What is Python?

Python is a **high-level**, **interpreted**, and **general-purpose programming language** known for its **simple and readable syntax**. It was created by **Guido van Rossum** and first released in **1991**.

Python is very popular among beginners and professionals because it is:

- Easy to learn and write
- Versatile (used in many fields)
- Open-source and has a large community
- Rich in libraries and frameworks

Uses of Python

Python is used in a wide range of areas:

1. Artificial Intelligence & Machine Learning

- Libraries: `TensorFlow`, `PyTorch`, `Scikit-learn`
- Used to build models that can learn and make predictions

2. Web Development

- Frameworks: `Django`, `Flask`, `FastAPI`
- Used to create dynamic websites and REST APIs

3. Data Science & Data Analysis

- Libraries: `Pandas`, `NumPy`, `Matplotlib`, `Seaborn`
- Used to analyze, visualize, and manipulate data

4. Automation & Scripting

- Automate boring or repetitive tasks
- Example: Web scraping, file management, email sending

5. Desktop Applications

- Libraries: `Tkinter`, `PyQt`
- Used to create GUI-based applications

6. Game Development

- Libraries: `Pygame`
- Used to build simple 2D games

What is NumPy?

NumPy (short for Numerical Python) is a powerful Python library used for numerical and scientific computing. It provides support for:

- Multidimensional arrays (ndarrays)
- Mathematical functions
- Linear algebra
- Random number generation

NumPy is fast and efficient, and it's the foundation of many other libraries like Pandas, TensorFlow, and Scikit-learn.

Key Features of NumPy

- Powerful **N-dimensional arrays**
- Very **fast** operations (written in C under the hood)
- Supports **vectorized operations** (no need for loops)
- Tools for working with **matrices, Fourier transforms, and statistics**

What is Pandas?

Pandas is a powerful **open-source Python library** used for **data analysis and data manipulation**. It is built on top of **NumPy** and makes working with **structured data** (like tables or spreadsheets) easy and efficient.

Core Data Structures in Pandas

1. **Series** – 1D labeled array (like a single column)
2. **DataFrame** – 2D table with rows and columns (like an Excel sheet)

Key Features of Pandas

Feature	Description
DataFrame Support	Handles labeled 2D data like SQL tables or Excel files
Data Cleaning	Handle missing data, filter, replace, or fill values
Data Aggregation	Grouping, summarizing, counting, mean, etc.
Data Analysis	Perform statistical operations and data exploration
File Handling	Easily read/write from CSV, Excel, JSON, SQL, etc.
Feature	**Description**
Time Series Handling	Built-in support for date/time functions and indexing

Data Visualization	Integrates well with libraries like Matplotlib and Seaborn for plotting
Fast Operations	Built on top of NumPy, optimized for performance

---

What is Matplotlib?

Matplotlib is a Python library used for **data visualization**. It lets you create **line charts, bar charts, scatter plots**, and many other types of graphs.

Key Features of Matplotlib

Feature	Description
Plotting Variety	Line, bar, scatter, pie, histogram, etc.
Customization	Control over colors, labels, legends, etc. Subplots
Support	Create multiple plots in one figure
Exporting	Save plots as PNG, PDF, SVG, etc. Annotating Graphs
	Add labels, arrows, and custom styling

What is Seaborn?

Seaborn is a Python visualization library built **on top of Matplotlib**. It provides a **high-level interface** for making beautiful and informative statistical graphics.

Key Features of Seaborn

Feature	Description
Easy Plotting	One-liner plots like histograms and box plots
Statistical Graphs	Built-in support for correlation and regression
Beautiful Themes	Pre-set styles and color palettes Works with
Pandas	Supports DataFrames directly Complex Plots Made
Easy	Heatmaps, pairplots, violin plots, etc.

What is SciPy?

SciPy (Scientific Python) is a Python library used for **scientific and technical computing**. It builds on NumPy and adds advanced features.

Key Features of SciPy

Feature	Description

Advanced Math	Integration, differentiation, optimization Statistics
	Probability distributions, statistical tests Linear Algebra Matrix operations, eigenvalues, etc.
Signal & Image Processing	Tools for filtering, image analysis, etc.
Performance	Fast calculations built in C/Fortran

What is Scikit-learn (sklearn)?

Scikit-learn is a powerful **machine learning library** in Python. It provides simple tools for **classification, regression, clustering**, and more.

Key Features of Scikit-learn

Feature	Description
Machine Learning	Classification, regression, clustering
Model Tools	Training, testing, model evaluation
Preprocessing	Feature scaling, encoding, normalization
Model Selection	Cross-validation, hyperparameter tuning
Pipeline Support	Combine multiple steps into one workflow

1.Write a Python program to calculate the Mean, Median, and Standard Deviation of a dataset using a NumPy array.

[1]:
```
import numpy as np
```

[14]:
```
marks = [23,23,12,45,50,22,21,45,34,36]
print(type(marks))
print(marks)
marks = np.array(marks)
print(marks)
print(type(marks))
```

```
<class 'list'>
[23, 23, 12, 45, 50, 22, 21, 45, 34, 36]
[23 23 12 45 50 22 21 45 34 36]
<class 'numpy.ndarray'>
```

[8]:
```
avg_marks = np.mean(marks)
middle_marks = np.median(marks)
stdv_marks = np.std(marks)
```

[12]:
```
print(f"Marks:\nAverage: {avg_marks} \nMiddle_marks: {middle_marks} \nDeviation_marks: {stdv_marks}")
```

```
Marks:
Average: 31.1
Middle_marks: 28.5
Deviation_marks: 12.070211265756702
```

[ ]:

2)Write a Python program to read a CSV file, clean the data by handling missing values, and perform basic data analysis using the Pandas library.

[36]:
```
import numpy as np
import pandas as pd
```

[5]:
```
medical_data = pd.read_csv('medical.csv')
```

[7]:
```
print("Total null values column vise:\n\n",medical_data.isnull().sum())
```

```
Total null values column vise:

 Patient_ID        0
Name               1
Age                2
Gender             1
Diagnosis          1
Blood_Pressure     1
Heart_Rate         1
Cholesterol        1
dtype: int64
```

Cleaning of data

Method1: Droping all the null value

[8]:
```
med_data_final = medical_data.dropna()
```

[9]:
```
med_data_final
```

[9]:

	Patient_ID	Name	Age	Gender	Diagnosis	Blood_Pressure	Heart_Rate	Cholesterol

Method1: Droping all the null value

[8]:
```
med_data_final = medical_data.dropna()
```

[9]:
```
med_data_final
```

[9]:

	Patient_ID	Name	Age	Gender	Diagnosis	Blood_Pressure	Heart_Rate	Cholesterol
0	101	Alice	29.0	F	Diabetes	120/80	78.0	190.0
1	102	Bob	45.0	M	Hypertension	140/90	85.0	220.0

Method2: Replacing with the Central tendencies

[10]:
```
medical_data
```

[10]:

	Patient_ID	Name	Age	Gender	Diagnosis	Blood_Pressure	Heart_Rate	Cholesterol
0	101	Alice	29.0	F	Diabetes	120/80	78.0	190.0
1	102	Bob	45.0	M	Hypertension	140/90	85.0	220.0
2	103	Charlie	34.0	M	Asthma	110/70	90.0	NaN
3	104	David	NaN	M	Diabetes	135/85	88.0	240.0
4	105	Eva	41.0	NaN	Heart Disease	125/80	NaN	210.0
5	106	Frank	50.0	M	NaN	145/95	80.0	200.0
6	107	NaN	38.0	F	Hypertension	NaN	76.0	230.0

[35]:

	Patient_ID	Name	Age	Gender	Diagnosis	Blood_Pressure	Heart_Rate	Cholesterol
0	101	Alice	29.0	F	Diabetes	120/80	78.0	190.0
1	102	Bob	45.0	M	Hypertension	140/90	85.0	220.0
2	103	Charlie	34.0	M	Asthma	110/70	90.0	205.0
3	104	David	39.8	M	Diabetes	135/85	88.0	240.0
4	105	Eva	41.0	M	Heart Disease	125/80	82.5	210.0
5	106	Frank	50.0	M	Diabetes	145/95	80.0	200.0
7	108	Hannah	39.8	F	Diabetes	130/85	72.0	180.0

[70]:
```
#funciton to check is data is cleaned or not
def Check_clean(data):
    d = dict(data.isnull().sum())
    null_values = []
    for key,value in d.items():
        if value != 0:
            null_values.append((key,value))
    return 'Cleaned' if data.isnull().sum().sum() == 0 else null_value
```

[71]:
```
Check_clean(data)
```

[71]:
```
'Cleaned'
```

3.) Create a Python program to plot a scatter plot, histogram, and boxplot using Matplotlib and Seaborn.

[5]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

[6]:
```
data = pd.read_csv("medical.csv")
```

[8]:
```
df = pd.DataFrame(data)
```

[11]:
```
def plotshow(data):
    # Set the plotting style
    sns.set(style="whitegrid")
    plt.figure(figsize=(16, 6))

    # Scatter Plot: Age vs Cholesterol
    plt.subplot(1, 3, 1)
    sns.scatterplot(data=df, x='Age', y='Cholesterol', hue='Gender')
    plt.title("Age vs Cholesterol")

    # Histogram: Heart Rate distribution
    plt.subplot(1, 3, 2)
    sns.histplot(df['Heart_Rate'].dropna(), kde=True, bins=5, color='orange')
    plt.title("Heart Rate Distribution")

    # Boxplot: Cholesterol by Gender
    plt.subplot(1, 3, 3)
    sns.boxplot(data=df, x='Gender', y='Cholesterol', palette='pastel')
    plt.title("Cholesterol Levels by Gender")

    plt.tight_layout()
    plt.show()
```

[12]:
```
plotshow(data)
```

```
C:\Users\Keshav Barawal\AppData\Local\Temp\ipykernel_2124\3806559445.py:18: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(data=df, x='Gender', y='Cholesterol', palette='pastel')
```