Pierre Troulet

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

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

Sommaire

* 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)

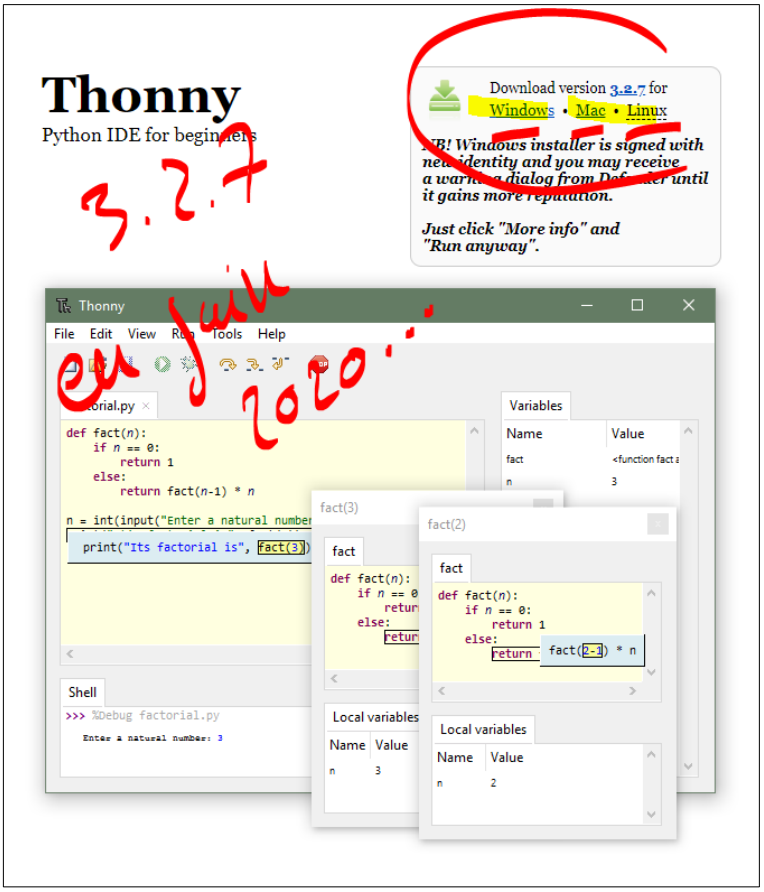
  Au regard de la protection des données personnelles, l'Appli génère des annonces, et l'environnement en ligne pourra poser problème dans le cadre de l'utilisation des bibliothèques d'ou mon conseil d'installer en local un environnement de développement.

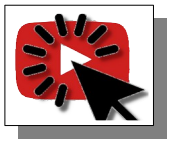
3 - Présentation de l'IDE Thonny

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

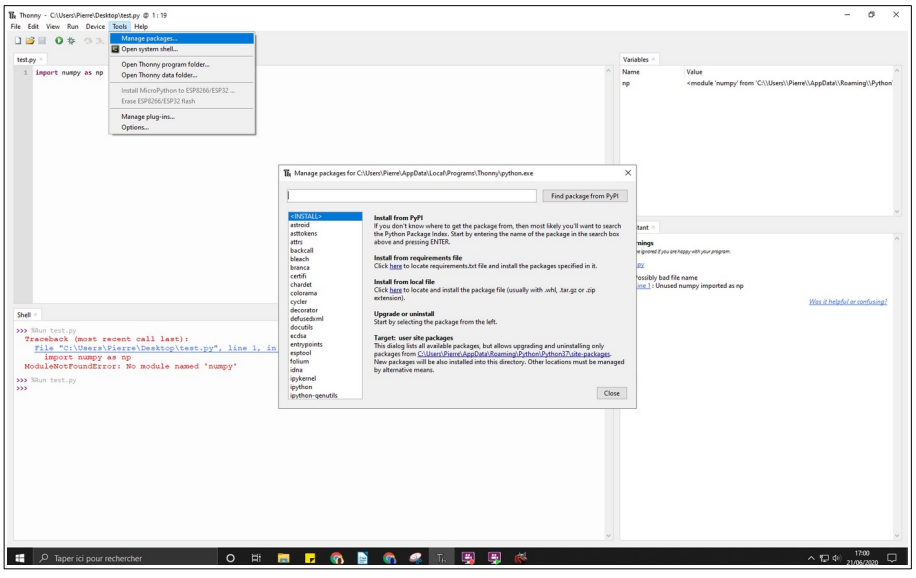
Rq :

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

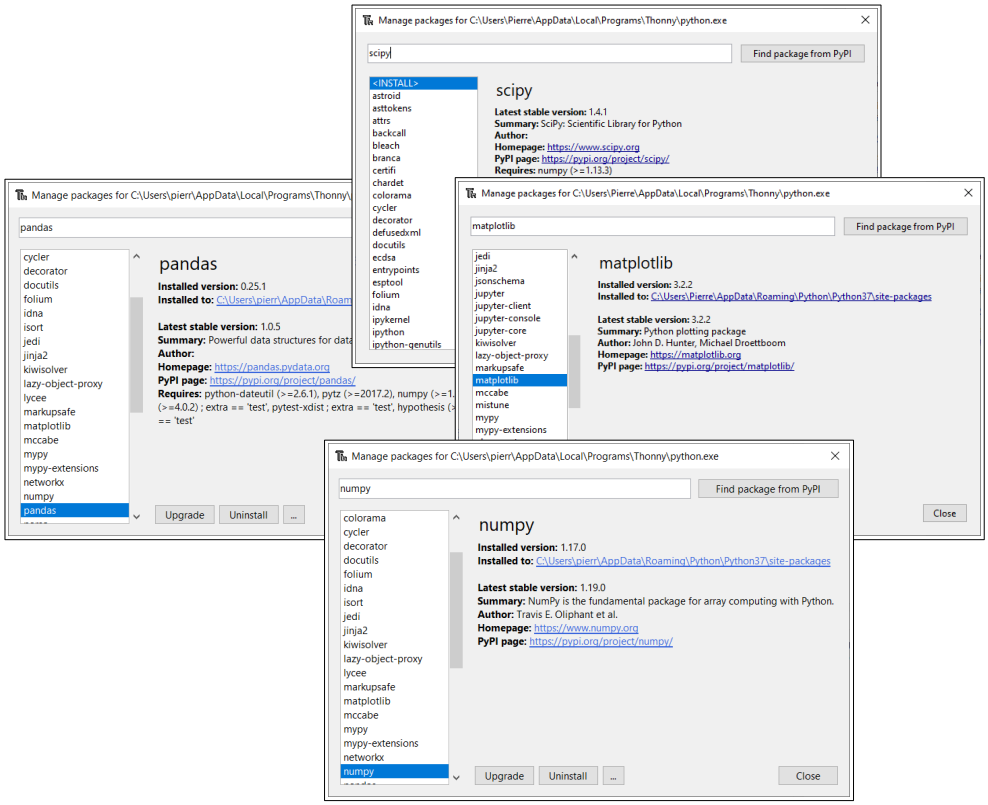
Menu Tools / Manage packages...

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

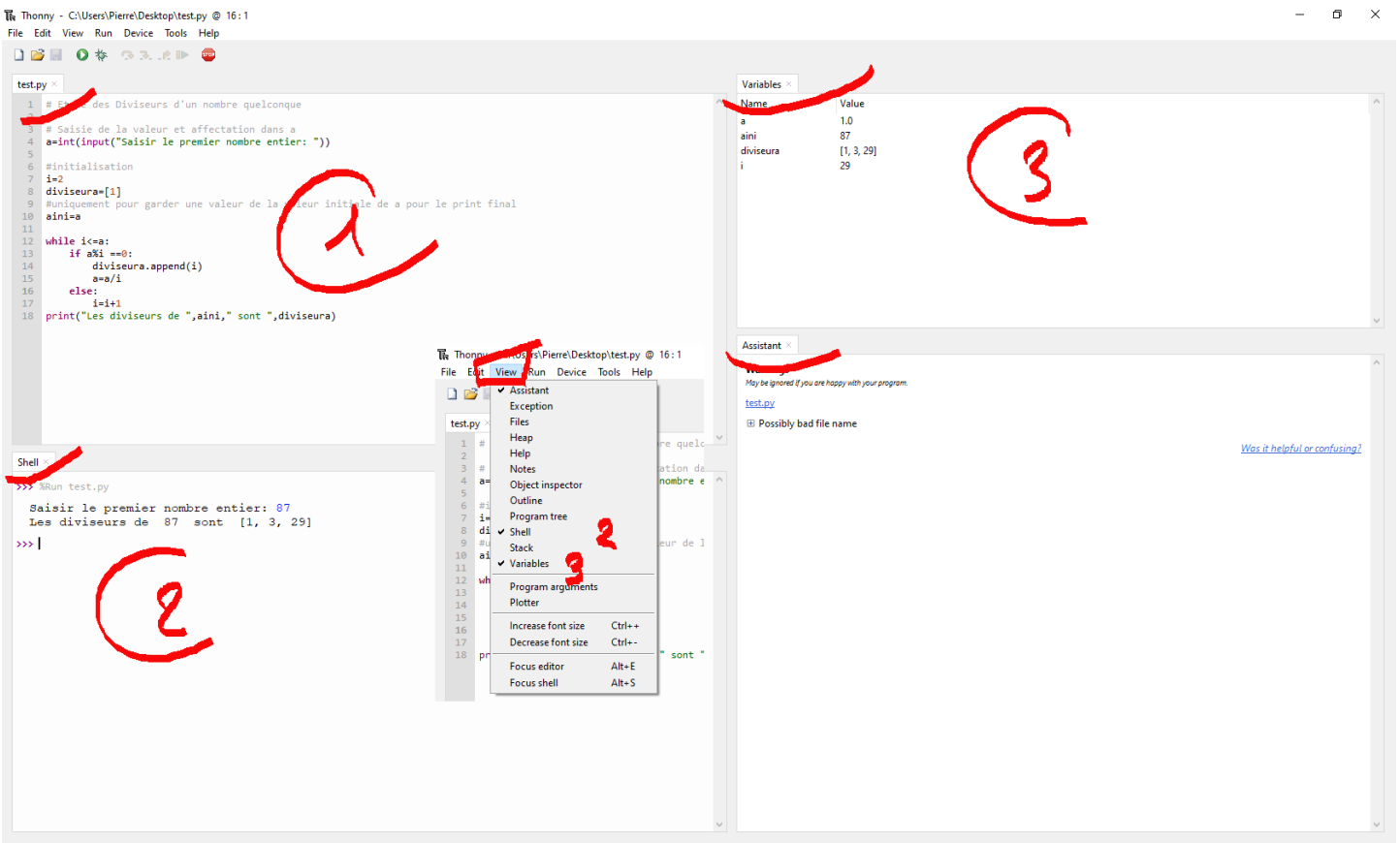
3.4.1 - Présentation en vidéo

Tuto vidéo

- Présentation de l'éditeur
- Visualisation de l'exécution du programme dans le Shell
- Présentation du mode débug 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.



Script du dessous à copier / coller dans la zone 1 puis executer 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
diviseura=[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:
      diviseura.append(i)
      a=a/i
   else:
      i=i+1
print("Les diviseurs de ",aini," sont ",diviseura)
```

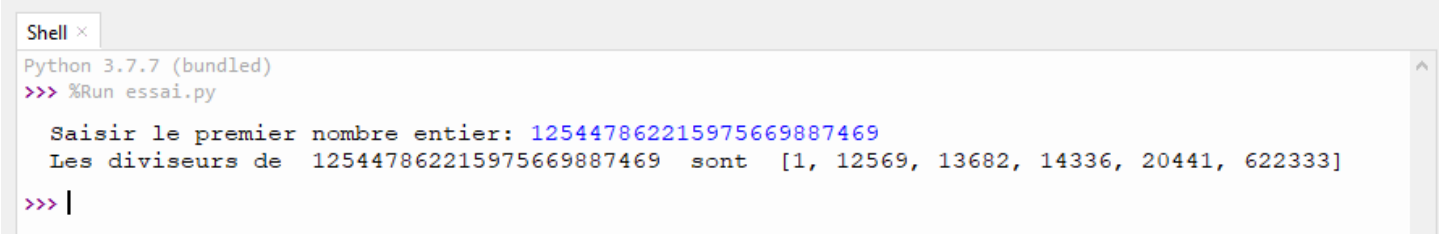
3.4.2.1 - L'éditeur de code – Zone 1

C'est là que l'on travail. 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.

3.4.2.2 - Le shell – Zone 2

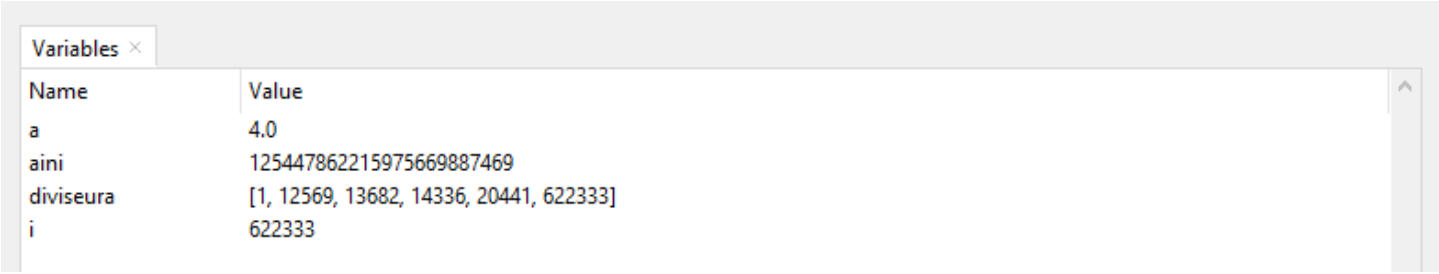
Le shell représente la partie visible du programme (l'enveloppe, la coquille). Dans le shell votre programme s'incarne ;). Noter que l'on peut directement travailler dans le shell !

Clic droit / Clear pour effacer les données du shell

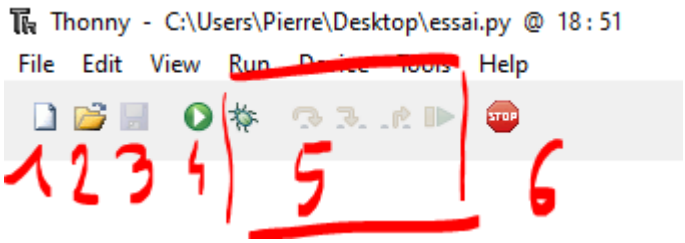


3.4.2.3 - La fenêtre des variables – Zone 3

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



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 débug 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 "

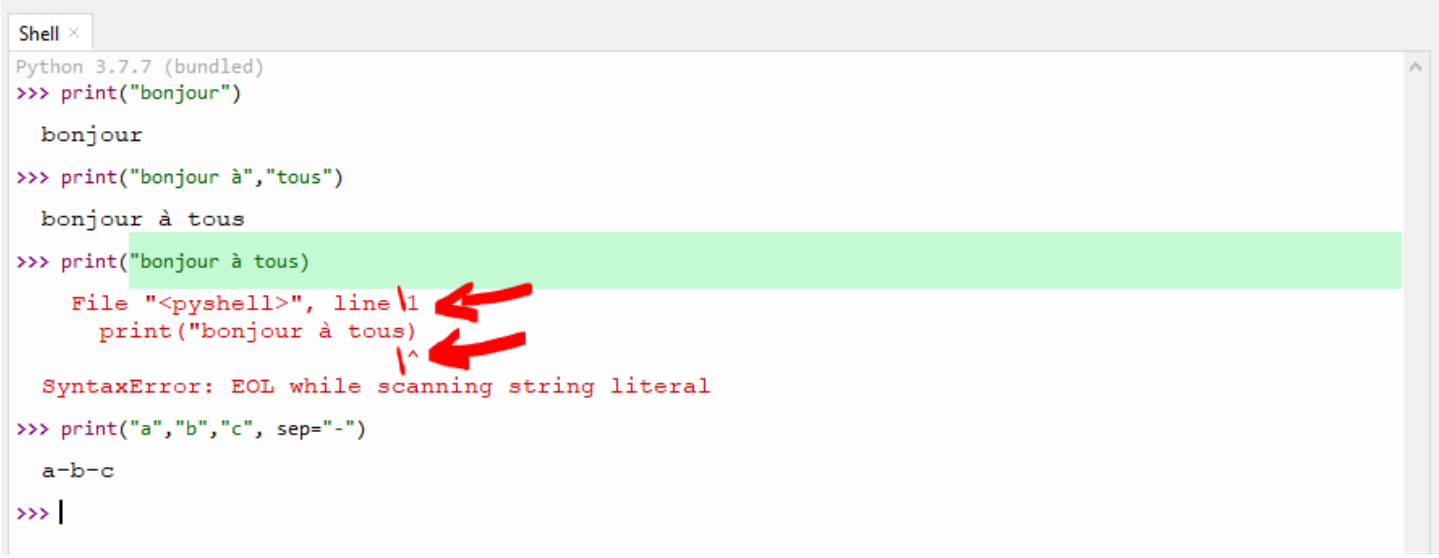
4.2 - Exercices

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 ^



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 **V**ariable= est différent de **v**ariable=.

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

5.2 - Exercices

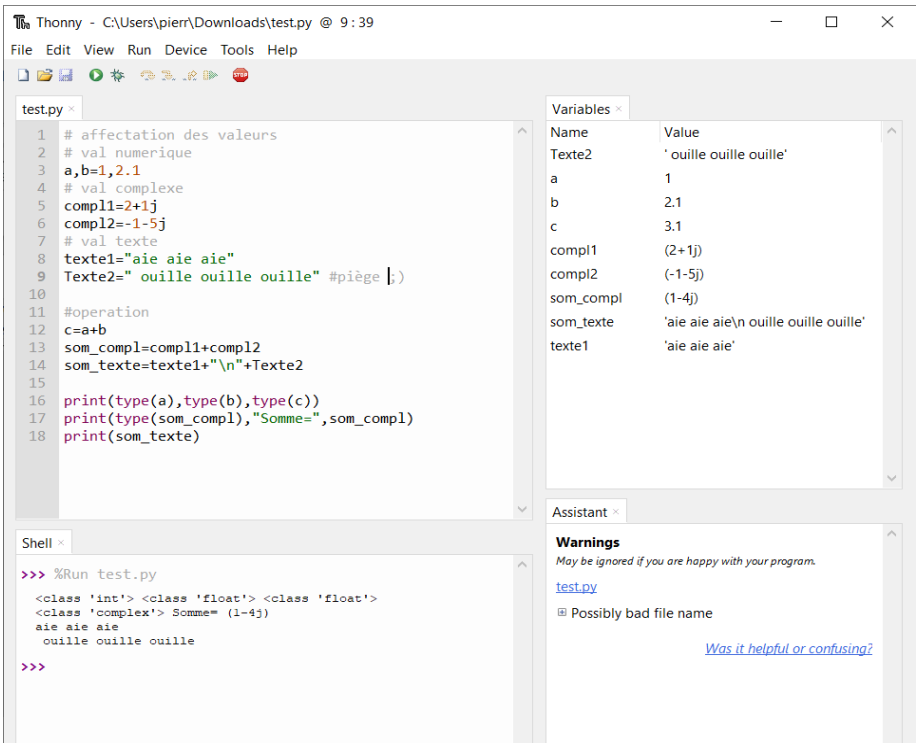
5.2.1 - Exo – variables et affichage

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 :

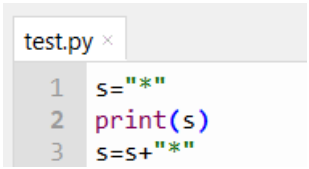
- que le piège est grossier;)
- que le type de c est float car lors de la somme de a et b il ne peut faire autrement à cause de b
- que le i étant souvent utilisé dans les scripts, l'imaginaire est j et qu'il faut déclarer 1j
- que python peut faire des calculs avec du texte !!
- que jeter un œil dans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



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 →



Coller votre proposition dans la zone texte du dessous !

```
#  Votre script que vous collez ici avec une extrême fierté
```

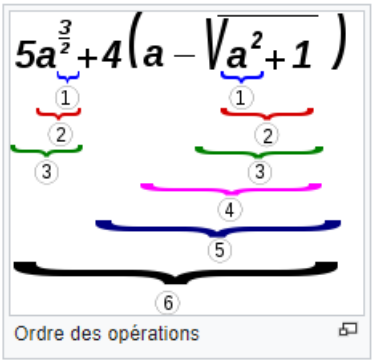
5.3 - Vos notes

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

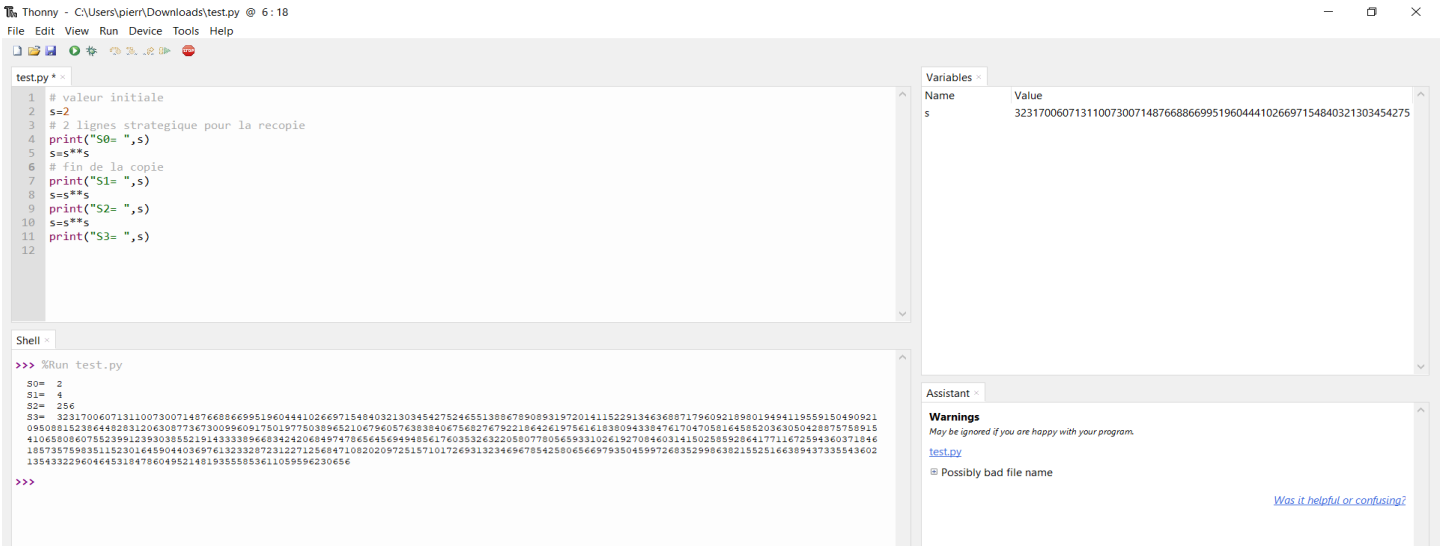
Dans l'éditeur, saisir un script permettant d'afficher les valeurs successives de $S_n = S_{n-1}^{S_{n-1}}$ pour n allant de 0 à 3. (*lire* S_n *est égale à* S_{n-1} *à la puissance* S_{n-1}) sachant que :

$S_0 = 2$ 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



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

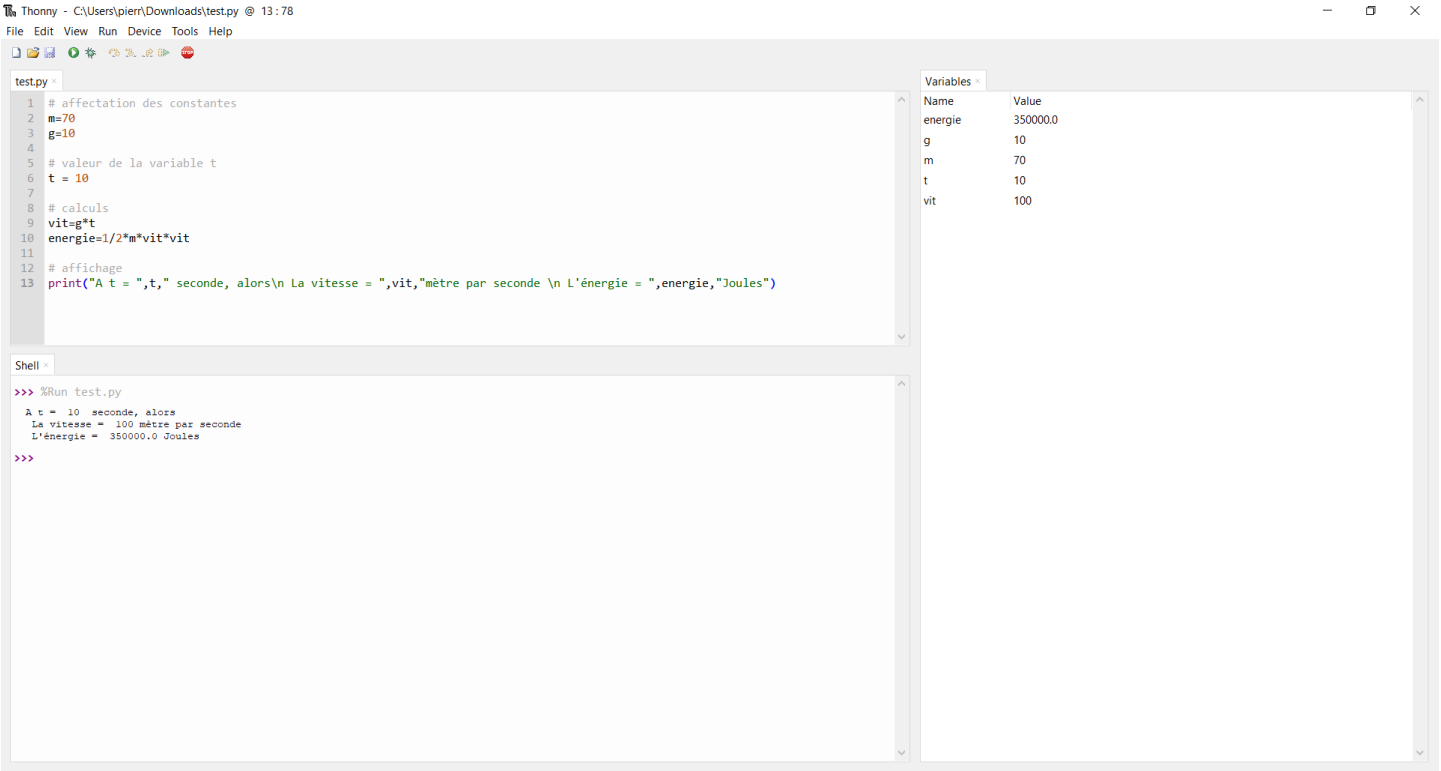
Soit les relations $\left| \begin{array}{c} Energie\,cinétique = \frac{1}{2}.masse.vitesse^2 \\ Ec = \frac{1}{2}.m.V^2 \end{array} \right.$ et $\left| \begin{array}{c} Vitesse = cte\,de\,gravitation.temps \\ V = g.t \end{array} \right.$

On donne $\left| \begin{array}{l} m = 70\,kg \\ g = 10\,m.s^{-2} \end{array} \right.$

Dans l'éditeur, écrire un programme permettant d'afficher pour t = 10s 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



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 - Énonce

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

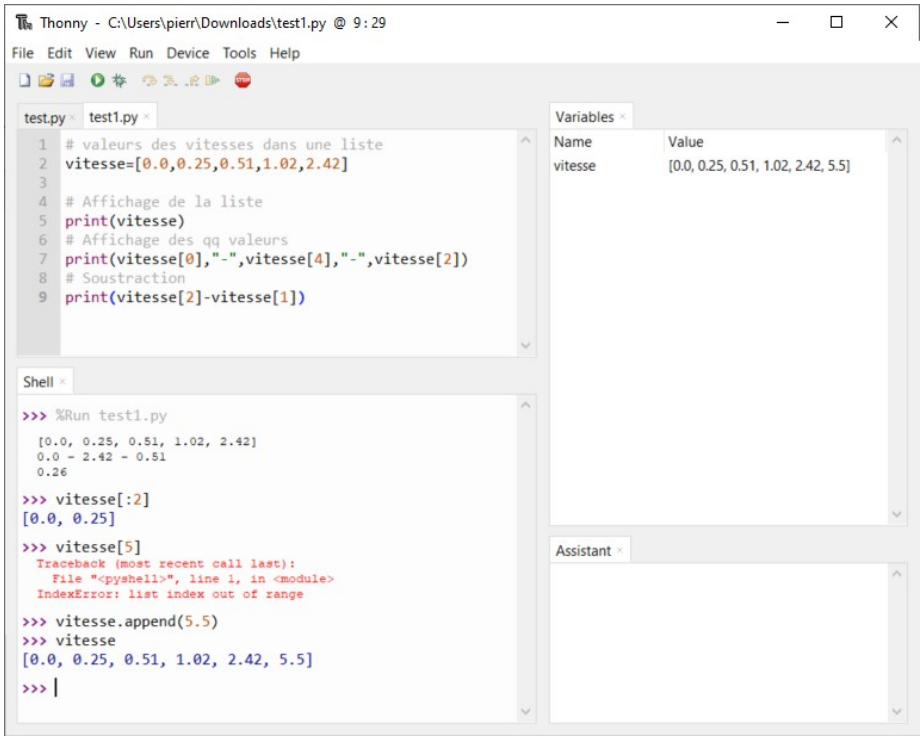
- avec un print(), afficher le contenu de la liste vitesse
- avec un print() faire apparaître la première valeur, la dernière valeur et celle du milieu (la 3ème). Vos valeurs seront séparées par "-"
- avec un print() faire apparaître la différence de vitesse entre 0,51 et 0,25 en appelant les valeurs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

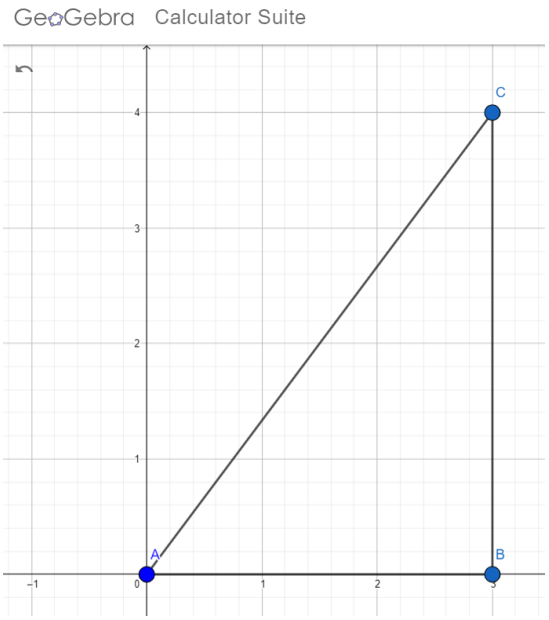
- Noter que dans le shell le print peut être omis
- vitesse[:2] affiche les 2 premières valeurs (indice 0 et 1)
- vitesse[5] n'existe pas car les indices utilisés dans cette liste sont :
  0,1,2,3,4



7.2.2 - Exo – liste, sous-liste et append

7.2.2.1 - Énonce

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 :

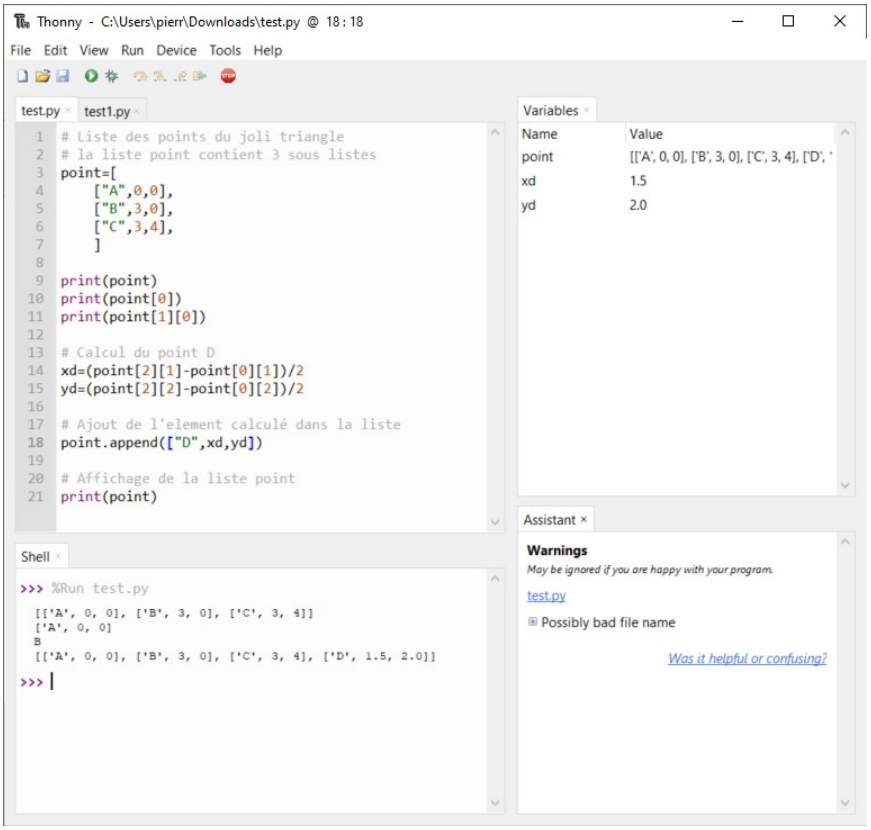
- modifier le script du dessus pour ajouter dans la liste point un nouvel élément - le point D - qui aura les coordonnées du milieu de [A,C].

  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 évidement !!!



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 !

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 • < 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

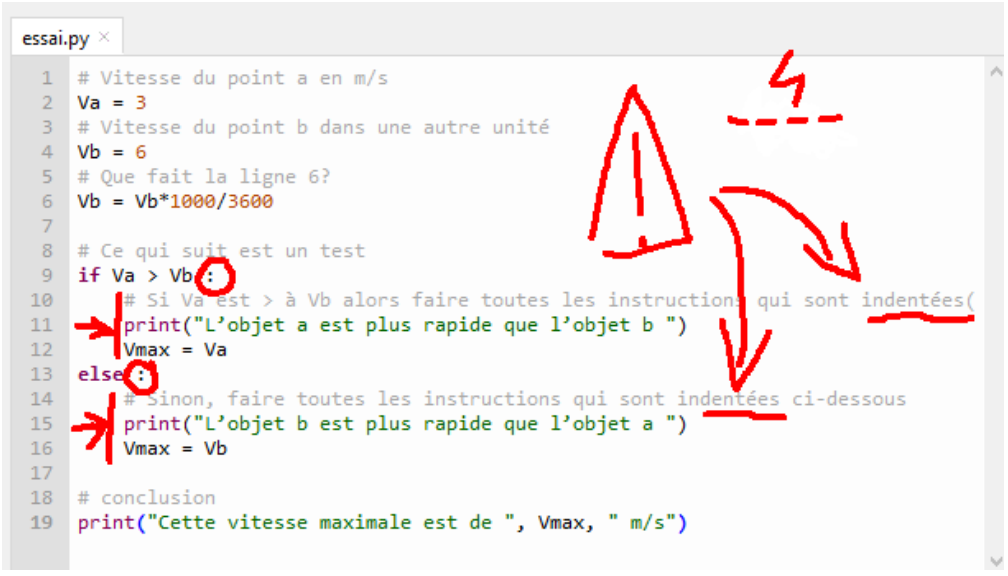
Dans l'éditeur :

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



Dans l'éditeur MAIS en mode **Débug :**
(la bestiole - touche 1)

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

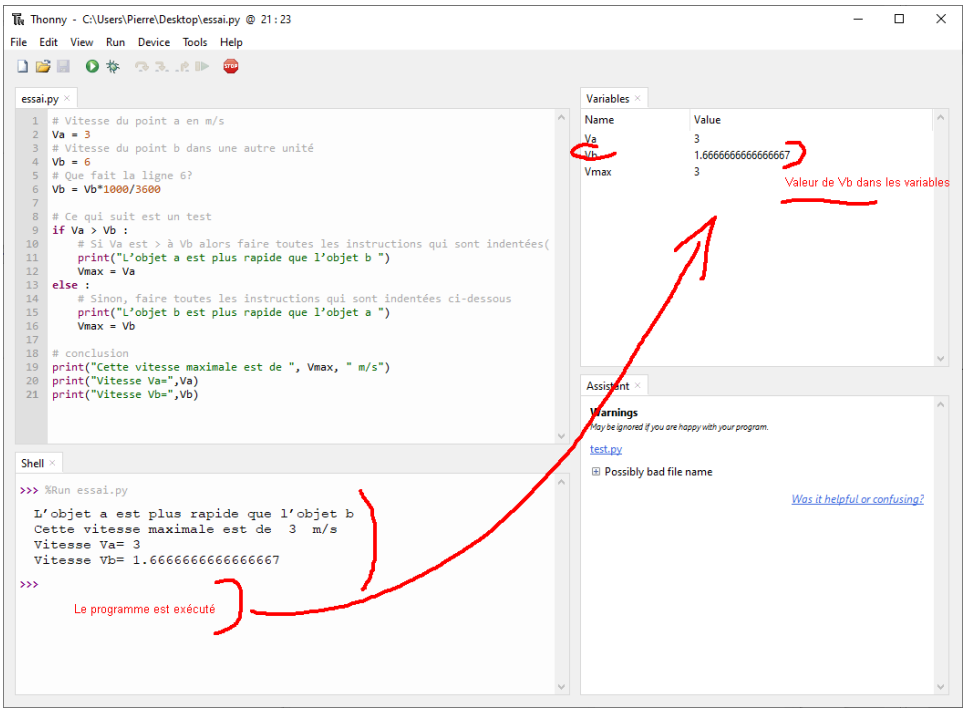
Dans l'éditeur :

- 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

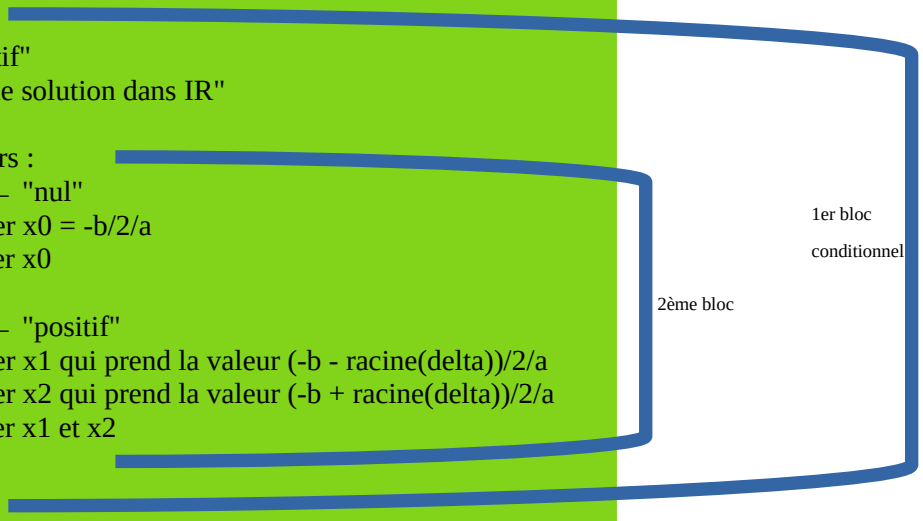
Étude des solutions des fonctions du second degré. Pour la théorie voir ici si besoin :
https://www.maths-et-tiques.fr/telech/Secondegre2ESL.pdf

En dessous nous étudions : $f(x)=2x^2+2x-12$ *donc* (*nous en déduisons que*) $a=2$, $b=2$ *et* $c=-12$

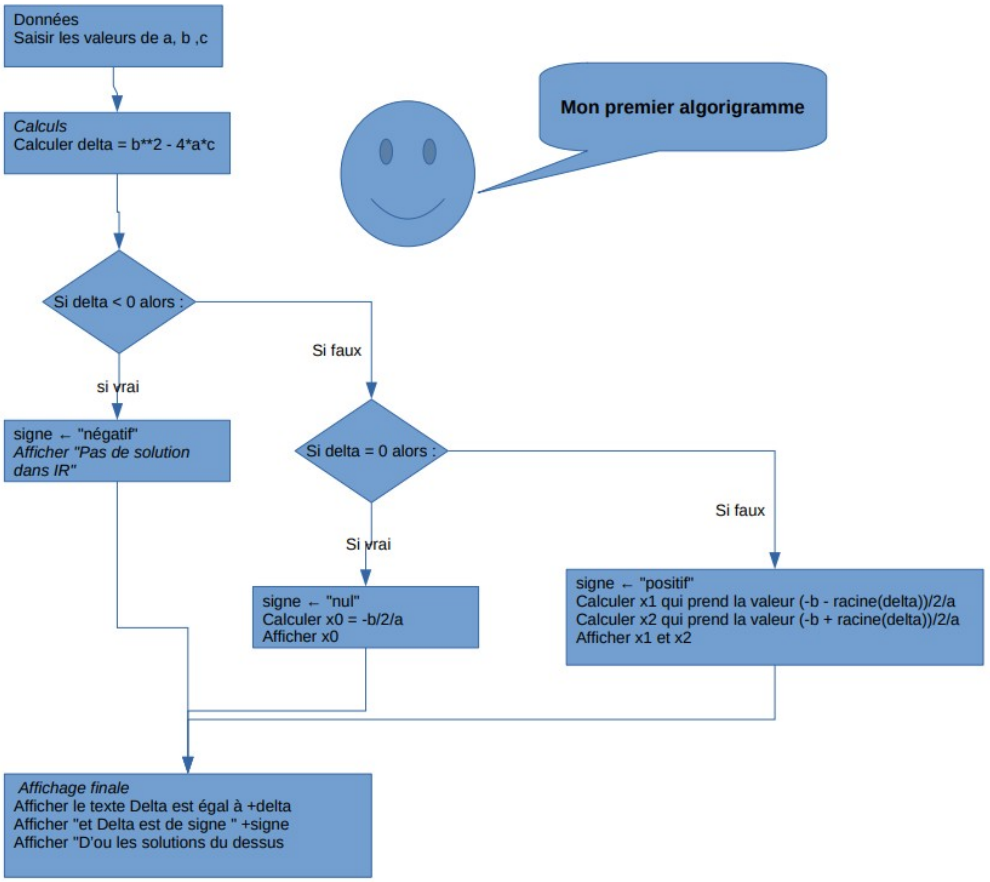
Attention, vous avez besoin de la **racine carrée** dans cet exercice donc dans l'état actuel de nos connaissances je rappelle que $\sqrt[2]{4}=4^{0.5}=2$ donc pour le calcul de la racine nous utiliserons la **puissance 0,5** !

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 - 4*a*c

Test(s)
Si delta < 0 alors :
        signe ← "négatif"
        Afficher "Pas de solution dans IR"
sinon :
        Si delta = 0 alors :
                signe ← "nul"
                Calculer x0 = -b/2/a
                Afficher x0
        sinon :
                signe ← "positif"
                Calculer x1 qui prend la valeur (-b - racine(delta))/2/a
                Calculer x2 qui prend la valeur (-b + racine(delta))/2/a
                Afficher x1 et x2
        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
```



Attention ceci n'est pas un script mais un algorigramme



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

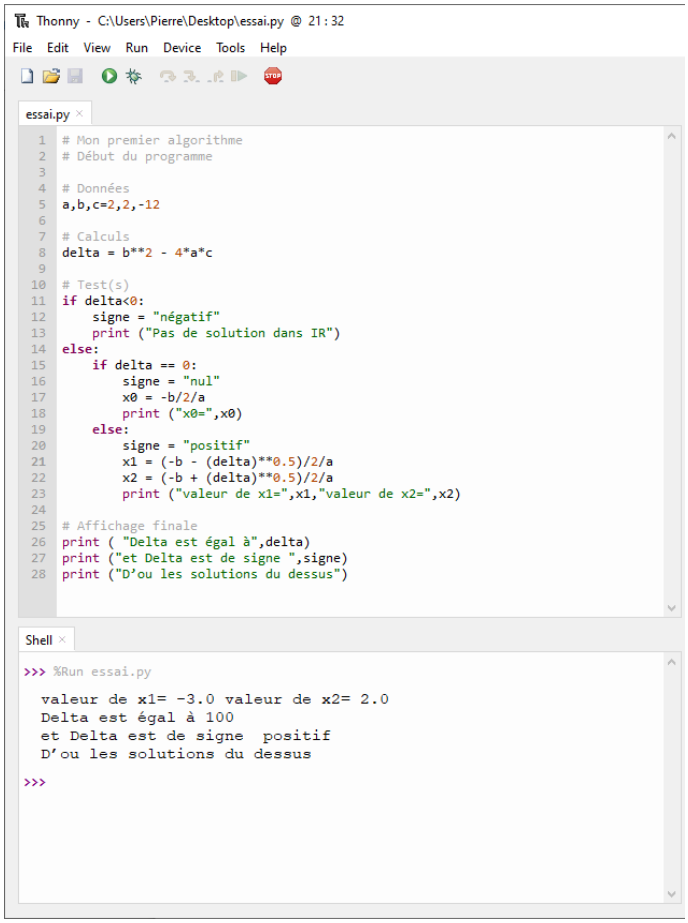


Figure 1: 2 fonctions {SI + SINON} imbriquées

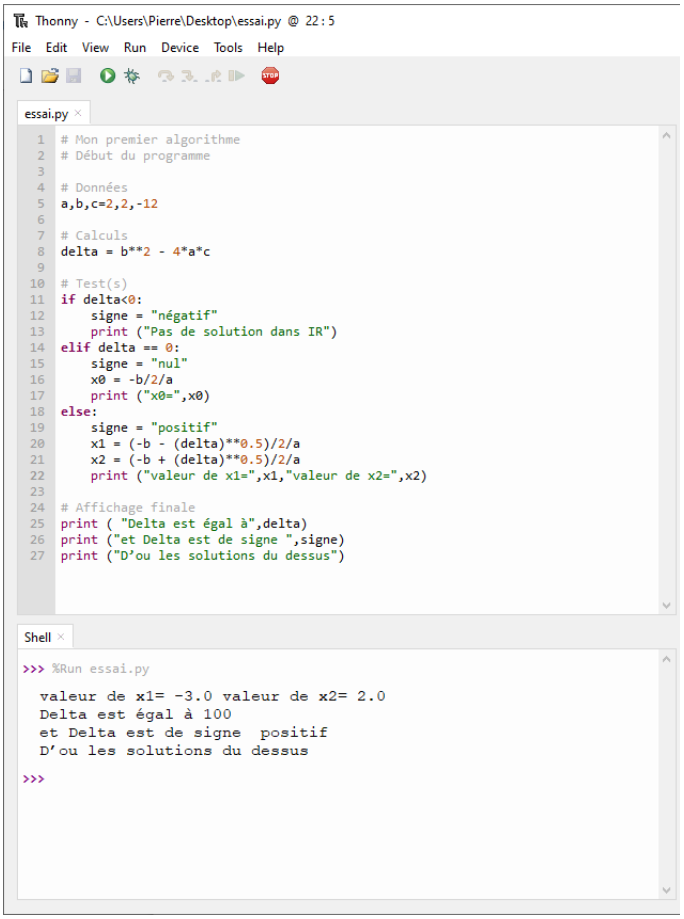


Figure 2: Fonction {SI + SINON-SI + SINON}

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'ou 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
for	Pour ...
in	… Allant
:	Début du bloc après les :
indentation	4 espaces
len() range() ou range(start, stop, step)	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 artihmétique : $Premier\,terme\ A_0 = 2$ $Raison\ r = 3$ $Relation\,de\,reccurence\ A_n = A_{n-1} + r$	*Suite géométrique* : $Premier\,terme\ G_0 = 2$ $Raison\ q = 3$ $Relation\,de\,reccurence\ G_n = G_{n-1} * q$

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 géometrique
# 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. artihmétique=",A)
print("Valeur finale de S. géométrique=",G)
```

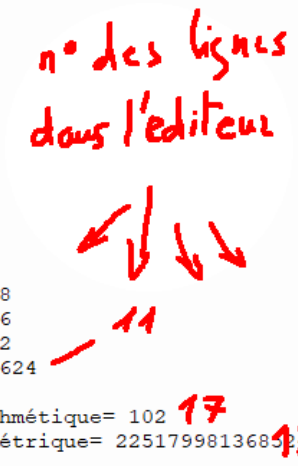
essai.py

```
1  # Suite arithmétique et géometrique
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. artihmétique=",A)
18 print("Valeur finale de S. géométrique=",G)
```

Mon shell me donne →

Shell

```
 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
Valeur finale de S. artihmétique= 102
Valeur finale de S. géométrique= 2251799813685248
>>>
```



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. artihmétique= 102 *(ici c'est bien la dernière valeur calculée)*
  Valeur finale de S. géométrique= 2251799813685248 *(idem)*

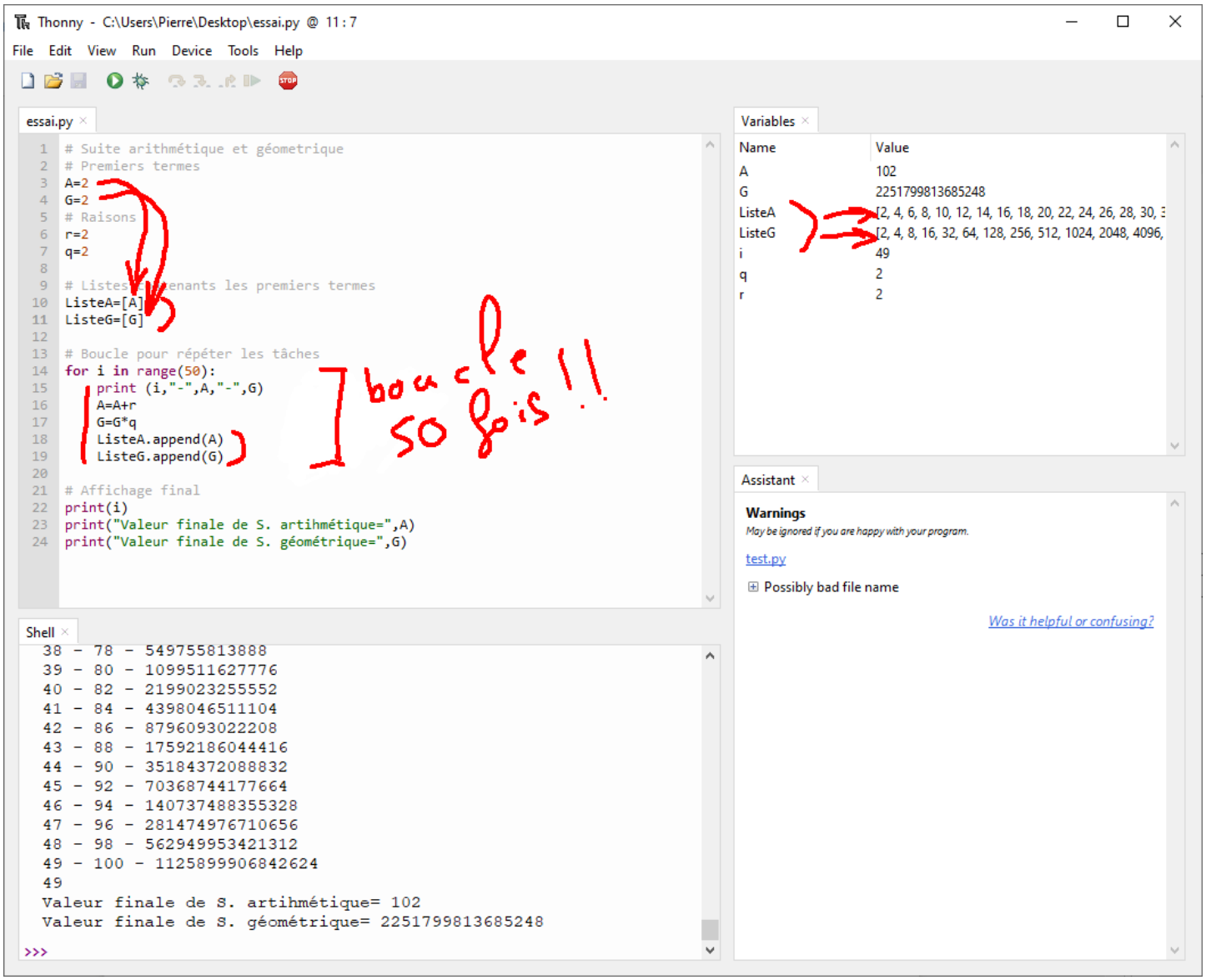
Dans l'éditeur :

- 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



Rq :

Oui je sais… ListeA et ListeG contienne 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 affirmation 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.

$$1 + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \frac{1}{5^2} + \cdots = \sum_{k=1}^{+\infty} \frac{1}{k^2} = \frac{\pi^2}{6}$$

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;)

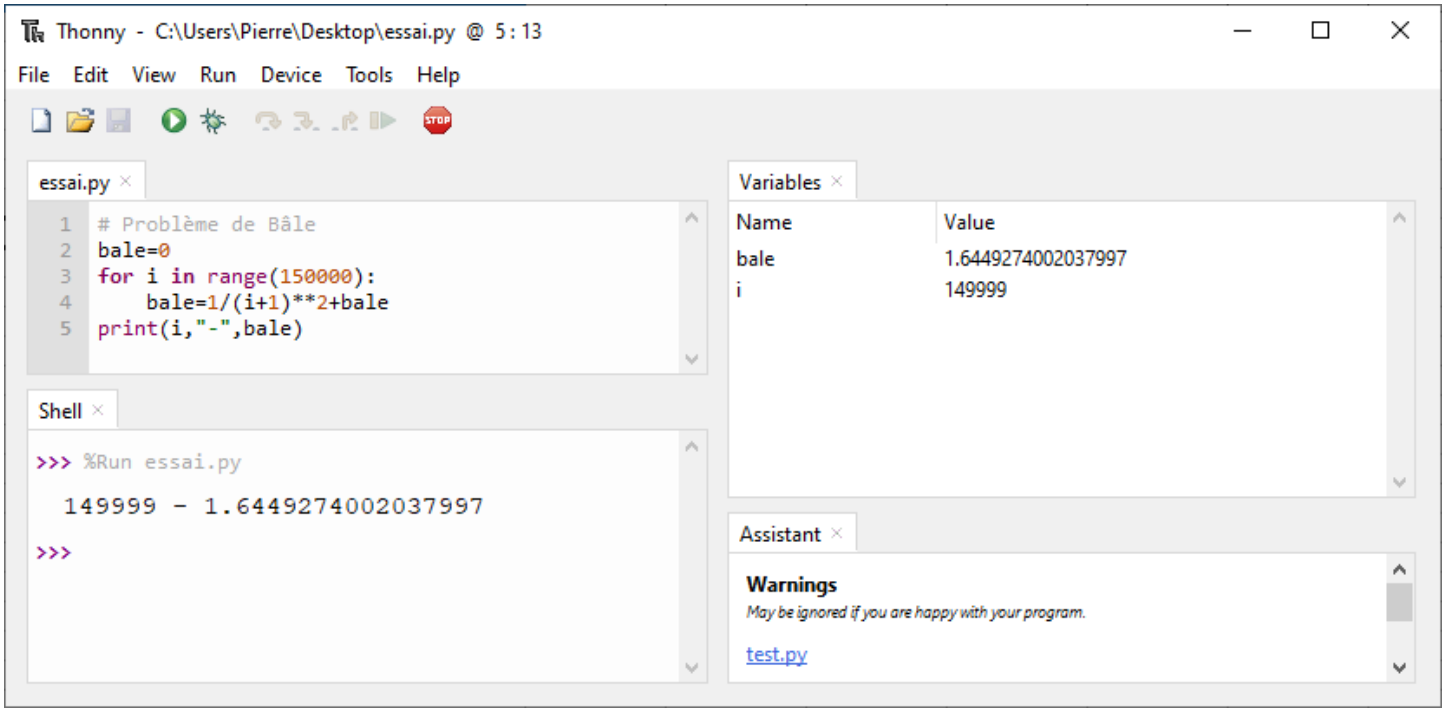
  Je donne $\frac{\pi^2}{6} = 1{,}6449340668\ldots$

* 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



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 parallélépipède 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 :

Vitesse moyenne	*Accélération moyenne*
$v[i]=\frac{x[i+1]-x[i-1]}{t[i+1]-t[i-1]}$	$a[i]=\frac{v[i+1]-v[i-1]}{t[i+1]-t[i-1]}$

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 !

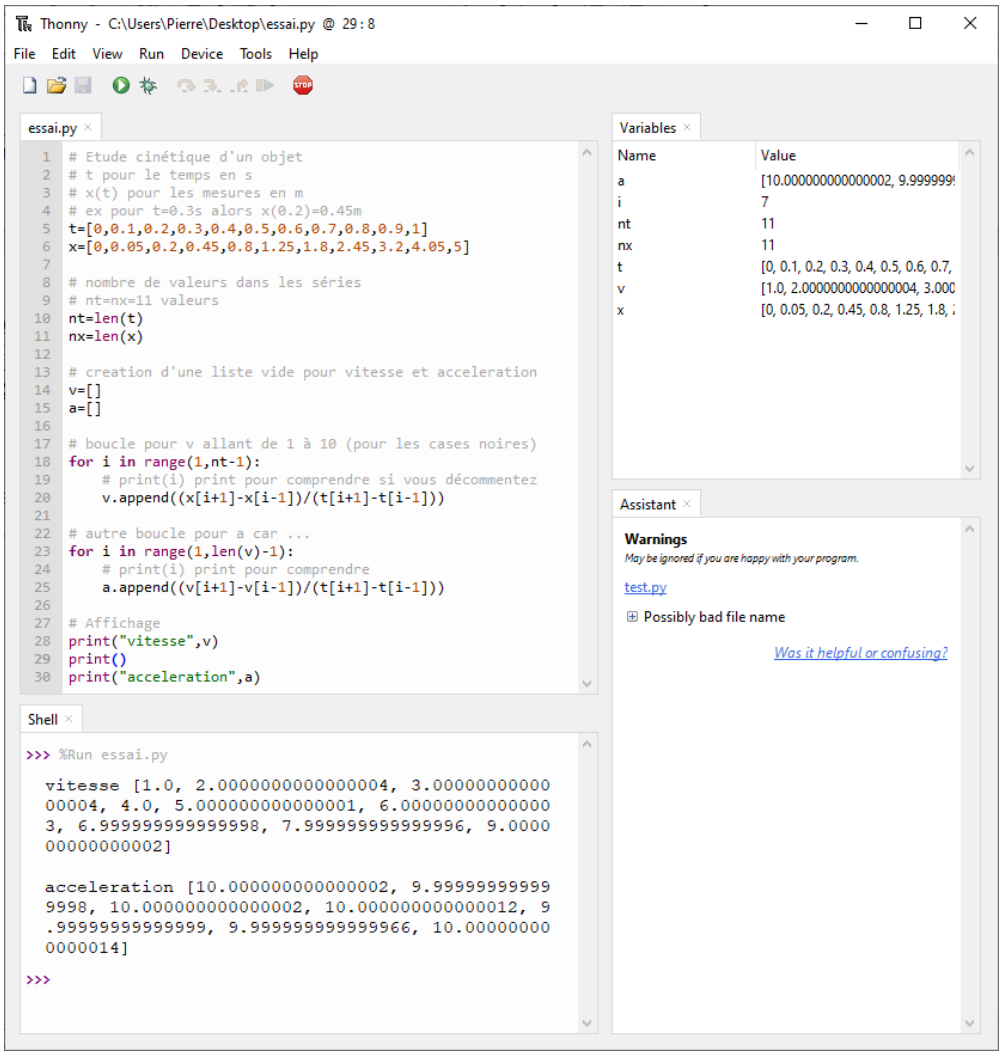
indice i	0	1	2	3	4	5	6	7	8	9	10
t en s	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
x en m	0	0,05	0,2	0,45	0,8	1,25	1,8	2,45	3,2	4,05	5
v en m/s											
a en m/s²											

- 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(delat_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.

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 intiale
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 ! Un 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 :

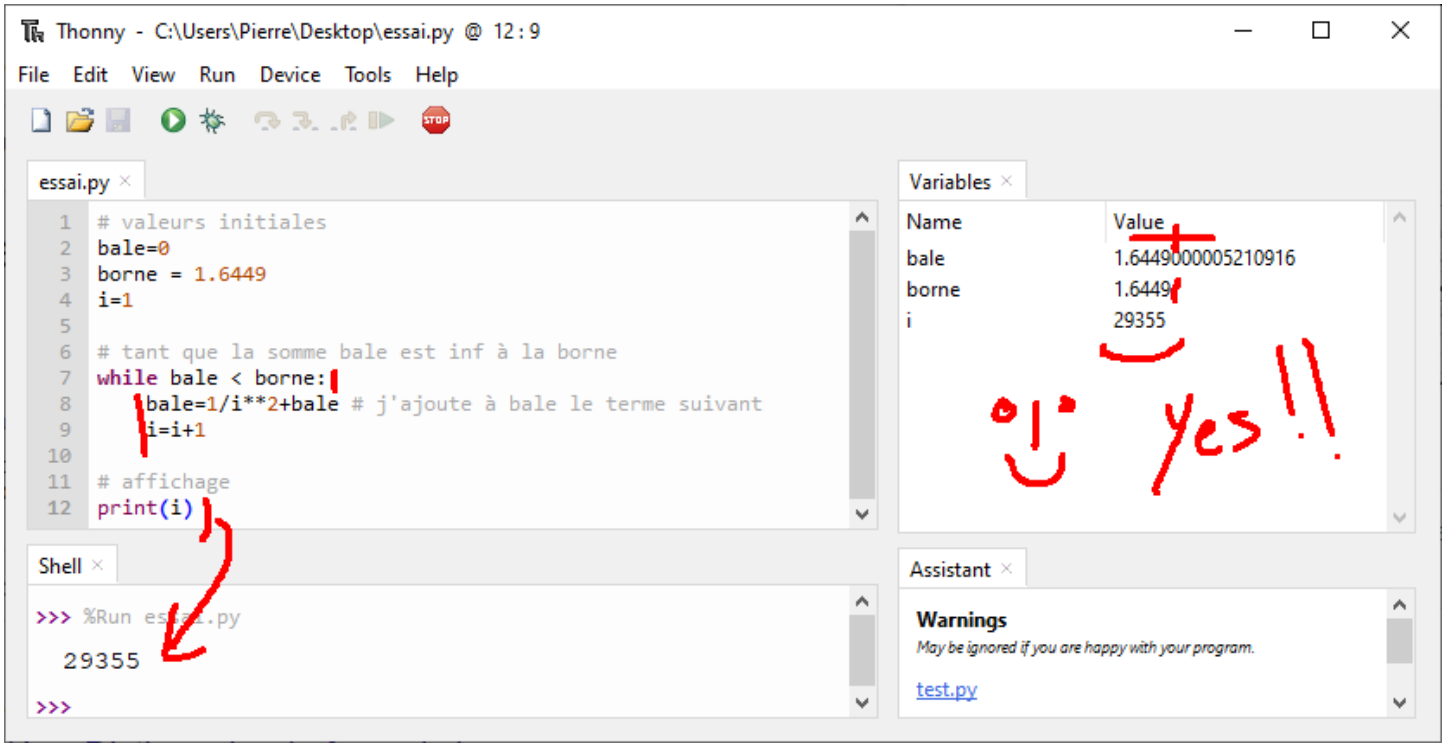
$$1+\frac{1}{2^2}+\frac{1}{3^2}+\frac{1}{4^2}+\frac{1}{5^2}+\cdots=\sum_{k=1}^{+\infty}\frac{1}{k^2}=\frac{\pi^2}{6}$$ avec $\frac{\pi^2}{6}=1{,}6449340668\ldots$

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



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 !?**

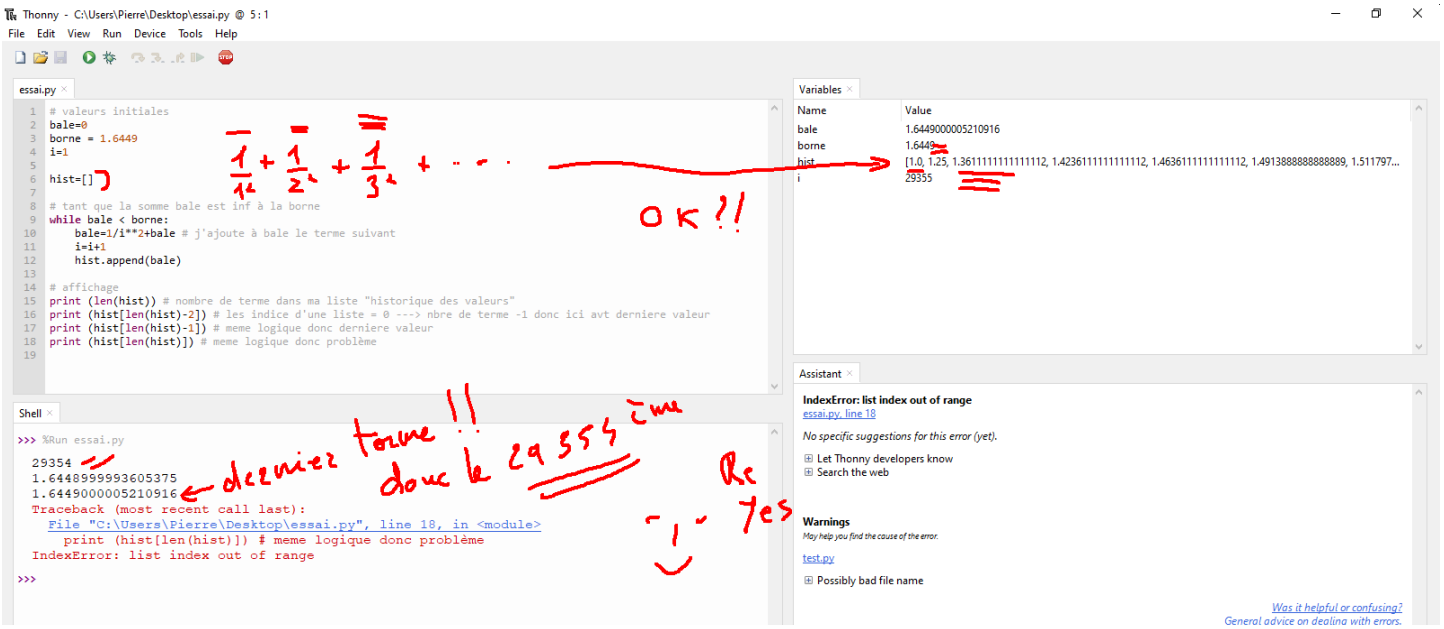
10.2.2 - Exo – while et listes

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
```



- 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**
def	Abréviation de define en EN = définition en FR
return	Retourne le résultat de la fonction (si pas de paramètres alors pas de return)
:	comme d'hab
indentation	idem
input()	Demande de saisie – la valeur saisie sera de type texte
int()	Transforme le type d'une variable en valeur numérique entière
float()	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 $f(x) = \dfrac{2x^4 - \frac{5}{7}}{x+3}$

```
# 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.
  Évidement 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

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

# boucle pour avoir des x de -10 à 10
for x in range (-10,11):
    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 :

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

# boucle pour avoir des x de -10 à 10
for x in range (-10,11):
    # le si pour supprimer la valeur interdite
    if x==-3:
        x+1
    else:
        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

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

# déclaration de ma liste vide pour mémo les points
point=[]

# boucle pour avoir des x de -10 à 10
for x in range (-10,11):
    # le si pour supprimer la valeur interdite
    if x==-3:
        x+1
    else:
        point.append([x,f(x)])
        print(x,"-",f(x))

print(point)
```

Et j’obtiens cela

Shell

```
>>> %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 :

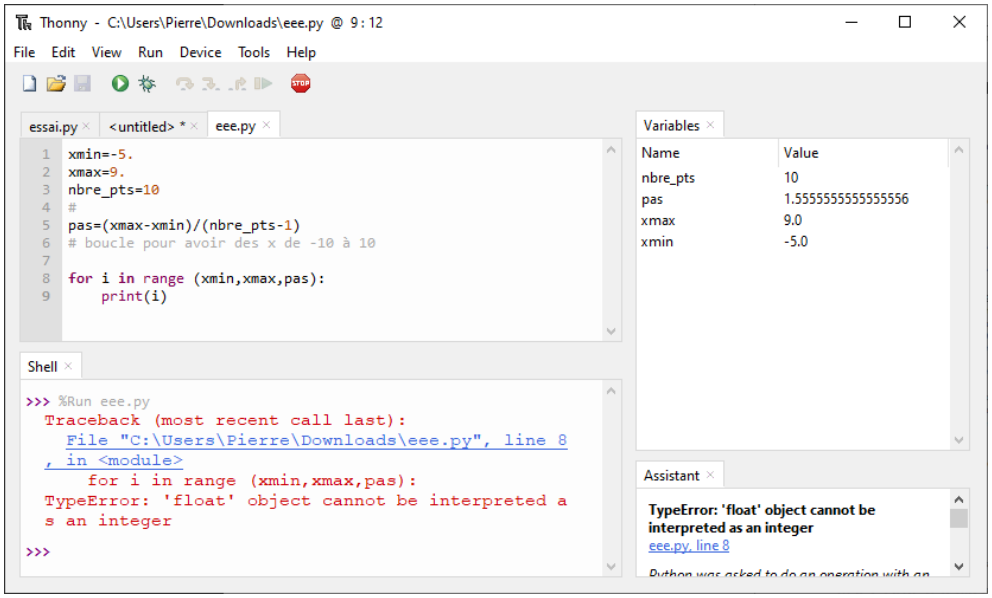
Créer une boucle affichant les valeurs des x désirées. Je déclare des valeurs arbitraires pour xmin, xmax, nbre de points

Essai n°1 :

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

=>

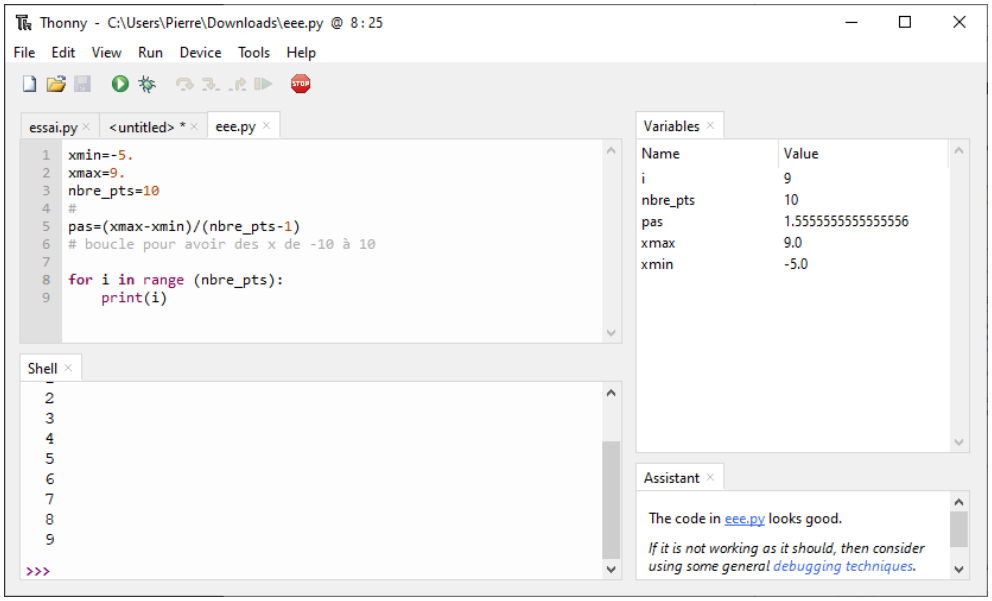
Ne fonctionne pas ligne 8 car i est de type "int" et la valeur de mon pas est "float"



Piste n°1 :

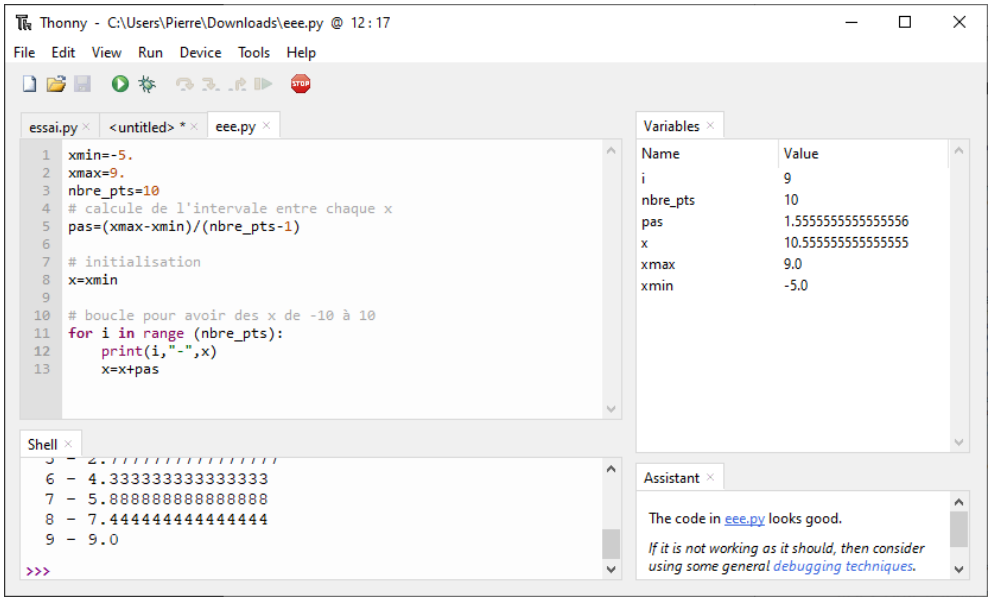
Utiliser le nombre de points dans la boucle car lui sera toujours entier !

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



Essai n°2 :

Calculer tous les x, donc dans la boucle

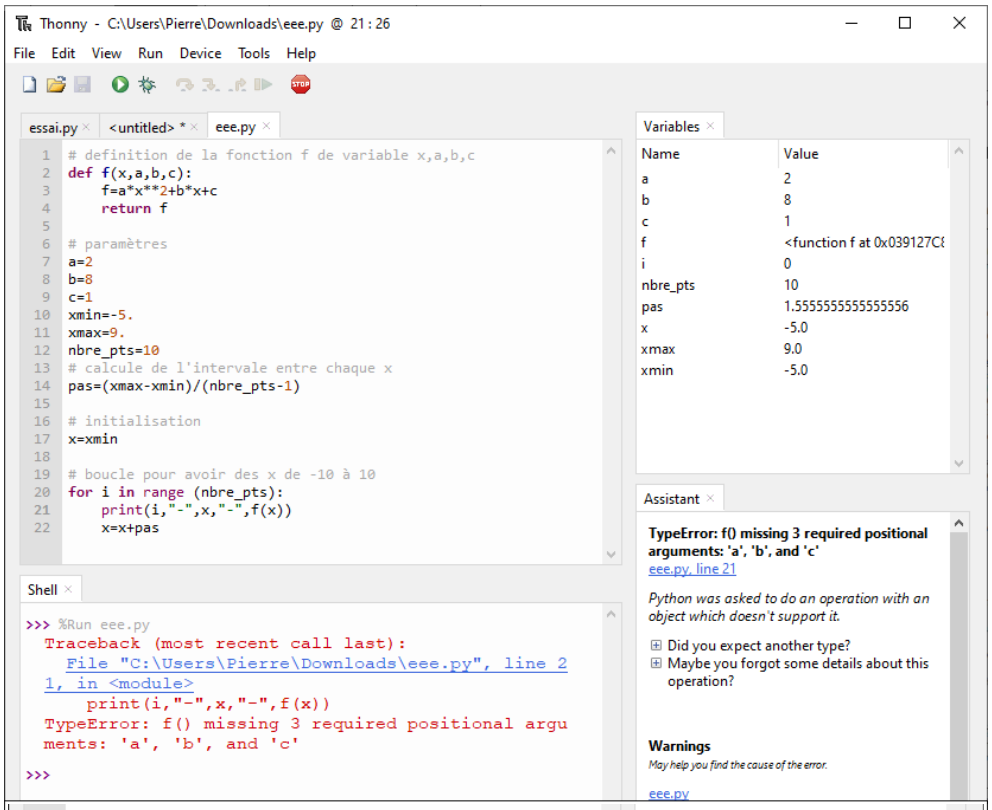


Essai n°3 :

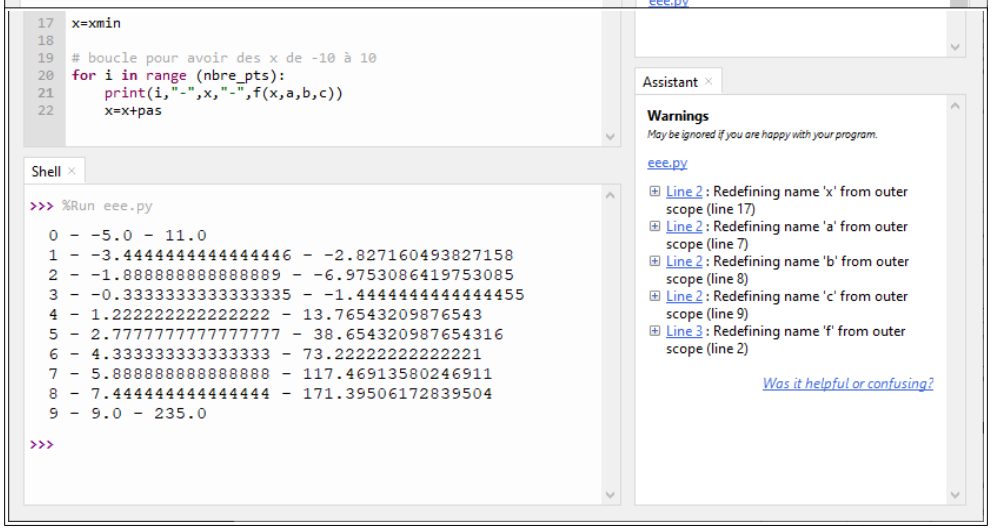
Définir la fonction avec ses paramètres afin de pouvoir ajouter dans ma liste les valeurs de f(x)

=> Erreur indiquée ligne 21!

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



Erreur corrigée ligne 21



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…

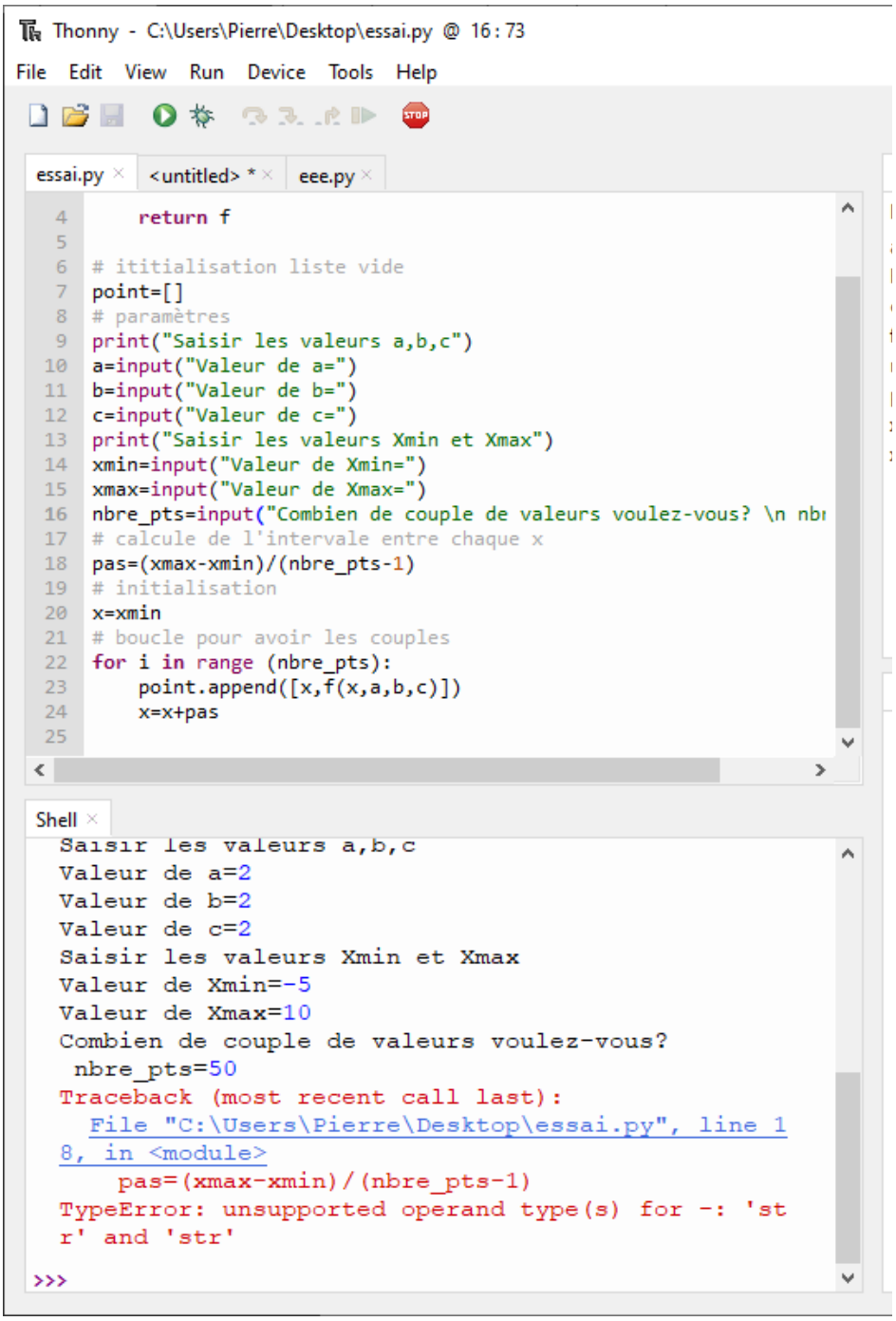
Evidement, toutes mes variables numériques via input auront le même problème !!!!

Donc :

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

MAIS

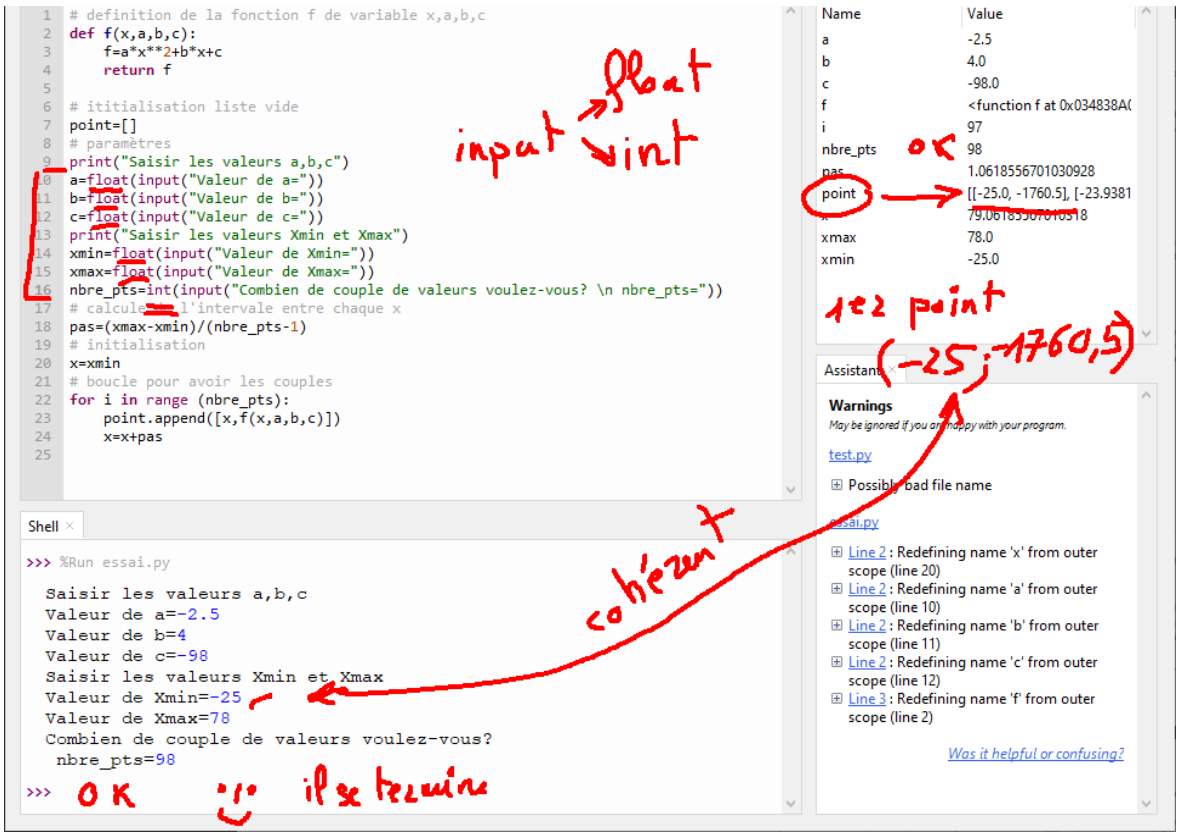
nbre_pts de type **int()**



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

# ititialisation 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="))
# calcule 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
```



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 différentiées
  m.sin() et r.sin()

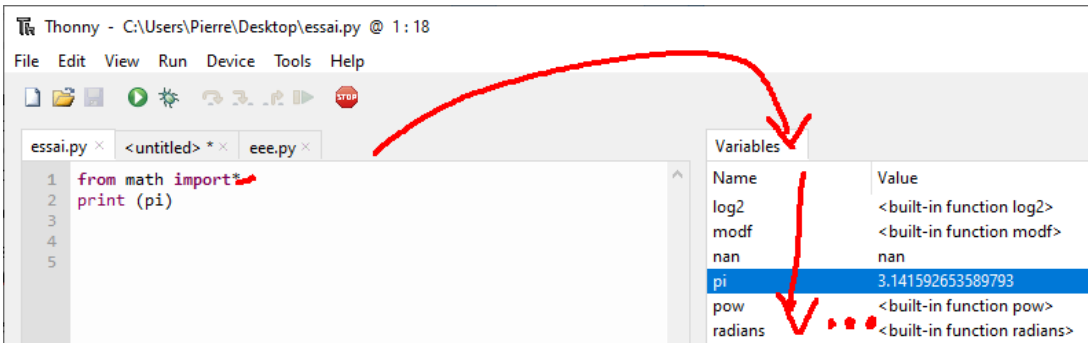
ou

- 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.

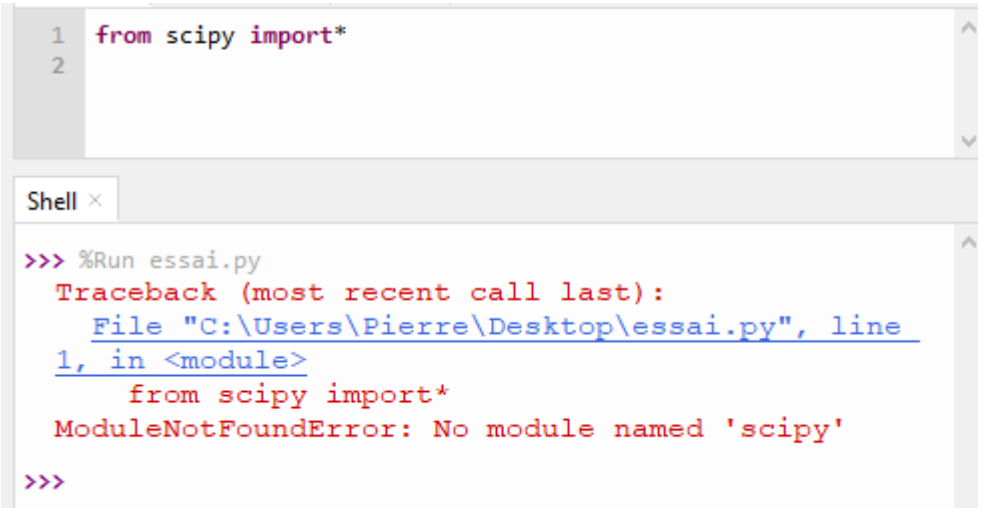
Rq :

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 panda as pd • https://pandas.pydata.org/pandas-docs/stable/getting_started/intro_tutorials/index.html • guide 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 • https://docs.python.org/fr/3/library/math.html	fonctions trigo exp, ln, pi pgcd ...
import cmath as cm • https://docs.python.org/fr/3.5/library/cmath.html	gestion des nombres complexes
import matplotlib.pyplot as plt • https://matplotlib.org/tutorials/introductory/pyplot.html • guide sympa MATHPLOTLIB	Permet de tracer et visualiser des données
import numpy as np • 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 • 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 • https://python-visualization.github.io/folium/quickstart.html	Permet de visualiser des objets géolocalisés avec des coord GPS sur une carte OpenStreetMap

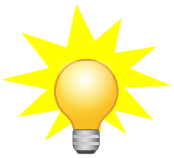
12.2 - Commandes

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

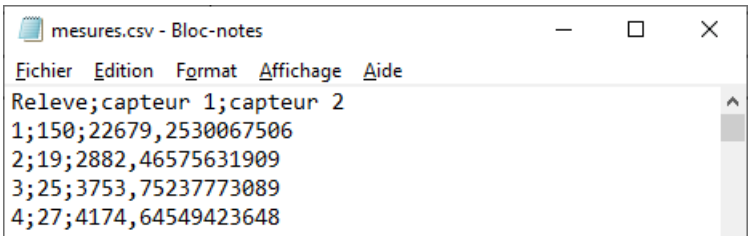
Commande ou Symbole	Opération
import csv source=open('nomdevotrefichier.csv','r',encoding='utf-8') for row in csv.reader(source,delimiter=';'): |...	Via bibliothèque csv Importation des données d'un fichier csv dans une liste de type string
import pandas as pd mesures=pd.read_csv('nomdevotrefichier',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



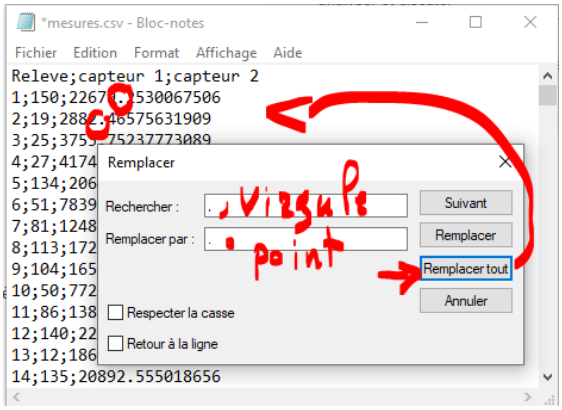
  - Vous observez que le séparateur de champs est le ; - c'est ok
  - Vous observez aussi que le séparateur de décimale est la , et là cela posera un problème car pour python le séparateur de décimale est le .

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



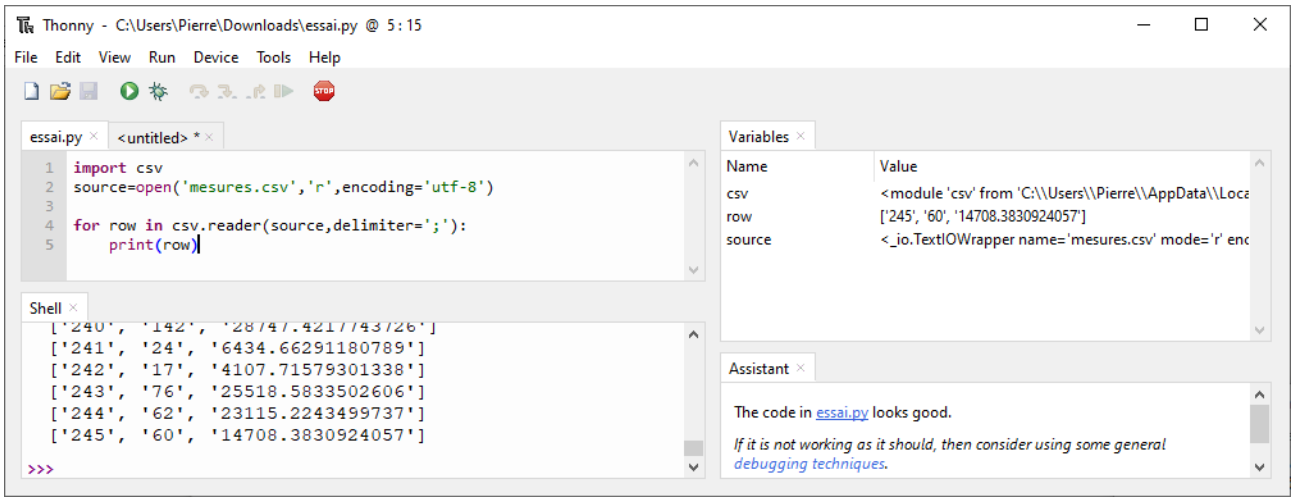
Avec l'éditeur :

- 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

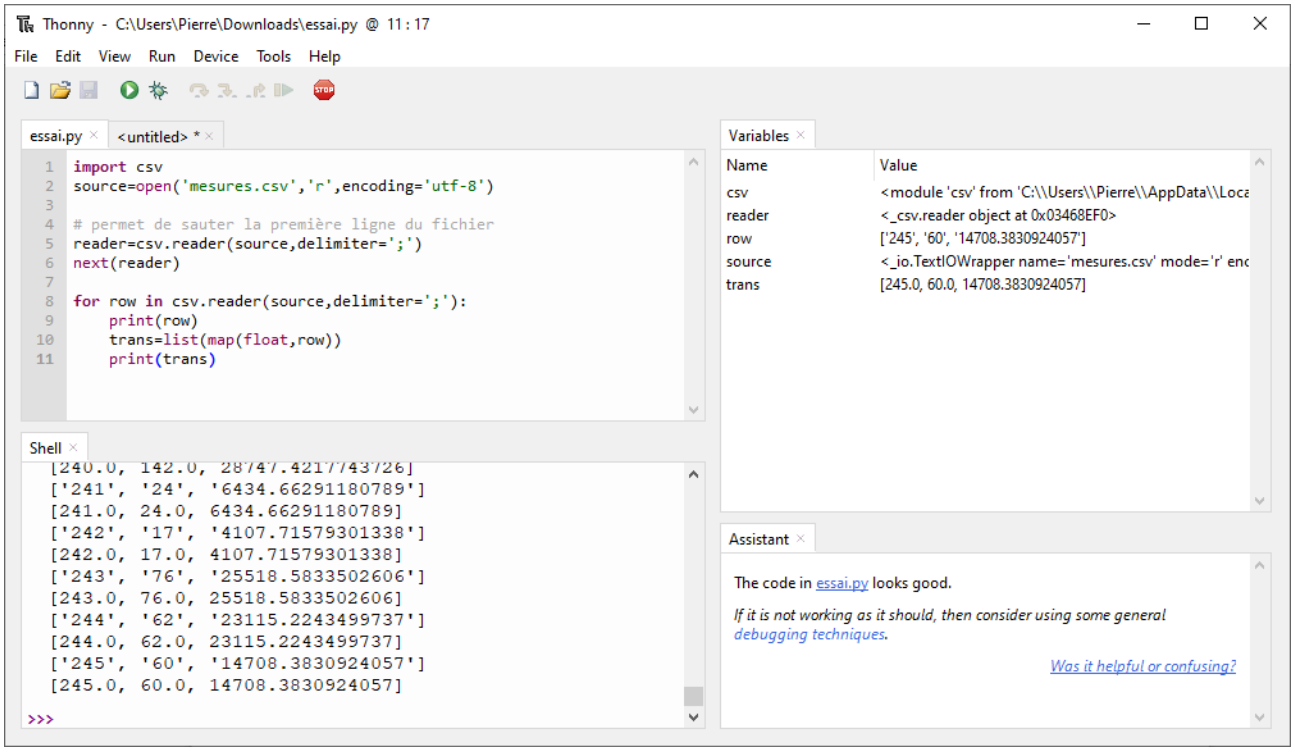
Essai 2

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



- 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

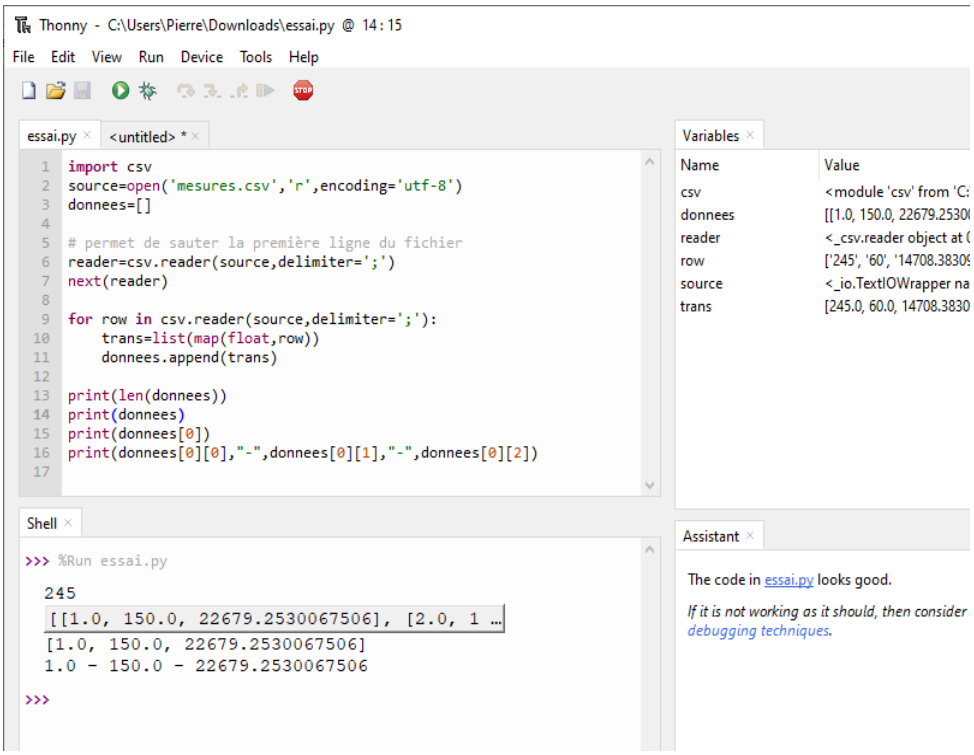
Essai 3

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



Remarque :

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

Bilan

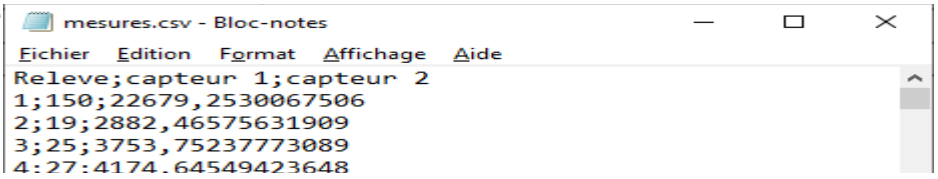
Je retiens donc ce début de script pour créer une liste manipulable sous python en partant de données importées d'un fichier csv.

```
#
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)

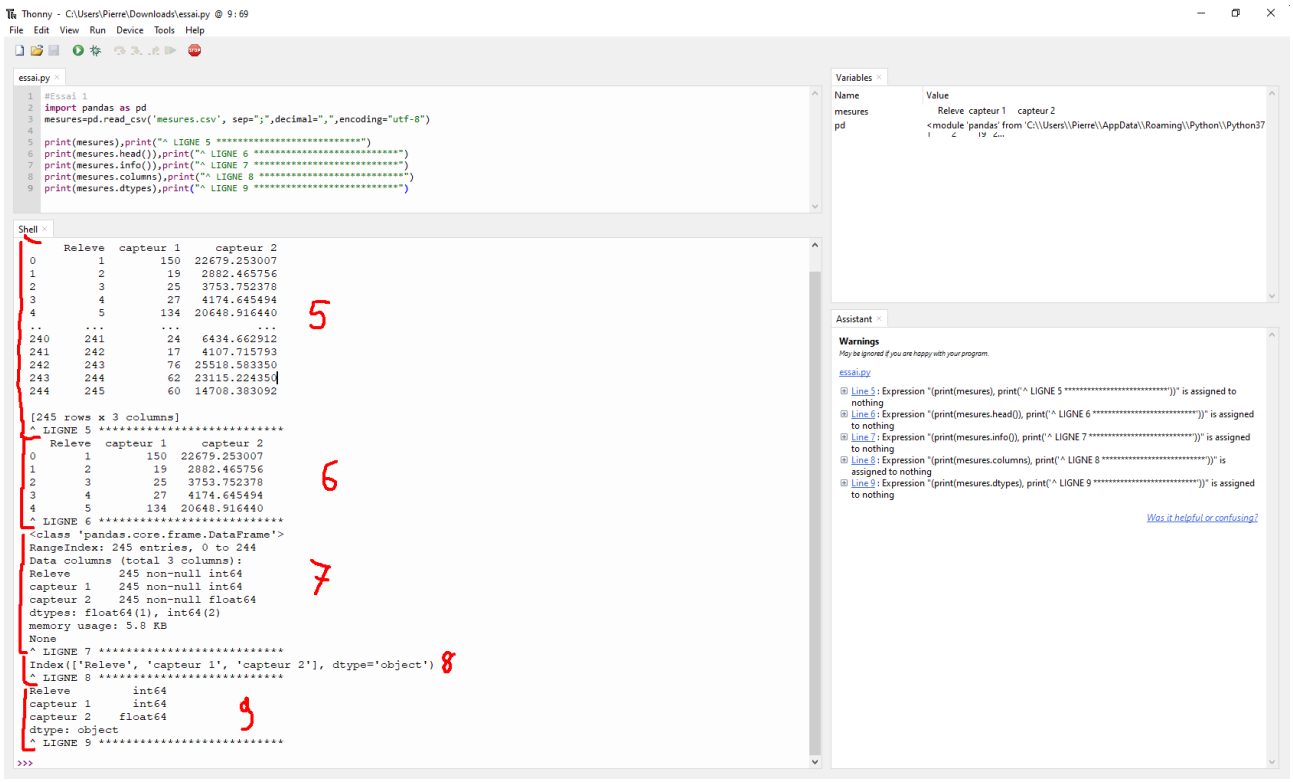


- Dans l'éditeur, copier/coller le script suivant

```
#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



- **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

```
>>> print(type(mesures))
<class 'pandas.core.frame.DataFrame'>
```

- 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

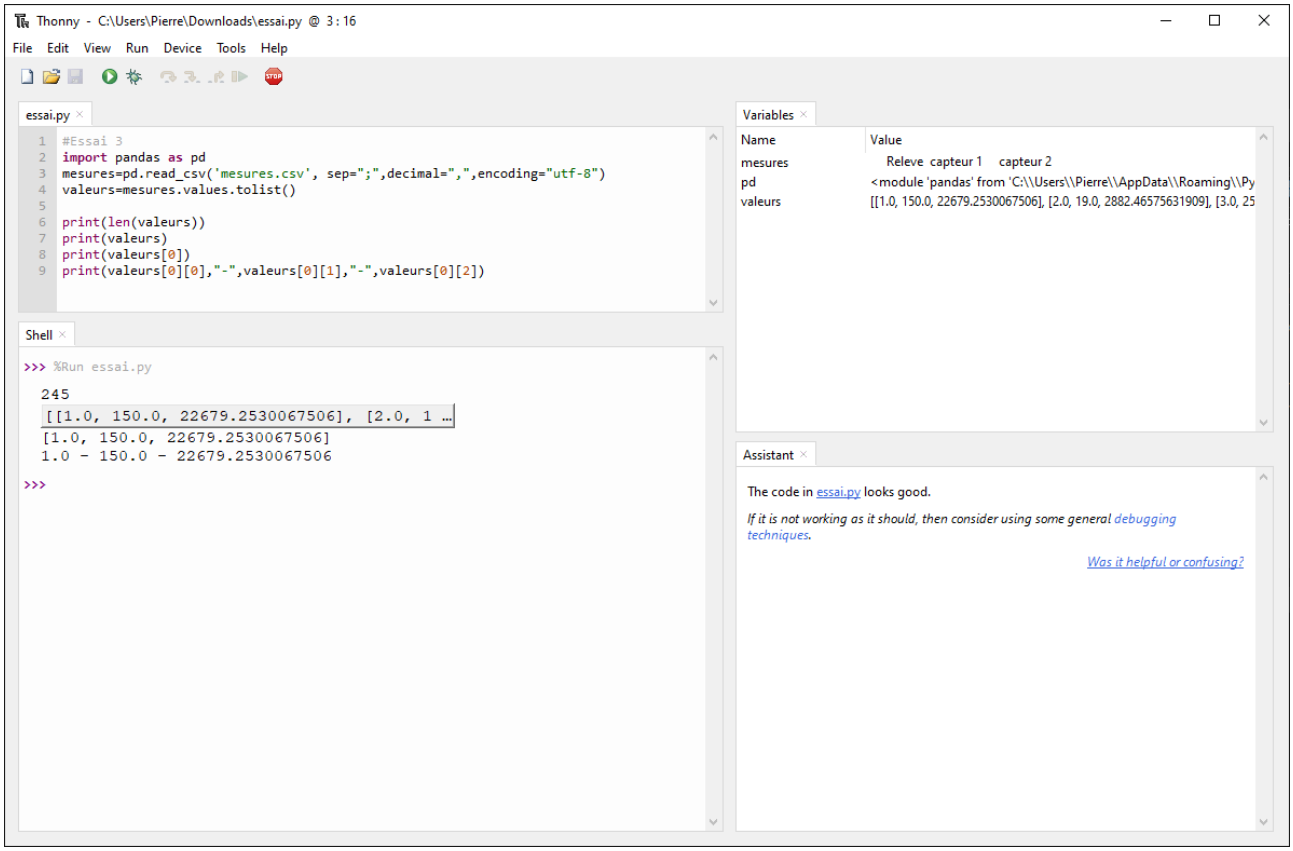
=> MAIS à ce stade je ne gère pas une liste

- 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



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 pourvoir 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
>>> from math import *
>>> dir()
['__annotations__', '__builtins__', '__doc__', '__file__', '__loader__', '__name__', '__pa
ckage__', '__spec__', 'acos', 'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil',
'copysign', 'cos', 'cosh', 'degrees', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factori
al', 'floor', 'fmod', 'frexp', 'fsum', 'gamma', 'gcd', 'hypot', 'inf', 'isclose', 'isfinit
e', '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
import math as m	Importe la bibliothèque math. Les fonctions appelées seront précédées de m.
from math import sin,pi	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 *

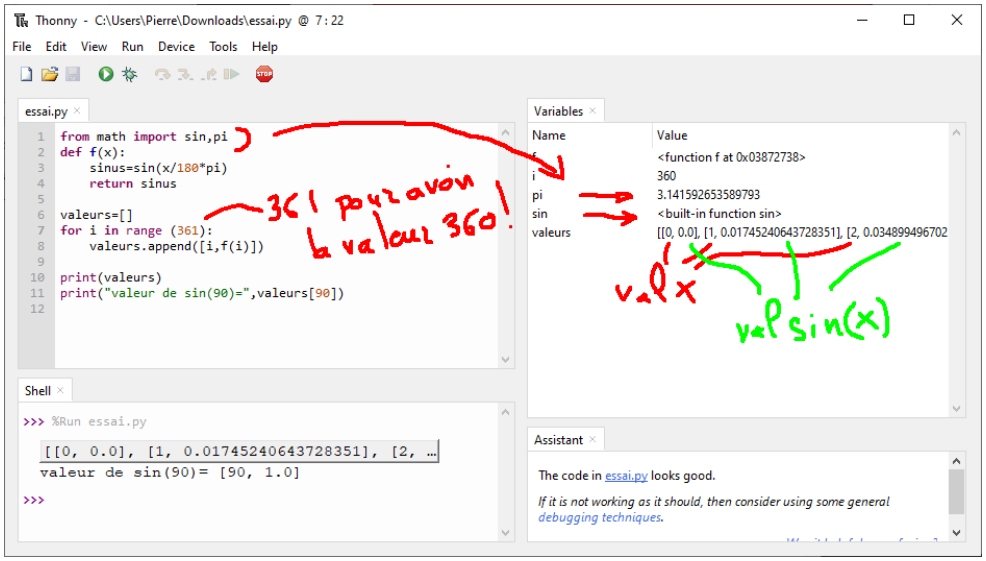
Ma piste : import de math * - une def de fonction – une liste vide – une boucle et ajout des valeurs dans la liste avec append

```
# Votre script
```

14.2.1.2 - Un exemple de correction

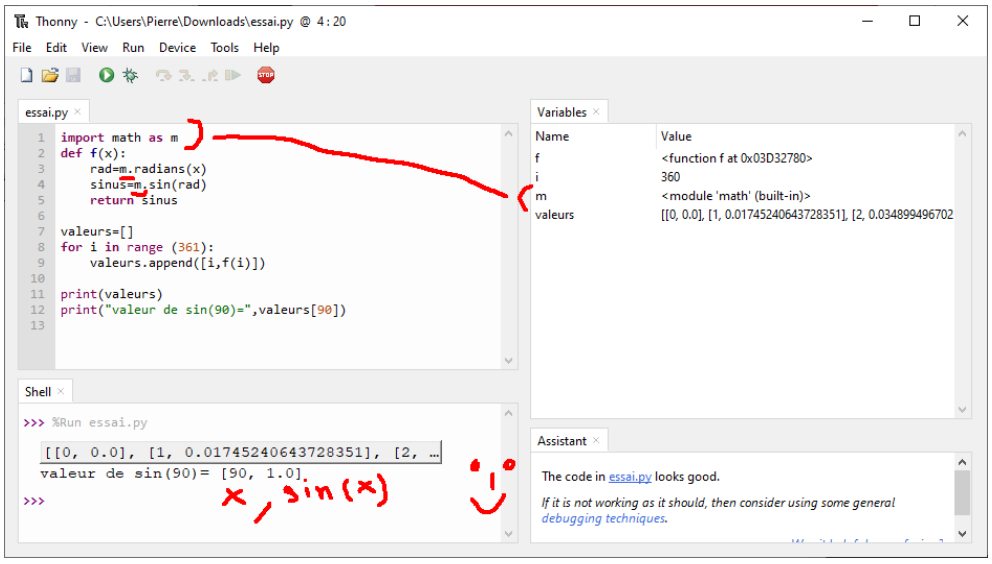
Utilisation de pi pour convertir les degrés en radians, je dois donc l'importer aussi !!

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



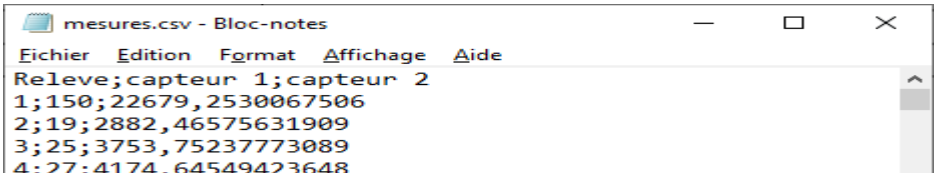
Utilisation de la fonction radians pour la conversion. Notez qu'ici j'importe tout le module donc il me suffit de mettre m. devant les fonctions !

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



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

14.2.2.1 - Énoncé



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 :

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

Étape 1

Testez ce script

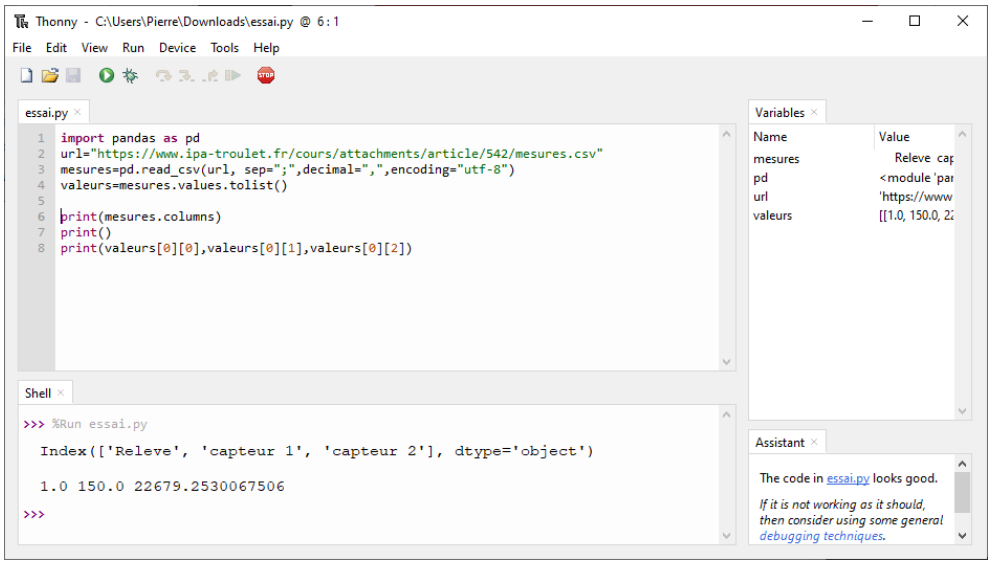
```
#
import pandas as pd
url="https://www.ipa-troulet.fr/cours/attachments/article/542/mesures.csv"
mesures=pd.read_csv(url, sep=";",decimal=",",encoding="utf-8")
valeurs=mesures.values.tolist()

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 !



Étape 2

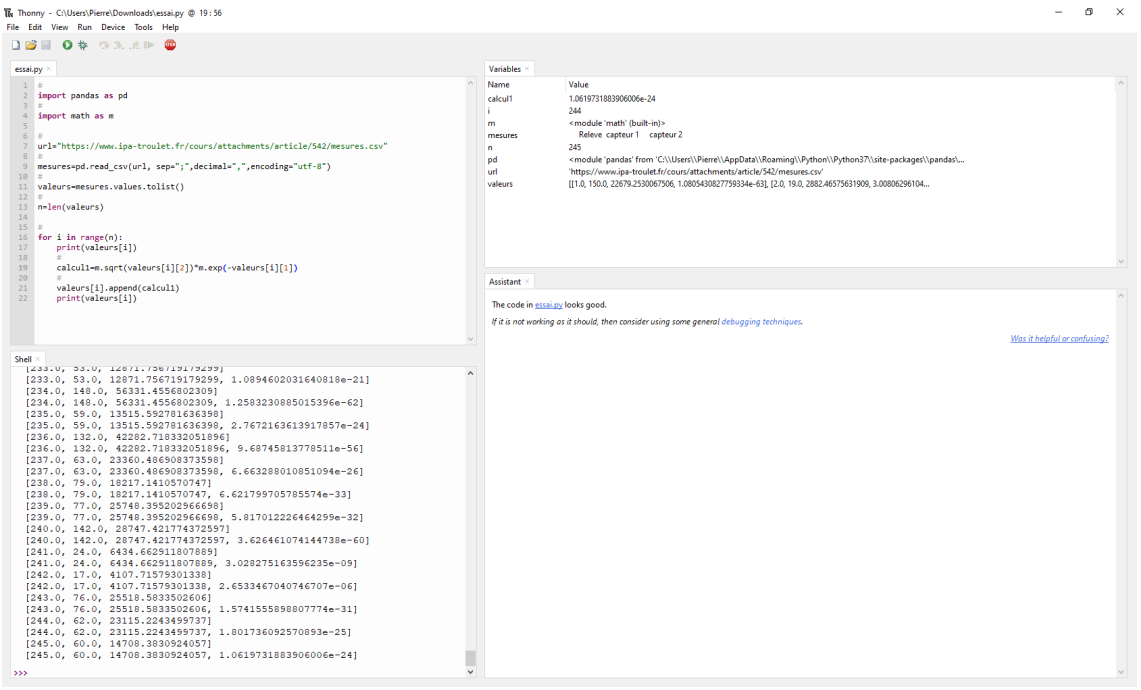
Testez ce script puis **commentez le !**

```
#
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;)



14.3 - Vos notes

Si besoin

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**
import matplotlib.pyplot as plt	Import de la bibliothèque
plt.axis([xmin, xmax, ymin, ymax]) plt.plot(x, y) plt.title("titre") plt.xlabel("abscisses") plt.ylabel("ordonnees") plt.axhline(y=0,color='black') plt.axvline(x=0,color='black')	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

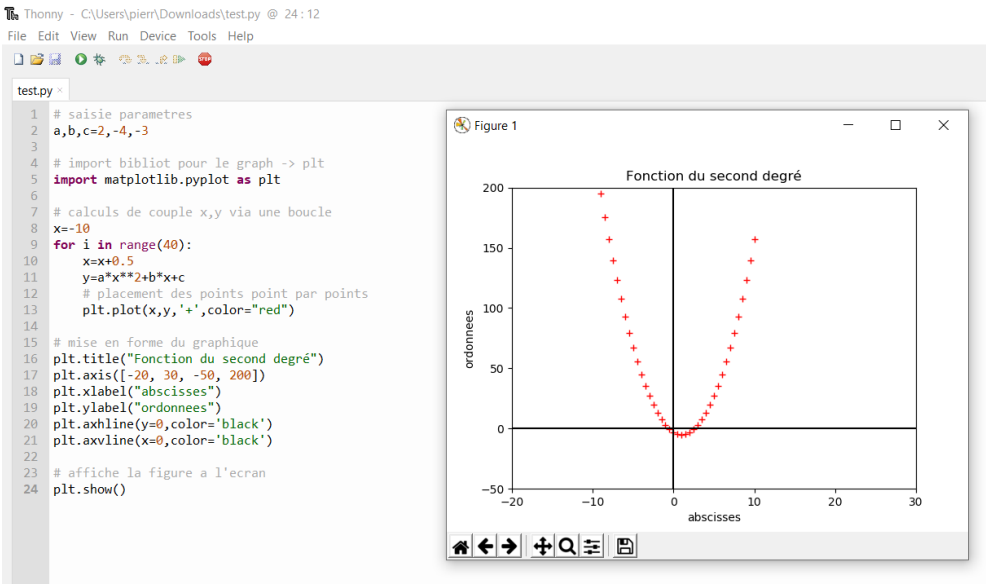
Dans l'éditeur :

- 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 ?	

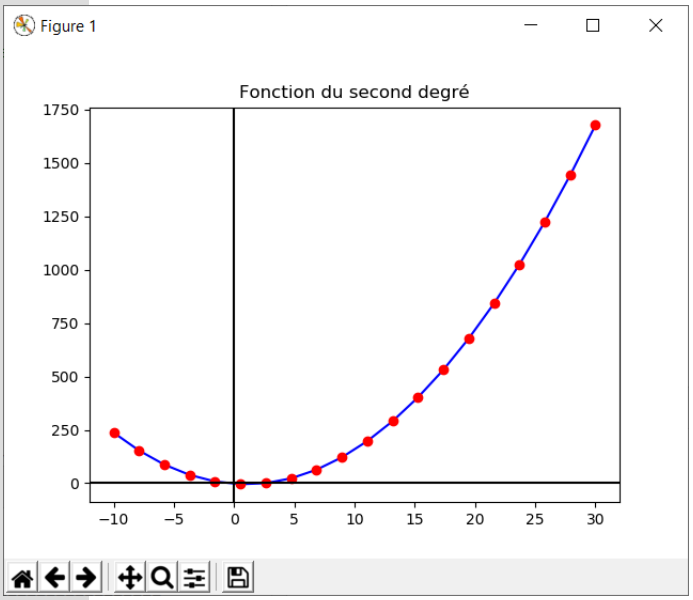
15.2.2 - Exo : Faire des graphiques avec matplotlib et numpy

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
```

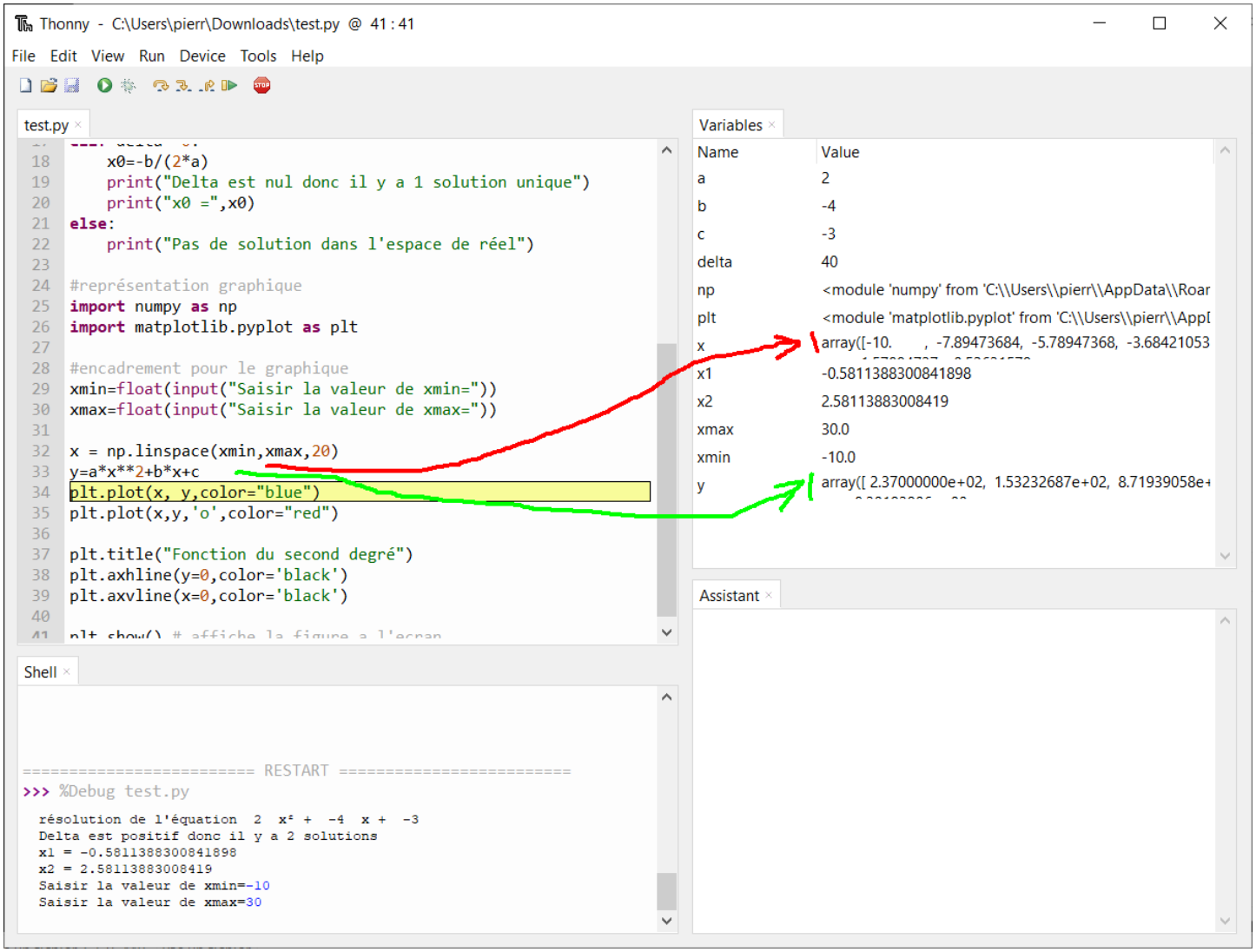
Dans l'éditeur :

- 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 !



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.

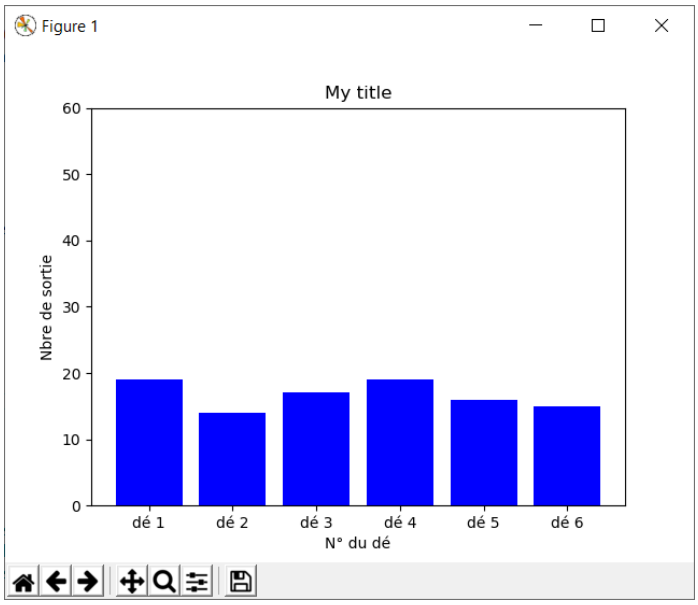
Ce qui serait sympa, c'est de laisser le choix à l'utilisateur du nombre de jet

15.2.3.1 - Énoncé

Dans l'éditeur :

copier/coller le script suivant puis exécutez le !


```
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 résultat 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 ?	

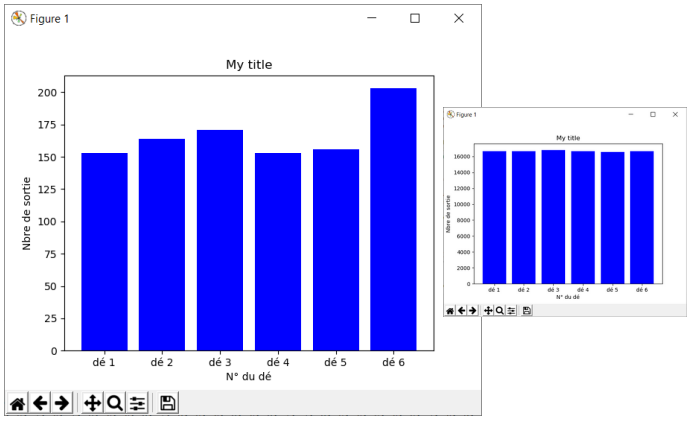
15.2.3.3 - Amélioration du script

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 :

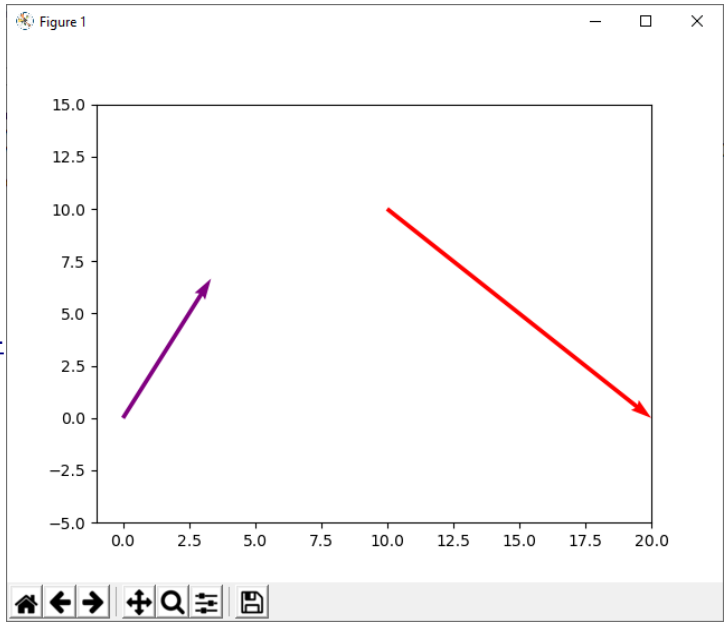
```
#
import matplotlib.pyplot as plt

# 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

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

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

  Accélération

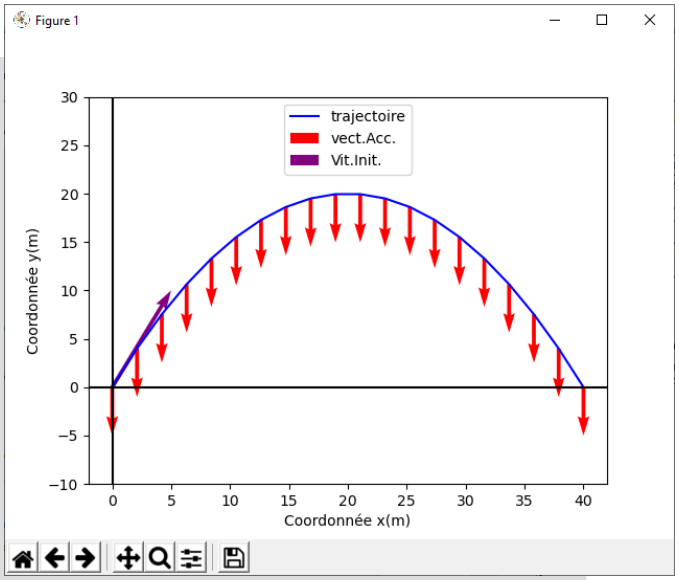
  $a_x(t)=0$

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

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

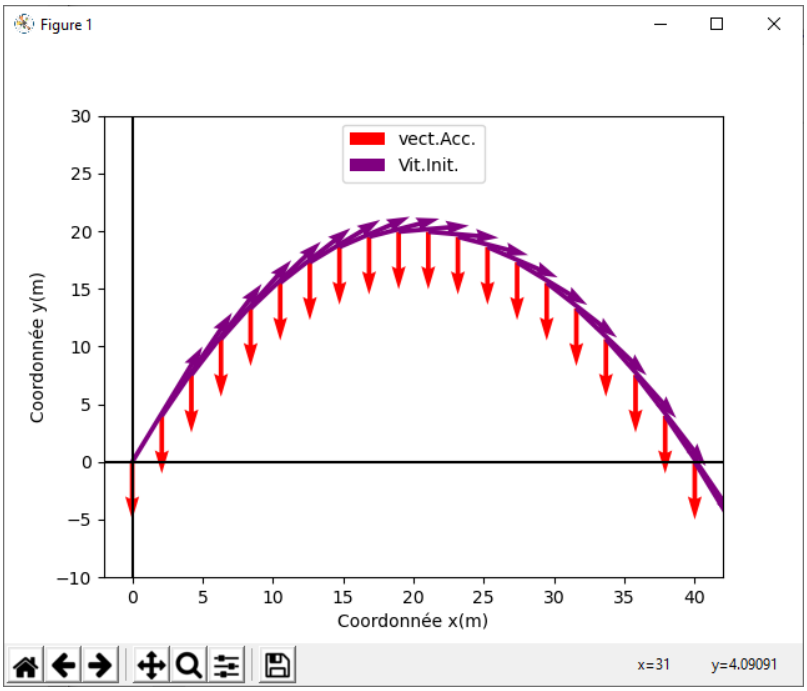
```
1  #
2  import matplot
3  import numpy a
4
5  # Initialisati
6  xo = 0 # m
7  yo = 0 # m
8  vox = 10 # m.s
9  voy = 20 # m.s
10 g = 10 # N.kg-
11 m = 1 # kg
12 duree = 4 # s
13
14 tps = np.linsp
15
16 # Déclaration
17 def x(t) :
18     return vox
19 def y(t) :
20     return -0.
21 def ax(t) :
22     return 0
23 def ay(t) :
24     return -g
25
26 # Corps du pro
27 X = [ x(t) for
28 Y = [ y(t) for
29 Ax = [ ax(t) f
30 Ay = [ ay(t) f
31
32 fig, v = plt.s
33 trajectoire =
34 vecAcc = v.qui
35 vecVit = v.qui
36
37 v.axhline(y=0,
38 v.axvline(x=0,
39 v.axis([ -2, 4
40 v.set_xlabel("
41 v.set_ylabel("
42 v.legend(loc='
43
44 fig.show()
```

Répondre aux questions suivantes :	
Expliquez vox et voy aux lignes 8 et 9.	
Expliquez pour vous avez 20 vecteurs rouge .	
Expliquer 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

```python
#
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
def vx(t):
    return vox
def vy(t):
    return -g*t+voy

# 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 ]
Vx = [ vx(t) for t in tps ]
Vy = [ vy(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(X, Y, Vx, Vy, 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()
```

15.2.5 - 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 scypi

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
from scipy import linregress	Importation du sous-module fftpack pour la transformée de Fourier
from scipy import integrate	… techniques d'intégration
from scipy import interpolate	… techniques d'extrapolation

16.2 - Exercices

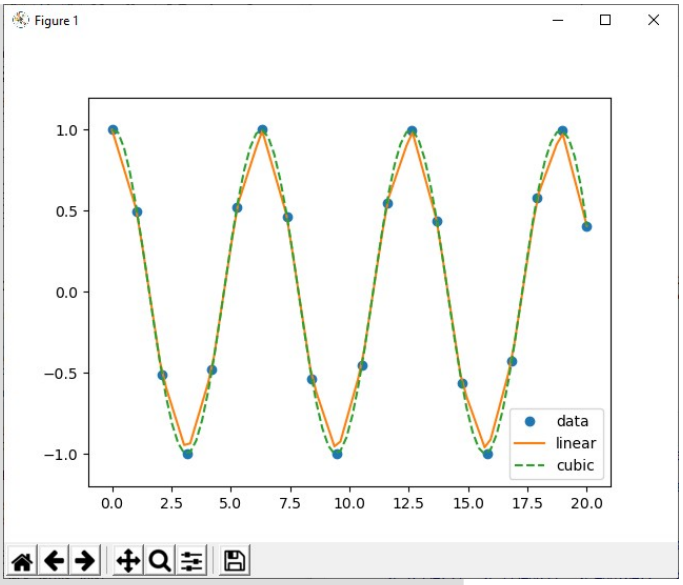
16.2.1 - Exo : Extrapoler une courbe en partant de points

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 ?	

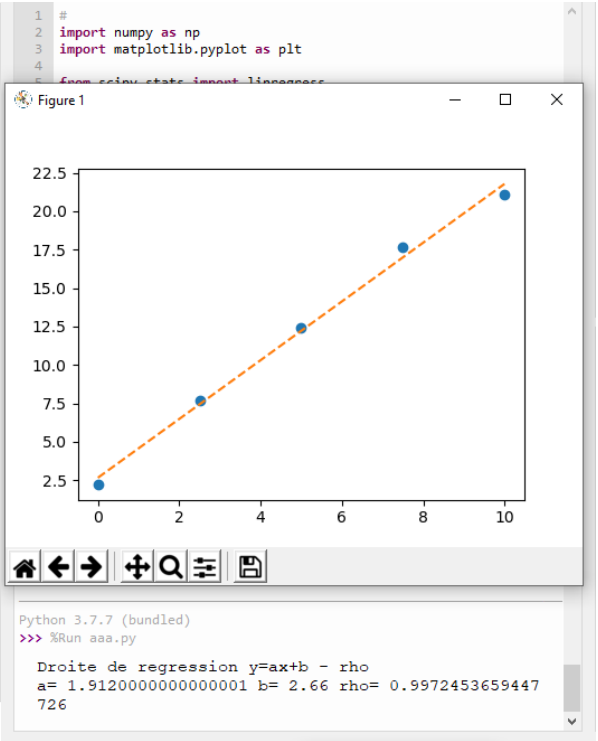
16.2.2 - Exo : Calculer une droite de régression en partant de points

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