TP n°5 : Représentations graphiques d'une suite récurrente

1 Eléments de base sur les listes

Une liste est une variable qui contient plusieurs variables (pas nécessairement de même type) et qui permet de les stocker et de les ordonner. Dans ce TP, on a pour objectif de stocker les termes d'une suite. Une liste se note entre crochets : `[...]`, la numérotation des éléments se fait de "gauche à droite" et commence à 0.

C'est un nouveau type de données, ce type est noté `list`. Attention c'est un objet de type différent des tableaux numpy.

1.1 Création d'une liste et manipulations

Exercice n°1 (Créations et manipulations de listes)

Dans la fenêtre de **commande**, exécuter ligne par ligne les instructions suivantes, puis répondre aux questions.

1. ```
   >>> L1=[]
   >>> L2=[2,44,'coucou',-3,-4]
   >>> L1
   >>> L2
   >>> L2[0]
   >>> L2[2]
   >>> L2[5]
   >>> L2[-1]
   >>> print('La variable',L1,'est de type',type(L1),'et possède',len(L1),'éléments')
   >>> print('La variable',L2,'est de type',type(L2),'et possède',len(L2),'éléments')
   ```
   Si `L` est une liste :
   - que renvoie la commande `len(L)` ?
   - que renvoie la commande `L[i]` ?
2. ```
   >>> L3=list(range(3,12,2))
   >>> L3
   >>> L4=[-3,-1]
   >>> L5=L3+L4
   >>> L5
   >>> L6=5*L4
   >>> L6
   ```
   - que renvoie la commande `list(range(a,b,p))` ?

   Si `L`, `G` sont deux listes et `n` est un nombre entier strictement positif :
   - que renvoie la commande `L+G` ?
   - que renvoie la commande `n*L` ?
3. ```
   >>> print('La liste L1 est',L1,'sa longueur est',len(L1))
   >>>  L1.append('bonjour')
   >>>  print('La liste L1 est devenue',L1,'sa longueur est',len(L1))
   >>> L1.append(123)
   >>> print('La liste L1 est devenue',L1,'sa longueur est',len(L1))
   >>> L1.insert(1,100)
   >>> print('La liste L1 est devenue',L1,'sa longueur est',len(L1))
   ```
   Si `L` est une liste et `x` une variable :
   - que renvoie la commande `L.append(x)` ?
   ```

2 Utilisation des listes pour stocker les termes d'une suite

On peut utiliser les listes pour stocker des données (en particulier les termes d'une suite) : on crée une liste vide `L=[]` puis, à chaque étape d'une boucle, si on souhaite ajouter la donnée `x` à la liste `L`, on écrit `L.append(x)` ou `L=L+[x]`.

Exemple : Pour créer une liste contenant les carrés des entiers positifs inférieurs ou égaux à 20, on peut par exemple utiliser le code Python :

```
L=[]
for k in range(21):
    L = L + [k**2] # ou L.append(k**2)
print(L)
```

Exercice n°2

Soit (u_n) une suite géométrique de premier terme $u_0 = 2$ et de raison 3.

1. Compléter le programme suivant pour que, pour tout n entier naturel, `liste(n)` affiche la liste des termes de la suite jusqu'au rang n.

```
def liste1(n):
     L=[u0]
     for k in range(1,n+1):
          .....
     return L
```

2. Afficher la liste des 10 premiers termes de la suite (u_n).

Exercice n°3

On considère la suite (u_n) définie par : $\forall n \in \mathbb{N}, \quad u_{n+1} = \dfrac{u_n}{u_n + 1}, \quad u_0 = 1.$

1. Ecrire une fonction `liste2(n)` prenant en argument un entier n et qui renvoie la liste des termes de la suite jusqu'au rang n.
2. Ecrire une fonction `somme2(n)` prenant un entier n et qui renvoie la valeur de $\displaystyle\sum_{k=0}^{n} u_k$.
3. Représenter graphiquement les 20 premiers termes de la suite (u_n). Pour cela, on utilisera les options de la commande `plot` suivantes : `plt.plot(x,y,color='k',linestyle='none',marker='.',markersize=10)` afin de ne pas relier les points de la représentation obtenue par les listes `x` et `y`.

Exercice n°4

1. Ecrire une fonction `liste3(N)` prenant en argument un entier `N` supérieur ou égal à 1 et qui renvoie la liste $[1, 1, 2, 4, 3, 9, ..., N, N^2]$ des N premiers entiers non nuls suivis de leur carré.
2. Ecrire une fonction `diviseur(N)` prenant en argument un entier `N` supérieur ou égal à 1 et qui renvoie la liste des diviseurs de ce nombre et le nombre de diviseurs.
   On pourra utiliser la commande `a%b` *donnant le reste dans la division euclidienne de a par b.*

3 Etude d'une suite récurrente

3.1 La suite logistique

On considère les suites récurrentes définies par :

$$\mu \in [0,4[, \quad u_0 \in [0,1], \quad \text{et} \quad \forall n \in \mathbb{N}, \quad u_{n+1} = \mu(1-u_n)u_n.$$

1. Ecrire une fonction `suite(u0,mu,n)` qui renvoie le terme u_n de la suite.
2. Ecrire une fonction `suiteListe(u0,mu,n)` qui renvoie la liste des n premiers termes de la suite.

3.2 Dynamique de la suite logistique

1. Tracer les 50 premiers termes de la suite pour $u_0 = 0,8$ et

   (a) $\mu = 0,9$ (c) $\mu = 2,9$ (e) $\mu = 3,5$

   (b) $\mu = 1,8$ (d) $\mu = 3,1$ (f) $\mu = 3,9$

   Conjecturer la nature des suites.

2. Pour voir le comportement dynamique d'une suite récurrente du type $u_{n+1} = f(u_n)$, on peut aussi tracer la courbe de f et celle de la fonction $x \mapsto x$ sur un même graphique.

   Ensuite, on place les différents points (u_0, u_0), (u_0, u_1), (u_1, u_1), (u_1, u_2),... que l'on relie par une ligne brisée.

   Définir une fonction `dynamique(u0,mu,n)` qui réalise un tel graphique (en se plaçant sur l'intervalle $[0,1]$).

   Réaliser ces graphiques avec les valeurs de u_0 et μ utilisées précédemment. Par exemple, pour $\mu = 1,8$, on doit trouver la Figure 1.

3.3 Diagramme de bifurcations

On observe que les comportements varient suivant les valeurs de μ. Pour étudier cela, nous allons nous placer assez loin sur la suite pour qu'elle soit proche de sa limite si elle converge et ne pas être dérangé par les premiers termes.

Par exemple, on fixe $n = 250$. Si la suite converge, alors tous les termes suivants, auront presque la même valeur que u_{250}, sinon, on observera un étalement des valeurs.

Tracer un diagramme dont l'abscisse corresponde aux différentes valeurs de $\mu \in [0,4[$ et avec en ordonnée (pour chaque valeur de μ) les 100 valeurs prises par u_n pour $n \in [\![250, 349]\!]$. On doit obtenir la Figure 2. Interpréter.

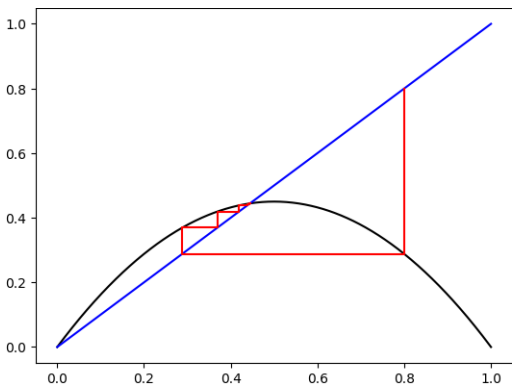

Figure 1

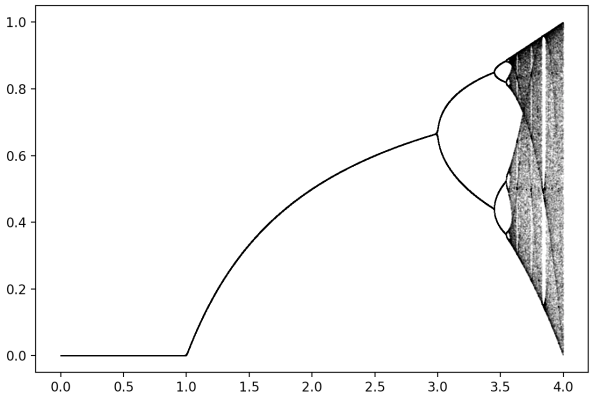

Figure 2