1 Initiation à Python

1.1 Types de Données et opérations basiques

Exercice 1. On commence par manipuler les structures de base.

1. Effectuer dans python les operations suivantes :
   (a) Définir l'entier $a = 1$ et le réel $b = 3$.
   (b) Calculer $a + b$, ab, a/b, $a/3$, $b/3$, b^3, $2^{(0.5)}$

   Rq : l'opérateur puissance en python est $**$.
   Pour utiliser des fonctions plus avancées, on doit importer le module `Numpy` de la manière suivante :

   ```
   import numpy
   ```

   On appelle ensuite les fonctions voulues en leur ajoutant le prefixe "numpy."
2. Calculer $\exp(a)$, $\cos(b\pi)$.

   Si l'on souhaite eviter d'écrire les préfixes, on peut utiliser la méthode suivante :

   ```
   from numpy import *
   ```

   Attention, en utilisant cette méthode, si le module contient une variable ou une fonction avec le même nom qu'une variable précédemment définie, la méthode sera écrasée. N'hésitez pas à essayer.
3. Définir le nombre complexe $z = 1 + i$, calculer le module et l'argument de z. Rq : En python, l'appel d'un nombre complexe se fait avec le mot-clé j et non i. De plus ce mot-clé doit toujours être accompagné d'un coefficient. i s'écrit par exemple $1.0j$.
4. Calculer $a + z$, et z^{14}.
5. Définir les listes $u = (1, 2, 3, 4, "hi")$ et $v = (1, 2, (0, 3))$.
6. Additionner les elements numériques de la liste v et stocker la somme dans le premier élément de u.

   Rq : Les listes python sont indicées à partir de 0. Elles sont aussi circulaires, c'est à dire que $u[-1]$ accède au dernier élément, $u[-2]$ à l'avant-dernier et ainsi de suite... .
7. Une fonction importante à maitriser est la fonction "range" qui permet de construire des listes. Essayez range(10), range(3,10), range(3,10,2). Le module `numpy` permet de faire des range avec un pas flottant avec la fonction `arange`.

1.2 Structures de contrôle et fonctions

Commençons par un petit rappel des structures.

— Les instructions de condition `if, elif, else` ont la syntaxe suivante :

```
if test :
    commands
    ...
elif  test :
    commands
    ...
else :
    commands
    ...
```

— L'instruction de boucle `for` a la syntaxe suivante :

```
for i in liste:
    commands
    ...
```

Une particularité de python est que l'instruction `for` fait toujours intervenir une liste, d'où l'importance de savoir utiliser la fonction `range()`.

— L'instruction de boucle `while` a la syntaxe suivante :

```
while test:
    commands
    ...
```

— Définir une fonction se fait de la manière suivante :

```
def FunctionName (arg_1,arg_2,...,arg_n):
    commands
    ....
    return result
```

Rq : N'oubliez pas les : à la fins des instructions de contrôle et d'indenter les instructions affectées.

Exercice 2. Définition de fonctions, boucles et tests.

1. Définir une fonction `prodsum` qui prend en argument deux réels et renvoie un vecteur contenant le produit et la somme des deux arguments.

2. Définir une fonction `fact` qui prend en argument un entier n et renvoie $n!$. Utiliser pour cela une boucle `for`.

3. Même question en utilisant une boucle `while`.

4. Même question en utilisant un appel récursif.

Exercice 3. Mise en évidence de la précision machine. Ecrire un programme qui met en œuvre l'algorithme suivant :
$\varepsilon = 1$;
$a = 1; b = 2$;
Tant que $a \neq b$, faire :
$\varepsilon = \varepsilon/2$;
$a = 1$;
$b = 1 + \varepsilon$;
Fin de tant que

Expliquer pourquoi cet algorithme calcule la précision machine. Le tester, pour obtenir cette valeur.

2 Numpy

On a vu que les listes en python sont très libres. On peut mettre plusieurs types de données distincts dans une même liste. Cela rend la définition de fonctions sur le calcul matriciel compliqué en l'état. On utilisera donc le module `numpy`, qui est dédié au calcul matriciel.

Pour transformer une simple liste python en vecteur ou matrice numpy, il suffit de la donner en argument de la fonction `array` du module.

Exercice 4. On note A, B et C les matrices suivantes

$$A = \begin{pmatrix} 1 & 3 & 2 \\ -5 & 3 & 1 \\ -10 & 0 & 3 \\ 1 & 0 & -2 \end{pmatrix}, \; B = \begin{pmatrix} 1 & -2 & 5 \\ 6 & 1 & -1 \end{pmatrix}, \; C = \begin{pmatrix} 10 & -5 \\ 3 & 1 \end{pmatrix}.$$

1. Définir ces matrices.
2. Définir la matrice $D = 0$ (de taille 12×20). (On utilisera la fonction `zeros` du module `numpy`.
3. Calculer (quand elles existent) les matrices AB, BA et AB^T. (On utilisera les fonctions `dot` ou @ et `transpose` du module.
4. Calculer la matrice $D = I_2 - BB^T$. (On pourra utiliser la fonction `identity` du module.)
5. Calculer les déterminants des matrices A, B, C, D et $E = AA^T$. On utilisera la fonction `det` du submodule `linalg`.
6. Calculer (quand elle existe) l'inverse des matrices A, B, C, D et $E = AA^T$. On utilisera la fonction `inv` du submodule `linalg`.
7. Calculer les valeurs propres de la matrice E. On utilisera la fonction `eigvals` du submodule `linalg`.
8. Déterminer les vecteurs propres de la matrice E. On utilisera la fonction `eig` du submodule `linalg`.

Exercice 5. On pose

$$A = \begin{pmatrix} 1 & -1 & 7 \\ -4 & 2 & 11 \\ 8 & 0 & 3 \end{pmatrix}, \; B = \begin{pmatrix} 3 & -2 & -1 \\ 7 & 8 & 6 \\ 5 & 1 & 3 \end{pmatrix}.$$

Que font les instructions suivantes ?

```
3*A,  A*B, A@B,  A/B,  A**B,  cos(A),  exp(B),  expm(A)
```

La dernière de ces instructions fait partie du submodule `linalg`.

3 Matplotlib

On va désormais utiliser le module `matplotlib`, et plus précisément son submodule `matplotlib.pyplot`.

Exercice 6. Affichage de courbes graphiques.

1. Exécuter les instructions suivantes (attention, les instructions `import module` ne sont pas indiquées et les nouvelles fonctions sont dans le submodule pyplot) (et expliquer le résultat obtenu) :
```
x=arange(-1,1,0.1)
y=x**2
plot(x,y)
xlabel("x")
ylabel("x^2")
title("quadrique")
show()
```
2. Exécuter les commandes suivantes (et expliquer le résultat obtenu) :
```
clf()
plot(x,sin(3*x),'r') // 'r'
plot(x,sin(2*x),'g--') // 'g' , style lignes brisees
show()
```
3. Exécuter les commandes suivantes (et expliquer le résultat obtenu) :
```
clf()
plot(x,x**2)
plot(x,x**3)
show()
```
4. Tracer la courbe de la fonction $x \mapsto \sin(x)/x$ sur l'intervalle $[0.1\ ,\ \pi]$.