Python Programming

Lab:- 28(Scipy Introduction)

Student Id:- AF0417098

Student Name:- Ankush

Introduction to SciPy

SciPy is a powerful Python library used for scientific and technical computing. It builds on top of NumPy and provides a wide range of functions and tools for mathematical computations, optimization, integration, interpolation, eigenvalue problems, and signal processing, among many other tasks.

SciPy is essential for data scientists, engineers, and researchers who require advanced computational methods. It is widely used for solving problems in scientific computing, mathematics, engineering, and machine learning.

1. SciPy vs. NumPy

SciPy builds upon NumPy, which is the fundamental package for array operations in Python. NumPy handles basic array operations like matrix and vector multiplication, while SciPy provides higher-level functions to perform operations such as optimization, signal processing, and solving differential equations.

- NumPy is mainly for numerical operations with arrays and matrices.
- SciPy extends NumPy and provides a collection of mathematical algorithms and convenience functions that make scientific computing easier.

2. SciPy Ecosystem

SciPy is part of the broader SciPy ecosystem which includes libraries like:

- NumPy: For numerical operations and array manipulation.
- Matplotlib: For data visualization.
- Pandas: For data manipulation and analysis.
- SymPy: For symbolic computation.
- scikit-learn: For machine learning.

3. Installation of SciPy

To install SciPy, you can use Python's package manager, pip:

```
pip install scipy
```

Once installed, you can import the library into your Python script:

```
import scipy
```

4. SciPy Sub-packages

SciPy is organized into sub-packages, each focusing on a specific scientific computing domain. Here are some of the key sub-packages:

- scipy.integrate: For integration (numerical integration, solving ordinary differential equations).
- scipy.optimize: For optimization and root finding (finding minimums and maximums).
- scipy.linalg: For linear algebra operations (matrix decompositions, solving systems of linear equations).
- scipy.fftpack: For Fourier transforms (used in signal processing and other areas).
- scipy.signal: For signal processing (filtering, convolution, etc.).

- scipy.spatial: For working with spatial data structures and algorithms (KD-trees, distance computations).
- scipy.stats: For statistical functions (probability distributions, hypothesis testing)

5.Real-World Applications of SciPy

- SciPy is versatile and widely used in several fields:
- Engineering: Used for solving differential equations, system modeling, signal processing, etc.
- Physics: For simulations, modeling dynamic systems, and solving physical equations.
- Machine Learning: SciPy is often used in feature extraction and data processing stages of machine learning pipelines.
- Finance: Optimization algorithms help solve financial modeling problems, portfolio optimization, and risk management.

6. Why Use SciPy?

- Ease of Use: SciPy provides high-level functions for complex mathematical operations, making it accessible even for non-experts in numerical computing.
- Performance: SciPy is optimized for performance, leveraging compiled libraries such as BLAS, LAPACK, and Fortran.
- Comprehensive: It has an extensive range of functions that cover most aspects of scientific computing.
-

Assignment Questions:-

Ques1:- task 1: To find Area of the circle

https://raw.githubusercontent.com/AnudipAE/DANLC/master/radius_data.csv

Program:-

```python
import pandas as pd
import math

# Load the radius data and calculate area in one step
radius_data = pd.read_csv("https://raw.githubusercontent.com/AnudipAE/DANLC/master/radius_data.csv")
radius_data['Area'] = math.pi * radius_data['Radius'] ** 2

# Display the updated DataFrame
print(radius_data)
```

Output:-

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  TEST RESULTS  PORTS

/Users/Raj Kumar/Desktop/python programming/lab28.py"
   CircleName    Radius        Area
0         SAY  3.798717   45.333960
1         PSN  9.958397  311.550720
2         JDP  5.142711   83.087197
3         AUO  3.319584   34.619210
4         OHG  1.138395    4.071325
..        ...       ...         ...
95        PVZ  7.798122  191.042457
96        SQR  5.133239   82.781400
97        NSM  9.761868  299.375156
98        SXE  6.774164  144.165471
99        JNT  2.823492   25.045121

[100 rows x 3 columns]
PS C:\Users\Raj Kumar\Desktop\python programming>
```

Ques 2. task 2: To Convert Celsius into Fahrenheit

https://raw.githubusercontent.com/d4dipdas/DANLC/main/city_temperatures.csv

Program:-

```python
import pandas as pd

# Load the temperature data
temperature_url = "https://raw.githubusercontent.com/d4dipdas/DANLC/main/city_temperatures.csv"
temperature_data = pd.read_csv(temperature_url)

# Check the column names
print(temperature_data.columns)  # To confirm the column names

# Rename the column to a simpler name for easier handling (optional)
temperature_data.rename(columns={'Temperature (°C)': 'Celsius'}, inplace=True)

# Convert Celsius to Fahrenheit
temperature_data['Fahrenheit'] = (temperature_data['Celsius'] * 9/5) + 32

# Display the updated DataFrame
print(temperature_data)
```

Output:-

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  TEST RESULTS  PORTS

PS C:\Users\Raj Kumar\Desktop\python programming> & "C:/Users/Raj Kumar/AppData/Local/Programs/Pyth
/Users/Raj Kumar/Desktop/python programming/lab28.py"
Index(['City', 'Temperature (°C)'], dtype='object')
            City  Celsius  Fahrenheit
0    Los Angeles     35.8       96.44
1        Chicago      8.1       46.58
2  San Francisco      0.2       32.36
Index(['City', 'Temperature (°C)'], dtype='object')
            City  Celsius  Fahrenheit
0    Los Angeles     35.8       96.44
1        Chicago      8.1       46.58
2  San Francisco      0.2       32.36
1        Chicago      8.1       46.58
2  San Francisco      0.2       32.36
3        Chicago     36.2       97.16
4         Boston     37.9      100.22
..           ...      ...         ...
95       Seattle     39.7      103.46
96        Boston      1.5       34.70
```