TP Informatique n° 2
Fonctions, listes, chaînes de caractères
Intégration numérique

1 Notion de fonction

Dans le langage Python, une fonction est une suite d'instructions dépendant de paramètres. Par exemple

Listing 1 – Fonction factorielle

```
def factorielle(n):
    """ calcule n! """
    fact=1

    for k in range(n):
        fact=(k+1)*fact

    return fact
```

On peut préciser ce que fait la fonction entre les guillemets qui suivent sa définition (c'est une chaîne de documentation), et

```
>>> help(factorielle)
```

affichera ces renseignements. Le paramètre `n` est passé par valeur, c'est à dire que seule la valeur prise par `n` au moment de l'appel de la fonction est connue dans la suite des instructions. ⚠ `n` n'est pas une variable, on ne peut donc pas modifier la valeur de `n` à l'intérieur de la fonction.

De plus, une fonction peut renvoyer une valeur (le résultat) grâce à l'instruction `return`.

■ **Exercice 1** Ecrire la fonction suivante :

Listing 2 – Passage par valeur

```
def incr(x):
    x=x+1
    return(x)
```

Prédire la valeur de `x` à chaque étape puis vérifier avec Python :

```
>>> x=3
>>> incr(x)
>>> x
>>> x=incr(x)
```

■

Remarque 1 A l'intérieur d'une fonction, il est possible d'utiliser des variables. Par défaut, ces variables sont dites locales : leur contenu n'est accessible qu'à l'intérieur de la fonction et pendant son exécution.

Listing 3 – Portée d'une variable

```
def g(x):
    a=x**2
```

```
    return(a)
```

Ainsi, la variable `a` n'existe pas hors de la fonction `g`. Si l'on souhaite pouvoir modifier le contenu d'une variable à l'intérieur d'une fonction, il faut lui donner une portée globale à l'aide du mot-clé `global`.

2 Les fonctions de bibliothèque

Python dispose de plusieurs bibliothèques de fonctions destinées à un usage spécifique. Il convient de les charger grâce à la commande `import` avant de pouvoir les utiliser. Détaillons celles qui nous seront le plus utiles pendant l'année.

2.1 La bibliothèque `math`

Prenons l'exemple de la bibliothèque `math` : elle contient les fonctions suivantes

`math.fabs(x)`, `math.factorial(x)`, `math.floor(x)`, `math.fsum(iterable)`, `math.exp(x)`

`math.sqrt(x)`, `math.acos(x)`, `math.asin(x)`, `math.atan(x)`, `math.cos(x)`, `math.hypot(x, y)`

`math.tan(x)`, `math.degrees(x)`, `math.radians(x)`, `math.cosh(x)`, `math.sinh(x)`, `math.tanh(x)`

`math.erf(x)`, `math.gamma(x)`, `math.log(x[, base])`, `math.sin(x)`

On retiendra que la fonction `math.factorial` peut désormais être utilisée !

2.2 Les bibliothèques `numpy` , `scipy` , `random` et `matplotlib`

Les bibliothèques `numpy` et `scipy` fournissent des outils pour le calcul scientifique. Il est possible de gagner du temps en utilisant des fonctions prédéfinies pour résoudre un problème (et se concentrer sur les aspects mathématiques par exemple), mais il est bon de savoir également ce que font ces fonctions prédéfinies : ce ne sont pas des boîtes noires.

Considérons le programme suivant :

Listing 4 – Etude graphique d'une suite récurrente

```
import numpy as np
import matplotlib.pyplot as plt
import math

def suiterec(f,u,n):

    x=np.linspace(0,5,200)
    y=f(x)

    U=np.zeros(n)
    U[0]=u
    for k in range(1,n):
        U[k]=f(U[k-1])

    Abscisse=np.zeros(2*n)
    Ordonnee=np.zeros(2*n)

    Abscisse[0]=u;
    Ordonnee[0]=u;

    for j in range(1,n):
```

```
        Abscisse[2*j]=U[j];Abscisse[2*j-1]=U[j-1]
        Ordonnee[2*j]=U[j];Ordonnee[2*j-1]=U[j]

    Abscisse[2*n-1]=U[n-1]
    Ordonnee[2*n-1]=f(U[n-1])

    return (x,y,Abscisse,Ordonnee,U)

u=2.0
n=10
(x,y,Abs,Ord,U)=suiterec(sin,u,n)

plt.plot(x,y,'b-',x,x,'k-',Abs[1:],Ord[1:],'red',label='$u_{n+1}=f(u_n)$')
plt.ylim(ymin = 0.)
plt.xlabel('$u_n$')
plt.ylabel('$f(u_n)$')
plt.title("suite recurrente")

plt.savefig("suite_rec.png")
plt.show()
```

Commenter le programme ligne à ligne ainsi que le résultat. Que doit-on retenir ?

3 Listes

Une liste est une suite finie d'éléments pouvant être de types différents. Ces éléments sont modifiables, contrairement aux tuples. On peut de plus ajouter ou enlever des éléments à une liste, ce qui permet de représenter des structures de données évoluant au cours du temps.

3.1 Définition d'une liste

Une liste peut être définie par énumération :

```
>>> L = [3,7,42]
```

ou en utilisant l'instruction `range`. On peut également la définir en compréhension (comme un ensemble) :

```
>>> M = [i**2 for i in range(10)]
```

Que donne le résultat de `L+M` ? De `[0,1]*10` ?

■ **Exercice 2** Exécuter et commenter le script suivant :

Listing 5 – Construction itérative d'une liste

```
L=[]
for k in range(10):
    L.append(k)
```

■

3.2 Opérations sur les listes

Etant donnée une liste L, on dispose des opérations suivantes :

Commande	Effet
`>>> len(L)`	longueur
`>>> L[0]`	premier élément
`>>> L[-1]`	dernier élément
`>>> L[i:j]`	liste extraite des éléments d'indices entre i (inclus) et j (exclus)
`>>> L[i:]`	liste extraite à partir de l'indice i (inclus)
`>>> L.append(v)`	ajoute l'élément v à la fin de la liste
`>>> L.extend(s)`	ajoute la liste s à la fin de la liste
`>>> L.instert(i,v)`	insert l'objet v à l'indice i
`>>> L.pop()`	supprime le dernier élément et retourne l'élément supprimé
`>>> L.reverse()`	retourne la liste
`>>> del L[i]`	supprime l'élément d'indice i

⚠ Pour recopier une liste `A` dans une autre liste `B`, on écrira

```
>>> B = A[:] # ou B=list(A)
```

L'instruction `B=A` est en effet impropre, car en modifiant B, la liste A sera également modifiée !

■ Exercice 3 Ecrire une fonction `maxi(L)` qui recherche le maximum de la liste L et renvoie tous les indices où ce maximum est atteint. ■

■ Exercice 4

1. Ecrire une fonction `appartient(x,L)` qui détermine si la liste L contient l'élément x
2. Ecrire une fonction `indice(x,L)` qui renvoie le premier indice où l'élément x apparait dans la liste L, et `None` si L ne contient pas x.

■

4 Intégration numérique

4.1 Méthode des rectangles

Soit f une fonction définie sur un intervalle $[a, b]$ et de classe $\mathcal{C}^1$, et n un entier non nul. Pour tout $i \in [[0, n]]$, on note $x_i = a + i\frac{b-a}{n}$.

On définit

$$S_n = \frac{b-a}{n} \sum_{i=0}^{n-1} f(x_i)$$

Proposition 1 En notant $M = \sup\{|f'(x)|, x \in [a, b]\}$, on a alors :

$$\left|S_n - \int_a^b f(t)dt\right| \leqslant M\frac{(b-a)^2}{2n}.$$

■ **Exercice 5**

1. Ecrire un programme demandant à l'utilisateur un réel `eps` et qui renvoie une valeur approchée à `eps` près de $\int_0^1 \frac{1}{1+t}dt$ calculée par la méthode des rectangles.
2. Comparer l'erreur commise.
3. Ecrire une fonction Scilab `rectangle(f,a,b,n)` prenant en paramètre une fonction `f`, et un entier `n`, et renvoyant la valeur de S_n.
4. Calculer avec la fonction précédente $\int_0^\pi \sin(t)dt$.
5. Démontrer la proposition.

■

4.2 Méthode des trapèzes

On suppose que f est de classe C^2 et on note :

$$T_n = \frac{b-a}{n} \sum_{i=0}^{n-1} \frac{f(x_i) + f(x_{i+1})}{2} = \frac{b-a}{n}\left[\frac{f(a)+f(b)}{2} + \sum_{i=1}^{n-1} f(x_i)\right].$$

Proposition 2 Soit $M = \sup\{|f''(x)|, x \in [a,b]\}$, on a alors l'inégalité suivante :

$$\left|T_n - \int_a^b f(t)dt\right| \leqslant M\frac{(b-a)^3}{12n^2}.$$

■ **Exercice 6**

1. Ecrire un programme demandant à l'utilisateur un réel `eps` et qui renvoie une valeur approchée à `eps` près de $\int_0^1 \frac{1}{1+t}dt$ calculée par la méthode des trapèzes.
2. Comparer le nombre d'opérations nécessaires avec le calcul obtenu par la méthode des rectangles.
3. Démontrer la proposition de l'énoncé. *Indication : on pourra effectuer deux IPP dans l'intégrale*

$$I = \int_a^b (b-t)(t-a)f''(t)dt$$

■

4.3 Représentation graphique

■ **Exercice 7** Ecrire une fonction `graphrect(f,a,b,n)` prenant en paramètres une fonction f et un entier naturel n, et qui superpose la courbe représentative de f sur $[a,b]$ avec la fonction en escalier associée à la subdivision régulière de pas $\frac{b-a}{n}$ dans la méthode des rectangles. Vérifier avec les fonctions des exercices précédents. ■