TP : «Quelques graphiques avec la bibliothèque matplotlib»

1 Découverte de la bibliothèque `matplotlib.pyplot`

Exercice 1 (Découverte de matplotlib.pyplot) On pourra au besoin consulter ce tutoriel en français : `http://matplotlib.free.fr`. Le principe est simple : on crée une liste d'abcisses, une liste d'ordonnées et on demande de tracer les points de coordonnées correspondantes. Par défaut, les points tracés sont reliés par des segments de droite, ce qui donne une courbe. On peut demander aussi à ce que les points ne soients pas reliés, on obtient un nuage de points.

1. Exécuter le script Python suivant puis commenter :

```
import matplotlib.pyplot as plt

X = [-1, 0, 1, 2]  # la liste des abscisses
Y = [3, 2, 4, 1] # la liste des ordonnées

plt.plot(X, Y) # pour créer le graphique
plt.show() # pour afficher le graphique
```

2. Insérer progressivement les commandes suivantes :

```
plt.grid()
plt.axis([-2,3,0,5])
plt.xlabel("mes abscisses")
plt.ylabel("mes ordonnées")
plt.title("courbe d'équation $y =f(x)$")
```

3. Remplacer la ligne `plt.plot(X, Y)` par `plt.plot(X, Y,'b-')`. Dans le troisième argument de la fonction plot, le `b` signifie blue et donc code la couleur et le tiret `-` représente le style de la courbe. On peut utiliser les différents styles ce courbes suivants : `['-', '--', '-.', ':']`

4. Pour créer un **nuage de points** : remplacer la ligne `plt.plot(X, Y)` par `plt.plot(X, Y,'r.')`.

   Dans le troisième argument de la fonction plot, le `r` signifie red et donc code la couleur et le point `.` représente le marqueur de points. On peut utiliser les différents marqueurs de points suivants : `marqueurs = ['o', '+', '.', 'x', '*']`

Exercice 2 Dessiner la maison ci-dessous où toutes les arêtes ont pour longueur 50 :

Exercice 3 Représenter un polygone régulier à 7 côtés.

Exercice 4 (Notion de subdivision et graphe de fonctions)

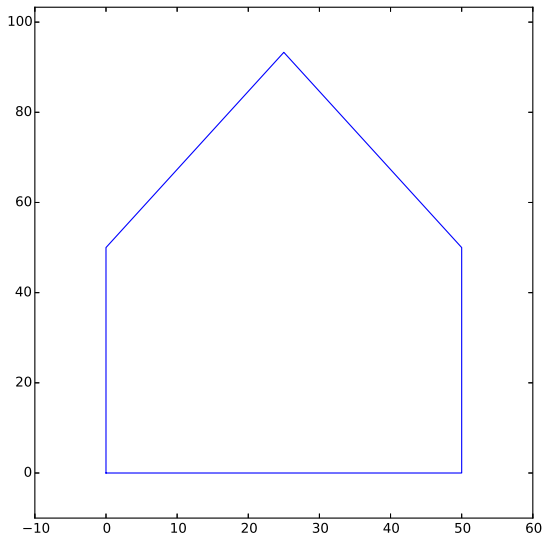

FIGURE 1 – Une maison

1. Créer une liste `X` de 101 points uniformément répartis dans l'intervalle $[0, 1]$. On parle de subdivision uniforme de pas $\frac{1}{100}$. On a ainsi découpé $[0, 1]$ en 100 intervalles de longueur $\frac{1}{100}$. Par exemple, $(0, 0.25, 0.5, 0.75, 1)$ est une subdivision de $[0, 1]$ en 4 intervalles de longueur $\frac{1}{4}$.
2. Créer la liste `Y` des images des points de `X` par la fonction carrée.
3. Tracer le graphe sur $[0, 1]$ de la fonction carrée.
4. Tester et commenter le script suivant :

```
import numpy as np
X0 = np.linspace(0,1,5)
print(X0)
```

5. Taper sur la console `X**2` et `X0**2`. Commenter.
6. Déterminer le type de la variable `X0` puis le type de ses éléments.

Exercice 5 Représenter sur $[0, 8\pi]$ la fonction 2π-périodique définie sur $[0, 2\pi[$ par $f(x) = x^2$ (on pourra utiliser avec profit l'opérateur modulo).

Exercice 6 (Représentations de plusieurs fonctions sur un même graphe) Représenter sur $[0, 2]$ le graphe des fonctions puissances $x \mapsto x^a$ pour $a \in \{0.5, 1, 2, 3\}$. On pourra créer 4 plots (et un seul show bien sûr).

Exercice 7 (Approximation de π par la méthode de Monte-Carlo) Nous allons approximer π en «tirant des fléchettes aléatoires et comptant celles qui atteignent la cible». Voici le principe : on simule n points dont les coordonnées sont des nombres aléatoires compris entre -1 et 1. On compte parmi ces points ceux qui sont dans le cercle unité. La proportion de points

tombés dans le cercle unité est une approximation de l'aire du cercle unité divisée par l'aire du carré donc de $\pi/4$. Pour simuler un nombre aléatoire compris entre -1 et 1, on pourra utiliser `random.uniform(-1, 1)`.

1. Écrire une fonction `simulation` qui prend en argument un entier $n \geqslant 1$, qui simule n points dont les coordonnées sont des nombres aléatoires compris entre -1 et 1 et qui renvoie la proportion de points dans le cercle unité multiplié par 4. Par exemple si $n = 100$, et que l'on a obtenu 80 points dans le cercle unité parmi les 100 points simulés, la proportion de points dans le cercle unité est de 0.8, `simulation(100)` renverra donc 4×0.8, c'est-à-dire `3.2`.
2. Écrire un script qui exécute 100 fois l'instruction `simulation(1000)` et qui calcule la moyenne des 100 estimations de π obtenues.
3. Représenter sur un même graphique le cercle unité ainsi que les points tirés au «hasard».

2 Graphe de suites récurrentes

Exercice 8 Représenter le graphe discret des onze premiers points de la suite récurrente u définie par $u_0 = -8.41$ et $u_{n+1} = -0.5u_n + 63$.

Exercice 9 (La suite logistique) Soit r un réel de $[0, 4]$. On considère la suite logistique définie par

$$u_{n+1} = ru_n(1 - u_n) \quad \text{et} \quad u_0 = 0.5.$$

Cette suite très célèbre permet de modéliser certaines dynamiques de population. Elle a un comportement très différent selon les valeurs de r. On se propose de tracer le nuage de points suivants.

1. Créer une subdivision uniforme `R` de l'intervalle $[0, 4]$ en 201 points.
2. Pour chaque élément r de R, représenter les points de coordonnées $(r, u_{100}), (r, u_{101}), \ldots, (r, u_{200})$.
3. Sauver votre graphique à l'aide de `plt.savefig('fonction_logistique.pdf')`.

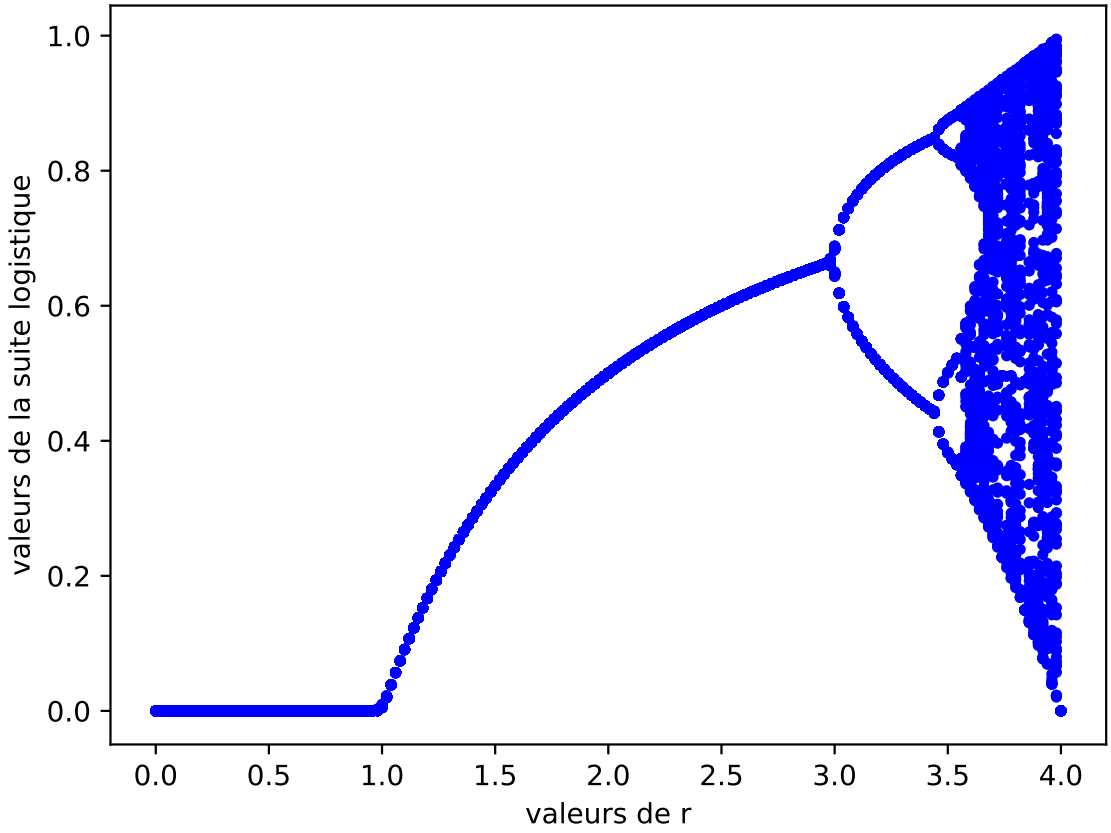

FIGURE 2 – Comportement de la suite logistique