Université Sultan Moulay Slimane
Ecole Supérieure de Technologie Béni Mellal
Département ITG
DUT- IDIA 1 - S2
M124 – Programmation en Python
Année universitaire: 2024-2025

Support de cours

PROGRAMMATION EN PYTHON

CHAPITRE I: INTRODUCTION AU LANGAGE PYTHON

Pr. Siham BAKKOURI
Email: s.bakkouri@usms.ma

Plan

- INTRODUCTION
- PYTHON EN INGÉNIERIE DES DONNÉES
- INSTALLATION ET CONFIGURATION
- SYNTAXE DE BASE, VARIABLES ET TYPES DE DONNÉES
- OPÉRATIONS MATHÉMATIQUES
- ENTRÉES ET SORTIES STANDARD
- MANIPULATION DES CHAÎNES DE CARACTÈRES
- EXERCICES

Introduction

Python est un langage de programmation interprété, polyvalent et facile à apprendre. Il est utilisé dans divers domaines, notamment **l'ingénierie des données**, le **développement web et l'intelligence artificielle**.

❑ Historique de Python

- Créé par **Guido van Rossum** en **1991**.
- Inspiré par les langages ABC et C, avec un objectif de simplicité et de lisibilité.

❑ Évolution et versions de Python

Python a évolué à travers plusieurs versions majeures :

- **Python 1.0 (1991)** : Premières fonctionnalités de base.
- **Python 2.0 (2000)** : Introduction du ramasse-miettes (garbage collector) et de la compatibilité Unicode.
- **Python 3.0 (2008)** : Rupture avec Python 2, amélioration de la gestion des chaînes de caractères et unification des types.

→ La dernière version stable de Python est la **3.13.2**, publiée le **4 février 2025**.
→ Site officiel : **https://www.python.org/**

❑ Caractéristiques de Python

- **Langage orienté objet**
- **Langage open-source** : Libre et gratuit
- **Langage interprété** : Exécution ligne par ligne sans compilation
- **Syntaxe claire et lisible** : Approche simplifiée par rapport à d'autres langages
- **Multiplateforme** : Compatible avec Windows, macOS et Linux
- **Large écosystème de bibliothèques** : Adapté au développement web, à l'IA, à la data science...
- **Modulable** : Python permet de structurer un programme en modules réutilisables dans d'autres projets Python
- **Langage dynamique** :
  - Pas de déclaration explicite du type des variables
  - Le type d'une variable peut changer en cours d'exécution

Python en ingénierie des données

Python est largement utilisé en ingénierie des données pour sa flexibilité et ses bibliothèques performantes. Il facilite la gestion, l'analyse et la visualisation des données.

→ Pourquoi Python pour l'ingénierie des données ?

- Manipulation efficace de grandes quantités de données.
- Automatisation des tâches de traitement et d'analyse.
- Intégration facile avec des bases de données et des outils de visualisation.

→ Applications concrètes

- Nettoyage et transformation des données.
- Analyse statistique et prédictive.
- Automatisation des processus de traitement des données.
- Développement de modèles d'apprentissage automatique.

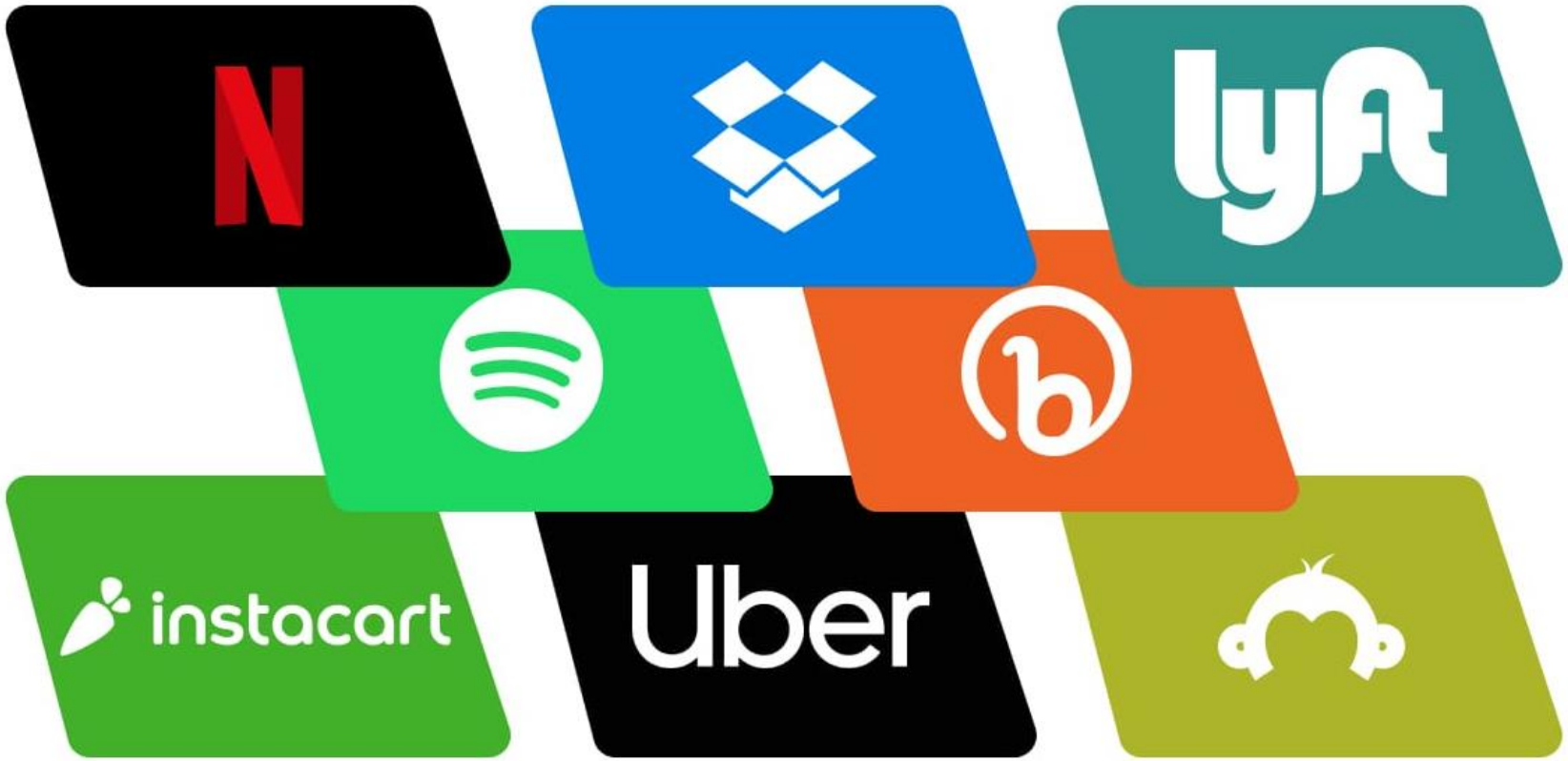
→ Bibliothèques essentielles

- **NumPy** : Calcul numérique et manipulation de tableaux.
- **Pandas** : Gestion et analyse de données sous forme de DataFrame.
- **Matplotlib & Seaborn** : Visualisation des données.
- **Scikit-learn** : Machine Learning et modélisation.

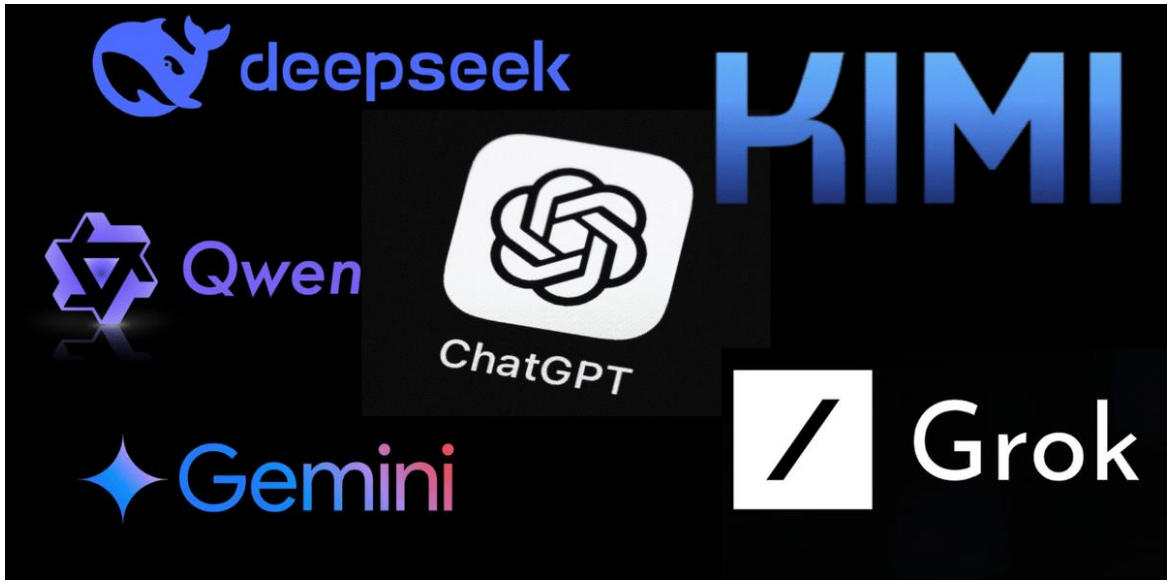
→ **Quelques applications connues utilisant Python pour ingénierie de données**

Python en ingénierie des données

→ **Quelques applications connues utilisant Python pour ingénierie de données**



→ Quelques applications connues utilisant Python pour ingénierie de données



Python est au cœur du développement de l'intelligence artificielle et de l'ingénierie des données. Il permet de :

- Manipuler et nettoyer d'énormes volumes de données textuelles.
- Construire et entraîner des modèles de deep learning.
- Déployer et optimiser les performances des modèles sur des infrastructures évolutives.
- …..

Installation et configuration

❑ Installation de Python

- Télécharger la dernière version de **Python** depuis le site officiel : Python.
- Sélectionner la version adaptée à votre système (**Windows, macOS, Linux**).
- Pendant l'installation, **cocher l'option "Add Python to PATH"** pour faciliter l'accès en ligne de commande.

❑ Environnements de Développement

Anaconda

- Idéal pour la data science et l'ingénierie des données.
- Installation via Anaconda

VS Code (Visual Studio Code): Installation des plugins recommandés :

- **Python Extension** (pour la coloration syntaxique et l'exécution).
- **Jupyter Notebook Extension** (pour travailler avec des notebooks).

PyCharm (JetBrains): Environnement avancé, idéal pour les projets complexes.

- Installation via PyCharm

Installation et configuration

❑ Environnements de Développement

→ Environnements en Ligne

- **Google Colab**: Basé sur Jupyter Notebook, accessible sur: Google Colab
  - Idéal pour la data science et le machine learning.
  - Permet d'exécuter du code sur GPU gratuitement.
- **Kaggle Notebooks**: Destiné aux scientifiques des données : Kaggle.
  - Permet d'exécuter du code avec accès à des datasets intégrés.
- **JupyterLite**: Version allégée de Jupyter Notebook accessible directement dans le navigateur : JupyterLite.

Installation et configuration

❑ Environnements de Développement

Environnement	Type	Avantages	Idéal pour
VS Code	Local	Flexible, extensible	Développement général
PyCharm	Local	Complet, outils avancés	Projets complexes
Anaconda (Jupyter Notebook)	Local	Visualisation interactive, data science	Analyse de données
Google Colab	En ligne	Exécution sur GPU, partage facile	Machine learning
Kaggle Notebooks	En ligne	Accès aux datasets, machine learning	Data science
JupyterLite	En ligne	Aucun besoin d'installation	Test rapide de code

❑ Gestion des Bibliothèques avec `pip`

`pip` est l'outil officiel de gestion des packages Python. Il permet d'installer des bibliothèques externes.

Exemple

```
pip install numpy
```

➔ Cette commande installe la bibliothèque ***NumPy***.

```
pip install pandas matplotlib
```

➔ Cette commande installe deux bibliothèques à la fois : ***Pandas*** et ***Matplotlib***.

→ Vérifier les packages installés

```
pip list
```

→ Mettre à jour `pip`

```
python -m pip install --upgrade pip
```

→ Tester l'Installation

- Ouvrir un terminal (ou cmd sous Windows) et taper : **`python –version`**

→ Syntaxe de Base

- Python est un langage à indentation obligatoire (***utilisation des espaces pour structurer le code***).
- Chaque instruction est écrite sur une nouvelle ligne, sans point-virgule (;).
- Les commentaires sont écrits avec # pour une ligne et """ """ pour plusieurs lignes.

Exemple

```
# Ceci est un commentaire sur une ligne
"""
Ceci est un commentaire
sur plusieurs lignes
"""
print("Bonjour, Python !")  # Affiche un message
```

→ Les variables

- Une variable est un espace mémoire qui stocke une valeur.
- Pas besoin de déclarer son type, Python le détecte automatiquement.
- On affecte une valeur avec =.

Exemple

```
nom = "Fatima"  # Chaîne de caractères
age = 18  # Entier
salaire = 9977.51  # Flottant
est_actif = True  # Booléen
```

→ Les Types de Données

Python gère plusieurs types de données :

Type	Description	Exemple
int	Entiers	10, -5, 1000
float	Nombres décimaux	3.14, -2.7, 0.99
str	Chaînes de caractères	"Python", 'Hello'
bool	Booléens	True, False
list	Liste (collection ordonnée)	[1, 2, 3]
tuple	Tuple (immuable)	(10, 20, 30)
dict	Dictionnaire (clé-valeur)	{"nom": "Alice", "âge": 25}
set	Ensemble (valeurs uniques)	{1, 2, 3, 4}

Opérations Mathématiques

→ Les opérateurs mathématiques de base

Python prend en charge les opérations arithmétiques classiques :

Opérateur	Description	Exemple	Résultat
+	Addition	10 + 5	15
-	Soustraction	10 - 5	5
*	Multiplication	10 * 5	50
/	Division (résultat flottant)	10 / 3	3.3333
//	Division entière	10 // 3	3
%	Modulo (reste de division)	10 % 3	1
**	Puissance	2 ** 3	8

Opérations Mathématiques

→ Les fonctions mathématiques de base

Python propose plusieurs fonctions utiles pour les calculs :

Fonction	Description	Exemple	Résultat
`abs(x)`	Valeur absolue	abs(-10)	10
`round(x, n)`	Arrondi à n décimales	round(3.14159, 2)	3.14
`max(x, y, ...)`	Maximum	max(5, 10, 3)	10
`min(x, y, ...)`	Minimum	min(5, 10, 3)	3
`pow(x, y)`	Puissance (x ** y)	pow(2, 3)	8

Opérations Mathématiques

→ Les fonctions mathématiques de base

Python propose un module **`math`** avec des fonctions avancées

- **Importation du module :** `import math`

Fonction	Description	Exemple	Résultat
math.sqrt(x)	Racine carrée	math.sqrt(16)	4.0
math.ceil(x)	Arrondi supérieur	math.ceil(3.1)	4
math.floor(x)	Arrondi inférieur	math.floor(3.9)	3
math.sin(x)	Sinus (radians)	math.sin(math.pi / 2)	1.0
math.cos(x)	Cosinus (radians)	math.cos(0)	1.0
math.log(x)	Logarithme naturel	math.log(10)	2.302

Entrées et Sorties Standard

→ Entrée avec `input()`

La fonction **`input()`** permet de récupérer une saisie de l'utilisateur sous forme de chaîne de caractères.

→ Sorties avec `print()`

La fonction **`print()`** permet d'afficher des informations à l'écran.

Exemple

```
nom = input("Entrez votre nom : ")
print("Bonjour", nom)
```

- On peut personnaliser l'affichage avec des séparateurs et des fins de ligne :

```
print("Python", "est", "puissant", sep="-", end="!\n")
```

```
Python-est-puissant!
```

Manipulation des Chaînes de Caractères

→ Définition et Création de Chaînes

En Python, une chaîne de caractères (string) est une séquence de caractères entourée de guillemets simples ('), doubles (") ou triples (""" """).

Exemple

```
texte1 = "Bonjour"
texte2 = 'Python'
texte3 = """Ceci est
une chaîne multilignes."""
```

→ Concaténation et Répétition

- **Concaténation** (+) : Fusionner deux chaînes.
- **Répétition** (*) : Répéter une chaîne plusieurs fois.

Exemple

```
nom = "Fatima"
message = "Bonjour, " + nom
print(message)  # Bonjour, Alice

print("Python " * 3)  # Python Python Python
```

Manipulation des Chaînes de Caractères

→ Accès aux Caractères et Sous-chaînes (Slicing)

- **Indexation** : Accéder aux caractères par leur position.
- Tronquer une chaîne (**slicing**) : Extraire une sous-chaîne.

Exemple

```
#Indexation
texte = "Python"
print(texte[0])    # P
print(texte[-1])   # n

#Slicing
print(texte[0:3])   # Pyt (de l'index 0 à 2)
print(texte[:4])    # Pyth (du début à l'index 3)
print(texte[2:])    # thon (de l'index 2 à la fin)
```

Manipulation des Chaînes de Caractères

→ Fonctions et Méthodes Utiles

- **Longueur d'une chaîne** ***len()*** : Retourne le nombre de caractères dans une chaîne.
- **Conversion en majuscules** ***upper()*** : Transforme tous les caractères en majuscules.
- **Conversion en minuscules** ***lower()*** : Transforme tous les caractères en minuscules.
- **Suppression des espaces** ***strip()*** : Supprime les espaces au début et à la fin d'une chaîne.
- **Remplacement de caractères** ***replace()*** : Remplace un mot ou un caractère par un autre.
- **Découpage d'une chaîne** ***split()*** : Sépare une chaîne en une liste de mots.
- **Vérification du début** ***startswith()*** : Vérifie si une chaîne commence par un certain mot.
- **Vérification de la fin** ***endswith()*** : Vérifie si une chaîne se termine par un certain mot.
- **Recherche d'une sous-chaîne** ***find()*** : Retourne la position d'un mot dans la chaîne.

Manipulation des Chaînes de Caractères

→ Fonctions et Méthodes Utiles

Exemple

```
texte = "  Bonjour tout le monde en Python  "

# Longueur de la chaîne
print("Longueur:", len(texte))

# Conversion en majuscules et minuscules
print("Majuscules:", texte.upper())
print("Minuscules:", texte.lower())

# Suppression des espaces
texte_sans_espaces = texte.strip()
print("Sans espaces:", texte_sans_espaces)

# Remplacement de caractères
texte_remplace = texte_sans_espaces.replace("Bonjour", "Salut")
print("Remplacement:", texte_remplace)

# Découpage de la chaîne
mots = texte_sans_espaces.split()
print("Découpage:", mots)

# Vérifications
print("Commence par 'Bonjour' ?", texte_sans_espaces.startswith("Bonjour"))
print("Se termine par 'Python' ?", texte_sans_espaces.endswith("Python"))
print("Position de 'monde':", texte_sans_espaces.find("monde"))
```

→ Formatage des Chaînes en Python

Le formatage des chaînes permet d'insérer des valeurs dynamiquement dans un texte. Voici deux méthodes courantes :

❑ Formatage avec *format()*

La méthode ***.format()*** remplace les **{}** par les valeurs fournies dans **.format()**

```
nom = "Fatima"
age = 20
print("Je suis {} et j'ai {} ans.".format(nom, age))
#On peut aussi préciser l'ordre des arguments :
print("Nom : {1}, Âge : {0}".format(age, nom))
```

```
Je suis Fatima et j'ai 20 ans.
Nom : Fatima, Âge : 20
```

→ Formatage des Chaînes en Python

Le formatage des chaînes permet d'insérer des valeurs dynamiquement dans un texte. Voici deux méthodes courantes :

❑ Formatage avec f-strings (*Python 3.6+*)

Les **f-strings** (***f"..."***) permettent d'insérer directement les variables entre **{}** sans ***.format().***

```
nom = "Fatima"
age = 20
print(f"Je suis {nom} et j'ai {age} ans.")
#Les f-strings permettent aussi d'évaluer des expressions :
print(f"L'année prochaine, j'aurai {age + 1} ans.")
```

```
Je suis Fatima et j'ai 20 ans.
L'année prochaine, j'aurai 21 ans.
```

Exercice 1

Un magasin vend des articles à un prix unitaire. Écris un programme qui :

1. Demande à l'utilisateur de saisir :
   - Le **nom du produit**
   - Le **prix unitaire** (en euros)
   - La **quantité achetée**
2. Calcule le **prix total** avant et après application d'une **réduction de 10%** si le montant dépasse **1000 Dh**.
3. Affiche les résultats en utilisant les **deux méthodes de formatage.**