**For running the code:**

<https://colab.research.google.com/drive/1nb5F3IBGPzHpOvYzpNV0ESrnL0tzVc3m#scrollTo=97IclqRYHLzx>

* **import** → tells Python: “Bring this library into my code so I can use it.”
* **numpy** → the name of the library (it handles math with arrays and matrices).
* **as np** → gives it a short nickname (**alias**) so instead of typing numpy.array() every time, you can just write np.array().
* **pandas** → the library for working with tables & data.
* **as pd** → short nickname so we can write pd.DataFrame() instead of pandas.DataFrame().

So:

* **NumPy** → works with **numbers & arrays**
* **Pandas** → works with **tables & dataframes**
* np and pd → just shortcuts so we type less when coding.

1. name = "Nouamat"

age = 20

print("Hello", name, "You are", age, "years old")

*output*: Hello Nouamat You are 20 years old

1. import numpy as np

# Create an array

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

# Multiply each element by 2

print(numbers \* 2)

# Reshape into 2 rows and 3 columns

arr2 = np.array([1, 2, 3, 4, 5, 6])

print(arr2.reshape(2, 3))

*output*: [ 2 4 6 8 10]

[[1 2 3]

[4 5 6]]

1. import pandas as pd

# Create a small table

data = {"Name": ["Ali", "Sara", "Omar"], "Age": [23, 21, 25]}

df = pd.DataFrame(data)

print(df) # Show table

print(df["Age"]) # Show only Age column

print(df["Age"].mean()) # Average Age

*output:* Name Age

0 Ali 23

1 Sara 21

2 Omar 25

0 23

1 21

2 25

Name: Age, dtype: int64

23.0

* DataFrame() = create table
* df["column"] = select a column
* .mean() = calculate average

dtype: int64 → means the data type is integer (64-bit).

1. url = "https://raw.githubusercontent.com/mwaskom/seaborn-data/master/tips.csv"

data = pd.read\_csv(url)

print(data.head())

print(data.describe())

*output:*

|  |
| --- |
| total\_bill tip sex smoker day time size |
| 0 16.99 1.01 Female No Sun Dinner 2 |
| 1 10.34 1.66 Male No Sun Dinner 3 |
| 2 21.01 3.50 Male No Sun Dinner 3 |
| 3 23.68 3.31 Male No Sun Dinner 2 |
| 4 24.59 3.61 Female No Sun Dinner 4 |
| total\_bill tip size |
| count 244.000000 244.000000 244.000000 |
| mean 19.785943 2.998279 2.569672 |
| std 8.902412 1.383638 0.951100 |
| min 3.070000 1.000000 1.000000 |
| 25% 13.347500 2.000000 2.000000 |
| 50% 17.795000 2.900000 2.000000 |
| 75% 24.127500 3.562500 3.000000 |
| max 50.810000 10.000000 6.000000 |

 Here, url is just a **string variable** that stores the link to a **CSV file**.

 **CSV** (Comma-Separated Values) is a simple file format for storing table data, like an Excel file but in plain text.

**data = pd.read\_csv(url)**

* pd.read\_csv() → a Pandas function that **reads CSV files** into a **DataFrame**.
* Since our CSV file is online, we pass the url instead of a local file path.
* Now, data contains the table from the CSV file.
* **count** → number of non-empty values
* **mean** → average value
* **std** → standard deviation (spread of the data)
* **min** → smallest value
* **25%**, **50%**, **75%** → percentiles (quarter, median, three-quarters)
* **max** → largest value

1. from sklearn.datasets import load\_iris

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

from sklearn.neighbors import KNeighborsClassifier

# Load data

iris = load\_iris()

X = iris.data

y = iris.target

# Split into training and test sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2)

# Create and train the model

model = KNeighborsClassifier(n\_neighbors=3)

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

# Check accuracy

print("Model Accuracy:", model.score(X\_test, y\_test))

*output:* Model Accuracy: 0.9666666666666667

 load\_iris → loads the **Iris flower dataset** (small, famous dataset used in ML).

 train\_test\_split → splits your dataset into a **training set** and a **testing set**.

 KNeighborsClassifier → a machine learning model that classifies data based on the **k-nearest neighbors algorithm**.

1. predictions = model.predict(X\_test)

print(predictions)

*output:* *[1 0 2 1 1 0 2 2 1 0 0 1 2 0 0 2 1 2 2 0 1 1 0 1 2 1 0 2 1 0]*