I - Introduction à Python

1) - Interpréteur de commandes ou shell (sous pyzo) ou console (sous spyder)

Python peut s'utiliser en mode interactif comme une calculatrice scientifique : on tape directement dans l'interpréteur de commandes (ou shell ou console ou terminal) des commandes utilisant des fonctions prédéfinies ou importées.
On peut aussi écrire des scripts (ou programmes implémentés) dans un fichier et les exécuter autant de fois que nécessaire. Cette deuxième façon de travailler sera la notre juste après la prise en main (exercices 1, 2 et 3). À terme les calculs effectués dans l'interpréteur de commandes seront occasionnels.

EXERCICE 1 : Prise en main rapide (calculs à effectuer dans l'interpréteur de commandes)

1. Calculer 53×98, $\frac{147}{32}$, $34^{2,4}$.
2. Tapez pi. Que remarquez-vous ?
3. Tapez exp(1). Que remarquez-vous ?
4. Tapez i*i. Que remarquez-vous ?

Pour être reconnues, les fonctions et constantes mathématiques usuelles doivent être importées, pour cela on tape : *import math* ou bien *from math import **
Dans le premier cas on utilise les fonctions ou constantes usuelles avec leurs noms abrégés précédés de math. (exemples math.exp, math.pi et math.cos). Dans le deuxième cas on les utilise simplement avec leurs noms abrégés (exemples exp, pi et cos).
Pour ceux qui connaissent les complexes, le nombre imaginaire i se note complex(0,1) ou bien 1j comme en physique.
Reprendre les trois dernières questions de l'exercice 1 après importation du module math et en remplaçant i par $1j$ ou bien par complex(0,1).

EXERCICE 2 : Un peu de fonctions

Calculer dans l'interpréteur de commandes $\sin^2\left(\frac{5\pi}{6}\right)$, $\left\lfloor \frac{e^3-5}{2-\sqrt{2}} \right\rfloor$, $|2\sqrt{2}-3|$. Les fonctions sinus, exponentielle, racine carrée, partie entière du module *math* s'écrivent `sin`, `exp`, `sqrt`, `floor`. La fonction native (disponible sans importation) valeur absolue s'écrit `abs`.

2) - Scripts

Pour pouvoir réutiliser des lignes de calcul (ou de code) sans avoir à les retaper, il faut les écrire dans un fichier à l'aide de l'éditeur de code. L'exécution de ces lignes de code s'obtient par le menu exécuter (run) ou par un raccourci clavier ou encore en cliquant sur l'icone d'exécution.

Exemple 1 : Dans le menu *Fichier*, sélectionner *Nouveau fichier* et créer le script suivant (bien observer le rôle de chaque instruction ou symbole) :

```
import math
print('Bonjour le monde.')
print('Un petit test :')
print('Ceci est la valeur approchée de pi :', math.pi)
```

Nommer ce fichier *TD1* et le sauvegarder dans un dossier. On codera tous les scripts du TD1 dans ce fichier.

3) - Variables

Une variable informatique est l'association entre un nom (un espace mémoire en fait) et une valeur, qui peut être de plusieurs types (float, integer, string, boolean, list, tuple, range,...). Pour affecter une valeur à une variable, on utilise tout simplement le symbole =, ainsi $a = 2$ signifie que l'on affecte la valeur 2 à la variable a. On peut également affecter à une variable une valeur obtenue à partir des opérations vues précédemment appliquées à des variables. Python fait la distinction entre majuscule et minuscule, on dit qu'il est sensible à la casse.

EXERCICE 3 : Calculs sur les variables

1. Affecter la valeur 2 à la variable a, et la valeur -5 à la variable b, sans afficher les résultats. Ensuite, calculer et afficher $a - b$ et $\frac{b}{a}$.
2. Observer ce qui se passe si l'on écrit $x = A + 3$; même chose si l'on écrit $x = a + 3$, puis print(x). Tant que la session n'est pas fermée, les variables affectées sont gardées en mémoire. La liste de toutes les variables s'obtient par l'affichage de la fenêtre *variable explorer*.

Fonction input

Dans un script, la commande *input('texte')* affiche 'texte' puis attend une valeur entrée par l'utilisateur. Le programme considère alors *input('texte')* comme étant cette valeur. Pour valoriser cette "entrée" dans le programme on capture la

valeur dans une variable `x` avec le code suivant : `x = eval(input('texte'))` si la valeur est numérique ou bien par `x = input('texte')` si la valeur est une chaîne de caractères.

EXERCICE 4 : Encore du calcul, à l'aide d'un script
Créer un script qui affiche "Entrez la valeur de x : ", attend une valeur saisie au clavier et affecte le nombre saisi à x, puis affecte la valeur 2^x à y, puis remplace y par y + x, et enfin affiche la valeur de x et de y.

EXERCICE 5 : Échange de deux variables
Créer un script qui demande de saisir les valeurs de 2 variables a et b, échange les contenus de ces 2 variables, puis les affiche. Dorénavant, pour permuter les contenus de deux variables, on pourra utiliser l'affectation parallèle : `a,b = b,a`.

Examinons maintenant les différents types de variables.

a) - Variables booléennes

Pas très passionnant mais fort utile, le type booléen ne contient que deux valeurs : True ou False (Vrai ou Faux), que python est capable d'assimiler à 1 ou 0 dans un calcul. Ce type de variables sera principalement utilisé dans des instructions conditionnelles ou des boucles conditionnelles abordées dans un prochain TD.

b) - Variables numériques (type int ou type float ou type complex)

Les variables de type *int* correspondent aux nombres entiers relatifs.
Les variables de type *float* sont des approximations des nombres réels.
Les variables de type *complex* sont des approximations des nombres complexes.
Un entier a peut être converti en un réel par l'instruction a=float(a).
Un entier ou un réel a peut être converti en un complexe par l'instruction a=complex(a).
Inversement, l'instruction a=int(a) transforme un réel a en un entier obtenu en enlevant ce qui est après la virgule.
Les chaînes de caractères sont transformés, lorsque c'est possible, en objets de type numérique par la fonction eval.

EXERCICE 6 : Calcul complexe mais simple (ceux qui n'ont pas vu les complexes peuvent sauter cet exercice)
Calculer sous forme algébrique le complexe $\dfrac{\mathrm{i}-2}{2-3\mathrm{i}}$. Calculer le module et un argument de ce nombre avec les fonctions `abs` et `cmath.phase` après avoir importé le module `cmath`.

c) - Variables chaînes de caractères

Ce sont des variables contenant du texte. Toute chaîne de caractère doit être délimitée par des apostrophes ou des guillemets ou des triples guillemets. Si l'on souhaite inclure une apostrophe (resp. des guillemets) dans une chaîne de caractères délimitée par ce même symbole il faut faire précéder l'apostrophe (resp. les guillemets) du symbole \.
Les triples guillemets seront plutôt réservées aux spécifications de fonctions python abordées dans un prochain TD.

Exemple 2 :
a='J\'aime dire : "Hello !"'
b="J'aime dire : \"Hello !\""
c="""J'aime dire : "Hello !" """
Afficher les chaînes de caractères a, b et c. Que remarque-t-on ?

II - Éléments de programmation

1) - Boucles

Les boucles permettent de répéter une suite d'instructions. Il existe deux types de boucles, selon que le nombre de répétitions est connu à l'avance ou non.

a) - Boucles for

Elles sont adaptées lorsqu'on connaît à l'avance le nombre de répétitions. Leur schéma est le suivant :
Pour i variant dans une suite **faire** une série d'instructions. *range(p,n)* est la suite des entiers de p (inclus) à n (exclu).
Exemple 3 : Essayons une première boucle toute simple :

```
for i in range(1,6):
    print('lorsque i vaut', i, '2i^2+1 vaut', 2*i**2+1)
    print('et i^2 vaut', i**2)
```

Exemple 4 : Une autre boucle un peu moins simple :

```
s = 0
for i in range(1,11):
      s = s + i**2
print(s)
```

Essayez de comprendre ce que fait cette boucle en construisant un tableau avec une ligne pour i et une ligne pour s dans lequel on inscrit la valeur de s pour chaque valeur de i. Le remplissage de ce tableau simule le déroulement de la boucle.
EXERCICE 7 : Factorielle (on pourra mettre à profit le fait que *range(p,n)* est vide lorsque $p \geqslant n$)
Écrire un script qui calcule à l'aide d'une boucle la factorielle d'un entier fourni par l'utilisateur par la fonction `input`.