

SYNTHÈSE DE QUELQUES NOTIONS

1. Boucle for

1.1 Programmer une liste

On présente un exemple qui devra être adapté pour répondre aux questions posées. Étant donné un entier n , on se propose de créer la liste L des carrés de 0^2 à n^2 :

- directement en compréhension

```
| L = [k*k for k in range(n+1)]
```

- ou à l'aide d'une boucle **for**

```
| L = [0]                                # initialisation
| for k in range(1, n+1): # 1 inclus , n+1 exclu
|     L.append(k**2)
```

Invariant de boucle.

À la fin du i ème tour de boucle, L est la liste $[0, \dots, i^2]$.

1.2 Programmer une somme

Pour créer $\sum_{k=0}^n a_k$ lorsque sont donnés une liste a et un entier n :

```
| s = a[0]                                # initialisation
| for k in range(1, n+1): # 1 inclus , n+1 exclu
|     s = s + a[k]
```

Invariant de boucle.

À la fin du i ème tour de boucle, s est la somme $a[0] + \dots + a[i]$.

1.3 Programmer un produit

Pour créer $\prod_{k=0}^n a_k$ lorsque sont donnés une liste a et un entier n :

```
| p = a[0]                                # initialisation
| for k in range(1, n+1): # 1 inclus , n+1 exclu
|     p = p * a[k]
```

Invariant de boucle.

À la fin du i ème tour de boucle, p est le produit $a[0] \times \dots \times a[i]$.

1.4 Programmer une suite récurrente

On considère la suite u définie par

$$\begin{cases} u_0 \in \mathbb{R} & (\text{valeur initiale}) \\ \forall n \in \mathbb{N}, u_{n+1} = f(u_n) \end{cases}$$

Étant donné un entier n et une fonction f :

- pour calculer le terme u_n

```

u = u0                # initialisation
for k in range(n):    # n tours de boucle
    u = f(u)

```

Invariant de boucle.

À la fin du i ème tour de boucle, u est le terme u_i .

- pour calculer **tous** les termes de u_0 à u_n et les stocker dans une liste L :

```

u = u0
L = [u]                # initialisations
for k in range(n):    # n tours de boucle
    u = f(u)
    L.append(u)

```

Invariant de boucle.

À la fin du i ème tour de boucle, u est le terme u_i et L est la liste $[u_0, \dots, u_i]$.

Remarque. La fonction f peut être issue d'une bibliothèque telle que `math` ou `numpy`. Sinon elle peut être créée en suivant l'exemple ci-dessous qui correspond à $f : x \mapsto x^3 - 5x + 7$

```

def f(x):
    return x**3 - 5*x + 7

```

2. Tracés

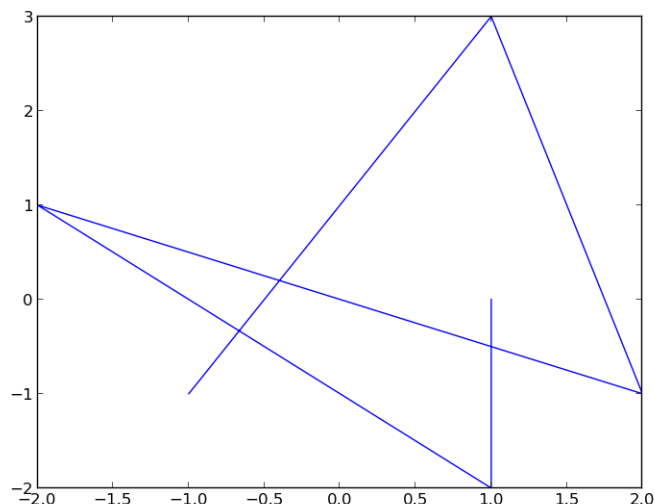
2.1 Tracé rudimentaire

Tout graphe en Python est une ligne brisée reliant des points dont on donne la liste des abscisses et la liste des ordonnées.

```

import matplotlib.pyplot as plt
X = [1, 1, -2, 2, 1, -1] # liste des abscisses
Y = [0, -2, 1, -1, 3, -1] # liste des ordonnées
plt.plot(X, Y)           # construction du graphe
plt.show()               # affichage du graphe

```



Cette ligne brisée relie les points de coordonnées $(1, 0)$, $(1, -2)$, $(-2, 1)$, $(2, -1)$, $(1, 3)$ et $(-1, -1)$.

Pour la courbe d'une fonction, plus il y a de points et plus l'impression de courbure est importante.

2.2 Graphe d'une liste en fonction de ses indices

Étant donnée une liste L de nombres, on souhaite tracer la courbe reliant les points de coordonnées $(k, L[k])$. Ce type de graphe peut servir quand L est constituée d'éléments d'une suite.

```
import matplotlib.pyplot as plt
n = len(L)
X = [k for k in range(n)] # liste des abscisses
plt.plot(X, L)            # construction du graphe
plt.show()                # affichage du graphe
```

2.3 Graphe d'une fonction

Étant données une fonction f et deux abscisses a et b , on souhaite tracer la courbe de f sur le segment $[a, b]$.

```
import matplotlib.pyplot as plt
import numpy as np
X = np.linspace(a, b, 100) # liste de 100 abscisses
Y = [f(t) for t in X]      # liste des ordonnées
plt.plot(X, Y)             # construction du graphe
plt.show()                 # affichage du graphe
```

Remarques.

1. Si la fonction f est issue de la bibliothèque `numpy` alors on peut se passer de la construction de la liste des ordonnées point par point et écrire directement :

```
import matplotlib.pyplot as plt
import numpy as np
X = np.linspace(a, b, 100) # liste de 100 ✓
    abscisses
Y = f(X)                    # liste des ordonnées
plt.plot(X, Y)              # construction du ✓
    graphe
plt.show()                  # affichage du graphe
```

2. La commande `X = np.linspace(a,b)` génère un array de 100 points régulièrement répartis entre a inclus et b inclus. Si on écrit `X = np.linspace(a,b)` alors on obtient liste de 50 abscisses incluant a et b . Cela donnera une courbe moins lisse qu'avec une liste de 100 abscisses ou plus.

2.4 Options

Le TP sur les graphes présente quelques options pour agrémenter le dessin telles qu'un titre, une légende, des styles de traits, des couleurs.