Ce TP a pour but de se familiariser avec l'environnement PYZO qui permet l'utilisation du langage PYTHON
Pour télécharger, il suffit de se rendre sur la page internet suivante : http :/www.pyzo.org/start.html puis exécuter :
•STEP 1 selon votre système d'exploitation. Vous téléchargez puis exécutez le fichier.
•STEP 2 je vous conseille Anaconda, puis télécharger la version correspondant à votre système d'exploitation. (il faudra créer un compte avec votre e-mail)
•STEP 3 vous pouvez lancez Pyzo :

Dans une des fenêtre sur la droite, vous devez voir : "Use this environnement". Cliquez dessus, un "Happy coding" doit s'afficher.
•STEP 4 Dans le shell, tapez **install numpy matplotlib** (ce sont les modules que nous serons amenés à utiliser)

I La fenêtre de commande

La fenêtre en haut à droite s'appelle le **shell** ou **console**. On y utilise Python en mode interactif. On rentre une instruction, et Python donne la réponse (ou un message d'erreur éventuellement).
Une introduction en anglais apparaît, puis >>>
Vous avez la main. (ici on attend la première instruction ou commande de la session).

I.1 Une grosse calculatrice

Dans cette console, vous pouvez utiliser Python comme une calculatrice. Pour cela, tapez une instruction après >>> : puis appuyez sur la touche "Entrée". :
Le symbôle ☞ vous indique l'instruction à taper dans le **shell**. Python donne sa réponse sur la ligne suivante.

Exercice 1: Tapez les instructions suivantes :

☞ $2+3$ ☞ $5*4+3$ ☞ $2**3$ ☞ $18/3$

Au lieu de tout taper à chaque fois, essayez la flèche du clavier ↑

Exercice 2: Tapez les instructions suivantes :

☞ $19/4$ ☞ $19//4$ ☞ $19\%4$

Le symbôle // permet d'effectuer une division dans $\mathbb{N}$. C'est la division euclidienne. On peut récupérer également le reste de cette division.

I.2 Utilisation de variables

Du point de vue de l'ordinateur, une **variable** est une zone de mémoire au contenu de laquelle on accède via un identificateur.
Du point de vue algorithmique, une **variable** a un nom fixe et une valeur qui peut changer au cours de l'exécution de l'algorithme
Nous pouvons définir des **variables** dans la console, on distinguera les variables numériques simples (entier, réel, complexe), les variables alphanumériques (chaînes de caractère) et les variables booléennes. Nous verrons aussi des variables structurées plus compliquées, notamment les listes.

I.2.1 Variables numériques

Exercice 3: Tapez les instructions suivantes :

☞ $a=4$ ☞ $a+3$ ☞ $a=-2$ ☞ $a+3$ ☞ $1=a$

On donne une valeur à une variable en utilisant le signe =. Attention à l'ordre, il faut écrire $a = valeur$. On peut modifier le contenu d'une variable. L'ancienne valeur est écrasée.

Exercice 4: Tapez les instructions suivantes :

☞ $a=8$ ☞ $b=-2$ ☞ $c=a+b$ ☞ $d=a-b$
☞ d ☞ A ☞ $b=c-6$ ☞ $h=3/b$

PYTHON respecte la casse et différencie les minuscules des majuscules.

Exercice 5: Tapez les instructions suivantes :

☞ c ☞ $print\ c$ ☞ $print(c)$ ☞ $print(c+d)$
☞ $a,b=1,3$ ☞ $print(a,b)$ ☞ $a,b=b,a$ ☞ $print(a,b)$

Pour afficher le contenu d'une variable, on tape le nom de la variable ou on utilise l'instruction "print()". Attention à ne pas oublier les parenthèses. Une affectation simultanée est possible et permet d'échanger le contenu de deux variables.

Exercice 6: Tapez les instructions suivantes :

☞ $a=1$ ☞ a ☞ $del(a)$ ☞ a

La commande *del* permet d'effacer des variables. Celles-ci ne sont alors plus définies. Si la commande *del* n'est pas utilisée, les variables sont conservées pendant toute la durée d'une session et suprimées à la fermeture de PYTHON.
Dans l'onglet *Shell*, vous avez la possibilité d'effacer l'écran en conservant cependant le contenu de vos variables ou de redémarrer une session qui écrase le contenu de toutes les variables.

Exercice 7: Tapez les instructions suivantes :

☞ $a = 3.0$ ☞ $type(a)$ ☞ $a = int(a)$ ☞ $type(a)$
☞ a ☞ $b = 25$ ☞ $b = float(b)$ ☞ $type(b)$

PYTHON distingue les entiers des réels. L'instruction *type* permet de savoir de quel "type" est la variable.On peut changer le type d'une variable avec les instructions *int* et *float*.

Exercice 8: Tapez les instructions suivantes :

☞ $a = 3$ ☞ $a = a + 1$ ☞ a ☞ $a+ = 1$ ☞ a

Nous avons souvent besoin d'incrémenter des variables, ie d'augmenter sa valeur d'une certaine quantité. On dispose de même de $* =, - =, / = et // =$.

On peut choisir des noms de variables à plusieurs lettres, mais les espaces ne sont pas permis et ainsi que les guillemets et apostrophes. Je vous conseille d'éviter les accents. On peut aussi utiliser des chiffres dans les noms de variables, mais le nom de la variable doit toujours commencer par une lettre.

Noms valables : toto, ptsi, longueur_1, $\cdots$. Noms erronés : j'aime, vive python, 1g4.

I.2.2 Variables alphanumériques

Python peut manipuler des chaînes de caractères. Une chaîne de caractères peut se noter entre apostrophes simples ou apostrophes doubles, indifféremment :

Exercice 9: Tapez les instructions suivantes :

☞ u='coucou' ☞ v=hello ☞ v="hello" ☞ u+v

Remarquons que l'addition de deux chaînes de caractères est une concaténation de celles-ci. Sans cotes, le texte est considéré comme le nom d'une variable. Si l'on a une apostrophe simple dans notre texte, on utilisera une apostrophe double pour délimiter notre chaînes de caractères et vice-versa.

On peut convertir un nombre en chaîne de caractères. Et réciproquement :

Exercice 10: Tapez les instructions suivantes :

☞ str(20)+str(22) ☞ str(2.5)+2.5 ☞ int('20')+int('22') ☞ float('2.5')

I.2.3 Variables booléennes

Les booléens, ou valeurs de vérité, peuvent prendre deux valeurs : True ou False (Attention à la majuscule). Les opérateurs de comparaison ($<, >, <=, >=$) permettent de comparer deux nombres :

Exercice 11: Tapez les instructions suivantes :

☞ $17 > 35$ ☞ $35 >= 17$ ☞ $35 == 35$ ☞ $35! = 35$
☞ $35 == 17$ ☞ $35! = 17$ ☞ $17 < 2$ *or* $3 < 4$ $17 < 35$ *and* $6! = 6$

Attention à ne pas confondre = et ==. Le premier sert à l'affectation d'une variable, le second permet de tester si deux variables ou deux nombres ont la même valeur. Dans ce cas, la réponse attendue est True ou False.

II Editeur de programmes

II.1 Mode d'emploi de l'éditeur

Pour l'instant, vous n'avez pas du tout pu enregistrer ce que vous avez fait. Pourtant, on peut vouloir utiliser une séquence d'instructions plusieurs fois de suite, voire à différentes sessions de PYTHON. Ceci est possible, via un éditeur de programmes. Nous allons distinguer les scripts (programmes simples) des fonctions (programmes possédant des variables d'entrées et/ou de sorties).

L'éditeur se présente sur la fenêtre à gauche de Pyzo. Si aucune page n'est ouverte, allez dans le menu *Fichier*, puis *Nouveau* ou bien tapez sur $\boxed{CTRL}$+N. Tant que votre travail n'est pas enregistré, un onglet $< tmp1 >$ se trouve en haut à gauche de la fenêtre.
Vous tapez alors une suite d'instructions, dans la page ouverte, puis vous sauvegardez, dans le menu *Fichier*, puis *Enregistrer sous* ou *Enregistrer*.
Je vous demande de créer un répertoire TPPYTHON sur votre clé USB , puis ensuite, vous pourrez créer des sous-répertoires pour les différents TP.
Notez que le nom du programme doit être en un seul mot, sans accent, sans espace, sans

tiret et sans point. Le caractère souligné est autorisé.

Pour exécuter un programme, vous vous placez dans l'éditeur, dans la page où se trouve votre script, et vous allez dans le menu *Exécution*, puis *Exécutez le fichier*. Notez qu'il existe deux raccourcis clavier avec la touche F5 ou $\boxed{CTRL}$+E. On peut aussi ne sélectionner qu'une partie du code et la faire tourner avec la touche F9 ou $\boxed{ALT}$+$\boxed{Entrée}$. Cela peut être pratique, notamment en cas de souci.
Dans le fichier, on peut organiser son travail par cellules. Une cellule démarre avec deux symbôles ##. Une ligne marron apparaît, où l'on peut mettre un titre. On peut alors seulement exécuter la cellule en cours avec $\boxed{CTRL}$+$\boxed{Entrée}$.

II.2 Commentaires

Au moment où vous écrivez un programme, vous savez exactement ce que vous faites. Mais si vous voulez le relire rien que 2 jours après, vous ne savez plus exactement ce que vous avez fait et vous perdez du temps à bien retrouver chaque étape de l'algorithme utilisé. De même, si vous demandez à votre voisin de corriger une erreur que vous n'arrivez pas vous-même à trouver, celui-ci va perdre beaucoup de temps à relire et comprendre le programme. Pour minimiser ces pertes de temps ultérieures, il est indispensable de prendre soin de la mise en page des programmes. Tout d'abord, il faut toujours aérer les programmes : ne pas mettre plus d'une instruction par ligne, sauter des lignes entre les grandes étapes, regrouper les instructions de même type (si possible)... Par ailleurs, il faut commenter vos programmes, c'est-à-dire mettre du texte que PYTHON ne lira pas mais vous oui, texte dans lequel vous dites en quoi consiste l'étape.

Pour cela on utilise # : vous pouvez remarquer que le texte qui suit s'affiche alors en marron.
Les commentaires permettent également de demander à PYTHON de ne pas faire certaines étapes. Ceci peut être particulièrement utile quand vous êtes à la recherche d'une erreur dans votre programme.

II.3 Les commandes *print-input*

L'instruction ***print()*** permet d'afficher le contenu d'une variable ou du texte.

Exercice 12: Tapez les instructions suivantes dans un script, puis exécutez-le :

```
1 """ Ceci est notre premier script """
2 print('bonjour')
3 a="hello"        # on peut aussi commenter à la fin d'une ligne
4 print(a)         # on affiche le contenu de la variable a.
```

Exercice 13: Tapez les instructions suivantes dans le **shell**

☞ $a = 2$ ☞ $print("J'ai ", a, "soeurs")$ ☞ $print("J'ai " + str(a) + " " + "soeurs")$
☞ ☞ $print("J'ai \{\} soeurs".format(a))$

Il est possible d'afficher simultanément des variables numériques et alphanumériques

L'instruction ***input()*** permet une interaction entre l'utilisateur et la machine, en donnant la main à l'utilisateur pour qu'il affecte une valeur à une variable. La syntaxe est la suivante : ☞ $x = input('Entrez\ une\ valeur : ')$
Tapez cette ligne dans le **shell**, vous constatez que le texte entre cotes s'affiche, et que vous avez la main. Tapez alors 5, puis $\boxed{Entrée}$.
⚠ la variable x est de type string, il faut la "typer" par rapport à ce qu'elle contient.
☞ $x + 3$ ☞ $x = int(x)$ $x + 3$
Vous constatez que PYTHON a affecté la valeur 5 à x, une fois x typé en entier.

II.4 Exemples de scripts

Exercice 14: Voici un algorithme-calcul écrit en Python, qui effectue quelques calculs

```
1 a = 2
2 b = a**3
3 b=3*b
4 a=a-b
5 b=b-4*a
6 print(b)         # on affiche le contenu de la variable b.
```

Exécuter ce programme en effectuant le suivi de l'évolution des valeurs de toutes les variables dans un tableau comme ci-dessous :

Ligne	1	2	3	4	5
Valeur de a	6				
Valeur de b					

Exercice 15: On considère l'algorithme suivant :

```
1 t = 5
2 t = t - 1
3 t = 8 * t
```

Après exécution de l'algorithme que contient la variable t ?
Réécrire le programme en n'utilisant qu'une seule affectation

Exercice 16: On considère les deux programmes ci-dessous :

Programme 1 :

```
1 a = 0
2 b = a - 6
3 c = b**2
4 d = c - 25
5 print(d)
```

Programme 2 :

```
1 a = 0
2 b = a**2
3 c = -12 * a
4 d = b + c + 11
5 print(d)
```

1. Exécuter chacun de ces programmes, puis en remplaçant la première ligne par $a = 7$ et enfin avec $a = -3$.
   Émettre une conjecture concernant ces programmes.
2. Prouver la conjecture émise à la question précédente.
3. Écrire un troisième programme qui a le même rôle que les deux précédents.

Exercice 17: On souhaite échanger le contenu de deux variables :

```
1 a = 3
2 b = 6
3 a = b
4 b = a
5 print(a,b)
```

1. Exécuter le programme avec un suivi de variables.
2. Corriger ce programme afin qu'il permute le contenu des deux variables.
3. Améliorer le programme en demandant à l'utilisateur d'entrer les valeurs de a et b.

Exercice 18: Pour la piscine, il y a trois tarifs : 0.5€pour les moins de 6 ans, 1,80€pour les 6-12 ans et 3,5€l'entrée pour les autres. Créer un petit programme qui demandera au client combien il veut de tickets à chaque prix et qui renverra le prix total à payer.

Exercice 19: Au 1er janvier 2024, deux populations (respectivement 1 et 2) comptaient respectivement 1000 et 700 individus. On considère que le taux d'accroissement de chaque population reste constant dans le temps et vaut 5% par an pour la population 1 et 7% par an pour la population 2. On voudrait savoir si dans les 5 années qui viennent la population 2 finira ou non par être plus nombreuse que la population 1. Ecrire un script qui calcule, en utilisant des variables les effectifs des deux populations au bout d'un an ; puis au bout de 5 ans. Faites le même calcul avec une population 2 qui comprend 800 individus au lieu de 700, et un taux d'accroissement de 10% au lieu de 7% (vous modifierez juste le premier programme que vous venez d'écrire)