**MODULE 1 : Introduction à Python et à la Data Science**

**Objectifs :**

* Comprendre la Data Science et ses applications.
* Apprendre les bases de Python.
* Installer l’environnement de travail.

1. **Introduction à la Data Science**

La **Data Science** (Science des données) est un domaine interdisciplinaire qui utilise des techniques **statistiques**, des **algorithmes informatiques** et des outils d'**intelligence artificielle** pour **analyser et exploiter les données** afin d'en extraire des informations utiles et prendre de meilleures décisions.

***1.1 Pourquoi la Data Science est-elle importante ?***

Les entreprises et organisations collectent des quantités massives de données (Big Data) à partir de diverses sources :

* Réseaux sociaux
* Transactions bancaires
* Capteurs IoT
* Logs de navigation internet
* Données médicales

Sans analyse appropriée, ces données restent **inutilisables**. La Data Science permet donc de **transformer ces données brutes en informations stratégiques**.

## *****1.2 Les Composantes de la Data Science*****

La Data Science repose sur plusieurs disciplines :

1. **Statistiques et Mathématiques**
   * Analyse des tendances et des distributions
   * Tests statistiques et modélisation
2. **Programmation (Python, R, SQL, etc.)**
   * Manipulation et nettoyage des données
   * Développement d’algorithmes
3. **Machine Learning et Intelligence Artificielle**
   * Prédictions et classifications
   * Automatisation des prises de décision
4. **Manipulation des Bases de Données**
   * Requêtes SQL et NoSQL
   * Stockage et traitement des Big Data
5. **Visualisation des Données**
   * Graphiques et dashboards interactifs
   * Aide à la prise de décision

## *****1.3 Applications de la Data Science*****

* **Finance** → Détection de fraudes, analyse des risques
* **Marketing** → Publicité ciblée, recommandation de produits
* **Santé** → Diagnostic médical assisté par IA
* **Industrie** → Maintenance prédictive des machines
* **Transport** → Optimisation des itinéraires (ex : Uber, Google Maps)

## *****1.4 Différence entre Data Science, Machine Learning et Deep Learning*****

Ces trois domaines sont souvent confondus, mais ils ont des différences clés. Voici une explication claire et détaillée.

**1.4.1 Data Science (Science des Données)**

La **Data Science** est un domaine qui englobe la collecte, l’analyse, l’exploration et l’interprétation des données pour **extraire des informations utiles** et aider à la prise de décision.

**Objectif :**  
Transformer des **données brutes** en **connaissances exploitables** grâce aux statistiques, à la programmation et à la visualisation des données.

**Outils et techniques :**

* **Manipulation des données** → Pandas, SQL
* **Statistiques et probabilités** → SciPy, Statsmodels
* **Visualisation des données** → Matplotlib, Seaborn
* **Big Data et Cloud** → Hadoop, Spark, Google Cloud

**Exemples d’applications :**

* Analyse des tendances de marché
* Prédiction des ventes
* Détection d’anomalies dans les données financières

**1.4.2 Machine Learning (Apprentissage Automatique)**

Le **Machine Learning** est une sous-discipline de la Data Science qui se concentre sur la création d'**algorithmes capables d’apprendre** automatiquement à partir des données sans programmation explicite.

**Objectif :**  
Permettre à une machine d’**apprendre à partir de données** et d’effectuer des **prédictions**.

**Types d'apprentissage :**

1. **Supervisé** (les données sont étiquetées) → Ex : classification d’e-mails (spam ou non spam)
2. **Non-supervisé** (les données ne sont pas étiquetées) → Ex : regroupement de clients en segments
3. **Apprentissage par renforcement** → Ex : entraînement d’un robot à jouer à un jeu vidéo

**Outils et technologies :**

* Scikit-learn (ML classique)
* TensorFlow, PyTorch (Deep Learning)
* XGBoost, LightGBM (Modèles avancés)

**Exemples d’applications :**

* Recommandations Netflix/Amazon
* Reconnaissance faciale
* Détection des fraudes bancaires

**1.4.3 Intelligence Artificielle (IA)**

L’**Intelligence Artificielle (IA)** est un domaine plus large qui englobe **toutes les techniques permettant à une machine d’imiter l’intelligence humaine**.

**Objectif :**  
Créer des machines capables de **raisonner, apprendre, comprendre le langage, et résoudre des problèmes** de manière autonome.

**Domaines de l’IA :**

* **Machine Learning** (une sous-partie de l’IA)
* **Vision par ordinateur** → Ex : reconnaissance d’images
* **Traitement du langage naturel (NLP)** → Ex : ChatGPT, assistants vocaux
* **Robots et agents intelligents** → Ex : robots autonomes

**Outils et technologies :**

* Deep Learning (réseaux neuronaux)
* OpenAI, IBM Watson
* GPT, BERT (modèles de NLP)

**Exemples d’applications :**

* Assistants virtuels (Siri, Alexa)
* Véhicules autonomes (Tesla)
* Traduction automatique (Google Translate)

**Tableau Comparatif : Data Science vs Machine Learning vs IA**

| **Critères** | **Data Science** | **Machine Learning** | **Intelligence Artificielle** |
| --- | --- | --- | --- |
| **Définition** | Analyse et exploitation des données | Algorithmes d'apprentissage à partir des données | Simulation de l’intelligence humaine |
| **Objectif** | Transformer les données en connaissances | Permettre à la machine d’apprendre et prédire | Créer des systèmes intelligents autonomes |
| **Méthodes** | Statistiques, analyse de données, visualisation | Algorithmes supervisés et non-supervisés | Réseaux neuronaux, NLP, robotique |
| **Exemples d’outils** | Pandas, SQL, Seaborn | Scikit-learn, XGBoost | TensorFlow, OpenAI |
| **Applications** | Études de marché, analyse financière | Détection de fraudes, recommandations | Voitures autonomes, assistants virtuels |

**Conclusion**

**Data Science** = Analyse et exploitation des données  
**Machine Learning** = Sous-domaine de la Data Science qui utilise des **algorithmes** pour apprendre à partir des données.  
**Intelligence Artificielle** = Domaine plus large qui inclut le Machine Learning et d'autres techniques avancées en cherchant à simuler l’intelligence humaine.

1. **Installation et Configuration**
   1. ***Installation de Python et Anaconda***

L’installation de **Python** et d’**Anaconda** est une étape essentielle pour commencer en **Data Science**. Voici un guide détaillé pour configurer ton environnement.

## ****2.1.1 Installation de Python****

Python est le langage principal utilisé en Data Science.

### **Étapes pour installer Python :**

1️-**Télécharger Python**

* Va sur le site officiel : <https://www.python.org/downloads/>
* Sélectionne la version **la plus récente** et compatible avec ton OS (Windows, macOS, Linux).

2️-**Lancer l’installation**

* Exécute le fichier téléchargé (python-<version>.exe).
* **Coche l’option "Add Python to PATH"** (important).
* Clique sur **Install Now** et attends la fin de l’installation.

3️-**Vérifier l’installation**

* Ouvre un terminal (cmd sous Windows ou Terminal sous macOS/Linux).
* Tape la commande :

*python --version*

Tu devrais voir quelque chose comme : Python 3.x.x

4️-**Installer pip (gestionnaire de paquets Python)**

* Pip est souvent installé avec Python, vérifie avec :

*pip --version*

* Si ce n’est pas le cas, installe-le avec :

*python -m ensurepip --default-pip*

## ****2.1.2 Installation d’Anaconda****

**Anaconda** est une distribution de Python qui contient **tous les outils nécessaires** pour la Data Science (Python, Jupyter Notebook, Pandas, NumPy, etc.).

### **Étapes pour installer Anaconda :**

1️-**Télécharger Anaconda**

* Va sur https://www.anaconda.com/download
* Choisis la version **Anaconda pour ton système (Windows/macOS/Linux)**.
* Sélectionne **Python 3.x** (la dernière version).

2️-**Lancer l’installation**

* Exécute le fichier téléchargé (Anaconda3-<version>.exe).
* **Suivre les étapes d’installation (Next, Next, Install…)**.

3️-**Vérifier l’installation**

* Ouvre un terminal (cmd sous Windows, Terminal sous macOS/Linux).
* Tape la commande suivante pour vérifier que conda est bien installé :

*conda --version*

Résultat attendu : conda 23.x.x (ou une autre version).

1. **Présentation de Jupyter Notebook et Google Colab**

Lorsque l'on travaille en **Data Science**, il est essentiel d'utiliser un environnement interactif pour **écrire, exécuter et visualiser** du code facilement. Deux outils très populaires sont :

* **Jupyter Notebook** (local, via Anaconda)
* **Google Colab** (en ligne, sans installation)

## *****3.1 Jupyter Notebook*****

Jupyter Notebook est un environnement interactif qui permet d’écrire du code **Python**, d’exécuter des cellules indépendamment et d'afficher des graphiques directement dans l'interface.

### **Avantages :**

* Permet d’écrire du code **Python interactif**
* Idéal pour **l’analyse de données et le Machine Learning**
* Supporte les **graphiques et visualisations intégrés**
* Compatible avec **Markdown** (pour structurer des notes)
* Fonctionne **localement** sur ton PC

### **Comment ouvrir Jupyter Notebook ?**

**Si tu as installé Anaconda** :

1. Ouvre **Anaconda Navigator** et clique sur **"Launch"** à côté de **Jupyter Notebook**
2. Un onglet s’ouvrira dans ton navigateur

**Via le terminal / cmd** :

1. Ouvre un terminal (cmd sous Windows, Terminal sous macOS/Linux)
2. Tape la commande suivante et appuie sur **Entrée** :

*jupyter notebook*

1. Un **navigateur web** s'ouvre avec Jupyter

**Créer un nouveau notebook** :

* Clique sur **New** → **Python 3**
* Un notebook vide s'ouvre et tu peux commencer à coder

### **Exécuter du code dans Jupyter**

Dans une cellule, tape :

*print("Hello, Data Science!")*

Puis appuie sur **Shift + Entrée** pour exécuter la cellule.

### **Insérer du texte en Markdown**

Dans une cellule, écris :

# Titre en gros

## Sous-titre

- Point 1

- Point 2

Puis change le type de cellule en **Markdown** et exécute-la pour voir le rendu.

## *****3.2 Google Colab (Google Colaboratory)*****

Google Colab est un environnement **basé sur Jupyter Notebook**, mais **hébergé sur le cloud**. Il permet d'exécuter du code **sans installation** et d'utiliser gratuitement des **GPU/TPU** pour le Machine Learning.

### **Avantages :**

* **Aucune installation** requise (fonctionne directement via Google Chrome)
* Accès à **des GPU et TPU gratuits** (utile pour le Deep Learning)
* Sauvegarde automatique sur **Google Drive**
* **Collaboration en temps réel** (partage avec d'autres utilisateurs)
* Compatible avec **Jupyter Notebook**

### **Comment ouvrir Google Colab ?**

1️-**Accéder à Colab**

* Ouvre https://colab.research.google.com
* Clique sur **"Nouveau Notebook"**

2️-**Créer un Notebook**

* Un environnement similaire à Jupyter Notebook s'ouvre
* Commence à écrire et exécuter du code Python directement

3️-**Exécuter une cellule**  
Dans une cellule, écris :

*print("Hello, Google Colab!")*

Puis appuie sur **Shift + Entrée**.

4️-**Accéder aux fichiers Google Drive**  
Tu peux lier ton Google Drive pour sauvegarder tes fichiers :

*from google.colab import drive*

*drive.mount('/content/drive')*

5️-**Activer un GPU** (Graphics Processing Unit) **pour le Deep Learning**  
Va dans **Runtime > Modifier le type d’exécution > Accélérateur matériel : GPU/TPU**

## ****Tableau Comparatif : Jupyter Notebook vs Google Colab****

| **Critères** | **Jupyter Notebook** | **Google Colab** |
| --- | --- | --- |
| **Installation requise** | Oui (via Anaconda) | Non (100% en ligne) |
| **Utilisation** | Localement sur ton PC | Cloud (accès depuis Chrome) |
| **Stockage** | Fichiers locaux | Google Drive |
| **GPU/TPU** | Non (nécessite une installation supplémentaire) | Oui (gratuit) |
| **Collaboration** | Partage manuel des fichiers | Collaboration en temps réel |

## ****Conclusion****

**Jupyter Notebook** est idéal si tu veux **travailler localement** et garder tes fichiers sur ton PC.  
**Google Colab** est parfait si tu veux **travailler en ligne sans installation** et profiter des **GPU gratuits**.

1. **Introduction aux bibliothèques essentielles : numpy, pandas, matplotlib, seaborn, scikit-learn.**

En **Data Science**, on utilise plusieurs bibliothèques Python pour manipuler, analyser et visualiser les données. Voici un aperçu des 5 bibliothèques essentielles :

* **NumPy** → Manipulation des tableaux de données (arrays)
* **Pandas** → Manipulation et analyse de données tabulaires
* **Matplotlib** → Visualisation basique (graphiques)
* **Seaborn** → Visualisation avancée et esthétique
* **Scikit-learn** → Machine Learning (modèles prédictifs)

***4.1 NumPy : Manipulation des tableaux de données***

* Permet de **créer et manipuler des tableaux multidimensionnels** (numpy.array).
* Plus rapide et efficace que les **listes Python classiques**.
* Offre des fonctions mathématiques avancées (moyenne, variance, etc.).

**Installation :**  
Si NumPy n'est pas installé, exécute :

*pip install numpy*

**Exemple d'utilisation :**

*import numpy as np*

*# Créer un tableau NumPy (array)*

*arr = np.array([1, 2, 3, 4, 5])*

*# Afficher le tableau et ses propriétés*

*print(arr)*

*print("Shape :", arr.shape)*

*print("Moyenne :", np.mean(arr))*

***4.2 Pandas : Manipulation et analyse de données***

* Manipule des **données sous forme de tableaux (DataFrame)**.
* Permet d'**importer/exporter des fichiers CSV, Excel, SQL**.
* Outils puissants pour **le nettoyage et la transformation des données**.

**Installation :**

*pip install pandas*

**Exemple d'utilisation :**

*import pandas as pd*

*# Charger un fichier CSV*

*df = pd.read\_csv("data.csv")*

*# Afficher les 5 premières lignes*

*print(df.head())*

*# Résumé statistique des données*

*print(df.describe())*

*# Sélectionner une colonne*

*print(df["NomColonne"])*

*# Filtrer les données*

*print(df[df["Age"] > 30])*

***4.3 Matplotlib : Visualisation des données (graphiques de base)***

* Permet de tracer des **graphiques de base** (courbes, barres, histogrammes, etc.).
* Offre un contrôle total sur l’apparence des graphes.

**Installation :**

*pip install matplotlib*

**Exemple d'utilisation :**

*import matplotlib.pyplot as plt*

*# Créer des données*

*x = [1, 2, 3, 4, 5]*

*y = [10, 15, 7, 12, 9]*

*# Tracer un graphique simple*

*plt.plot(x, y, marker="o", linestyle="-", color="blue")*

*# Ajouter des titres et labels*

*plt.xlabel("X-axis")*

*plt.ylabel("Y-axis")*

*plt.title("Exemple de graphique")*

*# Afficher le graphique*

*plt.show()*

***4.4 Seaborn : Visualisation avancée et esthétique***

* Plus **élégant** et puissant que Matplotlib.
* Permet de **visualiser des relations complexes** entre les données.
* Intégré avec **Pandas** pour un affichage facile.

**Installation :**

*pip install seaborn*

**Exemple d'utilisation :**

*import seaborn as sns*

*import matplotlib.pyplot as plt*

*# Charger un dataset intégré à Seaborn*

*df = sns.load\_dataset("tips")*

*# Créer un graphique en nuage de points*

*sns.scatterplot(x="total\_bill", y="tip", data=df, hue="sex")*

*# Afficher le graphique*

*plt.show()*

***4.5 Scikit-learn : Machine Learning (modèles prédictifs)***

* Permet d’**appliquer des algorithmes de Machine Learning** facilement.
* Contient des outils pour **le prétraitement des données, l'évaluation et l'entraînement des modèles**.

**Installation :**

*pip install scikit-learn*

**Exemple d'utilisation (Régression Linéaire) :**

*from sklearn.linear\_model import LinearRegression*

*import numpy as np*

*# Données d'entraînement*

*X = np.array([[1], [2], [3], [4], [5]])*

*y = np.array([2, 4, 5, 4, 5])*

*# Créer un modèle de régression linéaire*

*model = LinearRegression()*

*# Entraîner le modèle*

*model.fit(X, y)*

*# Prédire une nouvelle valeur*

*prediction = model.predict([[6]])*

*print("Prédiction pour X=6 :", prediction)*

**Tableau comparatif des bibliothèques :**

| **Bibliothèque** | **Utilité principale** | **Exemples d'utilisation** |
| --- | --- | --- |
| **NumPy** | Tableaux multidimensionnels | Opérations mathématiques, statistiques |
| **Pandas** | Manipulation de données tabulaires | Lire CSV, filtrer, grouper des données |
| **Matplotlib** | Graphiques de base | Courbes, histogrammes, camemberts |
| **Seaborn** | Visualisation avancée | Graphiques interactifs et esthétiques |
| **Scikit-learn** | Machine Learning | Régressions, classifications, clustering |

1. **Bases de Python pour la Data Science**

Avant de plonger dans la **Data Science**, il est essentiel de maîtriser les **bases de Python**. Voici un guide avec les **concepts fondamentaux** et des **exemples concrets**.

***5.1 Variables et types de données***

En **Python**, une variable est un espace mémoire qui stocke une valeur. Contrairement à d'autres langages, Python **ne nécessite pas de déclarer le type de variable** : il est défini automatiquement.

5.1.1 Déclaration et Affectation de Variables

En Python, une variable est créée lorsqu'on lui attribue une valeur avec le signe =.

# Déclaration de variables

nom = "Alice" # Chaîne de caractères (str)

âge = 25 # Entier (int)

taille = 1.75 # Flottant (float)

étudiant = True # Booléen (bool)

# Affichage des variables

print(nom)

print(âge)

print(taille)

print(étudiant)

**Remarque :** Python est un langage **dynamique**, donc une variable peut changer de type en cours d’exécution.

x = 10 # x est un entier

x = "texte" # x devient une chaîne de caractères

5.1.2 Vérification du Type d'une Variable

Pour connaître le type d'une variable, on utilise la fonction type().

print(type(nom)) # <class 'str'>

print(type(âge)) # <class 'int'>

print(type(taille)) # <class 'float'>

print(type(étudiant)) # <class 'bool'>

5.1.3 Principaux Types de Données

Python propose plusieurs types de données intégrés :

| **Type** | **Description** | **Exemple** |
| --- | --- | --- |
| int | Entiers (positifs ou négatifs) | x = 42 |
| float | Nombres décimaux | y = 3.14 |
| str | Chaînes de caractères | nom = "Alice" |
| bool | Booléens (True ou False) | est\_étudiant = True |
| list | Listes (collections modifiables) | fruits = ["Pomme", "Banane"] |
| tuple | Tuples (collections immuables) | coord = (10, 20) |
| dict | Dictionnaires (clé-valeur) | pers = {"nom": "Alice", "âge": 25} |
| set | Ensembles (collections uniques) | nombres = {1, 2, 3, 4} |

5.1.4 Conversion de Types (Type Casting)

On peut convertir une variable d'un type à un autre avec int(), float(), str(), etc.

# Conversion d'un entier en chaîne de caractères

age = 25

age\_str = str(age) # "25"

# Conversion d'une chaîne en entier

nombre = "10"

nombre\_int = int(nombre) # 10

# Conversion d'un nombre en flottant

nombre\_float = float(nombre) # 10.0

# Conversion d'un flottant en entier (arrondi vers le bas)

pi = 3.14

pi\_int = int(pi) # 3

print(type(age\_str), type(nombre\_int), type(nombre\_float), type(pi\_int))

**Remarque** : int("abc") déclenchera une erreur car la chaîne ne représente pas un nombre valide.

5.1.5 Manipulation des Chaînes de Caractères (str)

Les **chaînes de caractères** sont entourées par " " ou ' '.

# Déclaration d'une chaîne

texte = "Bonjour, Python !"

# Accéder aux caractères (indexation)

print(texte[0]) # B

print(texte[-1]) # !

# Découpage (slicing)

print(texte[:7]) # "Bonjour"

# Longueur d'une chaîne

print(len(texte)) # 17

# Mise en majuscules/minuscules

print(texte.upper()) # "BONJOUR, PYTHON !"

print(texte.lower()) # "bonjour, python !"

# Remplacement d'un mot

print(texte.replace("Python", "Data Science"))

# Vérification de présence d'un mot

print("Python" in texte) # True

## **5.1.6 Les Listes (**list**)**

Les **listes** sont des collections **modifiables** permettant de stocker plusieurs éléments.

*# Déclaration d'une liste*

*fruits = ["Pomme", "Banane", "Orange"]*

*# Accéder aux éléments*

*print(fruits[0]) # "Pomme"*

*# Modifier un élément*

*fruits[1] = "Mangue"*

*# Ajouter un élément*

*fruits.append("Kiwi")*

*# Supprimer un élément*

*fruits.remove("Orange")*

*# Trier la liste*

*fruits.sort()*

*print(fruits) # ['Kiwi', 'Mangue', 'Pomme']*

## **5.1.7 Les Dictionnaires (**dict**)**

Les **dictionnaires** stockent des paires **clé → valeur**.

*# Création d'un dictionnaire*

*personne = {*

*"nom": "Alice",*

*"âge": 25,*

*"ville": "Paris"*

*}*

*# Accéder aux valeurs*

*print(personne["nom"]) # Alice*

*# Modifier une valeur*

*personne["âge"] = 26*

*# Ajouter une clé*

*personne["email"] = "alice@gmail.com"*

*# Supprimer une clé*

*del personne["ville"]*

*print(personne)*

## **5.1.8 Les Booléens (**bool**) et Opérateurs Logiques**

Les valeurs booléennes sont **True** ou **False**.

*a = 10*

*b = 5*

*print(a > b) # True*

*print(a == b) # False*

*print(a != b) # True*

### **Opérateurs logiques :**

| **Opérateur** | **Description** | **Exemple** |
| --- | --- | --- |
| and | Vrai si les deux conditions sont vraies | (a > 5) and (b < 10) |
| or | Vrai si au moins une condition est vraie | (a > 5) or (b > 10) |
| not | Inverse une condition | not(a > 5) |

Exemple :

*x = True*

*y = False*

*print(x and y) # False*

*print(x or y) # True*

*print(not x) # False*

## ****Résumé des Concepts Importants****

| **Concept** | **Description** | **Exemple** |
| --- | --- | --- |
| **Variable** | Stocke une valeur | x = 10 |
| **Type dynamique** | Python détermine le type automatiquement | y = "texte" |
| **Conversion de type** | Changer un type en un autre | int("10") → 10 |
| **Chaînes (str)** | Texte modifiable | "Python".upper() |
| **Listes (list)** | Collection modifiable | fruits.append("Pomme") |
| **Dictionnaires (dict)** | Clé-Valeur | {"nom": "Alice"} |
| **Booléens (bool)** | True ou False | x > 5 → True |
| **Opérateurs logiques** | and, or, not | if a > 5 and b < 10: |

## ****Conclusion****

* En maîtrisant les **variables et types de données**, tu es prêt à manipuler des **données plus complexes** en Data Science.
* La prochaine étape est d’apprendre à **utiliser les bibliothèques Python** comme **NumPy et Pandas** pour manipuler des données efficacement.

***5.2 Structures de contrôle : boucles et conditions***

Les **structures de contrôle** permettent d’exécuter des blocs de code **sous certaines conditions** ou de **répéter** une action plusieurs fois.

**Principaux types :**

* **Conditions (if, elif, else)** → Exécuter un bloc selon une condition.
* **Boucles (for, while)** → Répéter une action.
* **Instructions de contrôle (break, continue, pass)** → Modifier le comportement des boucles.

## **5.2.1 Les Conditions (**if**,** elif**,** else**)**

Une condition permet **d’exécuter un bloc de code si une expression est vraie**.

*age = 20*

*if age >= 18:*

*print("Vous êtes majeur.") # S'exécute car 20 >= 18*

*else:*

*print("Vous êtes mineur.")*

### **Ajout d’un** else if **(Sinon Si)**

*note = 75*

*if note >= 90:*

*print("Excellent")*

*elif note >= 70:*

*print("Bien") # S'exécute car 75 >= 70*

*elif note >= 50:*

*print("Moyen")*

*else:*

*print("Échec")*

### **Conditions Multiples (**and**,** or**,** not**)**

| **Opérateur** | **Description** | **Exemple** |
| --- | --- | --- |
| and | Vrai si **toutes** les conditions sont vraies | if age > 18 and note > 70: |
| or | Vrai si **au moins une** condition est vraie | if age > 18 or note > 70: |
| not | Inverse une condition | if not is\_student: |

*age = 25*

*note = 60*

*if age >= 18 and note >= 50:*

*print("Admis") # Les* deux conditions sont vraies

## **5.2.2 Boucles (**for**,** while**)**

Les boucles permettent **de répéter des instructions**.

### **Boucle** for **: Itération sur une Séquence**

On utilise for pour parcourir une liste, un tuple, une chaîne, un dictionnaire ou un range.

**Exemple 1 : Boucle sur une liste**

*fruits = ["Pomme", "Banane", "Orange"]*

*for fruit in fruits:*

*print(fruit) # Affiche chaque fruit un par un*

**Exemple 2 : Utilisation de range()**

*for i in range(5): # De 0 à 4 (exclut 5)*

*print(i)*

**Exemple 3 : Boucle avec range(start, stop, step)**

*for i in range(1, 10, 2): # De 1 à 9 avec un pas de 2*

*print(i)*

### **Boucle** while **: Répétition tant qu’une Condition est Vraie**

*x = 0*

*while x < 5: # Tant que x est inférieur à 5*

*print(x)*

*x += 1 # Incrémentation*

## **5.2.3 Instructions de Contrôle**

Parfois, on souhaite **modifier l’exécution d’une boucle**.

| **Instruction** | **Description** | **Exemple** |
| --- | --- | --- |
| break | Arrête la boucle | if x == 3: break |
| continue | Passe à l’itération suivante | if x == 3: continue |
| pass | Place un bloc vide sans erreur | if x == 3: pass |

**Exemple 1 : break pour sortir d’une boucle**

*for i in range(10):*

*if i == 5:*

*break # Arrête la boucle quand i vaut 5*

*print(i)*

**Exemple 2 : continue pour ignorer une itération**

*for i in range(5):*

*if i == 2:*

*continue # Ignore le tour où i = 2*

*print(i)*

**Exemple 3 : pass pour éviter une erreur dans un bloc vide**

*for i in range(5):*

*if i == 3:*

*pass # Place réservé*

*print(i)*

## ****Résumé des Concepts Clés****

| **Concept** | **Description** | **Exemple** |
| --- | --- | --- |
| **Condition if** | Vérifie une condition | if age > 18: |
| **elif et else** | Gère plusieurs cas | elif note > 50: |
| **and, or, not** | Opérateurs logiques | if a > 5 and b < 10: |
| **Boucle for** | Itération sur une séquence | for x in liste: |
| **range()** | Générer une séquence numérique | range(1, 10, 2) |
| **Boucle while** | Répète tant qu’une condition est vraie | while x < 5: |
| **break** | Quitte une boucle | if x == 5: break |
| **continue** | Passe à l’itération suivante | if x == 2: continue |
| **pass** | Laisse un bloc vide | if x == 3: pass |

**Les conditions et les boucles sont essentielles** en Data Science pour filtrer, analyser et transformer les données.