**Liste des fonctions Python utilisées en Data Science**

**1. Fonctions Python natives utiles**

# print() : affiche une valeur à l'écran

print("Bonjour")

# type() : retourne le type de l'objet

print(type(10)) # <class 'int'>

# len() : retourne la longueur d'un objet (liste, chaîne, etc.)

data = [1, 2, 3]

print(len(data)) # 3

# range() : génère une séquence de nombres

for i in range(3):

print(i) # 0, 1, 2

# list(), dict(), set() : conversions de types

print(list((1, 2, 3))) # [1, 2, 3]

print(dict([("a", 1), ("b", 2)])) # {"a": 1, "b": 2}

print(set([1, 2, 2, 3])) # {1, 2, 3}

# zip() : assemble plusieurs listes

noms = ["Alice", "Bob"]

ages = [25, 30]

print(list(zip(noms, ages))) # [('Alice', 25), ('Bob', 30)]

# enumerate() : retourne index et valeur

for i, val in enumerate(["a", "b"]):

print(i, val)

# map() : applique une fonction à une séquence

print(list(map(lambda x: x \* 2, [1, 2, 3]))) # [2, 4, 6]

# filter() : filtre selon une condition

print(list(filter(lambda x: x > 1, [0, 1, 2]))) # [2]

# sum() : somme des éléments

print(sum([1, 2, 3])) # 6

# sorted() : trie une liste

print(sorted([3, 1, 2])) # [1, 2, 3]

# any() / all() : testent une condition sur une séquence

print(any([False, True])) # True

print(all([True, True])) # True

**2. Fonctions NumPy**

import numpy as np

# array() : créer un tableau NumPy

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

# mean() / median() / std() / var() : statistiques

print(np.mean(arr)) # Moyenne

print(np.median(arr)) # Médiane

print(np.std(arr)) # Écart-type

print(np.var(arr)) # Variance

# min() / max() / sum() / prod()

print(np.min(arr)) # Valeur minimale

print(np.max(arr)) # Valeur maximale

print(np.sum(arr)) # Somme

print(np.prod(arr)) # Produit

# linspace() / arange() : créer des séquences

print(np.linspace(0, 1, 5)) # [0. 0.25 0.5 0.75 1. ]

print(np.arange(1, 5)) # [1 2 3 4]

# reshape(), flatten()

m = np.array([[1, 2], [3, 4]])

print(m.reshape(4)) # Change de forme

print(m.flatten()) # Aplati le tableau

# unique(), sort(), argsort()

print(np.unique([1, 2, 2, 3])) # Valeurs uniques

print(np.sort([3, 1, 2])) # Trié

print(np.argsort([3, 1, 2])) # Indices du tri

**3. Fonctions Pandas**

import pandas as pd

df = pd.DataFrame({"Nom": ["Alice", "Bob"], "Age": [25, 30]}) # Création d'un DataFrame

# head(), tail(), shape, info(), describe()

print(df.head()) # Premières lignes

print(df.tail()) # Dernières lignes

print(df.shape) # (2, 2)

df.info() # Infos structurelles

print(df.describe()) # Statistiques descriptives

# isnull(), dropna(), fillna()

df2 = pd.DataFrame({"A": [1, None, 3]})

print(df2.isnull()) # Détection des NaN

print(df2.fillna(0)) # Remplace les NaN

print(df2.dropna()) # Supprime les NaN

# loc[], iloc[]

print(df.loc[0]) # Accès par étiquette

print(df.iloc[0]) # Accès par position

# apply(), map(), replace()

df["AgeX2"] = df["Age"].apply(lambda x: x \* 2)

print(df)

df["Nom\_maj"] = df["Nom"].map(str.upper)

df["Nom"] = df["Nom"].replace("Alice", "Alicia")

print(df)

# groupby(), agg(), pivot\_table()

df3 = pd.DataFrame({"Dépt": ["RH", "RH", "IT"], "Salaire": [3000, 3500, 5000]})

print(df3.groupby("Dépt").agg({"Salaire": "mean"})) # Moyenne par groupe

# sort\_values(), value\_counts(), duplicated(), drop\_duplicates()

print(df.sort\_values("Age")) # Tri

print(df["Nom"].value\_counts()) # Fréquence

print(df.duplicated()) # Doublons

print(df.drop\_duplicates()) # Supprime les doublons

# merge(), join(), concat()

df4 = pd.DataFrame({"ID": [1, 2], "Ville": ["Paris", "Lyon"]})

df5 = pd.DataFrame({"ID": [1, 2], "Nom": ["Alice", "Bob"]})

print(pd.merge(df4, df5, on="ID")) # Fusion

print(pd.concat([df4, df5], axis=1)) # Concaténation

**4. Fonctions Matplotlib**

import matplotlib.pyplot as plt

x = [1, 2, 3]

y = [2, 4, 6]

plt.figure()

plt.plot(x, y, label="Ligne") # Courbe

plt.scatter(x, y, color="red") # Nuage de points

plt.bar(x, y) # Diagramme en barres

plt.hist(y) # Histogramme

plt.title("Titre")

plt.xlabel("X")

plt.ylabel("Y")

plt.legend()

plt.show()

**5. Fonctions Seaborn**

import seaborn as sns

sns.histplot(data=df, x="Age", bins=5)

sns.boxplot(data=df, x="Age")

sns.scatterplot(data=df, x="Age", y="AgeX2")

sns.heatmap(df.corr(), annot=True)

plt.show()

**6. Fonctions Scikit-learn (Machine Learning)**

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error, accuracy\_score, confusion\_matrix

X = df[["Age"]] # Variables d'entrée

y = [100, 150] # Cible

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.5)

model = LinearRegression()

model.fit(X\_train, y\_train) # Apprentissage

pred = model.predict(X\_test) # Prédiction

print("Prédiction:", pred)

print("MSE:", mean\_squared\_error(y\_test, pred))

**7. Fonctions de traitement de texte**

df['Nom\_min'] = df['Nom'].str.lower() # Minuscule

print(df['Nom'].str.contains("bob")) # Vérifie si "bob" est présent

print(df['Nom'].str.replace("b", "B")) # Remplace b par B

df.to\_csv("donnees.csv", index=False) # Sauvegarde CSV

df\_loaded = pd.read\_csv("donnees.csv") # Chargement

print(df\_loaded.head())

Ces fonctions couvrent l’essentiel de l’analyse de données moderne. Elles permettent :

• Prétraitement

• Visualisation

• Modélisation

• Évaluation

**EXERCICES CORRIGES SIMPLE**

**Exercice 1 :** Statistiques descriptives de base avec NumPy

import numpy as np # Importation de la bibliothèque NumPy

data = np.array([15, 20, 35, 40, 50]) # Création d'un tableau NumPy avec des entiers

mean = np.mean(data) # Calcule la moyenne

print("Moyenne:", mean) # Affiche la moyenne : 32.0

median = np.median(data) # Calcule la médiane (valeur centrale)

print("Médiane:", median) # Affiche : 35.0

std\_dev = np.std(data) # Calcule l'écart-type (dispersion)

print("Écart type:", std\_dev)

variance = np.var(data) # Calcule la variance

print("Variance:", variance) # Affiche la variance

# Ces mesures aident à comprendre la tendance centrale et la dispersion des données.

**Exercice 2 :** Manipulation de DataFrame avec Pandas

import pandas as pd # Importation de la bibliothèque Pandas

df = pd.DataFrame({ # Création d'un DataFrame avec 3 colonnes

'Nom': ['Alice', 'Bob', 'Charlie', 'David'],

'Âge': [25, 30, 35, 40],

'Salaire': [50000, 60000, 70000, None] # Valeur manquante pour David

})

print(df.head(2)) # Affiche les deux premières lignes du DataFrame

df['Salaire'] = df['Salaire'].fillna(df['Salaire'].mean()) # Remplace le NaN par la moyenne des salaires

print(df) # Affiche le DataFrame mis à jour

df['Âge2'] = df['Âge'].apply(lambda x: x \* 2) # Nouvelle colonne avec l'âge multiplié par 2

salaires\_filtres = df[df['Salaire'] > 55000] # Filtre les lignes où le salaire > 55000

print(salaires\_filtres) # Affiche le résultat du filtre

**Exercice 3 :** Visualisation avec Matplotlib et Seaborn

import matplotlib.pyplot as plt # Importation de Matplotlib

import seaborn as sns # Importation de Seaborn

plt.figure(figsize=(5, 3)) # Définit la taille de la figure

sns.histplot(df['Âge'], bins=4, kde=True) # Crée un histogramme avec courbe de densité

plt.title("Distribution des âges") # Titre du graphique

plt.show() # Affiche le graphique

plt.figure(figsize=(5, 3)) # Nouvelle figure

sns.boxplot(x=df['Salaire']) # Boxplot des salaires

plt.title("Répartition des salaires") # Titre

plt.show() # Affiche

**Exercice 4 :** Régression linéaire simple avec Scikit-learn

from sklearn.linear\_model import LinearRegression # Modèle de régression

from sklearn.model\_selection import train\_test\_split # Pour diviser les données

from sklearn.metrics import mean\_squared\_error # Pour évaluer le modèle

X = df[['Âge']] # Variable explicative

y = df['Salaire'] # Variable à prédire

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.25) # Division des données

model = LinearRegression() # Création du modèle

model.fit(X\_train, y\_train) # Entraînement du modèle

y\_pred = model.predict(X\_test) # Prédiction sur le jeu de test

mse = mean\_squared\_error(y\_test, y\_pred) # Calcul de l'erreur quadratique moyenne

print("MSE:", mse) # Affiche l'erreur

plt.figure(figsize=(5, 3)) # Nouvelle figure

plt.scatter(X, y, color='blue') # Nuage de points

plt.plot(X, model.predict(X), color='red') # Droite de régression

plt.title("Régression Salaire vs Âge") # Titre

plt.xlabel("Âge") # Axe X

plt.ylabel("Salaire") # Axe Y

plt.show() # Affiche

**Exercice 5 :** Groupement et agrégation

df\_group = pd.DataFrame({ # Nouveau DataFrame

'Département': ['RH', 'IT', 'IT', 'RH', 'Finance'],

'Salaire': [45000, 70000, 80000, 47000, 65000]

})

agg = df\_group.groupby('Département').agg({'Salaire': 'mean'}) # Moyenne des salaires par département

print(agg) # Affiche le résultat

**Exercice 6 :** Sauvegarde et chargement

df.to\_csv('donnees.csv', index=False) # Sauvegarde du DataFrame dans un fichier CSV

df\_chargé = pd.read\_csv('donnees.csv') # Chargement du fichier

print(df\_chargé.head()) # Affichage des premières lignes du fichier chargé

**Exercice 7 :** Chaînes de caractères avec Pandas

df['Nom\_min'] = df['Nom'].str.lower() # Conversion des noms en minuscules

print(df[['Nom', 'Nom\_min']]) # Affiche les noms originaux et modifiés

**STATISTIQUE DESCRIPTIVE AVEC INTERPRÉTATION**

**Application 01**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

Objectif : analyser un jeu de données sur les revenus mensuels

revenus = [1800, 2200, 2100, 2500, 2700, 1900, 2300, 2000, 2400, 2800, 3100, 1800, 2000, 1950, 2750]

# Convertir en tableau numpy

revenus\_array = np.array(revenus)

**1. Mesures de tendance centrale**

mean = np.mean(revenus\_array)

median = np.median(revenus\_array)

mode = pd.Series(revenus\_array).mode()[0] # mode() retourne une série

print("Moyenne:", mean) # Représente le revenu moyen

print("Médiane:", median) # Revenu central (sépare les valeurs en 2 moitiés)

print("Mode:", mode) # Revenu le plus fréquent

***Interprétation :***

* La moyenne permet de connaître la valeur typique.
* La médiane est robuste aux valeurs extrêmes.
* Le mode montre le revenu qui revient le plus souvent (utile s'il y a une concentration).

**2. Mesures de dispersion**

std\_dev = np.std(revenus\_array)

variance = np.var(revenus\_array)

min\_val = np.min(revenus\_array)

max\_val = np.max(revenus\_array)

range\_val = max\_val - min\_val

print("Écart-type:", std\_dev)

print("Variance:", variance)

print("Min:", min\_val)

print("Max:", max\_val)

print("Étendue:", range\_val)

***Interprétation :***

* Plus l'écart-type est grand, plus les revenus sont dispersés.
* L'étendue permet de visualiser l'amplitude globale.

**3. Quartiles et résumé statistique**

q1 = np.percentile(revenus\_array, 25)

q3 = np.percentile(revenus\_array, 75)

iqr = q3 - q1

print("1er quartile (Q1):", q1)

print("3e quartile (Q3):", q3)

print("IQR:", iqr)

***Interprétation :***

* Q1 et Q3 permettent de connaître les bornes de la majorité des données.
* IQR (Interquartile Range) montre la dispersion centrale sans être affectée par les extrêmes.

**4. Visualisations graphiques**

plt.figure(figsize=(12, 5))

# Histogramme

plt.subplot(1, 2, 1)

sns.histplot(revenus\_array, kde=True, bins=8, color='skyblue')

plt.title("Histogramme des revenus")

plt.xlabel("Revenu mensuel")

plt.ylabel("Effectif")

# Boîte à moustaches (boxplot)

plt.subplot(1, 2, 2)

sns.boxplot(x=revenus\_array, color='orange')

plt.title("Boxplot des revenus")

plt.tight\_layout()

plt.show()

***Interprétation :***

* L'histogramme permet de visualiser la distribution : symétrique, biaisée, multimodale...
* Le boxplot permet de repérer les valeurs aberrantes (outliers) et la médiane.

**Conclusion**

print("\nConclusion analytique:")

print(f"La moyenne des revenus est de {mean:.2f} €, avec une dispersion modérée (écart-type {std\_dev:.2f}).")

print(f"La médiane ({median:.2f} €) est proche de la moyenne, indiquant une distribution relativement symétrique.")

print("Les visualisations confirment une distribution centrée sans valeurs extrêmes fortes.")

**Application 02**

# Import des bibliothèques nécessaires pour l'analyse

import numpy as np # Bibliothèque pour le calcul scientifique (tableaux, statistiques)

import pandas as pd # Bibliothèque pour la manipulation des données tabulaires

import matplotlib.pyplot as plt # Pour les graphiques

import seaborn as sns # Pour les graphiques avancés

**Objectif :** Analyser un dataset de revenus mensuels plus conséquent

np.random.seed(0) # Fixe le générateur aléatoire pour obtenir les mêmes résultats à chaque exécution

dataset\_size = 200 # On choisit de générer 200 individus dans l'échantillon

# Génère des revenus normalement distribués autour de 2500€ avec un écart-type de 500

revenus = np.random.normal(loc=2500, scale=500, size=dataset\_size).astype(int)

# Ajoute deux valeurs extrêmes pour simuler des outliers

revenus = np.append(revenus, [10000, 12000])

1. Calculs de tendance centrale

mean = np.mean(revenus) # Moyenne arithmétique des revenus

median = np.median(revenus) # Médiane : valeur centrale triée

mode = pd.Series(revenus).mode()[0] # Mode : valeur la plus fréquente

# Affichage des mesures de tendance centrale

print("Moyenne:", mean)

print("Médiane:", median)

print("Mode:", mode)

***Interprétation :***

* La moyenne est sensible aux valeurs extrêmes (ex. : 10000€)
* La médiane est plus robuste : coupe l'échantillon en deux
* Le mode indique les revenus les plus fréquents (utile pour des pics de densité)

2. Mesures de dispersion

std\_dev = np.std(revenus) # Écart-type : mesure de la dispersion autour de la moyenne

variance = np.var(revenus) # Variance : carré de l’écart-type

min\_val = np.min(revenus) # Valeur minimale

max\_val = np.max(revenus) # Valeur maximale

range\_val = max\_val - min\_val # Étendue : différence entre max et min

# Affichage des mesures de dispersion

print("Écart-type:", std\_dev)

print("Variance:", variance)

print("Min:", min\_val)

print("Max:", max\_val)

print("Étendue:", range\_val)

***Interprétation :***

* L'écart-type est élevé à cause des outliers
* L'étendue est large : 12000 - 1064 = grande dispersion

3. Quartiles et IQR

q1 = np.percentile(revenus, 25) # Premier quartile : 25% des valeurs en dessous

q3 = np.percentile(revenus, 75) # Troisième quartile : 75% des valeurs en dessous

iqr = q3 - q1 # IQR : intervalle interquartile (Q3 - Q1)

# Affichage des quartiles

print("1er quartile (Q1):", q1)

print("3e quartile (Q3):", q3)

print("IQR:", iqr)

***Interprétation :***

* 50% des revenus sont compris entre Q1 et Q3
* IQR utile pour détecter les outliers (hors de Q1 - 1.5\*IQR à Q3 + 1.5\*IQR)

4. Visualisation graphique

plt.figure(figsize=(14, 6)) # Définir la taille de la figure

# Histogramme des revenus avec courbe de densité

plt.subplot(1, 2, 1)

sns.histplot(revenus, kde=True, bins=20, color='skyblue')

plt.title("Histogramme des revenus")

plt.xlabel("Revenus mensuels")

# Boxplot pour détecter les valeurs extrêmes

plt.subplot(1, 2, 2)

sns.boxplot(x=revenus, color='orange')

plt.title("Boxplot des revenus")

# Affiche les deux graphiques

plt.tight\_layout()

plt.show()

***Interprétation :***

* Histogramme : distribution centrée autour de 2500€, mais avec queue droite (outliers)
* Boxplot : les deux extrêmes apparaissent comme des points isolés (outliers)

**Conclusion**

print("\nConclusion analytique:")

print(f"Moyenne : {mean:.2f} €")

print(f"Médiane : {median:.2f} €")

print(f"Écart-type : {std\_dev:.2f} €")

print("Les revenus suivent une distribution normale avec quelques valeurs extrêmes qui augmentent la moyenne.")

print("La médiane est plus représentative dans ce cas.")

**Application 03**

Statistique descriptive avec un dataset CSV

# Importation des bibliothèques

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

Objectif : Analyser un dataset réel provenant d’un fichier CSV

Supposons que le fichier s'appelle "revenus\_population.csv" et contient une colonne 'revenu'

1. Aller sur Google et rechercher un dataset

2. Télécharger le fichier sur votre ordinateur

3. Renommer le fichier en "revenus\_population.csv"

4. Placer ce fichier dans le même dossier que ce script Python

**Le fichier doit être dans le même dossier que ce script ou indiquer un chemin absolu**

try:

df = pd.read\_csv("revenus\_population.csv")

print(" Fichier chargé avec succès!")

except FileNotFoundError:

print(" Le fichier 'revenus\_population.csv' est introuvable. Vérifiez le chemin ou le nom du fichier.")

raise

# Afficher les premières lignes

print("\nAperçu du jeu de données:")

print(df.head())

**Statistiques descriptives**

On suppose que la colonne s'appelle 'revenu'

revenus = df['revenu']

mean = revenus.mean() # Moyenne

median = revenus.median() # Médiane

mode = revenus.mode()[0] # Mode

std\_dev = revenus.std() # Écart-type

variance = revenus.var() # Variance

min\_val = revenus.min() # Valeur minimale

max\_val = revenus.max() # Valeur maximale

range\_val = max\_val - min\_val # Étendue

q1 = revenus.quantile(0.25) # 1er quartile

q3 = revenus.quantile(0.75) # 3e quartile

iqr = q3 - q1 # IQR

# Affichage

print("\n--- Statistiques descriptives ---")

print(f"Moyenne : {mean:.2f} €")

print(f"Médiane : {median:.2f} €")

print(f"Mode : {mode:.2f} €")

print(f"Écart-type : {std\_dev:.2f} €")

print(f"Variance : {variance:.2f} €")

print(f"Minimum : {min\_val:.2f} €")

print(f"Maximum : {max\_val:.2f} €")

print(f"Étendue : {range\_val:.2f} €")

print(f"Q1 : {q1:.2f} €")

print(f"Q3 : {q3:.2f} €")

print(f"IQR : {iqr:.2f} €")

**Visualisation graphique**

plt.figure(figsize=(14, 6))

# Histogramme avec courbe KDE (Kernel Density Estimate) : est une estimation lissée de la distribution des données.

plt.subplot(1, 2, 1)

sns.histplot(revenus, kde=True, bins=20, color='skyblue')

plt.title("Histogramme des revenus")

plt.xlabel("Revenus mensuels")

# Boîte à moustaches (boxplot)

plt.subplot(1, 2, 2)

sns.boxplot(x=revenus, color='orange')

plt.title("Boxplot des revenus")

plt.tight\_layout()

plt.show()

print("\nConclusion:")

print("Les statistiques décrivent la tendance centrale, la dispersion et les valeurs extrêmes.")

print("Les visualisations permettent de repérer d’éventuels outliers et de comprendre la distribution globale des revenus.")