TP 1 : Découverte de Python

Dans ce chapitre, on découvrira le fonctionnement général de Python.

Exercice 1 *Avant de commencer*

Démarrer le logiciel Spyder puis, dans le menu « Fichier » créer un nouveau fichier puis le sauvegarder avec un nom sous la forme TP1_votrenom. Dans la suite, on sauvegardera régulièrement son travail. On utilisera les combinaisons de touches `Ctrl + C`, `Ctrl + V` et `Ctrl + S`, `Insér.` et `F5`.

I Quelques commandes simples en Python

Dans toute cette partie, pour des instructions successives (notamment lorsque celles-ci sont présentées sur plusieurs lignes), on exécutera chaque ligne successivement dans la console en tapant sur entrée à chaque ligne. Par exemple, si l'énoncé vous dit de taper

```
t=4
2*5
```

vous aller taper, en face du `>>>` :

```
>>>t=4
```

appuyer sur entrée puis taper

```
>>>2*5
```

puis appuyer sur entrée.

Exercice 2 *Utilisation de la console comme calculatrice*

1. À l'aide de la console, calculer $54 + 23$ et $24 - 7$.
2. À quoi correspondent les opérations sur des nombres entiers : $*$, //, % et $**$.
3. Effectuer deux calculs sur une même ligne en les séparant par une virgule puis par un point virgule. Que remarque-t-on ?

Exercice 3 *Utilisation de l'éditeur de texte comme une sauvegarde*

Taper, dans **l'éditeur de texte**, la ligne (en revenant à la ligne) :

```
5*4
print(5*4)
2*2
```

puis sauvegarder avec `Ctrl+S` puis taper sur F5. Expliquer ce que vous avez fait.

Dans toute la suite, lorsqu'on ne définit pas de fonction, on peut très bien taper directement les instructions dans la console ou alors les mettre dans l'éditeur de texte (fichier texte) pour les exécuter ensuite. La seconde solution permet de garder une trace de ce qui a été tapé.

Exercice 4 *Utilisation d'un module*

1. Entrer dans l'aide et rechercher `math`.
2. Essayer de calculer arcsin(1). Si cela ne fonctionne pas, entrer `import math` dans la console puis refaire cette question en s'aidant de ce nouveau module.
3. Calculer $\sqrt{1 + \arcsin^2(\sqrt{2}/2)}$.
4. Calculer des valeurs approchées de $\arccos(1/3)$ et de $\sqrt{2 + \sqrt{2}}$.

Exercice 5 *Variables*

Une variable en Python est une suite de caractères qui renvoie à une adresse mémoire où a été créé un objet. Pour assigner la valeur numérique 1 à la variable x, on écrit `x=1`.

1. Expliquer, en les tapant dans Python, le résultat de la succession de commandes
   ```
   x=1
   y=2
   x=x+y
   y=x**y
   y
   ```
2. Expliquer la commande `print(x,y)`.
3. Expliquer la commande `a,b=1,2`
4. Deviner ce que fait la suite d'instructions suivantes :
   ```
   x = 10
   y = 2
   aux = x
   x=y
   y=aux
   del(aux)
   ```
   Vérifier ensuite en Python que votre raisonnement est exact.

Exercice 6 *Limitation de calculs*

Calculer `0.3-0.2-0.1` puis expliquer.

II Représentations graphiques

Pour effectuer des représentations graphiques, on va utiliser deux autres modules : `matplotlib.pyplot` et `numpy`. Pour éviter d'avoir à taper des noms de modules trop long, on peut abréger ces noms en exécutant `import matplotlib.pyplot as pypl`. Ensuite, les fonctions du module seront écrites sous la forme `pypl.`$\cdots$.

Exercice 7 *Représentation graphique de* sin *et* cos.

1. Importer le module `numpy` sous le nom `np`.
2. Taper les commandes (dans l'éditeur de texte avant d'exécuter)
   ```
   abscisse = [-pi+k*4*pi/24 for k in range(0,25)]
   ordonneecos = [np.cos(x) for x in abscisse]
   pypl.plot(abscisse,ordonneecos)
   ```
   Qu'obtient-t-on ? On expliquera notamment les deux premières lignes de commandes.
3. Améliorer le « lissage » de la courbe précédente.
4. Faire figurer la fonction sin sur le graphique (on pourra ajouter à la suite des instructions un autre `pypl.plot`$\cdots$).
5. Tester dans la console la commande `np.linspace(-pi,3*pi,25)`. Qu'avez-vous obtenu ?
6. Tester la ligne `pypl.clf()`. On n'oubliera d'effacer le graphique avant de commencer un autre exercice de représentation graphique.

Exercice 8 *Inégalité de convexité*

Représenter graphiquement la fonction exponentielle sur $[-4, 4]$ (pour par exemple 100 points) ainsi que sa tangente en 0. Que constate-t-on graphiquement qu'on a démontré en cours ?

Exercice 9 *Représentation d'une fonction définie en Python*

1. En s'inspirant de la fin du III du chapitre 2, définir une fonction Python `g` qui prend comme argument un réel x et qui retourne $x^5 + x + 1$.
2. Représenter graphiquement cette fonction sur $[-2, 4]$.
3. Ajouter successivement les différentes lignes après les lignes de commande de la ligne précédente.
   (a) `pypl.grid()` ;
   (b) `pypl.axhline(color='black')` ;
   (c) `pypl.axvline(color='black')` ;
   (d) `pypl.legend(['fonction g'],loc='upper left')`.

III Listes

Exercice 10 *Premières listes*

On appelle « liste » un objet du type `[1,2,3,5,12]` qui est une liste de valeurs.

1. Taper `liste = [1,2,3,5,12]` dans la console puis tester les fonction `len(liste)`, `max(liste)`, `min(liste)`, `sum(liste)` ainsi que `liste[2]` en les expliquant.
2. Définir une fonction Python appelée `moyenne` qui pour une liste donnée, renvoie la moyenne des valeurs de la liste.
3. Taper désormais `liste = sort(liste)` puis expliquer.
4. Définir une nouvelle fonction appelée `mediane` qui pour une liste donné, renvoie la valeur médiane des valeurs de la liste (lorsqu'on a un nombre pair de valeur, prend la moyenne des deux valeurs « centrales »).
5. Tester vos deux fonctions sur les listes

```
L1 = [10, 12, 14, 10, 10, 12, 10, 14, 12, 16, 8, 10]
L2 = [10, 12, 14, 10, 10, 12, 10, 14, 12, 18, 5, 10]
```

   Que constate-t-on ?

Exercice 11 *Une étude de suite*

En revoyant la façon d'écrire `ordonneecos` et en utilisant `sum` dans un des exercices précédents, créer une fonction Python `serie(alpha,n)` qui pour un entier $n \in \mathbb{N}^*$ et un réel α, retourne la valeur de

$$S_n = 1 + \frac{1}{2^\alpha} + \cdots + \frac{1}{n^\alpha} = \sum_{k=1}^{n} \frac{1}{k^\alpha}.$$

Déterminer la comportement de la suite (S_n) (ceci dépend de α).