Chapitre 2
Python et Analyse de Données pour la Business Intelligence

1 Introduction générale

La Business Intelligence (BI) a pour objectif principal de transformer des données brutes en informations pertinentes afin d'aider les décideurs à prendre des décisions stratégiques. Dans ce contexte, les outils informatiques jouent un rôle fondamental, en particulier les langages capables de collecter, traiter, analyser et visualiser les données.

Parmi ces outils, le langage Python s'est imposé comme une référence incontournable grâce à sa simplicité, sa flexibilité et son riche écosystème de bibliothèques dédiées à l'analyse de données. Ce chapitre présente les bases du langage Python et explique son rôle central dans le processus d'analyse de données appliqué à la Business Intelligence.

2 Business Intelligence : cadre conceptuel et enjeux

La Business Intelligence peut être définie comme un ensemble de technologies et de méthodologies visant à collecter, intégrer, analyser et restituer des données afin de soutenir la prise de décision stratégique. Elle repose sur une vision globale des données de l'entreprise et vise à améliorer la qualité, la rapidité et la pertinence des décisions.

2.1 Enjeux stratégiques de la Business Intelligence

Les enjeux de la BI sont multiples :

— amélioration de la performance organisationnelle,
— anticipation des tendances du marché,
— optimisation des processus internes,
— réduction des risques décisionnels,
— valorisation du patrimoine informationnel.

Dans un environnement concurrentiel, la capacité à exploiter efficacement les données constitue un avantage compétitif décisif.

FIGURE 1 – Domaines en relation à l'analyse des données

2.2 BI, aide à la décision et gouvernance des données

La BI ne se limite pas à la production de rapports. Elle s'inscrit dans une démarche globale de gouvernance des données, visant à garantir leur qualité, leur cohérence et leur traçabilité. Les outils de BI doivent ainsi être intégrés dans une architecture cohérente, alignée sur les objectifs stratégiques de l'organisation.

3 Architecture d'un système de Business Intelligence

Un système de Business Intelligence repose sur une architecture multicouche, structurée autour de plusieurs composants interdépendants.

3.1 Sources de données

Les données peuvent provenir de sources internes (bases de données opérationnelles, ERP, CRM) ou externes (fichiers, API, open data, capteurs).

3.2 Processus ETL

Les processus ETL (Extract, Transform, Load) permettent :

— l'extraction des données depuis les sources,

— leur transformation (nettoyage, normalisation, agrégation),

— leur chargement dans un entrepôt de données.

Python est fréquemment utilisé pour implémenter ces processus grâce à sa capacité à automatiser des traitements complexes.

3.3 Entrepôt de données

L'entrepôt de données constitue le socle du système BI. Il centralise les données historiques et permet des analyses multidimensionnelles.

3.4 Couche analytique et restitution

La couche analytique comprend les outils d'analyse statistique, de data mining et de visualisation. Python intervient principalement à ce niveau.

4 Place de Python dans un système de Business Intelligence

Un système de Business Intelligence repose généralement sur plusieurs couches :

— la collecte des données à partir de différentes sources,

— le stockage dans des bases de données ou des entrepôts de données,

— l'analyse des données,

— la restitution des résultats sous forme de rapports et tableaux de bord.

Python intervient principalement dans les phases d'analyse et de visualisation, mais il est également utilisé dans les processus d'extraction, de transformation et de chargement des données (ETL). Grâce à ses bibliothèques spécialisées, Python constitue un lien efficace entre les bases de données, les modèles analytiques et les outils de restitution décisionnelle.

5 Historique et philosophie du langage Python

Python a été créé en 1991 par Guido van Rossum. Il s'agit d'un langage interprété dont la conception met l'accent sur la lisibilité du code et la simplicité de la syntaxe. Contrairement à certains langages plus complexes, Python privilégie une écriture claire et concise, facilitant ainsi l'apprentissage, la maintenance et la collaboration entre développeurs.

L'apparition de Python 3 a marqué une évolution majeure du langage, avec des améliorations significatives en termes de performances, de gestion de la mémoire et de typage. Aujourd'hui, Python est largement utilisé dans les domaines de la data science, de l'intelligence artificielle et de la Business Intelligence.

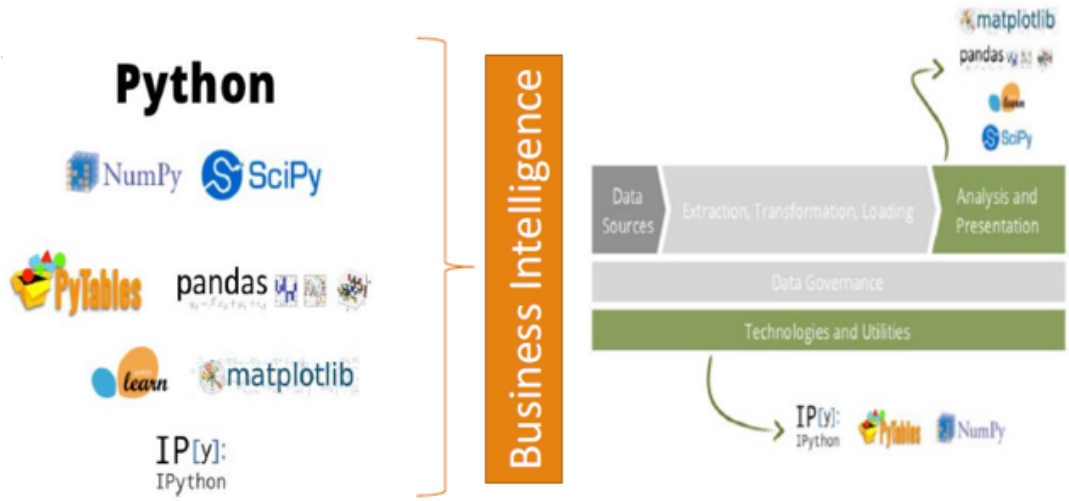

FIGURE 2 – Python et BI

FIGURE 3 – Langage Python

6 Présentation générale du langage Python

Python est un langage polyvalent qui permet :

— la manipulation de différents types de données,

— la structuration du code à l'aide de fonctions et de classes,

— le découpage des programmes en modules,

— l'interaction avec des bases de données et des services externes.

Son mode d'exécution repose sur l'interprétation de fichiers scripts portant l'extension `.py`, ce qui facilite le prototypage rapide et les phases de test.

6.1 Avantages de Python pour la BI

Python présente plusieurs avantages dans un contexte de Business Intelligence :

— langage libre et open source,

— portabilité sur différents systèmes d'exploitation,

— syntaxe simple et lisible,

— typage dynamique favorisant la rapidité de développement,
— vaste écosystème de bibliothèques analytiques,
— adoption massive dans le monde professionnel.

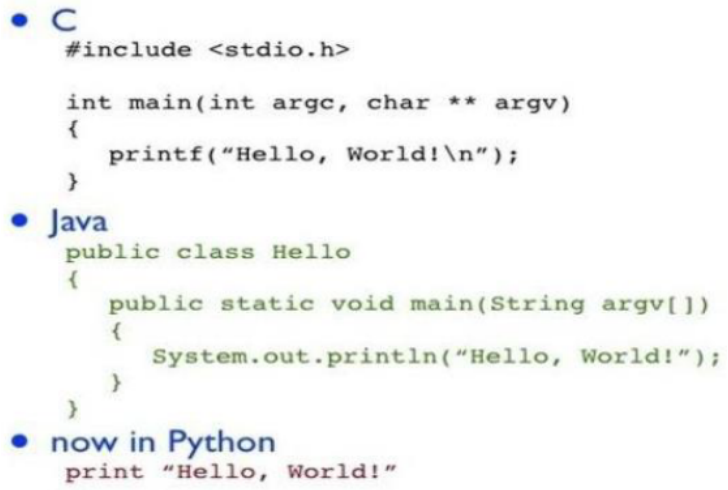

FIGURE 4 – Python vs autres langages

7 Environnements de développement Python

Plusieurs environnements permettent de développer en Python :
— IDLE,
— PyCharm,
— Visual Studio Code,
— Jupyter Notebook.
— Colab

Dans le cadre de l'analyse de données et de l'enseignement, Jupyter Notebook est particulièrement adapté car il permet de combiner texte explicatif, code Python et visualisations graphiques.

Python offre une approche intégrée combinant traitement, analyse et visualisation.

7.1 Mode interactif et mode script

Python peut être utilisé selon deux modes principaux :
— **Mode interactif** : exécution des instructions ligne par ligne, idéal pour les tests et l'exploration.
— **Mode script** : exécution d'un programme complet à partir d'un fichier, adapté aux projets structurés.

7.2 Commentaires, aide et modularité

Les commentaires améliorent la lisibilité du code :

— # pour un commentaire sur une seule ligne,

— ''' ''' pour un commentaire multiligne.

Python propose également une aide intégrée via la fonction `help()` et une grande modularité grâce au mécanisme d'importation des modules.

7.3 Bibliothèques Python utilisées en Business Intelligence

-NumPy

NumPy est utilisé pour le calcul numérique et matriciel.

-Pandas

Pandas est la bibliothèque principale pour la manipulation des données.

-Matplotlib et Seaborn

Ces bibliothèques sont utilisées pour la visualisation des données.

-Scikit-learn

Scikit-learn fournit des algorithmes de machine learning et des outils de prétraitement.

8 Analyse de données : méthodologie complète

L'analyse de données constitue le cœur de la Business Intelligence. Elle vise à transformer des données brutes en connaissances exploitables. Elle constitue un élément central de la Business Intelligence.

8.1 Importance de l'analyse de données

L'analyse de données permet notamment de :

— identifier des tendances,

— comprendre des comportements,

— optimiser les processus organisationnels,

— appuyer les décisions stratégiques.

9 Processus d'analyse de données

Le processus d'analyse de données peut être décomposé en cinq étapes principales.

9.1 Inspection des données

Cette étape permet d'évaluer la structure et la qualité des données.

```
import pandas as pd
data = pd.read_csv("data.csv")
data.info()
data.describe()
```

9.2 Nettoyage des données

Le nettoyage permet de garantir la fiabilité des résultats analytiques en supprimant les incohérences..

```
data = data.drop_duplicates()
data = data.dropna()
```

9.3 Transformation et préparation

La transformation permet d'adapter les données aux besoins de l'analyse.

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
data_scaled = scaler.fit_transform(data)
```

9.4 Modélisation

La modélisation permet d'extraire des tendances, des relations ou des prédictions.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

9.5 Interprétation et prise de décision

Les résultats sont interprétés à l'aide de visualisations.

```
import matplotlib.pyplot as plt
plt.hist(data["variable"])
plt.show()
```

10 Exemple d'utilisation de Python dans la chaîne décisionnelle en Business Intelligence

Dans cette section, nous présentons un exemple illustratif de l'utilisation du langage Python à travers les différentes étapes de la chaîne décisionnelle en Business Intelligence, allant de la planification jusqu'à la restitution des résultats.

10.1 Planification

Objectif : définir les besoins décisionnels et analyser les tendances afin d'anticiper l'évolution de la demande.

Python permet d'analyser des données historiques et de produire des visualisations simples facilitant la phase de planification.

```
import pandas as pd
import matplotlib.pyplot as plt

data = {
    "Mois": ["Jan", "Fév", "Mars"],
    "Demande": [100, 150, 130]
}

df = pd.DataFrame(data)

plt.plot(df["Mois"], df["Demande"], marker='o')
plt.xlabel("Mois")
plt.ylabel("Demande")
plt.title("Prévision de la demande")
plt.show()
```

10.2 ETL : Extraction, Transformation et Chargement

Objectif : extraire les données, les transformer afin d'améliorer leur qualité, puis les charger vers un système cible.

```
import pandas as pd
from sqlalchemy import create_engine

# Extraction
df = pd.read_csv("donnees.csv")

# Transformation
df["Prix_TTC"] = df["Prix_HT"] * 1.2

# Chargement
engine = create_engine("sqlite:///database.db")
df.to_sql("Produits", con=engine, if_exists="replace", index=False)
```

10.3 Stockage des données

Objectif : stocker les données de manière structurée afin de faciliter leur exploitation ultérieure.

```
import sqlite3

conn = sqlite3.connect("entreprise.db")
cursor = conn.cursor()

cursor.execute("""
CREATE TABLE IF NOT EXISTS Stock (
    id INTEGER PRIMARY KEY,
    produit TEXT,
    quantite INTEGER
)
""")

cursor.execute(
    "INSERT INTO Stock (produit, quantite) VALUES (?, ?)",
    ("Ordinateur", 10)
)

conn.commit()
conn.close()
```

10.4 Analyse des données

Objectif : analyser les données stockées afin d'extraire des indicateurs utiles à la prise de décision.

```
import numpy as np

quantites = np.array([10, 15, 20, 25])
moyenne_stock = np.mean(quantites)

print(f"Stock moyen : {moyenne_stock}")
```

10.5 Restitution des résultats

Objectif : présenter les résultats analytiques sous forme de graphiques ou de rapports compréhensibles par les décideurs.

```
import matplotlib.pyplot as plt

labels = ["Produit A", "Produit B", "Produit C"]
sizes = [40, 35, 25]

plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=140)
plt.title("Répartition des ventes")
plt.show()
```

Bonnes pratiques professionnelles

Une analyse BI efficace repose sur :

— la qualité et la traçabilité des données,

— la documentation du code,

— la reproductibilité des analyses,

— l'interprétation critique des résultats.

11 Exercice

1. Charger un jeu de données réel et analyser sa structure.
2. Nettoyer et transformer les données.
3. Visualiser les résultats et proposer une interprétation décisionnelle.

12 Conclusion

Dans ce chapitre, nous avons présenté le langage Python comme un outil central de l'analyse de données en Business Intelligence. Après avoir introduit les fondements du langage et son écosystème, nous avons mis en évidence son rôle à travers les différentes étapes du processus décisionnel, depuis la planification et les traitements ETL jusqu'à l'analyse et la restitution des résultats.

Grâce à ses bibliothèques spécialisées telles que Pandas, NumPy, Matplotlib et SQLAlchemy, Python permet de manipuler efficacement de grands volumes de données, de réaliser des analyses exploratoires et d'automatiser des traitements analytiques complexes. Sa simplicité syntaxique et sa flexibilité en font un langage accessible, tout en répondant aux exigences professionnelles des systèmes de Business Intelligence.