
- Shell:** Shows the execution of the script. Red annotations include:
 - A box around the input "Valeur de Xmin=-25" with the note "OK".
 - A box around the input "nbre_pts=98" with the note "OK" and a smiley face.
 - A note "il se termine" at the end of the output.
- Variable Viewer:** Shows the state of variables:

Name	Value
a	-2.5
b	4.0
c	-98.0
f	<function f at 0x034838A...
i	97
nbre_pts	98
pas	1.0618556701030928
point	[[-25.0, -1760.5], [-23.9381...
xmax	78.0
xmin	-25.0

11.3 - Vos notes

Si besoin

12 - Présentation des bibliothèques – bienvenue dans l’Alexandrie du XXIème siècle

Résumons, en installant Thonny, vous installez un environnement de développement ET un langage de programmation (python). Vous disposez alors de certaines commandes ou fonction (**print**, **len**, **etc.**). Avec la commande **def** vous pouvez définir vos propres fonctions dans programme. Et il y a plus fort, en important des bibliothèques, vous pouvez importer des fonctions créées par d’autres !!!

L’histoire ne commencera pas par **il était une fois** mais par :

- `import math as m`
`import random as r`

méthode conseillée car si vous appelez plusieurs bibliothèques, même si elles possèdent un nom de fonction commun celle-ci seront différenciées
`m.sin()` et `r.sin()`

ou

- `from math import *` (import de toutes les fonctions)
`from math import pi` (import de la fonction pi)

méthode déconseillée pour les mêmes raisons que celles du dessus.

Rq :

Avec Thonny si vous importez tout avec * alors la liste des fonctions de la bibliothèque apparaît dans les variables



Lors de l’appel de la bibliothèque, si elle n’est pas installée vous aurez un message d’erreur. Voir le chapitre 3.2 de ce cours pour l’installation des bibliothèques.

Ici il me manque le module (bibliothèque) scipy dans Thonny !



12.1 - Bibliothèques

Les liens du dessous pointent sur des pages en anglais mais un clic droit / traduire peut aider.

Le "problème" des bibliothèques c'est qu'il faut connaître un peu de ses fonctionnalités pour s'amuser avec. Mais n'oubliez jamais qu'internet est votre ami !

Le problème que nous allons avoir est que ces bibliothèques ont des fonctionnalités redondantes... donc nos choix seront cornéliens...

Bibliothèque	Pour quoi faire
import csv	Permet d'importer les données d'un fichier csv
import pandas as pd <ul style="list-style-type: none">https://pandas.pydata.org/pandas-docs/stable/getting_started/intro_tutorials/index.htmlguide sympa PANDAS	Permet l'import de données, et la mise en tableau avec étiquettes de colonnes et de lignes, calculs des indicateurs statistiques
import math as m <ul style="list-style-type: none">https://docs.python.org/fr/3/library/math.html	fonctions trigo exp, ln, pi pgcd ...
import cmath as cm <ul style="list-style-type: none">https://docs.python.org/fr/3.5/library/cmath.html	gestion des nombres complexes
import matplotlib.pyplot as plt <ul style="list-style-type: none">https://matplotlib.org/tutorials/introductory/pyplot.htmlguide sympa MATHPLOTLIB	Permet de tracer et visualiser des données
import numpy as np <ul style="list-style-type: none">https://numpy.org/devdocs/user/quickstart.html	Permet de gérer et calculer des tableaux matriciels de nombres et de les trier, classer...
import scipy as sc <ul style="list-style-type: none">https://docs.scipy.org/doc/scipy/reference/	SciPy est un projet visant à unifier et fédérer un ensemble de bibliothèques Python à usage scientifique. Scipy utilise les tableaux et matrices du module NumPy
import folium <ul style="list-style-type: none">https://python-visualization.github.io/folium/quickstart.html	Permet de visualiser des objets géolocalisés avec des coord GPS sur une carte OpenStreetMap

12.2 - Commandes

Commandes	Opération
import ...	Import de la bibliothèque ...
from ... import ...	De la bibliothèque... importer la fonction ... (ou *)
dir()	Liste les noms définis par un module

12.3 - Vos notes

Si besoin

13 - Import de données d'un fichier csv



A lire pour les enseignants

L'importation de valeurs via un fichier csv est la porte d'entrée de l'interopérabilité entre les TP et l'analyse. En ExAO vous exportez vos mesures au format csv : séparateur ; codage UTF 8 puis vous analysez les données en vous appuyant sur la puissance de python !

13.1 - Commandes

Commande ou Symbole	Opération
<pre>import csv source=open('nomdevotrefichier.csv','r',encoding='utf-8') for row in csv.reader(source,delimiter=';'): ...</pre>	Via bibliothèque csv Importation des données d'un fichier csv dans une liste de type string
<pre>import pandas as pd mesures=pd.read_csv('nomdevotrefichier', sep=";",decimal=",",encoding="utf-8") valeurs=mesures.values.tolist()</pre>	Via bibliothèque pandas Importation des données d'un fichier csv dans une DataFrame que l'on transforme en liste

13.2 - Exercices

13.2.1 - Exo : import csv – Importer les valeurs numériques d'un fichier CSV qui a une 1ère ligne d'en-tête

13.2.1.1 - Énoncé



- Télécharger le fichier csv suivant et localiser le dossier où il est enregistré : [Fichier mesures.csv \(données bidons\)](#)
- Ouvrir ce fichier avec le bloc note afin de visualiser sa structure
 - Vous observez 3 types de données
 - Releve
 - capteur 1
 - capteur 2
 - Vous observez que le séparateur de champs est le ; - c'est ok
 - Vous observez aussi que le séparateur de décimale est la , et là cela posera un problème car pour python le séparateur de décimale est le .

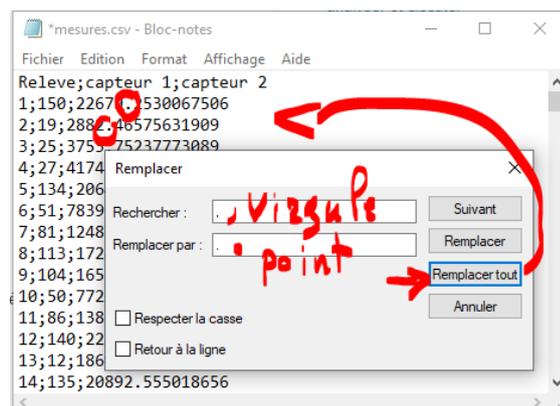
```
mesures.csv - Bloc-notes
Fichier Edition Format Affichage Aide
Releve;capteur 1;capteur 2
1;150;22679,2530067506
2;19;2882,46575631909
3;25;3753,75237773089
4;27;4174,64549423648
```

L'objectif va être de lire ce fichier, et de mémoriser les données dans une liste pour pouvoir en faire des calculs numériques

Essai 1

Avec le Bloc Notes :

- Ouvrir le csv, Edition / Remplacer et transformer la , en .
- Écraser le fichier mesure



Avec l'éditeur :

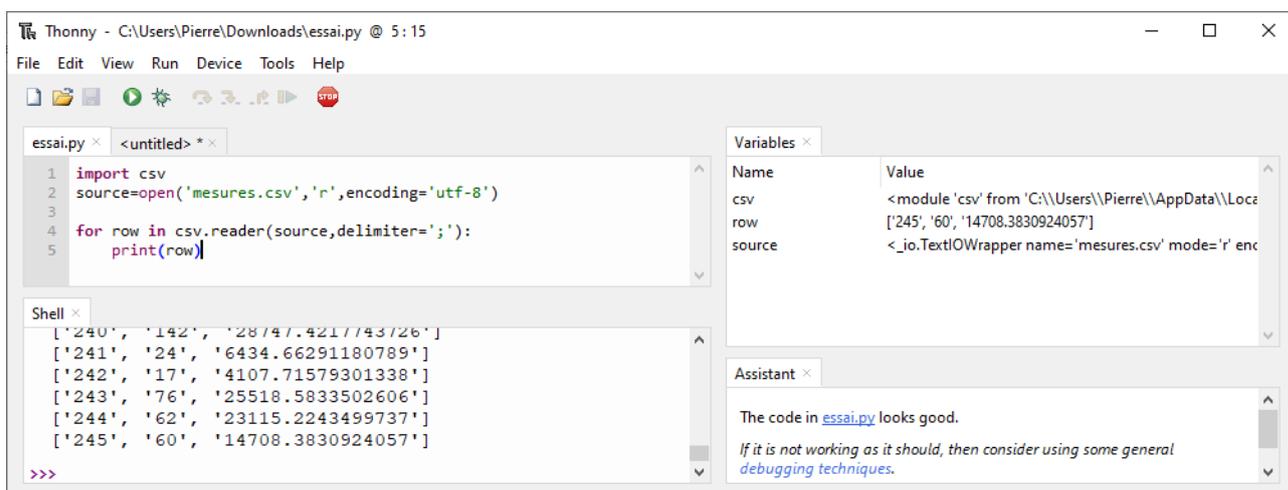
- Copier/coller le script suivant

```
# Essai 1
import csv
source=open('mesures.csv','r',encoding='utf-8')

for row in csv.reader(source,delimiter=','):
    print(row)
```

- Enregistrer le programme dans le même dossier que le fichier csv !
- Lancer le programme

Résultat de l'essai 1



- Ligne 1, import de la bibliothèque csv afin de pouvoir utiliser par exemple le paramètre open de la ligne suivante
- Ligne 2, ouverture du fichier mesures.csv en lecture et un codage utf-8
- Ligne 4, lecture de chaque ligne et je sépare les données entre ;
 - Ligne 5, j'imprime chaque ligne une par une.
(je ne conserve aucune donnée, sauf celle de la dernière ligne – voir la fenêtre variable pour s'en convaincre)
- Je constate que mes données sont toutes de type string ('245' ou '60' ou...)
=> problème

Essai 2

Avec l'éditeur :

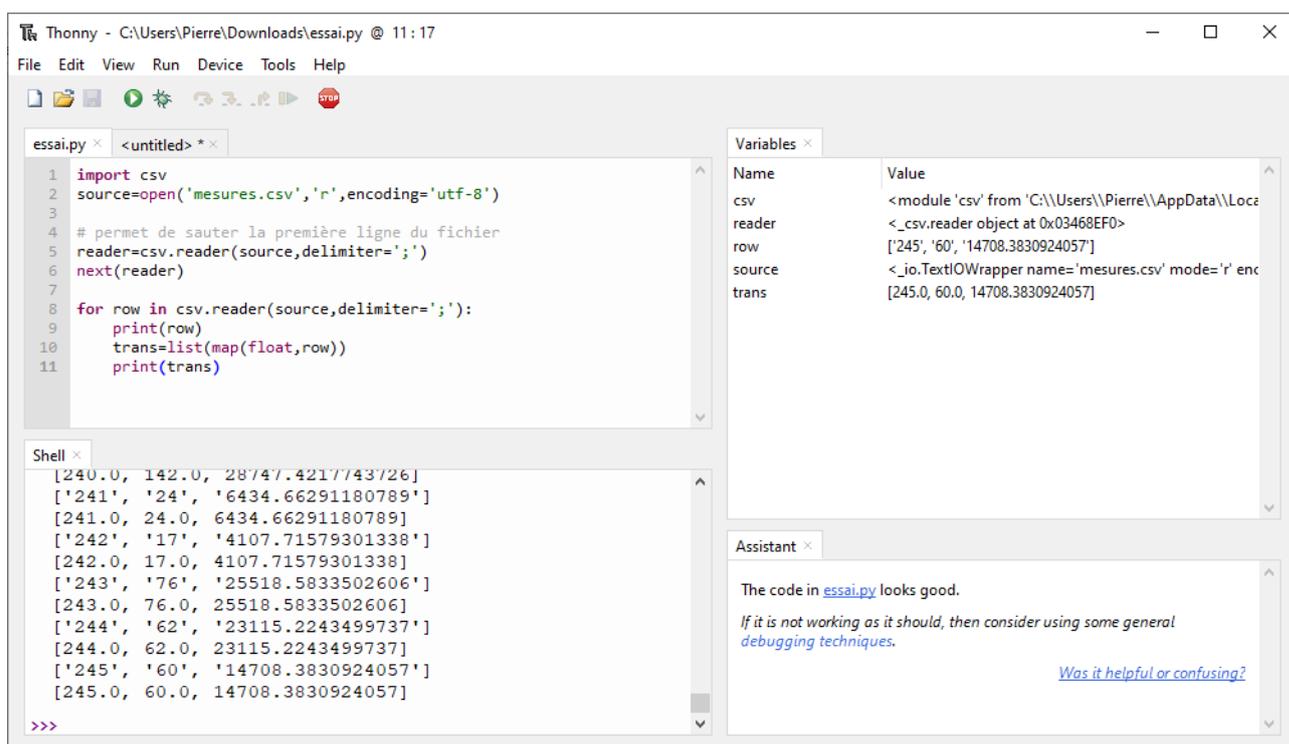
- Copier/coller le script suivant puis lancer le

```
# Essai 2
import csv
source=open('mesures.csv','r',encoding='utf-8')

# permet de sauter la première ligne du fichier
reader=csv.reader(source,delimiter=',')
next(reader)

for row in csv.reader(source,delimiter=','):
    print(row)
    trans=list(map(float,row))
    print(trans)
```

Résultat de l'essai 2



The screenshot shows the Thonny Python IDE interface. The main editor window displays the Python script from the previous block. The Shell window at the bottom shows the output of the script, which includes a row of strings and a row of floats. The Variables window on the right shows the state of the program, including the csv module, the reader object, the current row, the source file, and the trans list. The Assistant window at the bottom right provides feedback on the code execution.

```
Shell x
[240.0, 142.0, 28747.4217743726]
['241', '24', '6434.66291180789']
[241.0, 24.0, 6434.66291180789]
['242', '17', '4107.71579301338']
[242.0, 17.0, 4107.71579301338]
['243', '76', '25518.5833502606']
[243.0, 76.0, 25518.5833502606]
['244', '62', '23115.2243499737']
[244.0, 62.0, 23115.2243499737]
['245', '60', '14708.3830924057']
[245.0, 60.0, 14708.3830924057]
>>>
```

Name	Value
csv	<module 'csv' from 'C:\\Users\\Pierre\\AppData\\Local\\Programs\\Python\\Python38-64\\Lib\\site-packages\\csv.py'>
reader	<_csv.reader object at 0x03468EF0>
row	['245', '60', '14708.3830924057']
source	<_io.TextIOWrapper name='mesures.csv' mode='r' encoding='utf-8'>
trans	[245.0, 60.0, 14708.3830924057]

Assistant x

The code in [essai.py](#) looks good.

If it is not working as it should, then consider using some general [debugging techniques](#).

[Was it helpful or confusing?](#)

- Les lignes 4 à 6 permettent de sauter la ligne d'étiquette de mon csv car cette ligne ne pourra pas être transformée en valeur numérique
- La ligne 10 transforme (map) le contenu de chaque ligne (boucle) et transforme son contenu en valeurs numériques flottantes puis la commande list transforme le tout en liste et l'affecte à la variable trans
- Je constate bien dans le shell la ligne avec des données type string et la ligne du dessous avec les mêmes données de type float

=> je suis proche de la solution

Essai 3

Avec l'éditeur :

- Copier/coller le script suivant puis lancer le

```
#Essai 3
import csv
source=open('mesures.csv','r',encoding='utf-8')

#creation liste vide
donnees=[]

#permet de sauter la première ligne du fichier
reader=csv.reader(source,delimiter=';')
next(reader)

for row in csv.reader(source,delimiter=';'):
    trans=list(map(float,row))
    donnees.append(trans) #remplissage de la liste

print(len(donnees)) #nbre de lignes de données
print(donnees)
print(donnees[0]) #première ligne
print(donnees[0][0],"-",donnees[0][1],"-",donnees[0][2]) #elements de la première ligne
```

Résultat de l'essai 3

```
Thonny - C:\Users\Pierre\Downloads\essai.py @ 14:15
File Edit View Run Device Tools Help

essai.py x <untitled> * x
1 import csv
2 source=open('mesures.csv','r',encoding='utf-8')
3 donnees=[]
4
5 # permet de sauter la première ligne du fichier
6 reader=csv.reader(source,delimiter=';')
7 next(reader)
8
9 for row in csv.reader(source,delimiter=';'):
10     trans=list(map(float,row))
11     donnees.append(trans)
12
13 print(len(donnees))
14 print(donnees)
15 print(donnees[0])
16 print(donnees[0][0],"-",donnees[0][1],"-",donnees[0][2])
17

Variables x
Name Value
csv <module 'csv' from 'C:
donnees [[1.0, 150.0, 22679.25300
reader <_csv.reader object at (
row ['245', '60', '14708.3830
source <_io.TextIOWrapper na
trans [245.0, 60.0, 14708.3830

Shell x
>>> %Run essai.py
245
[[1.0, 150.0, 22679.2530067506], [2.0, 1 ...
[1.0, 150.0, 22679.2530067506]
1.0 - 150.0 - 22679.2530067506
>>>
```

Remarque :

La ligne 16 me prouve que je peux maintenant utiliser les données de ma liste pour d'autres calculs à venir

Bilan

Je retiens donc ce début de script pour créer une liste manipulable sous python en partant de données importées d'un fichier csv.

```
#
import csv
source=open('nomdevotrefichier.csv','r',encoding='utf-8')

#creation liste vide
donnees=[]

#permet de sauter la première ligne du fichier
reader=csv.reader(source,delimiter=',')
next(reader)

#remplissage de la liste donnees
for row in csv.reader(source,delimiter=','):
    trans=list(map(float,row))
    donnees.append(trans)
```

13.2.2 - Exo : import pandas – Importer données d'un fichier csv

13.2.2.1 - Énoncé



- Télécharger à nouveau le même fichier csv que dans l'exo précédent (il est peut-être préférable de supprimer votre précédent téléchargement) et localiser le dossier où il est enregistré : [Fichier mesures.csv \(données bidons\)](#)

```
mesures.csv - Bloc-notes
Fichier Edition Format Affichage Aide
Releve;capteur 1;capteur 2
1;150;22679,2530067506
2;19;2882,46575631909
3;25;3753,75237773089
4:27:4174.64549423648
```

- Dans l'éditeur, copier/coller le script suivant

```
#Essai 1
import pandas as pd
mesures=pd.read_csv('mesures.csv', sep=";",decimal=",",encoding="utf-8")

print(mesures),print("^ LIGNE 5 *****")
print(mesures.head()),print("^ LIGNE 6 *****")
print(mesures.info()),print("^ LIGNE 7 *****")
print(mesures.columns),print("^ LIGNE 8 *****")
print(mesures.dtypes),print("^ LIGNE 9 *****")
```

- Enregistrer le programme dans le même dossier que le fichier csv !
- Lancer le programme

Résultat de l'essai 1

```
1 #Essai 1
2 import pandas as pd
3 mesures=pd.read_csv("mesures.csv", sep=";", decimal=".", encoding="utf-8")
4
5 print(mesures).print("^ LIGNE 5 *****")
6 print(mesures.head()).print("^ LIGNE 6 *****")
7 print(mesures.info()).print("^ LIGNE 7 *****")
8 print(mesures.columns).print("^ LIGNE 8 *****")
9 print(mesures.dtypes).print("^ LIGNE 9 *****")

Shell
Releve  capteur 1  capteur 2
0      1      150  22679.253007
1      2      19   2882.465756
2      3      25   3753.752378
3      4      27   4174.645494
4      5      134  20648.916440
...
240    241      24   6434.662912
241    242      17   4107.715793
242    243      76  25518.583350
243    244      62  23115.224350
244    245      60  14708.383092

[245 rows x 3 columns]
^ LIGNE 5 *****
Releve  capteur 1  capteur 2
0      1      150  22679.253007
1      2      19   2882.465756
2      3      25   3753.752378
3      4      27   4174.645494
4      5      134  20648.916440
^ LIGNE 6 *****
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 245 entries, 0 to 244
Data columns (total 3 columns):
Releve      245 non-null int64
capteur 1   245 non-null int64
capteur 2   245 non-null float64
dtypes: float64(1), int64(2)
memory usage: 5.8 KB
None
^ LIGNE 7 *****
Index(['Releve', 'capteur 1', 'capteur 2'], dtype='object') 8
^ LIGNE 8 *****
Releve      int64
capteur 1   int64
capteur 2   float64
dtypes: object
^ LIGNE 9 *****
>>>
```

- En 2 lignes, et sans boucle...
- Ligne 1, importation de la bibliothèque pandas
- Ligne 2, lecture du fichier mesures.csv, déclaration du séparateur comme étant le ; déclaration aussi du séparateur de décimale comme étant la , et encodage utf-8. Les données sont enregistrées dans l'élément mesures qui est une DataFrame = Données tabulaires bidimensionnelles, mutables en taille, potentiellement hétérogènes
- Les lignes 5 à 8 nous affiches les données et je comprends :
 - le bloc 5 est le résultat de la ligne 5 et présente les 1ères et dernières lignes des données (DataFrame)
 - le bloc 6 est le résultat de la ligne 6
 - ...

=> A ce stade, je peux utiliser toutes les fonctions de pandas, et elles sont nombreuses ! https://www.ipa-troulet.fr/cours/attachments/article/542/DOC_Manipulation_Pandas.pdf

=> MAIS à ce stade je ne gère pas une liste

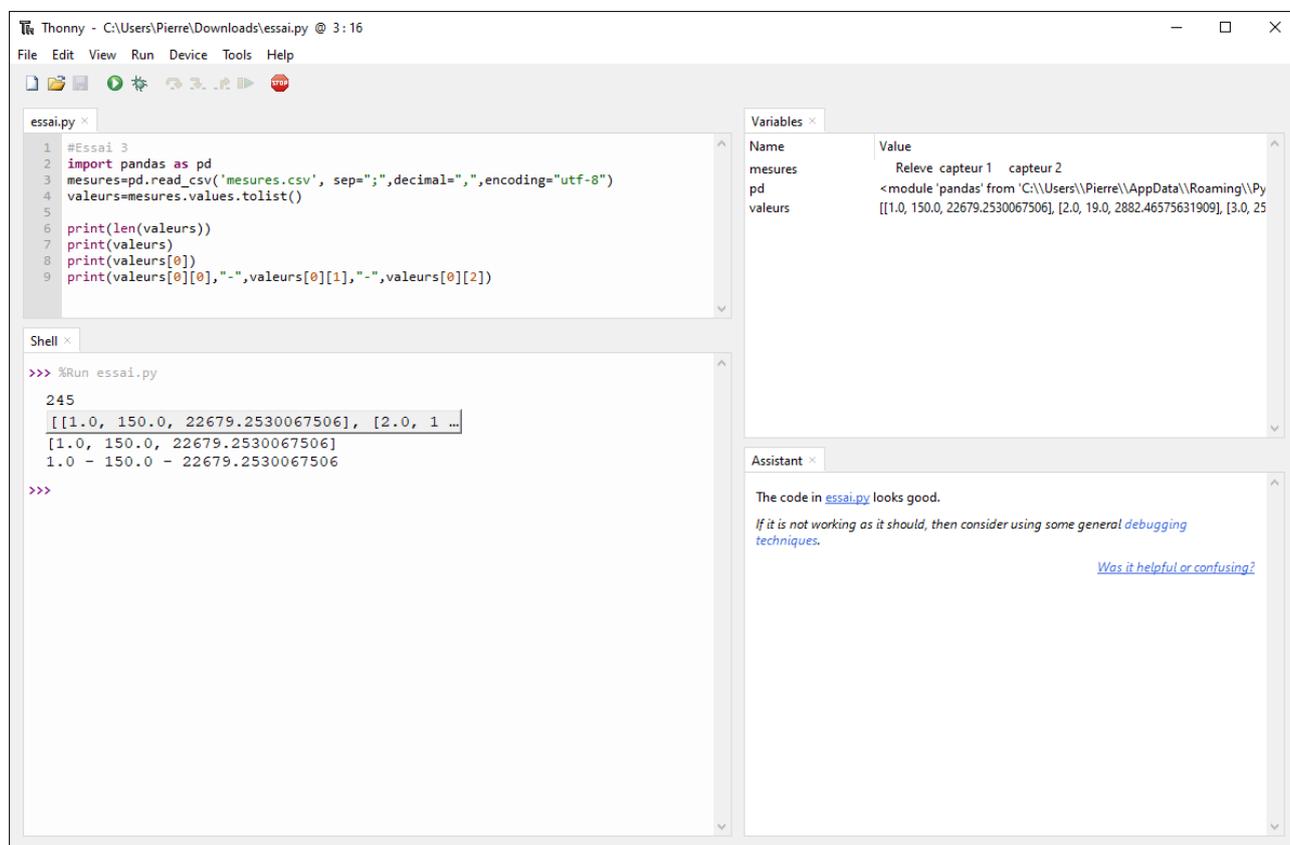
- Dans l'éditeur, copier/coller le script suivant

```
#Essai 2
import pandas as pd
mesures=pd.read_csv('mesures.csv', sep=";",decimal=".",encoding="utf-8")
valeurs=mesures.values.tolist()

print(len(valeurs))
print(valeurs)
print(valeurs[0])
print(valeurs[0][0],"-",valeurs[0][1],"-",valeurs[0][2])
```

- Enregistrer le programme dans le même dossier que le fichier csv !
- **Lancer le programme**

Résultat de l'essai 2



```
Thonny - C:\Users\Pierre\Downloads\essai.py @ 3:16
File Edit View Run Device Tools Help

essai.py
1 #Essai 3
2 import pandas as pd
3 mesures=pd.read_csv('mesures.csv', sep=";",decimal=".",encoding="utf-8")
4 valeurs=mesures.values.tolist()
5
6 print(len(valeurs))
7 print(valeurs)
8 print(valeurs[0])
9 print(valeurs[0][0],"-",valeurs[0][1],"-",valeurs[0][2])

Shell
>>> %Run essai.py
245
[[1.0, 150.0, 22679.2530067506], [2.0, 1 ...
[1.0, 150.0, 22679.2530067506]
1.0 - 150.0 - 22679.2530067506
>>>

Variables
Name Value
mesures Releve capteur1 capteur 2
pd <module 'pandas' from 'C:\Users\Pierre\AppData\Roaming\Py
valeurs [[1.0, 150.0, 22679.2530067506], [2.0, 19.0, 2882.46575631909], [3.0, 25

Assistant
The code in essai.py looks good.
If it is not working as it should, then consider using some general debugging techniques.
Was it helpful or confusing?
```

Remarque :

En 3 lignes je retrouve les données en listes. La ligne 9 me prouve que je peux maintenant utiliser les données de ma liste pour d'autres calculs à venir

Bilan

Je retiens donc ce début de script pour créer une liste manipulable sous python en partant de données importées d'un fichier csv.

```
#  
import pandas as pd  
mesures=pd.read_csv('nomdevotrefichier.csv', sep=";",decimal="," ,encoding="utf-8")  
valeurs=mesures.values.tolist()
```

13.3 - Vos notes

Si besoin

14 - Utilisation de fonctions en maths et calculs

Cette bibliothèque permet d'accéder à un grand nombre de fonctions mathématiques (sinus, racine carrée, exponentielle, ...) Pour pouvoir utiliser ce module, il faut taper l'instruction :

```
import math as m  
ou  
from math import *  
ou  
from math import sin, pi
```

Pour info voici la liste des noms des fonctions disponibles le shell en 2 étapes :

- Appel de toutes les fonctions avec le `from math import *`
- `dir()`

```
Shell x  
>>> from math import *  
>>> dir()  
['_annotations_', '__builtins__', '__doc__', '__file__', '__loader__', '__name__', '__package__', '__spec__', 'acos', 'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil', 'copysign', 'cos', 'cosh', 'degrees', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial', 'floor', 'fmod', 'frexp', 'fsum', 'gamma', 'gcd', 'hypot', 'inf', 'isclose', 'isfinite', 'isinf', 'isnan', 'ldexp', 'lgamma', 'log', 'log10', 'log1p', 'log2', 'modf', 'nan', 'pi', 'pow', 'radians', 'remainder', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'tau', 'trunc']  
>>> |
```

14.1 - Commandes

Commande ou Symbole	Opération
<code>import math as m</code>	Importe la bibliothèque math. Les fonctions appelées seront précédées de m.
<code>from math import sin,pi</code>	Importe les fonction sin et pi. Attention à ne pas nommer des variables avec le même nom !

14.2 - Exercices

14.2.1 - Exo : Utilisation des fonctions sinus et pi (ou radians) provenant de math

14.2.1.1 - Énoncé



Dans l'éditeur :

Créer un programme qui enregistre dans une liste les valeurs des sinus pour x allant de 0° à 360°. Je rappelle que la valeur de x dans sin(x) doit être en radian !

Vous avez donc besoins de la bibliothèque math et comme le problème est simple (Petit nombre de variables donc peu de risque de nommer une variable avec un nom de fonction) je vous conseille ceci :

```
from math import *
```

Ma piste : import de math * - une def de fonction – une liste vide – une boucle et ajout des valeurs dans la liste avec append

```
# Votre script
```

14.2.1.2 - Un exemple de correction

Utilisation de pi pour convertir les degrés en radians, je dois donc l'importer aussi !!

- Évidemment vous comprenez le $x/180 \cdot \pi$ ligne 3
- Notez que j'ai été économe en mémoire en n'appelant que les fonctions sin et pi mais cela aurait évidemment fonctionné avec **from math import ***

```
essai.py x
1 from math import sin, pi
2 def f(x):
3     sinus=sin(x/180*pi)
4     return sinus
5
6 valeurs=[]
7 for i in range (361):
8     valeurs.append([i,f(i)])
9
10 print(valeurs)
11 print("valeur de sin(90)=",valeurs[90])
12
```

Shell x

```
>>> %Run essai.py
[[0, 0.0], [1, 0.01745240643728351], [2, ...]
valeur de sin(90)= [90, 1.0]
>>>
```

Utilisation de la fonction radians pour la conversion. Notez qu'ici j'importe tout le module donc il me suffit de mettre m. devant les fonctions !

- Vous avez compris le
m.radians
m.sin
?

```
essai.py x
1 import math as m
2 def f(x):
3     rad=m.radians(x)
4     sinus=m.sin(rad)
5     return sinus
6
7 valeurs=[]
8 for i in range (361):
9     valeurs.append([i,f(i)])
10
11 print(valeurs)
12 print("valeur de sin(90)=",valeurs[90])
13
```

Shell x

```
>>> %Run essai.py
[[0, 0.0], [1, 0.01745240643728351], [2, ...]
valeur de sin(90)= [90, 1.0]
>>>
```

14.2.2 - Exo : Utilisation des fonctions racine et exponentielle, mais pas que...

14.2.2.1 - Énoncé



```
mesures.csv - Bloc-notes
Fichier Edition Format Affichage Aide
Releve;capteur 1;capteur 2
1;150;22679,2530067506
2;19;2882,46575631909
3;25;3753,75237773089
4;27;4174,64549423648
```

A l'adresse suivant "<https://www.ipa-troulet.fr/cours/attachments/article/542/mesures.csv>" se trouve le fichier csv contenant des mesures des 2 capteurs. Vous connaissez déjà son contenu.

Avec l'aide de **pandas** et de **math**, créez une série supplémentaire "calcul 1" contenant pour chaque ligne le résultat de :

$$\text{calcul 1} = (\sqrt{\text{capteur 2}}) * \exp(-\text{capteur 1})$$

Étape 1

Testez ce script

```
#
import pandas as pd
url="https://www.ipa-troulet.fr/cours/attachments/article/542/mesures.csv"
mesures=pd.read_csv(url, sep=";", decimal=",", encoding="utf-8")
valeurs=mesures.values.tolist()

print(mesures.columns)
print()
print(valeurs[0][0],valeurs[0][1],valeurs[0][2])
```

Impressionnant !

Je comprends que :

- url contient une url;
- que le `pd.read_csv()` lit le contenu du fichier à distance !!
- le reste je savais déjà...

Au fait, nous sommes tous d'accord, nous utiliserons la liste "valeurs" pour la suite, car valeur est une liste !

```
Thonny - C:\Users\Pierre\Downloads\essai.py @ 6:1
File Edit View Run Device Tools Help

essai.py
1 import pandas as pd
2 url="https://www.ipa-troulet.fr/cours/attachments/article/542/mesures.csv"
3 mesures=pd.read_csv(url, sep=";", decimal=",", encoding="utf-8")
4 valeurs=mesures.values.tolist()
5
6 print(mesures.columns)
7 print()
8 print(valeurs[0][0],valeurs[0][1],valeurs[0][2])

Shell
>>> %Run essai.py
Index(['Releve', 'capteur 1', 'capteur 2'], dtype='object')
1.0 150.0 22679.2530067506
>>>

Variables
Name      Value
mesures   Releve cap
pd         <module 'par
url        'https://www
valeurs    [[1.0, 150.0, 22679.2530067506]]

Assistant
The code in essai.py looks good.
If it is not working as it should, then consider using some general debugging techniques.
```

Étape 2

Testez ce script puis **commentez le !**

```
#
import pandas as pd
#
import math as m

#
url="https://www.ipa-troulet.fr/cours/attachments/article/542/mesures.csv"
#
mesures=pd.read_csv(url, sep=";",decimal=".",encoding="utf-8")
#
valeurs=mesures.values.tolist()
#
n=len(valeurs)

#
for i in range(n):
    print(valeurs[i])
    #
    calcul1=m.sqrt(valeurs[i][2])*m.exp(-valeurs[i][1])
    #
    valeurs[i].append(calcul1)
    print(valeurs[i])
```

Rq :

Elle est belle la ligne
21

Notez que la 19 est
bien aussi;)

```
Thierry - C:\Users\Pierre\Downloads\essai.py @ 19:56
File Edit View Run Device Tools Help

essai.py
1 #
2 import pandas as pd
3 #
4 import math as m
5
6 #
7 url="https://www.ipa-troulet.fr/cours/attachments/article/542/mesures.csv"
8 #
9 mesures=pd.read_csv(url, sep=";",decimal=".",encoding="utf-8")
10 #
11 valeurs=mesures.values.tolist()
12 #
13 n=len(valeurs)
14
15 #
16 for i in range(n):
17     print(valeurs[i])
18     #
19     calcul1=m.sqrt(valeurs[i][2])*m.exp(-valeurs[i][1])
20     #
21     valeurs[i].append(calcul1)
22     print(valeurs[i])
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
```

15 - Visualisation des données avec matplotlib

Matplotlib est une bibliothèque qui sert à tracer et visualiser des données. En effet, elle permet d'obtenir des graphiques complets et propres avec peu de lignes de code.

Pour importer matplotlib, il faut taper :

```
import matplotlib.pyplot as plt
```

On importe matplotlib.pyplot sous le nom plt et ainsi, toutes ses fonctions seront appelées en les faisant précéder de plt.

15.1 - Commandes

Commande ou Symbole	Opération
<code>import matplotlib.pyplot as plt</code>	Import de la bibliothèque
<code>plt.axis([xmin, xmax, ymin, ymax])</code> <code>plt.plot(x, y)</code> <code>plt.title("titre")</code> <code>plt.xlabel("abscisses")</code> <code>plt.ylabel("ordonnees")</code> <code>plt.axhline(y=0,color='black')</code> <code>plt.axvline(x=0,color='black')</code>	Intervalle du graphe Représente le point ... https://courspython.com/introduction-courbes.html

15.2 - Exercices

15.2.1 - Exo : Mise en forme d'un graphique avec matplotlib

15.2.1.1 - Énoncé



Dans l'éditeur :

- Testez le script et comprenez !

```
# saisie parametres
a,b,c=2,-4,-3

# import bibliot pour le graph -> plt
import matplotlib.pyplot as plt

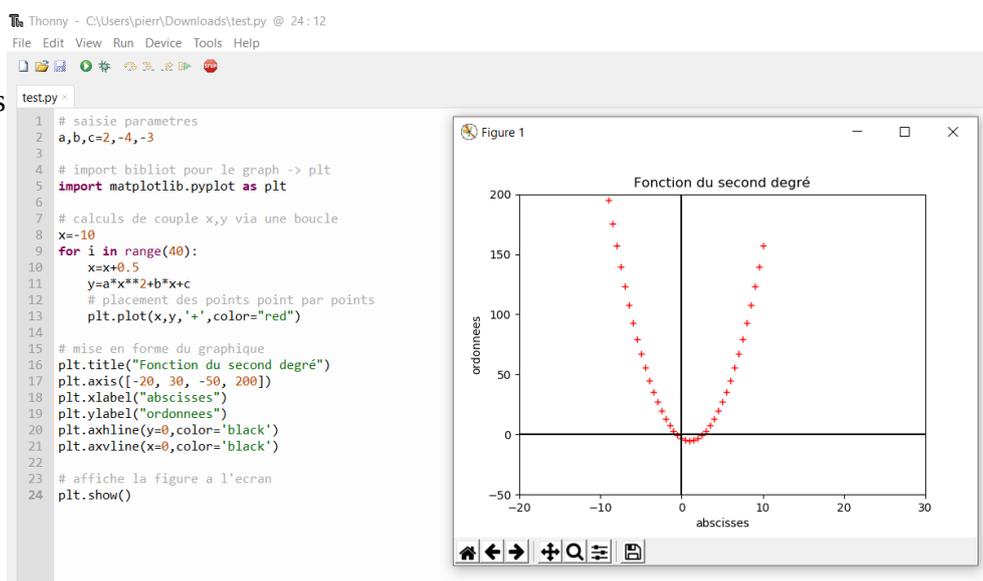
# calculs de couple x,y via une boucle
x=-10
for i in range(40):
    x=x+0.5
    y=a*x**2+b*x+c
    # placement des points point par points
    plt.plot(x,y,'+',color="red")

# mise en forme du graphique
plt.title("Fonction du second degré")
plt.axis([-20, 30, -50, 200])
plt.xlabel("abscisses")
plt.ylabel("ordonnees")
plt.axhline(y=0,color='black')
plt.axvline(x=0,color='black')

# affiche la figure a l'ecran
plt.show()
```

Remarque :

Ici `plt.plot(x,y)` crée les points dans la boucle donc un par un !



15.2.1.2 - Votre analyse du script

Répondre aux questions suivantes	
Quelle est la valeur de départ des x ?	
Le pas de calcul de x est de combien ?	
Comment faire si je veux des . en bleu et non des croix ?	
Quelles lignes font que je visualise les axes ox et oy de mon graphique ?	
Quelle ligne gère les valeurs max du graphe ?	
Puis-je après exécution du programme appeler une valeur de x ou de y ?	

15.2.2 - Exo : Faire des graphiques avec matplotlib et numpy

15.2.2.1 - Énoncé



```
# saisie param
a,b,c=2,-4,-3

#calcul de delta
delta=b**2-4*a*c

#affichage
print("résolution de l'équation ",a," x2 + ",b," x + ",c)

# condition sur delta dans cet ordre >0 puis ==0 puis <0
if delta>0:
    x1=(-b-delta**0.5)/(2*a)
    x2=(-b+delta**0.5)/(2*a)
    print("Delta est positif donc il y a 2 solutions")
    print("x1 =",x1)
    print("x2 =",x2)
elif delta==0:
    x0=-b/(2*a)
    print("Delta est nul donc il y a 1 solution unique")
    print("x0 =",x0)
else:
    print("Pas de solution dans l'espace de réel")

#représentation graphique
import numpy as np
import matplotlib.pyplot as plt

#encadrement pour le graphique
xmin=float(input("Saisir la valeur de xmin="))
xmax=float(input("Saisir la valeur de xmax="))

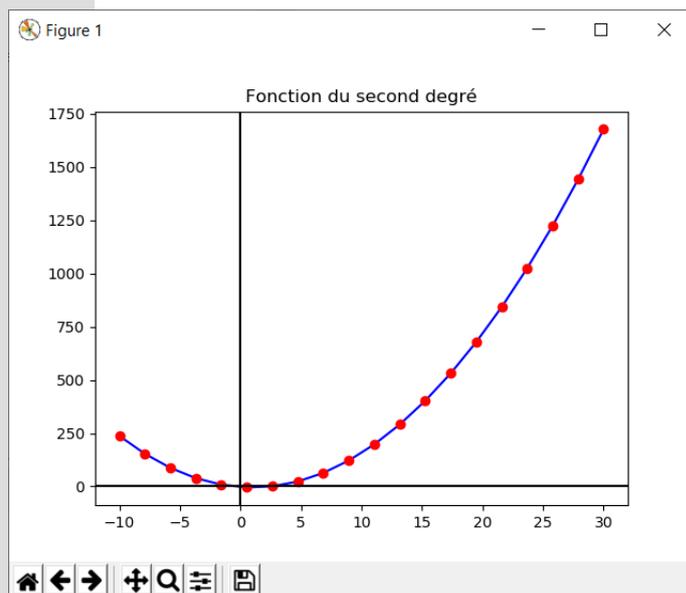
x = np.linspace(xmin,xmax,20)
y=a*x**2+b*x+c
plt.plot(x, y,color="blue")
plt.plot(x,y,'o',color="red")

plt.title("Fonction du second degré")
plt.axhline(y=0,color='black')
plt.axvline(x=0,color='black')

plt.show() # affiche la figure a l'ecran
```

Dans l'éditeur :

- Copier/coller le script puis exécuter le !
- En mode **débug** avec la commande Step over (F6) voyez ce qui se passe à la ligne 32 et 33 et comprenez !



Remarque :

Ici `plt.plot(x,y)` représente les valeurs des matrices (array) `x` et `y` donc pas de boucle !

The screenshot shows the Thonny Python IDE with a script named 'test.py'. The script defines a quadratic equation $ax^2 + bx + c = 0$ with $a=2$, $b=-4$, and $c=-3$. It calculates the discriminant $\Delta = b^2 - 4ac = 40$. Since $\Delta > 0$, it finds two real solutions: $x_1 \approx -0.5811388300841898$ and $x_2 \approx 2.58113883008419$. The script then plots the parabola $y = 2x^2 - 4x - 3$ using matplotlib. The Variables window shows the current state of variables, and the Shell window shows the execution output.

```

18     x0=-b/(2*a)
19     print("Delta est nul donc il y a 1 solution unique")
20     print("x0 =",x0)
21 else:
22     print("Pas de solution dans l'espace de réel")
23
24 #représentation graphique
25 import numpy as np
26 import matplotlib.pyplot as plt
27
28 #encadrement pour le graphique
29 xmin=float(input("Saisir la valeur de xmin="))
30 xmax=float(input("Saisir la valeur de xmax="))
31
32 x = np.linspace(xmin,xmax,20)
33 y=a*x**2+b*x+c
34 plt.plot(x, y,color="blue")
35 plt.plot(x,y,'o',color="red")
36
37 plt.title("Fonction du second degré")
38 plt.axhline(y=0,color='black')
39 plt.axvline(x=0,color='black')
40
41 plt.show() # affiche la figure a l'ecran
    
```

Variables window:

Name	Value
a	2
b	-4
c	-3
delta	40
np	<module 'numpy' from 'C:\Users\pierr\AppData\Roar
plt	<module 'matplotlib.pyplot' from 'C:\Users\pierr\AppData
x	array([-10. , -7.89473684, -5.78947368, -3.68421053
x1	-0.5811388300841898
x2	2.58113883008419
xmax	30.0
xmin	-10.0
y	array([2.37000000e+02, 1.53232687e+02, 8.71939058e+

Shell window:

```

===== RESTART =====
>>> %Debug test.py
résolution de l'équation 2 x² + -4 x + -3
Delta est positif donc il y a 2 solutions
x1 = -0.5811388300841898
x2 = 2.58113883008419
Saisir la valeur de xmin=-10
Saisir la valeur de xmax=30
    
```

15.2.2.2 - Votre analyse du script

Répondre aux questions suivantes	
Les valeurs du graphique sont elles enregistrées dans la mémoire ?	
Ligne 29, le float est utile dans quel cas ?	
Ligne 32 linspace est une fonction de quelle bibliothèque ?	
Ligne 33 la variable y est de quel type ?	
Pourquoi y a-t-il deux plt.plot ligne 34 et 35 ?	

15.2.3 - Exo : Créer un programme permettant de visualiser graphiquement un tirage aléatoire avec remise d'un dé à 6 faces.

Ce qui serait sympa, c'est de laisser le choix à l'utilisateur du nombre de jet

15.2.3.1 - Énoncé



Dans l'éditeur :

copier/coller le script suivant puis exécutez le !

```
histo.py * x   aaa.py x
1 import matplotlib.pyplot as plt
2 import numpy as np
3 import numpy.random as npr
4
5 ''' exemple de commentaire
6 la fonction randint génère un nbre aléatoire entre 1 et 6
7 puis la méthode append ajoute cette valeur dans la variable resultat
8 La liste resultat est par la suite affichée dans le shell
9 '''
10 resultat=[]
11 for i in range(100):
12     resultat.append(npr.randint(1,7)) # tirage aleatoire entre 1 et 6
13     print(resultat)
14
15 ''' commenter tout le bloc lignes 18-21
16
17 '''
18 total=[]
19 for i in range(6):
20     total.append(resultat.count(i+1))
21     print(total)
22
23
24 ''' commenter tout le bloc ligne 27
25
26 '''
27 des=["dé 1","dé 2","dé 3","dé 4","dé 5","dé 6"]
28
29
30 ''' commenter tout le bloc lignes 32-34
31
32 '''
33 plt.title('My title')
34 plt.xlabel('N° du dé')
35 plt.ylabel('Nbre de sortie')
36
37 ''' commenter tout le bloc lignes 39-41
38
39 '''
40 plt.ylim(0,60)
41 plt.bar(des,total,color="blue")
42 plt.show()
```

```
import matplotlib.pyplot as plt
import numpy as np
import numpy.random as npr

''' exemple de commentaire attendu:
la fonction randint génère un nbre aléatoire entre 1 et 6
puis la méthode append ajoute cette valeur dans la variable resultat
La liste resultat est par la suite affichée dans le shell
'''
resultat=[]
for i in range(100):
    resultat.append(npr.randint(1,7)) # tirage aleatoire entre 1 et 6
    print(resultat)

''' commenter tout le bloc lignes 18-21

'''
total=[]
for i in range(6):
    total.append(resultat.count(i+1))
print(total)

''' commenter tout le bloc ligne 27

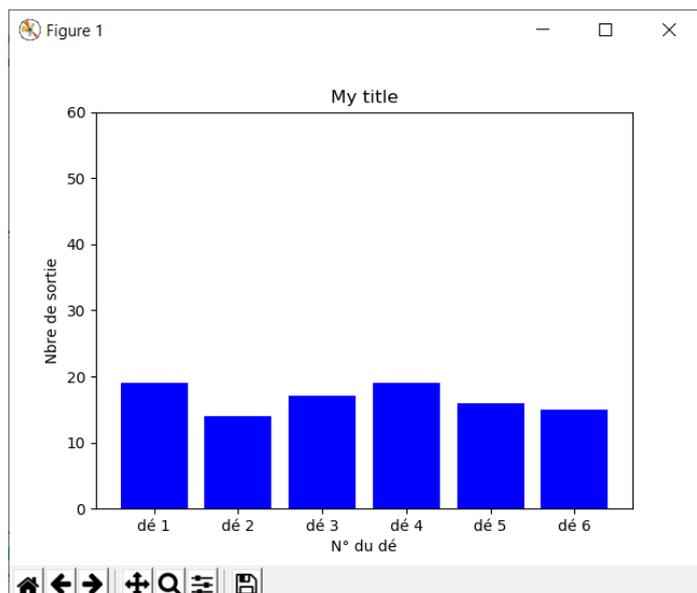
'''
des=["dé 1","dé 2","dé 3","dé 4","dé 5","dé 6"]

''' commenter tout le bloc lignes 32-34

'''
plt.title('My title')
plt.xlabel('N° du dé')
plt.ylabel('Nbre de sortie')

''' commenter tout le bloc lignes 39-41

'''
plt.ylim(0,60)
plt.bar(des,total,color="blue")
plt.show()
```



15.2.3.2 - Votre analyse du script

Répondre aux questions suivantes	
Combien de bibliothèques sont importées ?	
Quel est le type de la variable résultat ?	
Quel est le type de la variable total ?	
Quel est le type de la variable des ?	
Expliquez la ligne 41 ?	

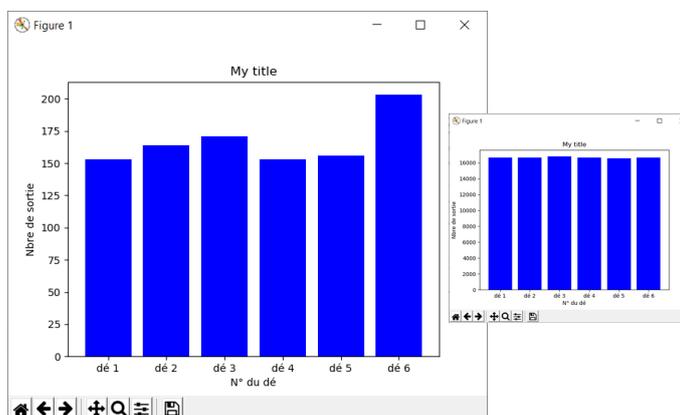
15.2.3.3 - Amélioration du script

Modifier ce script pour laisser le choix à l'utilisateur du nombre de jets avec un input ! Vous devrez probablement supprimer la ligne `plt.ylim(0,60)` mais pour quelle raison ?

Remarque :

Si vous testez 1000 ou 10000 tirages, ça passe !

Pour 100000, j'obtiens le graphique mais après je dois fermer Thonny !!!



15.2.3.4 - Script commenté

Votre script commenté



15.2.4 - Exo : Dessiner des vecteurs avec matplotlib

La commande est quiver. Un vecteur nécessite les coordonnées de son point de départ puis la projection sur l'axe des x et des y.

15.2.4.1 - Énoncé



Dans l'éditeur :

- copier/coller puis exécuter le script suivant :

```
#
import matplotlib.pyplot as plt

# Initialisation
xo = 0 # m
yo = 0 # m
vox = 10 # m.s-1
voy = 20 # m.s-1

fig, ax = plt.subplots()
vecVit = ax.quiver(xo,yo,vox,voy, angles='xy', scale_units='xy', scale=3, label="vitesse", color="purple")
vecVit = ax.quiver(xo+10,yo+10,20,-20, angles='xy', scale_units='xy', scale=2, label="vitesse", color="red")

ax.axis([-1, 20, -5, 15 ])

fig.show()
```

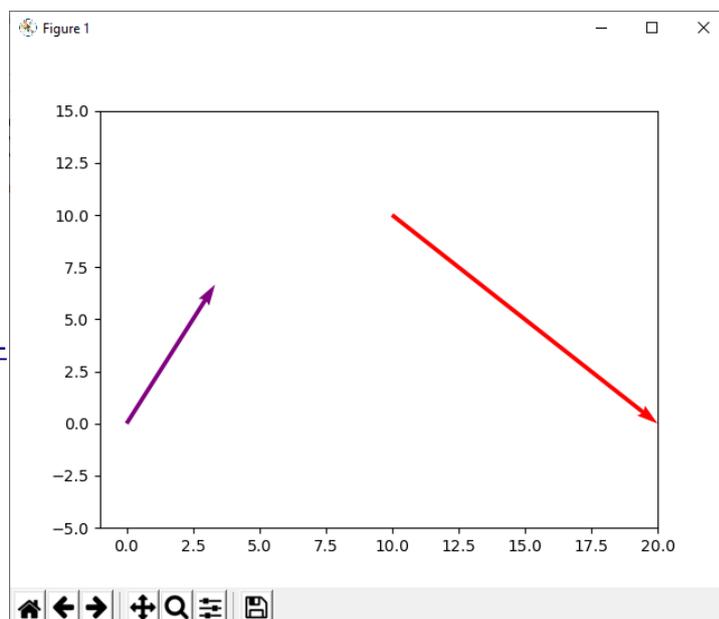
Pour comprendre :

- https://matplotlib.org/3.1.1/api/as_gen/matplotlib.pyplot.quiver.html

Pour impressionner :

- https://matplotlib.org/3.1.1/gallery/images_contours_and_fields/trigradient_demo.html#sphx-glr-gallery-images-contours-and-fields-trigradient-demo-py

15.2.4.2 - Vos remarques





Étude idéalisée d'un tir de projectile dans le vide (idéal car pas de frottement de l'air et gravité constante en tout point de la trajectoire)

Les bases théoriques sont :

- Vidéo démontrant les relations (attention son $z(t)$ est notre $y(t)$)
<https://www.youtube.com/watch?v=UUghMeBqn-g>

- Système d'équations horaires

Equations horaires:

Distance

$$x(t) = V_{0x} = V_0 \cos(\alpha) * t$$

$$y(t) = -\frac{1}{2}gt^2 + V_{0y} = -\frac{1}{2}gt^2 + V_0 \sin(\alpha) * t$$

Vitesse

$$V_x(t) = V_{0x} = V_0 \cos(\alpha) = cte$$

$$V_y(t) = -gt + V_{0y} = -gt + V_0 \sin(\alpha)$$

Accélération

$$a_x(t) = 0$$

$$a_y(t) = -g = cte$$

- A rapprocher des équations horaires définies dans le script par :

```
1 #
2 import matplotlib.pyplot as plt
3
4 # Initialisation
5 xo = 0 # m
6 yo = 0 # m
7 vox = 10 # m.s-1
8 voy = 20 # m.s-1
9 g = 9.81 # N.kg-1
10 m = 1 # kg
11 duree = 15 # s
12 t_cal = [ k/10 for k in range(0,duree*10) ] # liste des temps à calculer
13
14 # Déclaration des fonctions-équations horaires
15 def x(t) :
16     return vox*t + xo
17 def y(t) :
18     return -g*t**2 + voy*t + yo
19 def ax(t) :
20     return 0
21 def ay(t) :
22     return -g
```

Handwritten notes in red:

-) Au départ
- 1.2 position
- 3.4 accélération
- 1, 2, 3, 4 (marking lines 16, 18, 20, 22)

Dans l'éditeur :

- copier/coller puis exécuter le script suivant :

```
#
import matplotlib.pyplot as plt
import numpy as np

# Initialisation
xo = 0 # m
yo = 0 # m
vox = 10 # m.s-1
voy = 20 # m.s-1
g = 10 # N.kg-1
m = 1 # kg
duree = 4 # s

tps = np.linspace(0,duree,20)

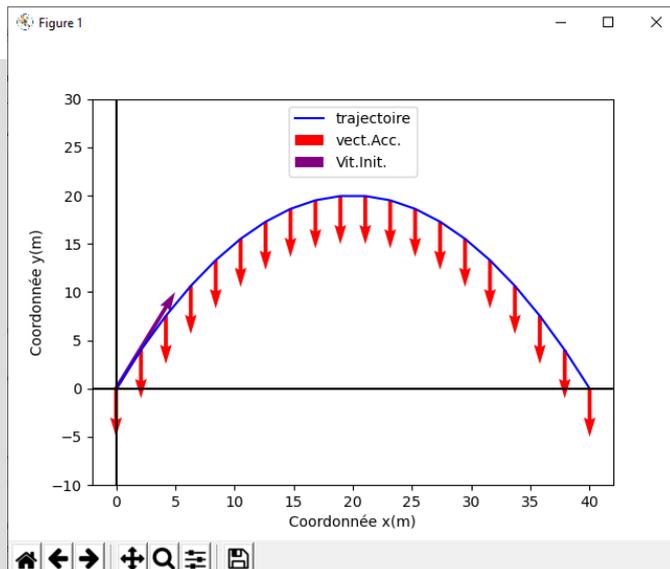
# Déclaration des fonctions-équations horaires
def x(t) :
    return vox*t + xo
def y(t) :
    return -0.5*g*t**2 + voy*t + yo
def ax(t) :
    return 0
def ay(t) :
    return -g

# Corps du programme
X = [ x(t) for t in tps ] # X et Y sont les listes des coordonnées de l'objet
Y = [ y(t) for t in tps ]
Ax = [ ax(t) for t in tps ] # Ax et Ay sont les listes contenant les différentes valeurs de ax et ay
Ay = [ ay(t) for t in tps ]

fig, v = plt.subplots()
trajectoire = v.plot(X,Y, label="trajectoire", color="blue")
vecAcc = v.quiver(X, Y, Ax, Ay, angles='xy', scale_units='xy', scale=2, label="vect.Acc.", color="red")
vecVit = v.quiver(xo, yo, vox, voy, angles='xy', scale_units='xy', scale=2, label="Vit.Init.", color="purple")

v.axhline(y=0,color='black')
v.axvline(x=0,color='black')
v.axis([-2, 42, -10, 30 ])
v.set_xlabel("Coordonnée x(m)")
v.set_ylabel("Coordonnée y(m)")
v.legend(loc='upper center')

fig.show()
```



15.2.4.3 - Votre analyse du script

```

1 #
2 import matplotlib
3 import numpy as np
4
5 # Initialisation des paramètres
6 xo = 0 # m
7 yo = 0 # m
8 vox = 10 # m.s
9 voy = 20 # m.s
10 g = 10 # N.kg-1
11 m = 1 # kg
12 duree = 4 # s
13
14 tps = np.linspace(0, duree, 20)
15
16 # Déclaration des fonctions
17 def x(t) :
18     return xo + vox*t
19 def y(t) :
20     return yo + voy*t - 0.5*g*t**2
21 def ax(t) :
22     return -g
23 def ay(t) :
24     return -g
25
26 # Corps du programme
27 X = [ x(t) for t in tps ]
28 Y = [ y(t) for t in tps ]
29 Ax = [ ax(t) for t in tps ]
30 Ay = [ ay(t) for t in tps ]
31
32 fig, v = plt.subplots()
33 trajectoire = v.plot(X, Y, 'b-')
34 vecAcc = v.quiver(X, Y, Ax, Ay, color='r')
35 vecVit = v.quiver(X, Y, vox, voy, color='m')
36
37 v.axhline(y=0, color='k')
38 v.axvline(x=0, color='k')
39 v.axis([-2, 40, -10, 30])
40 v.set_xlabel("Coordonnée x(m)")
41 v.set_ylabel("Coordonnée y(m)")
42 v.legend(loc='best')
43
44 fig.show()
    
```

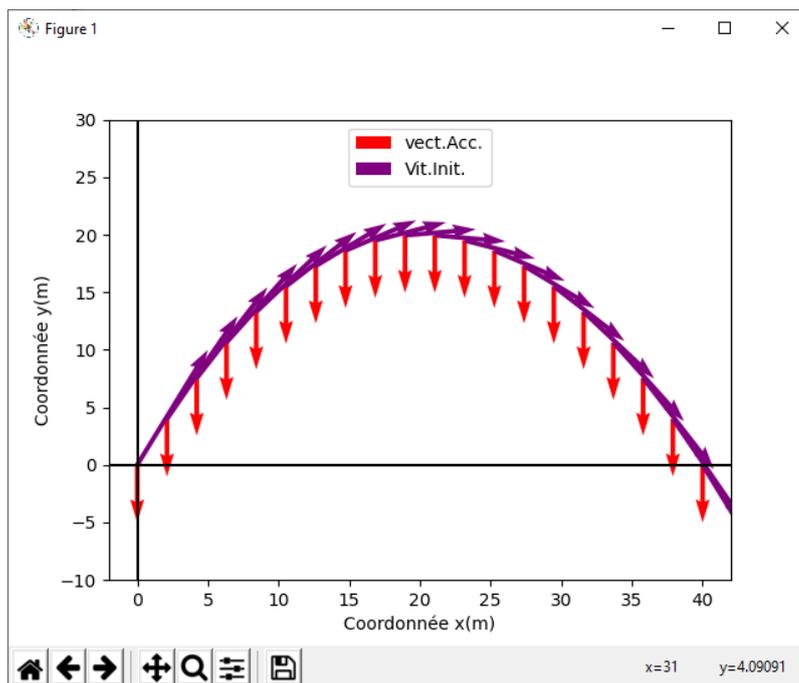
Répondre aux questions suivantes :

Expliquez v_{ox} et v_{oy} aux lignes 8 et 9.	
Expliquez pourquoi vous avez 20 vecteurs rouge.	
Expliquez pourquoi le vecteur vitesse n'est représenté qu'une fois.	

15.2.4.4 - Amélioration du script – ReD

Vous aussi vous trouvez dommage qu'il n'y ait pas la représentation des vitesses pour tous les points X,Y de la trajectoire...

Modifier le script pour obtenir :



15.2.4.5 - Proposition de correction

```
1 #
2 import matplotlib.pyplot as plt
3 import numpy as np
4
5 # Initialisation
6 xo = 0 # m
7 yo = 0 # m
8 vox = 10 # m.s-1
9 voy = 20 # m.s-1
10 g = 10 # N.kg-1
11 m = 1 # kg
12 duree = 4 # s
13
14 tps = np.linspace(0,duree,20)
15
16 # Déclaration des fonctions-équations horaires
17 def x(t) :
18     return vox*t + xo
19 def y(t) :
20     return -0.5*g*t**2 + voy*t + yo
21 def ax(t) :
22     return 0
23 def ay(t) :
24     return -g
25 def vx(t):
26     return vox
27 def vy(t):
28     return -g*t+voy
29
30 # Corps du programme
31 X = [ x(t) for t in tps ] # X et Y sont les listes des coordonnées de l'objet
32 Y = [ y(t) for t in tps ]
33 Ax = [ ax(t) for t in tps ] # Ax et Ay sont les listes contenant les différentes valeurs de ax et ay
34 Ay = [ ay(t) for t in tps ]
35 Vx = [ vx(t) for t in tps ]
36 Vy = [ vy(t) for t in tps ]
37
38 fig, v = plt.subplots()
39 #trajectoire = v.plot(X,Y, label="trajectoire", color="blue")
40 vecAcc = v.quiver(X, Y, Ax, Ay, angles='xy', scale_units='xy', scale=2, label="vect.Acc.", color="red")
41 vecVit = v.quiver(X, Y, Vx, Vy, angles='xy', scale_units='xy', scale=2, label="Vit.Init.", color="purple")
42
43 v.axhline(y=0,color='black')
44 v.axvline(x=0,color='black')
45 v.axis([ -2, 42, -10, 30 ])
46 v.set_xlabel("Coordonnée x(m)")
47 v.set_ylabel("Coordonnée y(m)")
48 v.legend(loc='upper center')
49
50 fig.show()
```

voilà les fonctions de début exercice

15.2.5 - Exo : Créer des animations avec matplotlib.animation

15.2.5.1 - Énoncé



Dans l'éditeur :

- copier/coller puis exécuter le script suivant :

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.animation as animation

k = 2*np.pi
w = 2*np.pi
dt = 0.01

xmin = 0
xmax = 3
nbx = 100

x = np.linspace(xmin, xmax, nbx)

fig = plt.figure() # initialise la figure
line, = plt.plot([],[])
plt.xlim(xmin, xmax)
plt.ylim(-1,1)

# fonction à définir quand blit=True
# crée l'arrière de l'animation qui sera présent sur chaque image
def init():
    line.set_data([],[])
    return line,

def animate(i):
    t = i * dt
    y = np.cos(k*x - w*t)
    line.set_data(x, y)
    return line,

ani = animation.FuncAnimation(fig, animate, init_func=init, frames=100, blit=True,
interval=20, repeat=False)

plt.show()
```

15.2.5.2 - Votre analyse du script

15.2.6 - Exo : Créer des animations en créant un objet puis en l'effaçant

15.2.6.1 - Énoncé



Dans l'éditeur :

- copier/coller puis exécuter le script suivant :

```
import matplotlib.pyplot as plt
import math
plt.ion() # force la mise à jour de l'affichage du graphique apres chaque modification
for i in range(100):
    plt.cla() #efface l'image
    plt.axis([-5, 5, -5, 5])
    plt.quiver(0, 0, 4*math.cos(i/10), 4*math.sin(-i/10), angles='xy', scale=1, scale_units='xy')
    plt.pause(0.01) #la valeur de la pause
plt.show()
```

15.2.6.2 - Votre analyse du script

15.3 - Vos notes

Si besoin

16 - Faire de la physique avec scipy

SciPy est une collection d'algorithmes mathématiques et de fonctions pratiques construits sur l'extension NumPy de Python. Il ajoute une puissance significative à la session Python interactive en fournissant à l'utilisateur des commandes et des classes de haut niveau pour manipuler et visualiser les données.

Pour plus d'info :

<https://docs.scipy.org/doc/scipy/reference/index.html>

16.1 - Commandes

Commande ou Symbole	Opération
<code>from scipy import linregress</code>	Importation du sous-module <code>fftpack</code> pour la transformée de Fourier
<code>from scipy import integrate</code>	... techniques d'intégration
<code>from scipy import interpolate</code>	... techniques d'extrapolation

16.2 - Exercices

16.2.1 - Exo : Extrapoler une courbe en partant de points

16.2.1.1 - Énoncé



Dans l'éditeur :

- copier/coller ce script puis l'exécuter

```
from scipy.interpolate import interp1d
import numpy as np
import matplotlib.pyplot as plt

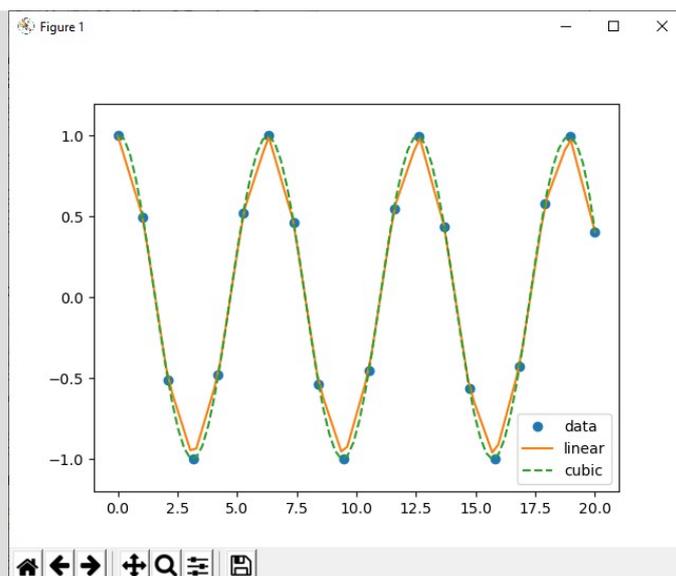
x = np.linspace(0, 20, 20)
y = np.cos(x)

f = interp1d(x, y)
f2 = interp1d(x, y, kind='cubic')

xnew = np.linspace(0, 20, 80)

plt.plot(x,y,'o',xnew,f(xnew),'-', xnew, f2(xnew),'--')
plt.legend(['data', 'linear', 'cubic'], loc='best')
plt.ylim(-1.2,1.2)

plt.show()
```



16.2.1.2 - Votre analyse du script

Répondre aux questions suivantes :

Que contient la liste x ?

Que contient la liste y ?

Que représente graphiquement f ?

Que représente graphiquement f2 ?

16.2.2 - Exo : Calculer une droite de régression en partant de points

16.2.2.1 - Énoncé



Dans l'éditeur :

- copier/coller le script puis exécuter le.

```
#
import numpy as np
import matplotlib.pyplot as plt

from scipy.stats import linregress

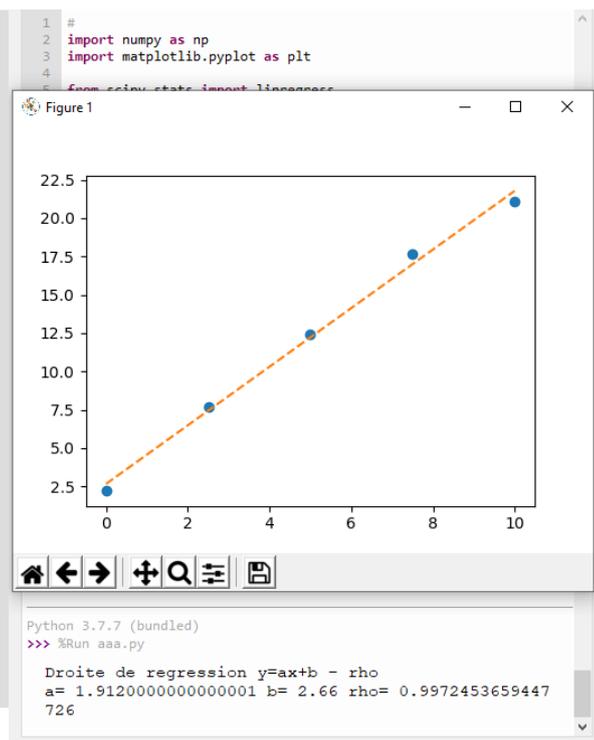
x=np.array([0,2.5,5,7.5,10])
y=np.array([2.2,7.7,12.4,17.7,21.1])

(a,b,rho,_,_)=linregress(x,y)
print( "Droite de regression y=ax+b - rho")
print( 'a=',a,'b=',b,'rho=',rho)

plt.plot(x,y,'o')

y=a*x+b
plt.plot(x,y,'--')

plt.show()
```



16.2.2.2 - Votre analyse du script

Répondre aux questions suivantes :

Que représente une droite de régression ?

Quel sens donnez-vous à rho ?

Expliquez ce que représente graphiquement le premier plt.plot ?

Expliquez ce que représente graphiquement le second plt.plot ?	
--	--

16.3 - Vos notes

Si besoin

Table des matières

1 - Sources inspirantes.....	2
2 - Introduction.....	2
3 - Présentation de l'IDE Thonny.....	3
3.1 - Téléchargement et installation.....	3
3.2 - Ajout des bibliothèques.....	3
3.2.1 - Procédure.....	3
3.2.2 - Liste des bibliothèques à installer.....	4
3.3 - Vos notes.....	4
3.4 - Présentation de l'environnement de développement intégré – EDI ou IDE en anglais.....	5
3.4.1 - Présentation en vidéo.....	5
3.4.2 - Les fenêtres de l'environnement.....	5
3.4.3 - L'exécution d'un programme.....	7
3.5 - Vos notes.....	7
4 - Afficher des données dans le shell et parler à python.....	8
4.1 - Commandes.....	8
4.2 - Exercices.....	8
4.2.1 - Exo – print.....	8
4.2.2 - Exo – print et input.....	9
4.3 - Vos notes.....	10
5 - Stocker des données dans des variables.....	11
5.1 - Commandes.....	11
5.2 - Exercices.....	12
5.2.1 - Exo – variables et affichage.....	12
5.2.2 - Exo – variables et pile mémoire.....	13
5.3 - Vos notes.....	13
6 - Calculer.....	14
6.1 - Opérateurs et symboles.....	14
6.2 - Exercices.....	14
6.2.1 - Exo - calculs et affectation des valeurs.....	14
6.2.2 - Exo – calculs et affectation des valeurs.....	15
6.3 - Vos notes.....	16
7 - Les séries de données – une liste pour vos mesures.....	17
7.1 - Commandes et symboles.....	17
7.2 - Exercices.....	18
7.2.1 - Exo – liste et indice.....	18
7.2.2 - Exo – liste, sous-liste et append.....	19
7.3 - Vos notes.....	20
8 - Le test booléen – avec des si.....	21
8.1 - Commandes.....	21
8.2 - Exercices.....	22
8.2.1 - Exo – Test avec if, si vrai... sinon.....	22
8.2.2 - Exo – Plusieurs tests if / elif.....	24
8.3 - Vos notes.....	26
9 - Boucle bornée - les tâches répétitives mais ayant une fin.....	27
9.1 - Commandes.....	27

9.2 - Exercices.....	27
9.2.1 - Exo – for et suite.....	27
9.2.2 - Exo – for et somme de série.....	30
9.2.3 - Exo – for, listes, gestion des indices et append.....	31
9.2.4 - Exo – for, liste et calculs.....	34
9.3 - Vos notes.....	34
10 - Boucle non bornée – les tâches répétitives dont je ne connais pas le terme.....	35
10.1 - Commandes.....	35
10.2 - Exercices.....	35
10.2.1 - Exo – while.....	35
10.2.2 - Exo – while et listes.....	37
10.3 - Vos notes.....	38
11 - Les fonctions – pour faire comme en maths.....	39
11.1 - Commandes.....	39
11.2 - Exercices.....	39
11.2.1 - Exo – def fonction à une variable.....	39
11.2.2 - Exo – def fonction avec plusieurs variables, input() et int().....	43
11.3 - Vos notes.....	48
12 - Présentation des bibliothèques – bienvenue dans l’Alexandrie du XXIème siècle.....	49
12.1 - Bibliothèques.....	50
12.2 - Commandes.....	50
12.3 - Vos notes.....	51
13 - Import de données d’un fichier csv.....	52
13.1 - Commandes.....	52
13.2 - Exercices.....	52
13.2.1 - Exo : import csv – Importer les valeurs numériques d’un fichier CSV qui a une 1ère ligne d’en-tête.....	52
13.2.2 - Exo : import pandas – Importer données d’un fichier csv.....	56
13.3 - Vos notes.....	59
14 - Utilisation de fonctions en maths et calculs.....	60
14.1 - Commandes.....	60
14.2 - Exercices.....	60
14.2.1 - Exo : Utilisation des fonctions sinus et pi (ou radians) provenant de math.....	60
14.2.2 - Exo : Utilisation des fonctions racine et exponentielle, mais pas que.....	62
14.3 - Vos notes.....	63
15 - Visualisation des données avec matplotlib.....	64
15.1 - Commandes.....	64
15.2 - Exercices.....	64
15.2.1 - Exo : Mise en forme d’un graphique avec matplotlib.....	64
15.2.2 - Exo : Faire des graphiques avec matplotlib et numpy.....	67
15.2.3 - Exo : Créer un programme permettant de visualiser graphiquement un tirage aléatoire avec remise d’un dé à 6 faces.....	69
15.2.4 - Exo : Dessiner des vecteurs avec matplotlib.....	72
15.2.5 - Exo : Créer des animations avec matplotlib.animation.....	77
15.2.6 - Exo : Créer des animations en créant un objet puis en l’effaçant.....	78
15.3 - Vos notes.....	78
16 - Faire de la physique avec scypi.....	79
16.1 - Commandes.....	79

16.2 - Exercices.....	79
16.2.1 - Exo : Extrapoler une courbe en partant de points.....	79
16.2.2 - Exo : Calculer une droite de régression en partant de points.....	80
16.3 - Vos notes.....	81