

27-10-2024

Training Day – 25

***Topic:* Introduction to SciPy**

SciPy is an open-source Python library used for scientific and technical computing. It builds on NumPy, providing a wide range of functionalities for mathematics, science, and engineering. SciPy includes modules for optimization, integration, interpolation, eigenvalue problems, signal processing, linear algebra, and more. It is designed to work efficiently with NumPy arrays, allowing users to perform complex computations with minimal code.

- Explored optimization and integration functions in SciPy.
- Example: Used `scipy.integrate.quad` for numerical integration.
- SciPy builds on NumPy and provides additional modules for optimization, integration, and statistics.

Key Features of SciPy

1. **Linear Algebra:** Tools for solving linear systems, eigenvalues, and singular value decompositions.
2. **Optimization:** Algorithms for optimization, including curve fitting and minimization.
3. **Integration:** Functions for numerical integration.
4. **Interpolation:** Methods for data interpolation.
5. **Statistics:** A wide range of statistical functions and random distributions.
6. **Signal Processing:** Tools for filtering, spectral analysis, and more.

EXAMPLE

```
from scipy.integrate import quad
def func(x):
    return x**2
result, error = quad(func, 0, 1)
print("Integral of x^2 from 0 to 1:", result)
```

28-10-2024

Training Day – 26

***Topic:* Introduction to Pandas**

- Learned about Pandas DataFrame and Series.
- Example: Created a DataFrame from a dictionary and accessed its rows and columns.

Pandas provides Series and DataFrame structures for efficient data manipulation and analysis.

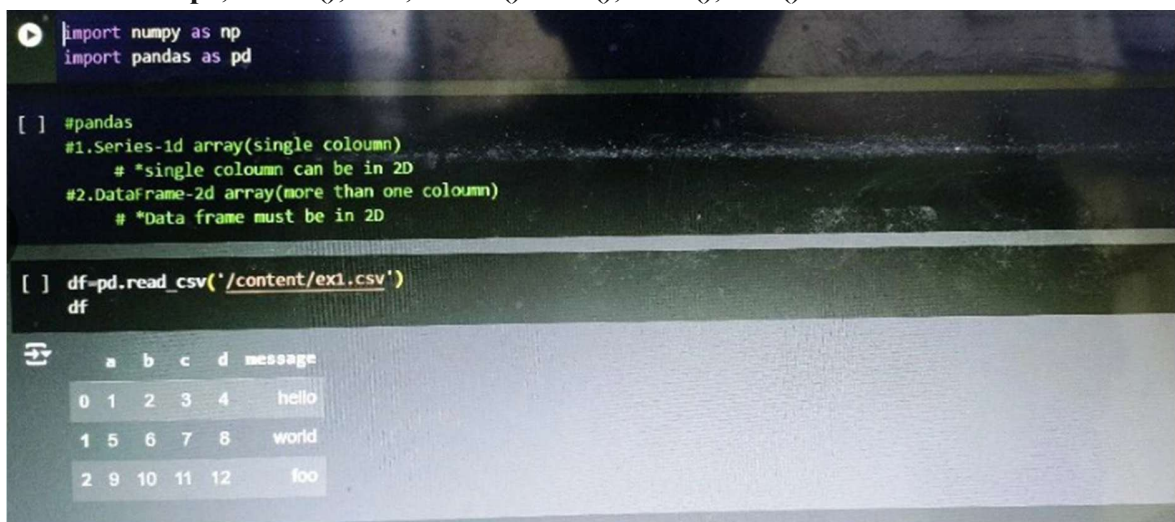
In today's session we worked little bit more on Broadcasting on numpy library, by solving a mathematical operation on matrices.

- Also learn about "fromfunction" through the help of numpy.
- Topic Array creation also covered in today's session, and completed with solving one example of 'Array Creation' using static method.

EXAMPLE

```
import pandas as pd
data = {"Name": ["Alice", "Bob"], "Age": [25, 30]}
df = pd.DataFrame(data)
print(df)
```

- After importing I used some functions of pandas to read and analyse the data set, like - . shape, . isnull(), . info(), . sum(), . head(), . tail()



```
import numpy as np
import pandas as pd

[ ] #pandas
    #1.Series-1d array(single column)
    # *single column can be in 2D
    #2.DataFrame-2d array(more than one column)
    # *Data frame must be in 2D

[ ] df=pd.read_csv('/content/ex1.csv')
df
```

	a	b	c	d	message
0	1	2	3	4	hello
1	5	6	7	8	world
2	9	10	11	12	foo

```
[ ] df5.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 3 entries, 0 to 2  
Data columns (total 6 columns):  
#   Column      Non-Null Count  Dtype  
---  ---  
0   something    3 non-null      object  
1   a            3 non-null      int64  
2   b            3 non-null      int64  
3   c            2 non-null      float64  
4   d            3 non-null      int64  
5   message      2 non-null      object  
dtypes: float64(1), int64(3), object(2)  
memory usage: 272.0+ bytes
```

```
[ ] df5.isnull().sum()
```

```
0  
something 0  
a          0  
b          0  
c          1  
d          0  
message    1
```

29-10-2024

Training Day – 27

October 25, Friday*

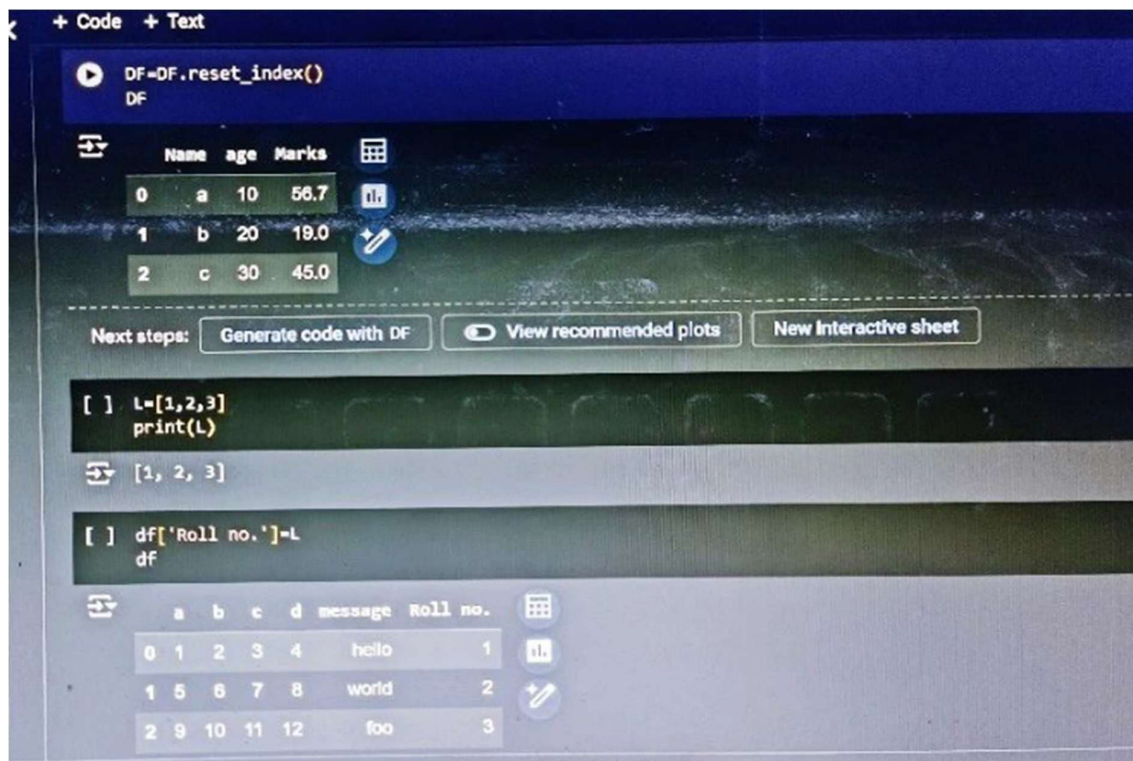
- *Topic:* Descriptive Analysis with Pandas

- Summarized data using .describe(), .mean(), and .sum().
- Example: Analyzed a dataset's central tendencies and spread.

Today I made my own data set and convert it into DataFrame and apply some functions of **pandas library**. Like - .info(), .set_index(), .reset_index(), adding an column name 'Roll no.', .drop().

•Also done 'Indexing' and 'Slicing' on large dataset for reading the data according to given rows and column detail, using '.iloc' function.

•Also practice for feeling NAN values by identifying outliers, with the help of mean, median, mode.



The screenshot shows a Jupyter Notebook interface with a dark theme. At the top, there are tabs for '+ Code' and '+ Text'. The main area displays a code cell with the following content:

```
DF=DF.reset_index()
DF
```

Below the code cell, the output is shown as a DataFrame with columns 'Name', 'age', and 'Marks'. The data is as follows:

	Name	age	Marks
0	a	10	56.7
1	b	20	19.0
2	c	30	45.0

Below the DataFrame, there are three buttons: 'Generate code with DF', 'View recommended plots', and 'New Interactive sheet'. Below these buttons, there is a code cell with the following content:

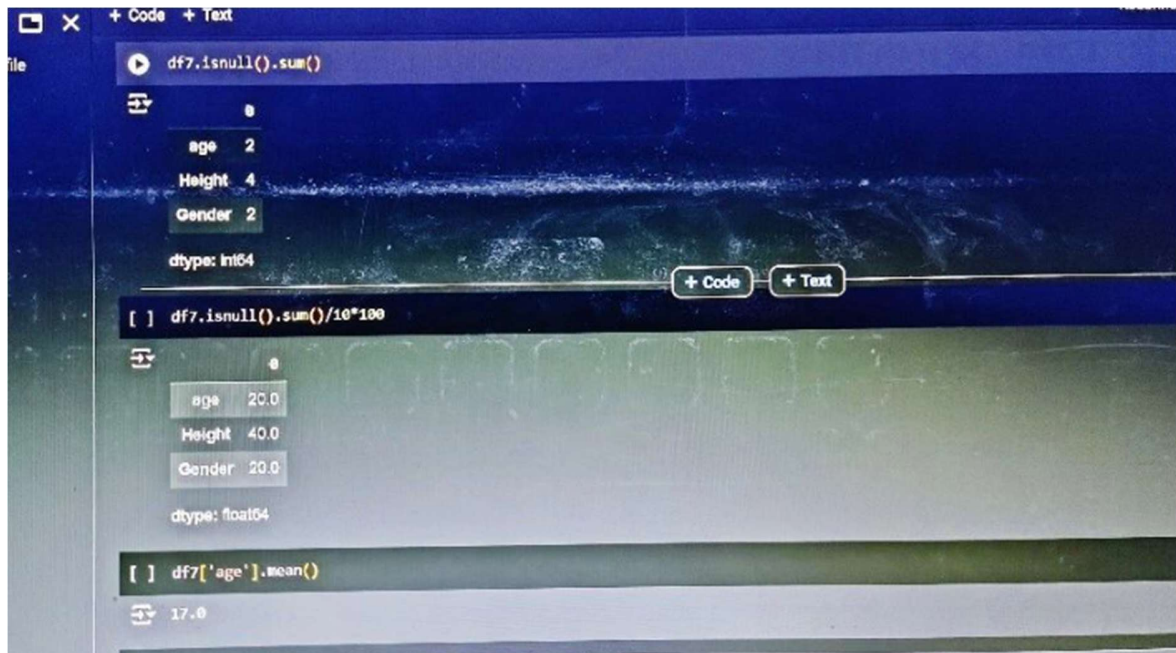
```
[ ] L=[1,2,3]
print(L)
```

The output of this code cell is shown as a list: [1, 2, 3]. Below this, there is another code cell with the following content:

```
[ ] df['Roll no.']=L
df
```

The output of this code cell is shown as a DataFrame with columns 'a', 'b', 'c', 'd', 'message', and 'Roll no.'. The data is as follows:

	a	b	c	d	message	Roll no.
0	1	2	3	4	hello	1
1	5	6	7	8	world	2
2	9	10	11	12	foo	3



The screenshot shows a Jupyter Notebook interface with three code cells. The first cell contains `df7.isnull().sum()` and its output is a Series with values 0 for age, 4 for Height, and 2 for Gender, with a dtype of int64. The second cell contains `df7.isnull().sum()/10*100` and its output is a Series with values 0.0 for age, 40.0 for Height, and 20.0 for Gender, with a dtype of float64. The third cell contains `df7['age'].mean()` and its output is 17.0.

```
df7.isnull().sum()
```

age	0
Height	4
Gender	2

dtype: int64

```
df7.isnull().sum()/10*100
```

age	0.0
Height	40.0
Gender	20.0

dtype: float64

```
df7['age'].mean()
```

17.0

```
df = pd.DataFrame({"Values": [10, 20, 30, 40, 50]})  
print(df.describe())
```


30-10-2024

Training Day – 28

Topic:* Pandas Input-Output Operations

- Read and wrote data to CSV and Excel files.
- Example: Saved a DataFrame to output.csv and loaded it back using pd.read_csv().

```
df=pd.read_csv("latest Covid-19 India Status1 (2).xls")
```

```
df.head()
```

	State/UTs	Total Cases	Active	Deaths	Active Ratio (%)	Death Ratio (%)
0	Maharashtra	6122893	117869	123857	1.93	2.02
1	Kerala	3011694	108400	14108	3.60	0.47
2	Karnataka	2862338	39626	35601	1.38	1.24
3	Tamil Nadu	2506848	34076	33196	1.36	1.32
4	Andhra Pradesh	1911231	32356	12919	1.69	0.68

Reading and writing data from/to CSV and Excel files.

Example:

```
df.to_csv("data.csv", index=False)
loaded_df = pd.read_csv("data.csv")
print(loaded_df)
```

02-11-2024

Training Day – 29

- ***Topic:** Pandas Data Manipulation

- Performed filtering, sorting, and adding new columns.
- Example: Filtered rows where column values exceeded a threshold.

Import necessary libraries

```
import pandas as pd
```

```
# Create a sample dataset
```

```
data = {  
    'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eve'],  
    'Age': [25, 32, 18, 45, 22],  
    'Salary': [50000, 60000, 32000, 78000, 45000],  
    'Department': ['HR', 'IT', 'Finance', 'Marketing', 'HR']  
}
```

```
# Convert to a DataFrame
```

```
df = pd.DataFrame(data)
```

```
# Display the original dataset
```

```
print("Original Dataset:")  
display(df)
```

```
# 1. Filtering rows where Salary > 45000
```

```
filtered_df = df[df['Salary'] > 45000]  
print("\nFiltered Dataset (Salary > 45000):")  
display(filtered_df)
```

```
# 2. Sorting by Age (ascending)
```

```
sorted_df = df.sort_values(by='Age')  
print("\nDataset Sorted by Age:")  
display(sorted_df)
```

```
# 3. Adding a new column 'Seniority'
```

```
# Seniority is 'Senior' if Age > 30, otherwise 'Junior'
```

```
df['Seniority'] = ['Senior' if age > 30 else 'Junior' for age in df['Age']]  
print("\nDataset with New Column 'Seniority':")
```


```
display(df)
```

```
# Exporting to CSV for further analysis if needed
```

```
df.to_csv('pandas_manipulation_example.csv', index=False)
```

1. Original Dataset:


markdown

 Copy code

	Name	Age	Salary	Department
0	Alice	25	50000	HR
1	Bob	32	60000	IT
2	Charlie	18	32000	Finance
3	David	45	78000	Marketing
4	Eve	22	45000	HR

2. Filtered Dataset (Salary > 45000):

markdown

 Copy code

	Name	Age	Salary	Department
0	Alice	25	50000	HR
1	Bob	32	60000	IT
3	David	45	78000	Marketing

3. Sorted Dataset (by Age):

