Chapitre 1. Introduction à la programmation orientée objets (POO)

Objectifs

À la fin de ce chapitre, vous serez capable de :

- Comprendre les principes fondamentaux de la POO (classes, objets, encapsulation, abstraction, etc.)
- Connaître la philosophie du langage Python
- Identifier les librairies les plus utilisées en Python
- Exécuter un programme Python dans différents environnements de développement (IDE)

1. Principe de la Programmation Orientée Objets

La POO (Programmation Orientée Objets) est un paradigme de programmation basé sur la modélisation du monde réel à l'aide d'objets**.**

Un objet regroupe :

- Des données → appelées *attributs*
- Des comportements → appelés *méthodes*

Exemple : Un voiture peut être représentée comme un objet :

- Attributs : marque, couleur, vitesse
- Méthodes : démarrer(), accélérer(), freiner()

2. Les 4 piliers de la POO

Principe	Description	Exemple
Encapsulation	Protection des données internes d'un objet	Variables privées dans une classe
Abstraction	Cacher les détails internes et ne montrer que l'essentiel	Méthode publique utilisant des méthodes internes
Héritage	Une classe peut hériter des propriétés d'une autre	class VoitureElectrique(Voiture)
Polymorphisme	Une méthode peut avoir des comportements différents selon le contexte	accelerer() peut agir différemment selon le type de véhicule

3. Encapsulation et abstraction

3.1. Encapsulation

L'encapsulation est le principe qui consiste à regrouper dans une même entité (appelée *objet* ou *module*) les données et les opérations qui les manipulent, tout en masquant les détails internes de leur implémentation. Autrement dit, c'est le fait de protéger les données d'un objet contre

un accès direct depuis l'extérieur, en imposant l'utilisation de méthodes d'accès (getters, setters, etc.) pour les consulter ou les modifier.

Avantages :

- Empêche les modifications accidentelles ou non contrôlées des données internes.
- Simplifie l'utilisation d'un objet en cachant la complexité.
- Facilite la maintenance et l'évolution du système.

Exemple : Imagine une boîte noire : tu peux appuyer sur des boutons (méthodes) pour interagir avec elle, mais tu ne vois pas ni ne modifies directement ce qu'il y a à l'intérieur (les données).

Les attributs peuvent être :

- **Publics** : accessibles partout
- **Privés** : précédés de __
- **Protégés** : précédés de _

3.2. Abstraction

L'abstraction est le processus qui consiste à isoler les caractéristiques essentielles d'un concept tout en ignorant les détails non pertinents. Elle permet de représenter un système complexe de manière simplifiée, en ne conservant que ce qui est nécessaire à la compréhension ou à l'utilisation.

Avantages :

- Simplifie la conception et la compréhension des systèmes complexes.
- Permet de se concentrer sur *ce que fait* un objet plutôt que *comment il le fait*.
- Favorise la réutilisation du code et la modularité.

Exemple : Quand tu conduis une voiture, tu n'as pas besoin de connaître les détails du moteur pour l'utiliser. Tu interagis avec une **interface abstraite** (volant, pédales, tableau de bord) qui représente uniquement les actions essentielles.

3.3. Différence entre encapsulation et abstraction :

Concept	Ce qu'il fait	Ce qu'il cache	Objectif principal
Encapsulation	Regroupe données et comportements	Les détails internes d'un objet	Protection et organisation
Abstraction	Simplifie la représentation d'un concept	Les détails non essentiels	Compréhension et modélisation

4. Initiation au langage Python

Le langage de programmation Python a été créé en 1989 par Guido van Rossum, aux Pays-Bas. Le nom *Python* vient d'un hommage à la série télévisée *Monty Python's Flying Circus* dont G. van Rossum est fan. La première version publique de ce langage a été publiée en 1991. La Python Software Foundation (https://www.python.org/psf/) est l'association qui organise le développement de Python et anime la communauté de développeurs et d'utilisateurs. Ce langage de programmation présente de nombreuses caractéristiques intéressantes :

- Il est multiplateforme : C'est-à-dire qu'il fonctionne sur de nombreux systèmes d'exploitation : Windows, Mac OSX, Linux, Android, iOS, depuis les mini-ordinateurs Raspberry Pi jusqu'aux supercalculateurs.
- Il est gratuit : Vous pouvez l'installer sur autant d'ordinateurs que vous voulez (même sur votre téléphone !).
- C'est un langage de haut niveau : Il demande relativement peu de connaissance sur le fonctionnement d'un ordinateur pour être utilisé.
- C'est un langage interprété : Un script Python n'a pas besoin d'être compilé pour être exécuté, contrairement à des langages comme le C ou le C++.
- Il est orienté objet : C'est-à-dire qu'il est possible de concevoir en Python des entités qui miment celles du monde réel (une molécule d'ADN, une protéine, un atome, etc.) avec un certain nombre de règles de fonctionnement et d'interactions.
- C'est le langage de programmation le plus utilisé au monde. La figure (1-2) illustre le classement annuel 2024 des langages de programmation les plus populaires, établi respectivement par TIOBE et IEEE Spectrum.

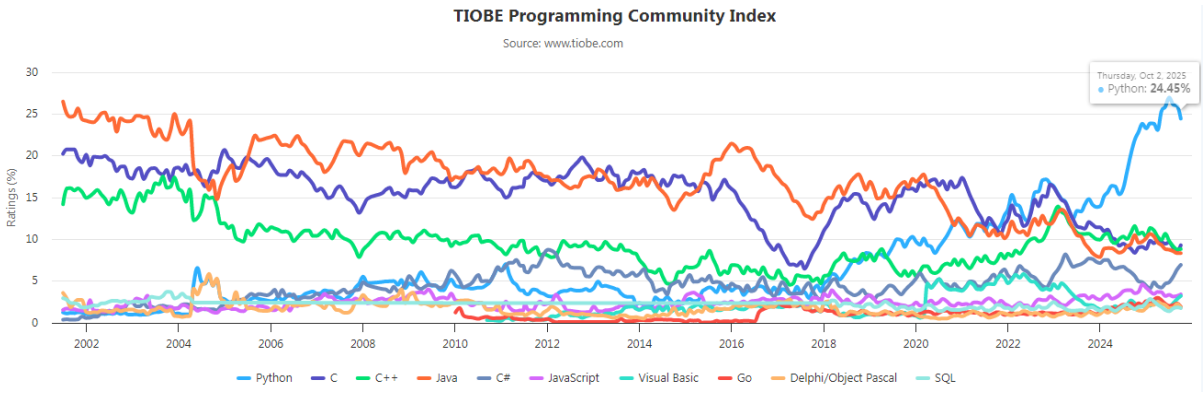

Figure 1 : Les classements TIOBE

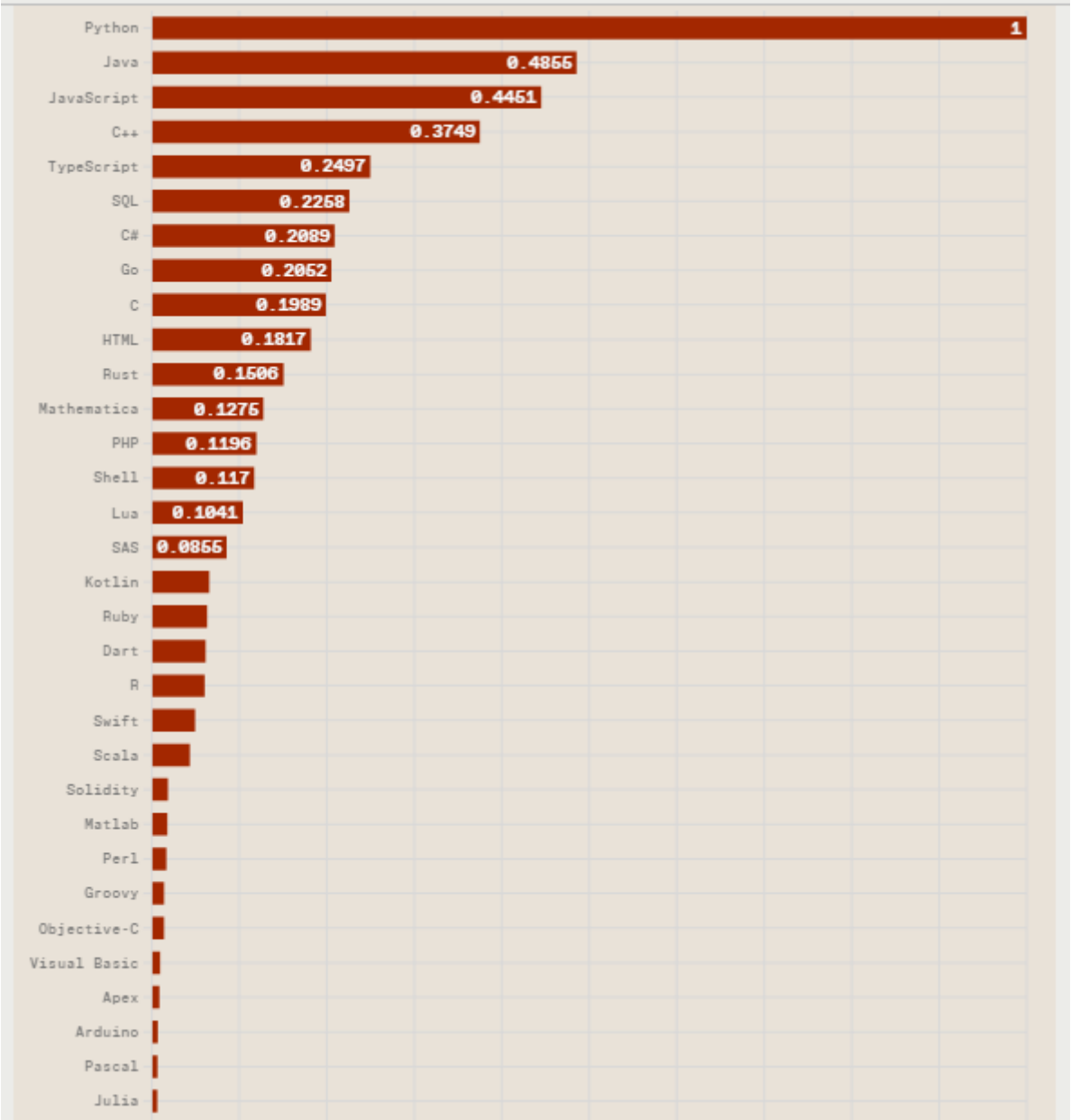

Figure 2 : Les classements TIOBE

Exemple de code simple :

```
print("Bonjour le monde !")
x = 5
y = 2
print("La somme est :", x + y)
```

Exécution :

Tapez dans le terminal : python mon_script.py

5. Librairies Python essentielles

Domaine	Librairie	Description
Traitement numérique	numpy	Calculs mathématiques rapides
Visualisation	matplotlib, seaborn	Graphiques et visualisation
Traitement de données	pandas	Tableaux et DataFrames
Apprentissage automatique	scikit-learn, tensorflow	Machine learning
Web	flask, django	Développement web
Système	os, sys	Interaction avec le système d'exploitation

6. Exécution des programmes Python

Méthodes d'exécution :

- Dans un terminal : python fichier.py
- Dans un IDE **(ex : PyCharm, VS Code, Spyder) :** Ouvrir le fichier, puis cliquer sur *Run*
- **En mode interactif :** python

  >>> print("Bonjour")
- **Depuis Jupyter Notebook :** Exécuter une cellule avec Shift + Enter.

7. . Environnements de développement Python

Environnement	Type	Particularité
IDLE	Basique (fourni avec Python)	Idéal pour débutants
PyCharm	Professionnel	Outils de debug, complétion, gestion de projets
Jupyter Notebook	Scientifique	Idéal pour l'analyse interactive et le data science
Spyder	Scientifique	Intégré à Anaconda, proche de MATLAB
VS Code	Polyvalent	Léger, très populaire, supporte Python, JS, etc.