1 Prise en Main

Lancer python

`python`

Définition d'une variable

```
a=1 #Defini un entier a de valeur unite
```

Opération de base

```
a=a+1
```

Affichage du résultat

```
print(a) #Affiche la valeur de a
```

2 Fonctions mathématiques

Le package `numpy` implémente un grand nombre de fonctions mathématiques (un peu à la manière de `Matlab`).

```
import numpy as np
```

Par exemple la fonction sinus.

```
np.sin(1)
```

Il permet également de générer des tableaux

```
x=np.linspace(0,np.pi,10)        #Tableau de 10 valeurs de 0 a pi
y=np.ones(np.size(x))   #Tableau ne contenant que des 1
print(x)
print(y)
```

et d'effectuer des opérations sur ces derniers

```
y=np.sin(x)
print(y)
```

Attention sur les tableaux `numpy`

```
y=x
print(y[0])
x[0]=10
print(y[0])
```

ne recopie que les pointeurs. Pour avoir une copie de `x` dans un nouveau tableau, il faut faire

```
y=x.copy()
```

`Python` permet également de générer des sorties graphiques , notamment des graphes de fonctions. De nombreux packages sont disponibles à cet effet. Par exemple `matplotlib`

```
import matplotlib.pyplot as plt
```

qui implémente une fonction `plot`

```
plt.plot(x,y)
plt.show()
```

3 Définition de fonctions

On peut définir de nouvelles fonctions

```
def f(x):
        y=np.sin(x)
        return y;
```

Il faut utiliser des espaces d'indentation ou des tabulations pour délimiter la portée de la fonction.

4 Les boucles

On peut définir des boucles `for` et `while`

```
for i in range(0,10):
        print(i)
```

ou encore

```
i=0
while (i<10):
        print(i)
        i+=1
```

La portée d'une boucle dépend, comme pour les fonctions, de l'indentation.

5 Les scripts

Il est possible (et plutôt conseillé d'ailleurs) d'écrire des scripts en `Python` avec un éditeur de texte. L'extension utilisée par `python` et `.py`. On peut exécuter le script de deux manières

1. Soit à partir d'une console de commande

   ```
   $ pyton script0.py
   ```

2. Soit à partir de l'interpréteur en ligne de `python`

   ```
   execfile("script0.py")
   ```

6 Fonction factoriel

Q1. Écrire une fonction `pyhton` nommée `fact` prenant en argument un entier n et renvoyant

$$n! = 1 \times 2 \times 3 \cdots \times n.$$

On rappelle que $0! = 1$. Il n'est pas demandé à la fonction définie de s'assurer que l'argument qui lui est transmis est bien un entier positif (il faudrait si on voulait implémenter cela correctement). Sauvegarder le script dans un fichier. Importer ce script dans l'interpréteur `python` en ligne afin de vérifier que la fonction ainsi définie fonctionne correctement.

7 Approximation de l'exponentielle

7.1 Méthode 1

On rappelle que pour tout $x \in \mathbb{R}$

$$u_n(x) = \left(1 + \frac{x}{n}\right)^n \to e^x.$$

Q2. Implémenter une fonction qui prend en argument un tableau de valeurs x ansi qu'un entier n et renvoie le tableau des valeurs de $u_n(x)$.
Q3. Vérifier graphiquement la convergence sur l'intervalle $[-4, 4]$ en choissant $n = 1, 2, 4, 10, 20, 40, 80, 160$.

7.2 Méthode 2

On va utiliser le développement limité de l'exponentielle on rappelle que

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots + \frac{x^n}{n!} + o(x^n).$$

On pose

$$v_n(x) = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots + \frac{x^n}{n!}.$$

Q4. Afficher le graphe de la fonction v_n sur l'intervalle $[-4, 4]$ pour $n = 1, 2, 4, 10, 20, 40, 80, 160$.

8 Autes Développements limités

On rappelle les développements limités suivants

$$\cos(x) = \sum_{n=0}^{N} (-1)^n \frac{x^{2n+1}}{(2n+1)!} + o(x^{2N+1})$$

$$\frac{1}{1-x} = \sum_{n=0}^{N} x^n + o(x^N)$$

Q5. Vérifier que le Développement limité de $\cos(x)$ est convergeant pour tout x.
Q6. Qu'en est-il pour celui de $1/(1-x)$ pour $|x| < 1$ et $|x| > 1$?

9 Calcul de π

Le développement limité de arcsin est

$$\arcsin(x) = \sum_{n=0}^{\infty} \frac{(2n)!}{(n!2^n)^2} \frac{x^{2n+1}}{2n+1} + o(x^{2N+1}).$$

Q7. Vérifier que le développement limité est convergeant sur $(-1, 1)$.
Q8. En déduire une formule permettant de calculer π.