T.P. III - Modules

Code Capytale : `00bd-786939`

I - Rendus graphiques

Les rendus graphiques en Python sont possibles à l'aide du module `matplotlib.pyplot`, importé ici avec le *surnom* `plt`. La fonction

- `plt.figure()` permet de créer un nouveau graphique,
- `plt.plot(abscisses, ordonnees)` permet de tracer les points dont la liste des abscisses est `abscisses` et la liste des ordonnées est `ordonnees`,
- `plt.show()`. permet d'afficher le graphique.

Pour tracer le graphe de la fonction exponentielle sur l'intervalle $[-5, 5]$, on utilisera ainsi :

```
import matplotlib.pyplot as plt
X = np.arange(-5, 5.1, 0.1)
Y = np.exp(X)

plt.figure()
plt.plot(X, Y)
plt.show()
```

Pour tracer un graphe, nous avons donc besoin de la liste des ordonnées, c'est-à-dire de la liste des images des abscisses par une fonction. Dans l'exemple précédent, `Y = np.exp(X)` permet de stocker dans `Y` la liste des images des éléments de `X` par la fonction `np`. Pour construire l'image des éléments de la liste `X` par la fonction `f`, il existe plusieurs solutions :

- si `f` est une fonction `numpy` *simple* (définie sans utiliser de conditionnelle), on peut écrire comme précédemment `Y = f(X)`.

```
def f(x):
    return x**2 * np.exp(x)

X = np.arange(1, 3, 0.5)
Y = f(X)
print(Y)
```

qui affiche

```
[ 2.71828183 10.08380041 29.5562244  76.14058725]
```

- sinon, on peut utiliser la notion de *liste par compréhension* : `Y = [f(x) for x in X]`. Ceci ce lit *Y est la liste des éléments* `f(x)` *lorsque* `x` *parcourt* `X`.

```
def f(x):
    if x < 2:
        return x
    else:
        return x + 1

X = np.arange(1, 3, 0.5)
Y = [f(x) for x in X]
print(Y)
```

qui affiche

```
[1.0, 1.5, 3.0, 3.5]
```

- enfin, on peut utiliser un initialiser `Y` avec un vecteur rempli de zéros à l'aide de la fonction `np.zeros` puis remplacer chacun des éléments par la *bonne* valeur :

```
def f(x):
    if x < 2:
        return x
    else:
        return x + 1

X = np.arange(1, 3, 0.5)
Y = np.zeros((len(X), 1))
for i in range(0, len(Y)):
    Y[i] = f(X[i])
```

```
print(Y)
```

qui affiche un vecteur colonne

```
[[1. ]
 [1.5]
 [3. ]
 [3.5]]
```

La fonction `len` renvoie la longueur d'un vecteur.

II - Exercices

Solution de l'exercice 1.

1. Ne pas oublier que `range(a, b)` est la liste des entiers compris entre `a` et `b-1`.

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

def u(n):
    u = 1
    for i in range(1, n+1):
        u = u / 2 + 3
    return u

n = 20
X = np.arange(0, n+1, 1)
Y = [u(n) for n in X]

plt.figure()
plt.plot(X, Y, 'o')
plt.show()
```

2.

```
n = 0
u = 1
while u <= 5.5:
    u = u/2 + 3
    n = n + 1

print("n", n)
```

□

Solution de l'exercice 2.

1. a)

```
def suite_geom(n, q):
    v = 1
    for i in range(1, n+1):
        v = q * v
    return v
```

b)

```
print("n = 10, q = 0.1", suite_geom(10, 0.1))
print("n = 100, q = 2", suite_geom(100, 2))
print("n = 110, q = 0.5", suite_geom(110, 0.5))
```

2. a) On utilise ici la fonction précédente (même si on cela induit de nombreux calculs inutiles).

```
def serie_geom(n, q):
    s = 0
    for i in range(0, n+1):
        s = s + suite_geom(i, q)
    return s
```

b)

```
import matplotlib.pyplot as plt
n = 101
s = [serie_geom(n, 0.01) for n in range(0, n+1)]

plt.plot(range(0, n+1), s, 'o')
plt.show()
```

c)

```
import matplotlib.pyplot as plt
n = 101
s = [serie_geom(n, 1.2) for n in range(0, n+1)]

plt.plot(range(0, n+1), s, 'o')
plt.show()
```

□