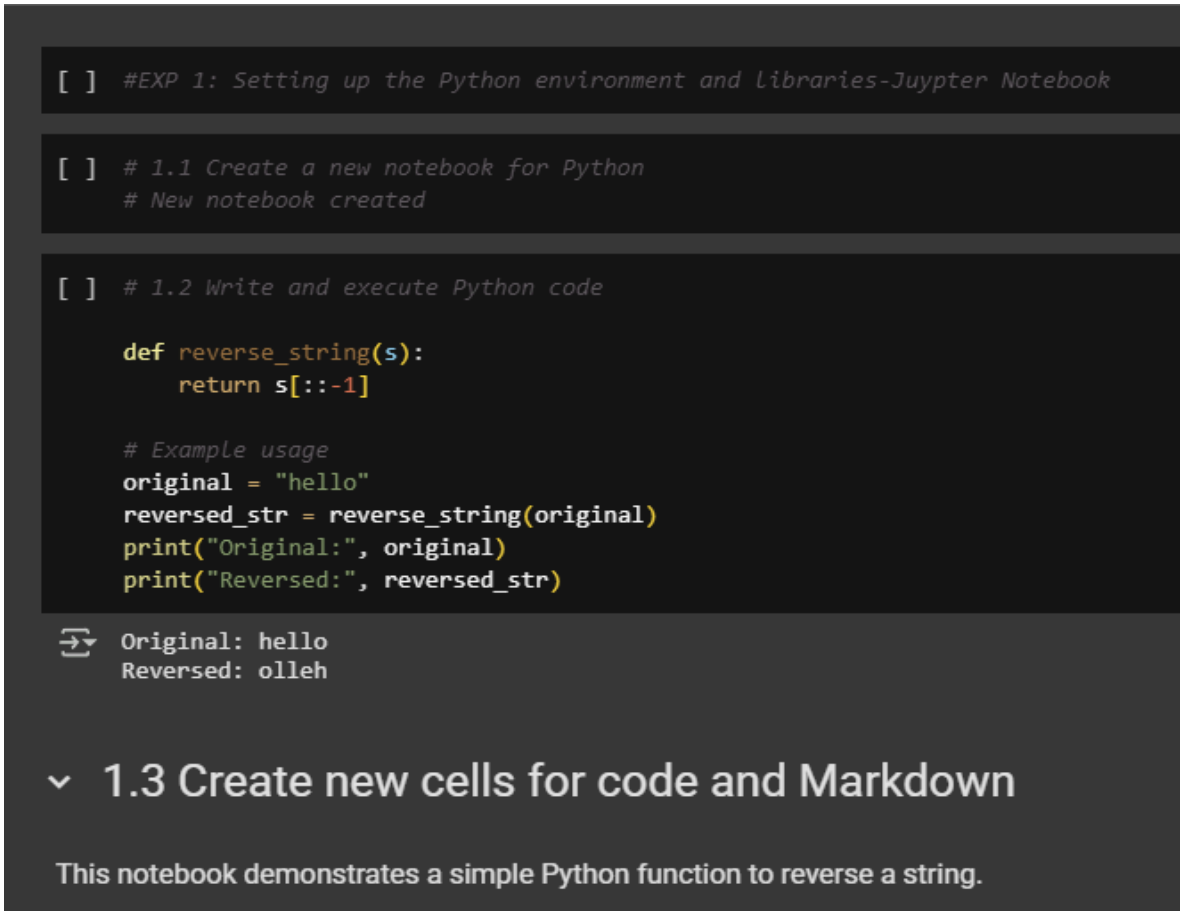
Framework for Data and Visual Analytics Lab Experiments

Name: C S Vishnupriya
Roll No: 231501185
Department: AIML
Subject Code: AD23632

Exp 1: Setting up the Python environment and libraries-Juypter Notebook



```
[ ]  #EXP 1: Setting up the Python environment and libraries-Juypter Notebook
```

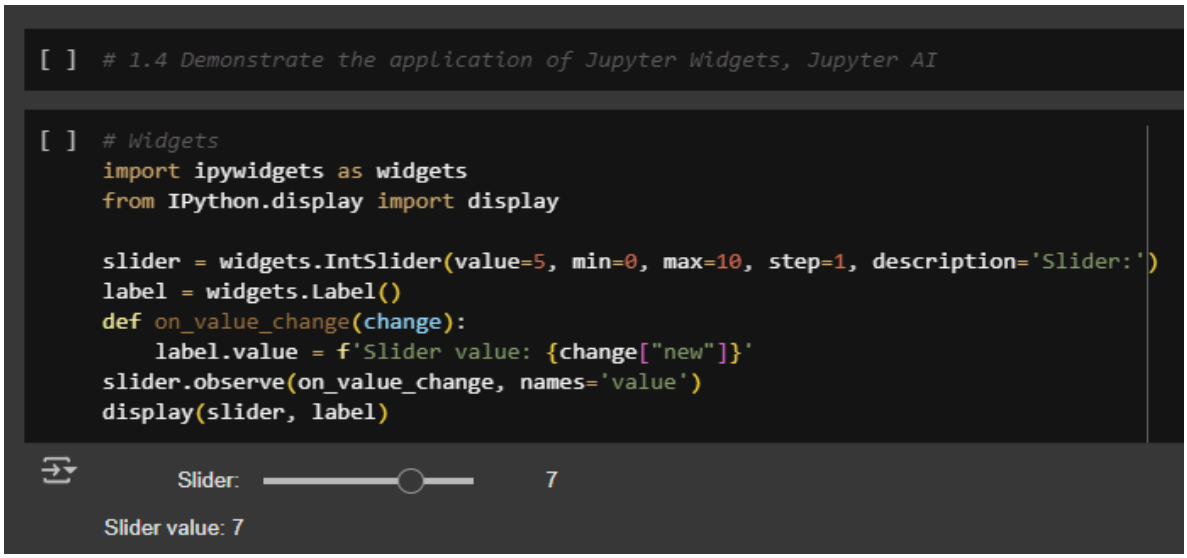
```
[ ]  # 1.1 Create a new notebook for Python
     # New notebook created
```

```
[ ]  # 1.2 Write and execute Python code

     def reverse_string(s):
         return s[::-1]

     # Example usage
     original = "hello"
     reversed_str = reverse_string(original)
     print("Original:", original)
     print("Reversed:", reversed_str)
```

```
Original: hello
Reversed: olleh
```

1.3 Create new cells for code and Markdown

This notebook demonstrates a simple Python function to reverse a string.



```
[ ]  # 1.4 Demonstrate the application of Jupyter Widgets, Jupyter AI
```

```
[ ]  # Widgets
     import ipywidgets as widgets
     from IPython.display import display

     slider = widgets.IntSlider(value=5, min=0, max=10, step=1, description='Slider:')
     label = widgets.Label()
     def on_value_change(change):
         label.value = f'Slider value: {change["new"]}'
     slider.observe(on_value_change, names='value')
     display(slider, label)
```

Slider: 7

Slider value: 7

Exp 2: EDA-Data Import and Export

```
[9]  # 2.1 Importing data from CSV, Excel, SQL databases, and web scraping
```

```
[4]  # CSV file

import pandas as pd

csv_url = "https://raw.githubusercontent.com/datasciencedojo/datasets/master/titanic.csv"

df_csv = pd.read_csv(csv_url)

print(df_csv.head())
```

```
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                Name     Sex   Age  SibSp  \
0                            Braund, Mr. Owen Harris    male  22.0      1
1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0      1
2                             Heikkinen, Miss. Laina  female  26.0      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                           Allen, Mr. William Henry    male  35.0      0

   Parch            Ticket     Fare Cabin Embarked
0      0         A/5 21171   7.2500   NaN        S
1      0          PC 17599  71.2833   C85        C
2      0  STON/O2. 3101282   7.9250   NaN        S
3      0            113803  53.1000  C123        S
4      0            373450   8.0500   NaN        S
```

```
[6]  # Excel file

import pandas as pd

df = pd.DataFrame({
    'Name': ['Alice', 'Bob', 'Charlie'],
    'Age': [25, 30, 35],
    'City': ['NY', 'LA', 'Chicago']
})

df.to_excel('sample.xlsx', index=False)
df_excel = pd.read_excel('sample.xlsx')
print(df_excel)
```

```
      Name  Age     City
0    Alice   25       NY
1      Bob   30       LA
2  Charlie   35  Chicago
```

```
[10]  # SQL data

import pandas as pd
import sqlite3

conn = sqlite3.connect(':memory:')
conn.execute('CREATE TABLE users (id INTEGER PRIMARY KEY, name TEXT, age INTEGER)')
conn.execute("INSERT INTO users (name, age) VALUES ('Alice', 25), ('Bob', 30), ('Charlie', 35)")
df_sql = pd.read_sql('SELECT * FROM users', conn)
print(df_sql)
```

```
   id     name  age
0   1    Alice   25
1   2      Bob   30
2   3  Charlie   35
```

```
[11] # Web Scraping

     import requests
     from bs4 import BeautifulSoup

     url = 'http://quotes.toscrape.com/'
     response = requests.get(url)
     soup = BeautifulSoup(response.text, 'html.parser')
     quotes = [quote.text.strip() for quote in soup.find_all('span', class_='text')]
     print(quotes[:5])

['“The world as we have created it is a process of our thinking. It cannot be changed without
```

```
[13] # 2.2 Handling different data formats

     import pandas as pd
     import sqlite3
     import requests
     from bs4 import BeautifulSoup

     # CSV
     pd.DataFrame({'Name': ['A', 'B'], 'Age': [20, 30]}).to_csv('sample.csv', index=False)
     print(pd.read_csv('sample.csv'))

     # Excel
     df = pd.DataFrame({'Name': ['X', 'Y'], 'City': ['NY', 'LA']})
     df.to_excel('sample.xlsx', index=False)
     print(pd.read_excel('sample.xlsx'))

     # SQL (avoid reserved word as table name)
     conn = sqlite3.connect(':memory:')
     df.to_sql('people', conn, index=False)  # renamed from 'table' to 'people'
     print(pd.read_sql('SELECT * FROM people', conn))

     # Web scraping
     res = requests.get('http://quotes.toscrape.com/')
     soup = BeautifulSoup(res.text, 'html.parser')
     print([q.text for q in soup.find_all('span', class_='text')[:2]])

  Name  Age
0    A   20
1    B   30
  Name City
0    X   NY
```

```
[14] # 2.3 Export a DataFrame to an Excel file

     import pandas as pd

     df = pd.DataFrame({
         'Name': ['Alice', 'Bob', 'Charlie'],
         'Age': [25, 30, 35],
         'City': ['New York', 'Los Angeles', 'Chicago']
     })

     df.to_excel('output.xlsx', index=False)
     print("DataFrame exported to 'output.xlsx'")

DataFrame exported to 'output.xlsx'
```

Exp 3: EDA-Data Cleaning

```
[15] #EXP 3: EDA-Data Cleaning
```

```
[17] import pandas as pd
     from sklearn.preprocessing import StandardScaler

     # Sample data with missing, duplicates, and inconsistent types
     df = pd.DataFrame({
         'Name': ['Alice', 'Bob', 'bob', None, 'David', 'Eva', 'Eva'],
         'Age': ['25', '30', '30', '22', None, '35', '35'],
         'Salary': [50000, 60000, 60000, 52000, 58000, None, None]
     })

     # 3.1 Handling missing values: detection, filling, and dropping
     df['Age'] = pd.to_numeric(df['Age'], errors='coerce')   # Convert Age to numeric
     df['Age'].fillna(df['Age'].mean(), inplace=True)
     df['Salary'].fillna(df['Salary'].median(), inplace=True)
     df['Name'].fillna('Unknown', inplace=True)

     # 3.2 Removing duplicates and unnecessary data
     df.drop_duplicates(inplace=True)

     # 3.3 Data type conversion and ensuring consistency
     df['Name'] = df['Name'].str.lower()  # Normalize text data to lowercase
     df['Age'] = df['Age'].astype(int)

     # 3.4 Normalize data (e.g., standardization, min-max scaling)
     scaler = StandardScaler()
     df[['Age', 'Salary']] = scaler.fit_transform(df[['Age', 'Salary']])

     print(df)
```

```
      Name       Age    Salary
0    alice -0.850963 -1.623280
1      bob  0.364698  0.939793
2      bob  0.364698  0.939793
3  unknown -1.580360 -1.110665
4    david  0.121566  0.427179
5      eva  1.580360  0.427179
```

Exp 4: EDA-Data Inspection and Analysis

```
[18] #EXP 4: EDA-Data Inspection and Analysis
```

```
[19] # 4.1 Viewing and inspecting DataFrames

import pandas as pd

data = {
    'Name': ['Alice', 'Bob', 'Charlie', 'David'],
    'Age': [25, 30, 35, 40],
    'Salary': [50000, 60000, 70000, 80000]
}
df = pd.DataFrame(data)
print(df.head())
print(df.tail())
print(df.info())
print(df.describe())
print(df.columns)
print(df.index)
```

```
      Name  Age  Salary
0    Alice   25   50000
1      Bob   30   60000
2  Charlie   35   70000
3    David   40   80000
      Name  Age  Salary
0    Alice   25   50000
1      Bob   30   60000
2  Charlie   35   70000
3    David   40   80000
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4 entries, 0 to 3
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Name    4 non-null      object
 1   Age     4 non-null      int64
 2   Salary  4 non-null      int64
dtypes: int64(2), object(1)
memory usage: 228.0+ bytes
None
             Age        Salary
count   4.000000      4.000000
mean   32.500000  65000.000000
std     6.454972  12909.944487
min    25.000000  50000.000000
25%    28.750000  57500.000000
```

[20]
```
# 4.2 Filtering and subsetting data using conditions

filtered_df = df[df['Age'] > 30]
print(filtered_df)
filtered_df2 = df[(df['Salary'] >= 55000) & (df['Salary'] <= 75000)]
print(filtered_df2)
filtered_df3 = df[df['Name'].str.startswith('A')]
print(filtered_df3)
```

```
      Name  Age  Salary
2  Charlie   35   70000
3    David   40   80000
      Name  Age  Salary
1      Bob   30   60000
2  Charlie   35   70000
    Name  Age  Salary
0  Alice   25   50000
```

[21]
```
# 4.3 Descriptive statistics: measures of central tendency (mean, median, mode)

mean_age = df['Age'].mean()
print(f"Mean Age: {mean_age}")
median_salary = df['Salary'].median()
print(f"Median Salary: {median_salary}")
mode_age = df['Age'].mode()
print(f"Mode Age: {mode_age.tolist()}")
range_age = df['Age'].max() - df['Age'].min()
print(f"Range of Age: {range_age}")
variance_salary = df['Salary'].var()
print(f"Variance of Salary: {variance_salary}")
std_salary = df['Salary'].std()
print(f"Standard Deviation of Salary: {std_salary}")
```

```
Mean Age: 32.5
Median Salary: 65000.0
Mode Age: [25, 30, 35, 40]
Range of Age: 15
Variance of Salary: 166666666.66666666
Standard Deviation of Salary: 12909.944487358056
```