List of Experiments:

Implement the lab experiments in Python with any real time example

1. Introduction to programming with Python.
2. Python programming basics
3. Conditional statements
4. Loops
5. Functions
6. Integrated Development Environments (IDEs).
7. How to structure Python code in a project.
8. How to manage libraries in Python using virtual environments.
9. Data Loading, Storage, and File Formats.
10. Data Cleaning and Preparation.
11. Data Manipulation with Pandas.
12. Data Wrangling: Join, Combine, and Reshape.
13. Plotting and Visualization.
14. Data Aggregation and Group Operations.
15. Advanced Numpy.
16. Matplotlib
17. Building and optimizing pipelines in scikit-learn.

Code:

1. Introduction to programming with Python

Program: Print "Welcome to Data Science"

```
print("Welcome to Data Science")
```

Output:
Welcome to Data Science

2. Python programming basics
Code:

```
a = 10
b = 5
print("Addition:", a + b)
print("Subtraction:", a - b)
print("Multiplication:", a * b)
```

Output:
Addition: 15
Subtraction: 5
Multiplication: 50

3. Conditional statements
Code:

```
num = 7
if num % 2 == 0:
   print("Even")
else:
   print("Odd")
```

Output:
Odd

4. Loops
Code:
Print numbers 1 to 5 using a loop

```
for i in range(1, 6):
   print(i)
```

Output:
1
2
3
4
5

5. Functions
Code:
```
def square(num):
   return num * num
print(square(4))
```

Output:
16

6. Use IDLE or Jupyter Notebook to run previous programs.

7. How to structure Python code in a project.
Code:
program: Split code into separate files for neatness.
File: greeting.py

```
def say_hello(name):
   return "Hello, " + name
```

File: main.py

```
from greeting import say_hello
print(say_hello("Roy"))
```

Output:

Hello, Roy

8. How to manage libraries in Python using virtual environments
No coding here, but simple commands:

```
# Create a virtual environment
python -m venv myenv

# Activate it (Windows)
myenv\Scripts\activate

# Install a package
pip install pandas
```

9. Program: Load a CSV file (you can use a sample CSV).

```
import pandas as pd
df = pd.read_csv("sample.csv")
print(df)
```

Sample Output:

```
Name  Age
0  Ali    22
1  Sara   20
```

10. Data Cleaning and Preparation.

Code:

```
import pandas as pd
data = {'Name': ['Ali', 'Sara', 'Zara'], 'Age': [22, None, 25]}
df = pd.DataFrame(data)
df.fillna(0, inplace=True)
print(df)
```

Output:

```
Name   Age
0  Ali   22.0
```

1 Sara 0.0

2 Zara 25.0

11. Data Manipulation with Pandas.

Code:

```
import pandas as pd
df = pd.DataFrame({'Student': ['Ali', 'Sara'], 'Marks': [80, 90]})
df.rename(columns={'Marks': 'Score'}, inplace=True)
print(df)
```

Output:

```
Student  Score
0    Ali     80
1   Sara     90
```

12. Data Wrangling: Join, Combine, and Reshape.

Code:

```
import pandas as pd
a = pd.DataFrame({'ID': [1, 2], 'Name': ['Ali', 'Sara']})
b = pd.DataFrame({'ID': [1, 2], 'Marks': [85, 90]})
result = pd.merge(a, b, on='ID')
print(result)
```

Output:

```
ID  Name  Marks
0   1   Ali     85
1   2  Sara     90
```

13. Plotting and Visualization.

Code:

```
import matplotlib.pyplot as plt
names = ['Ali', 'Sara']
scores = [90, 95]
```

```
plt.bar(names, scores)
plt.title("Student Scores")
plt.show()
```

14. Data Aggregation and Group Operations.

Code:

```
import pandas as pd
data = {'Dept': ['IT', 'HR', 'IT'], 'Salary': [30000, 25000, 40000]}
df = pd.DataFrame(data)
print(df.groupby('Dept').mean())
```

Output:

```
Salary
Dept
HR     25000.0
IT     35000.0
```

15. Advanced Numpy.

Code:

```
import numpy as np
arr = np.array([10, 20, 30])
print("Mean:", np.mean(arr))
print("Sum:", np.sum(arr))
```

Output:

```
Mean: 20.0
Sum: 60
```

16. Matplotlib

Code:

```
import matplotlib.pyplot as plt
x = [1, 2, 3]
y = [10, 20, 15]
plt.plot(x, y)
```

```
plt.title("Simple Line Chart")
plt.xlabel("X Axis")
plt.ylabel("Y Axis")
plt.show()
```

17. Building and optimizing pipelines in scikit-learn.

Code:

Program: Simple pipeline (no model training needed).

```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression

pipeline = Pipeline([
   ('scaler', StandardScaler()),
   ('model', LogisticRegression())
])
print(pipeline)
```

Output:

```
Pipeline(steps=[('scaler', StandardScaler()), ('model', LogisticRegression())])
```