Chapitre 1 : Rappels sur la programmation en Python

Dr. Ishak ABDI — ishak.abdi@univ-jijel.dz
Département de Génie civil et hydraulique — Université de Jijel

Python Programming 💻

- Les notions abordées suffisent pour débuter, mais pour créer de vraies applications, il faut connaître et utiliser des fonctionnalités supplémentaires :
  - Conditions : `if ... else`
  - Boucles `for` , Boucles `while`
  - `arrays`

Si vous connaissez déjà un ou plusieurs langages de programmation, ces concepts vous seront familiers.

Bibliothèque standard de Python 📚

- Python permet de découper un programme en modules, réutilisables dans d'autres programmes.
- Il inclut une grande collection de modules standard pouvant servir de base à vos programmes.
- Contenu de la bibliothèque standard
- Mathématiques de base
- Structures de données, dates, temps, etc.

Ces modules sont déjà inclus avec Python. Il suffit de les importer pour les utiliser :

```
import math
```

Exemple

- Si nous n'avons besoin que de la fonction `sin()` :

```
from math import sin
x = 3.14
y = sin (x)
print (y)
```

- Si nous avons besoin de quelques fonctions :

```
from math import sin , cos
x = 3.14
y = sin (x)
print (y)
y = cos (x)
print (y)
```

- Si nous avons besoin de nombreuses fonctions:

```
from math import *
x = 3.14
y = sin (x)
print (y)
y = cos (x)
print (y)
```

- Nous pouvons également utiliser cette alternative :

```
import math
x = 3.14
y = math.sin(x)
print(y)
```

```
import math as mt
x = 3.14
y = mt.sin(x)
print(y)
```

Utilisation des bibliothèques, packages et modules Python

- Python est extensible : fonctionnalités de base minimalistes, le reste via des packages.
- Nécessité : installer les packages puis importer les modules dans le code.
- Packages clés :
  - NumPy : calcul scientifique.
  - SciPy : calcul scientifique et technique avancé (optimisation, linéaire, FFT…).
  - Matplotlib : création de graphiques 2D.

Tracer des graphiques en Python

- Pour créer des graphiques ou des plots, Python utilise des bibliothèques externes.
- Matplotlib est la bibliothèque 2D la plus utilisée :
- Documentation : https://matplotlib.org
- Similaire à MATLAB pour les plots de base.
- Nécessite d'importer la bibliothèque ou certaines fonctions :

```
import matplotlib.pyplot as plt
```

Exemple de tracé avec Matplotlib

- On dispose de deux tableaux de données :
  - x : série temporelle
  - y : température en °C correspondante
- Objectif : tracer y en fonction de x

```
import matplotlib.pyplot as plt

x = [0, 1, 2, 3, 4, 5]       # temps
y = [22, 21, 23, 24, 22, 20] # température

plt.plot(x, y)
plt.xlabel('Temps (h)')
plt.ylabel('Température (°C)')
plt.title('Température en fonction du temps')
plt.show()
```

- ou bien

```
from matplotlib.pyplot import *

# Données
x = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
y = [5, 2, 4, 4, 8, 7, 4, 8, 10, 9]

# Création du graphique
plot(x, y)
xlabel('Time (s)')
ylabel('Temperature (degC)')
show()
```

Exemple 2

```
import matplotlib.pyplot as plt
import numpy as np
x_start = 0
x_stop = 2 * np.pi
increment = 0.1

x = np.arange(x_start, x_stop, increment)
y = np.sin(x)

plt.plot(x, y)
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```

Subplot

- La commande subplot permet d'afficher plusieurs graphiques dans la même fenêtre.

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

x = np.linspace(0, 2*np.pi, 100)
y1 = np.sin(x)
y2 = np.cos(x)

# Créer une figure avec 2 lignes et 1 colonne
plt.subplot(2, 1, 1)  # Ligne 1, colonne 1
plt.plot(x, y1)
plt.title('Sinus')
plt.grid()
plt.subplot(2, 1, 2)  # Ligne 2, colonne 1
plt.plot(x, y2)
plt.title('Cosinus')
plt.legend()
plt.tight_layout()  # Ajuste l'espacement automatiquement
plt.show()
```

If ... Else statement

- Une instruction si s'écrit avec le mot-clé `if`.
- Exemples d'utilisation en Python :

```
a = 5
b = 8

if a > b:
    print("a est plus grand que b")

if b > a:
    print("b est plus grand que a")

if a == b:
    print("a est égal à b")
```

- Utilisation de if - else

```
a = 5
b = 8

if a > b:
    print("a est plus grand que b")
else:
    print("b est plus grand que a ou a et b sont égaux")
```

- Utilisation de Elif :

```
a = 5
b = 8

if a > b:
    print("a est plus grand que b")
elif b > a:
    print("b est plus grand que a")
else:
    print("a est égal à b")
```

Exemple — Utilisation des boucles for en Python

```
data = [1.6, 3.4, 5.5, 9.4]

N = len(data)
print(N)

print(data[2])

data[2] = 7.3
print(data[2])

for x in data:
    print(x)

data.append(11.4)

N = len(data)
print(N)

for x in data:
    print(x)
```

- Boucle For

```
data = [1.6, 3.4, 5.5, 9.4]

for x in data:
    print(x)

carlist = ["Volvo", "Tesla", "Ford"]

for car in carlist:
    print(car)
```

boucle avec la fonction `range()`

```
N = 10

for x in range(N):
    print(x)
```

```
start = 4
stop = 12   # la valeur stop n'est pas incluse

for x in range(start, stop):
    print(x)
```

```
start = 4
stop = 12   # la valeur stop n'est pas incluse
step = 2

for x in range(start, stop, step):
    print(x)
```

Exemple — Utilisation de la boucle For pour la sommation des données

```
data = [1, 5, 6, 3, 12, 3]
# Calcul de la somme
somme = 0
for x in data:
    somme = somme + x
print("Somme =", somme)
```

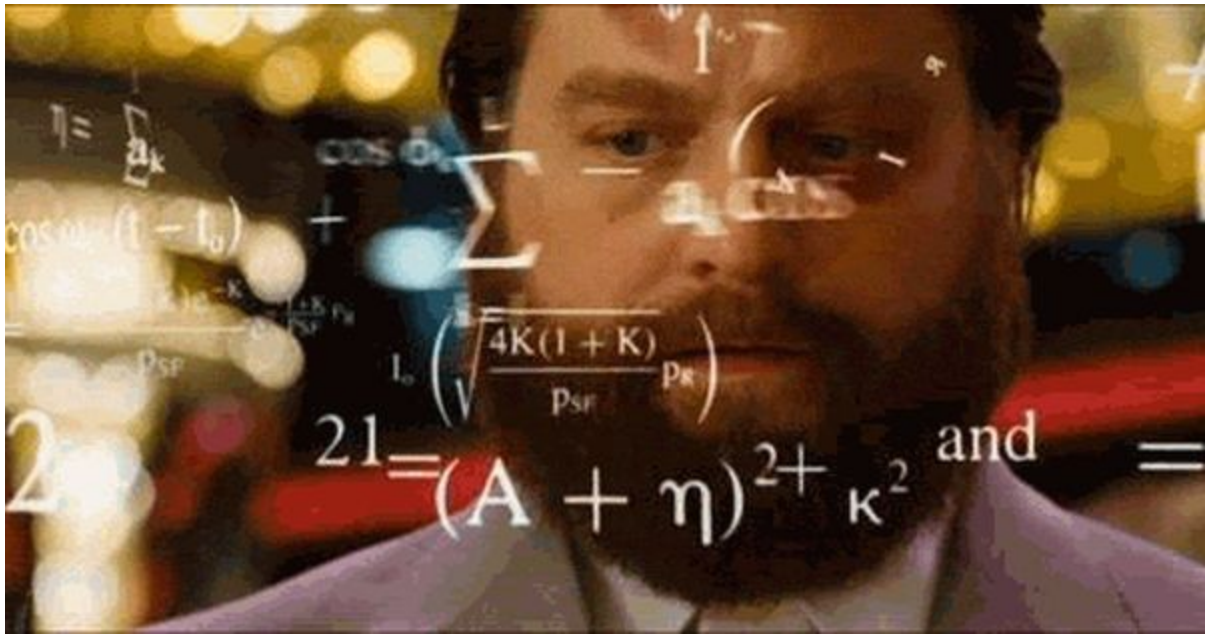
```
# Calcul de la somme
somme = 0
for x in data:
    somme += x  # somme = somme + x
    print("Somme =", somme)
```

```
# Calcul de la moyenne
N = len(data)
moyenne = somme / N

print("Moyenne =", moyenne)
```

Alternative numpy

```
import numpy as np

data = [1, 5, 6, 3, 12, 3]

somme=np.sum(data)
print("Somme =", somme)

moyenne = np.mean(data)
print("Moyenne =", moyenne)
```

à votre tour !!!



Exemple: Implémentation des nombres de Fibonacci avec une boucle for en Python

Les nombres de Fibonacci sont utilisés dans l'analyse des marchés financiers.Ils apparaissent également dans des contextes biologiques, tels que la ramification des arbres.

En mathématiques, les nombres de Fibonacci sont les éléments de la séquence suivante :

```
0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, …
```

Par définition, les deux premiers nombres de Fibonacci sont 0 et 1, et chaque nombre suivant est la somme des deux précédents.

En termes mathématiques, la suite (F_n) des nombres de Fibonacci est définie par la **relation de récurrence** :

$$F_n = F_{n-1} + F_{n-2}$$

C'est-à-dire que chaque nombre de Fibonacci est la somme des deux précédents.

avec les valeurs initiales (seed) :

$$F_0 = 0, \quad F_1 = 1$$

Exemples des premiers termes :

$$F_2 = F_1 + F_0 = 1 + 0 = 1$$
$$F_3 = F_2 + F_1 = 1 + 1 = 2$$
$$F_4 = F_3 + F_2 = 2 + 1 = 3$$

Nous allons écrire un script Python qui calcule les N premiers nombres de Fibonacci.

Nous allons écrire un script Python qui calcule les N premiers nombres de Fibonacci.

- Le script Python devient alors :

```
N = 10
f1 = 0
f2 = 1

print(f1)
print(f2)

for k in range(N-2):
    f_next = f2 + f1
    f1 = f2
    f2 = f_next
    print(f_next)
```

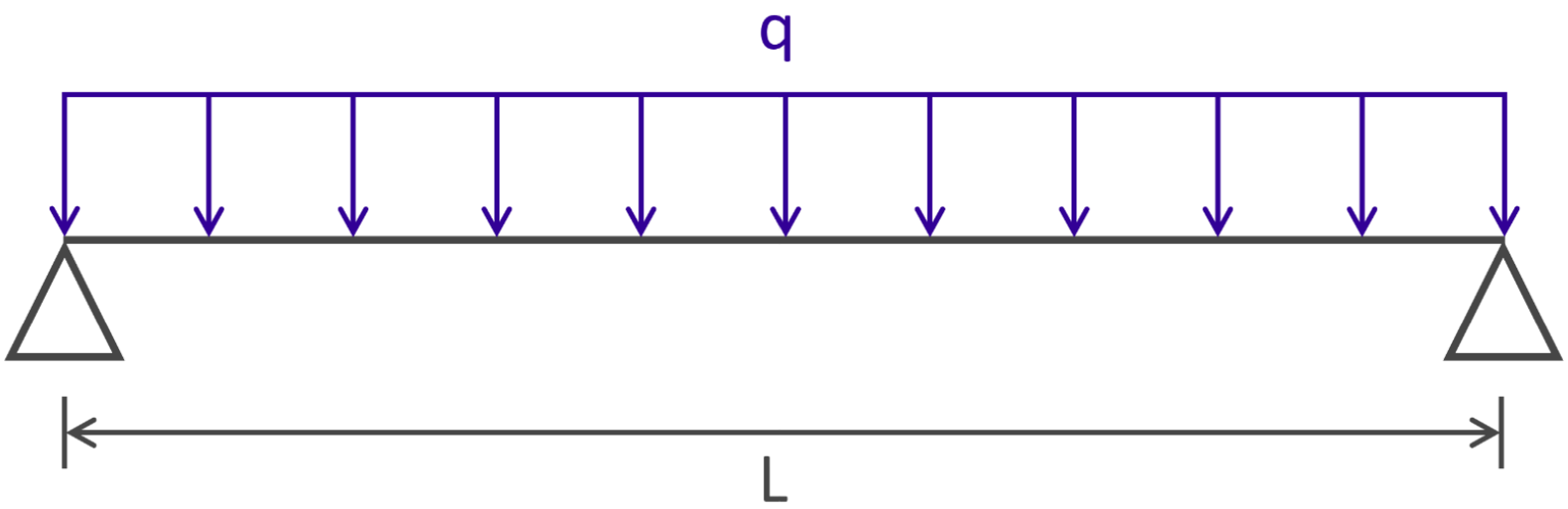
```
N = 10

fib = [0, 1]
for k in range(N-2):
    f_next = fib[k+1] + fib[k]
    fib.append(f_next)

print(fib)
```

```
N = 10
fib = []
# Initialisation de la liste avec des zéros
for k in range(N):
    fib.append(0)
# Définir les deux premiers nombres de Fibonacci
fib[0] = 0
fib[1] = 1
# Calcul des N-2 nombres suivants
for k in range(N-2):
    fib[k+2] = fib[k+1] + fib[k]
print(fib)
```

```
import numpy as np
N = 10

# Créer un tableau de zéros
fib = np.zeros(N)

# Définir les deux premiers nombres de Fibonacci
fib[0] = 0
fib[1] = 1

# Calcul des N-2 nombres suivants
for k in range(N-2):
    fib[k+2] = fib[k+1] + fib[k]

print(fib)
```

Exercice:

Une poutre simplement appuyée de longueur $L = 6m$ est soumise à une charge uniformément répartie $q = 10\,kN/m$



Question

- Écrire un script qui calcule le moment fléchissant en chaque point (discrétiser chaque travée comme vous le souhaitez).
- Tracer le diagramme des efforts tranchants et celui des moments fléchissants.

Formules importantes

- Effort tranchant à une distance x de l'appui gauche :

$$V(x) = \frac{qL}{2} - qx$$

- Moment fléchissant à une distance x :

$$M(x) = \frac{qL}{2}x - \frac{qx^2}{2}$$

où :

- L = longueur de la poutre
- q = charge uniformément répartie
- x = position le long de la poutre

-- coding: utf-8 --

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

# Données
L = 10.0         # longueur de la poutre en m
q = 15.0        # charge uniforme en kN/m
n = 70          # nombre de points de discrétisation

# Discrétisation de la poutre
x = np.linspace(0, L, n)

# Calcul des efforts tranchants et moments fléchissants
V = q*L/2 - q*x
M = q*L/2*x - q*x**2/2
```

```
# Affichage des résultats
print("Position (m) | Effort tranchant V (kN) | Moment fléchissant M (kNm)")
for xi, Vi, Mi in zip(x, V, M):
    print(f"{xi:10.2f} | {Vi:21.2f} | {Mi:23.2f}")

# Création des subplots
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(8, 10))

# Effort tranchant
ax1.plot(x, V, color='r')
ax1.plot(x, 0*x, color='k')
ax1.set_title("Effort tranchant le long de la poutre")
ax1.set_xlabel("Position x (m)")
ax1.set_ylabel("V (kN)")
ax1.grid(True)

# Moment fléchissant
ax2.plot(x, M,  color='b')
ax2.set_title("Moment fléchissant le long de la poutre")
ax2.set_xlabel("Position x (m)")
ax2.set_ylabel("M (kNm)")
ax2.grid(True)
ax2.invert_yaxis()  # inversion de l'axe y pour le moment fléchissant

plt.tight_layout()
plt.show()
```