

G Pulla Reddy Engineering College (Autonomous), Kurnool
Electronics and Communication Engineering Department
DATA SCIENCE LAB (DSL(P))
Lab Component (Scheme-20)

List of Experiments	GPREC/ECE/DSL LAB COMPONENT
	DATE: 12/07/2023

1. Python Environment setup to work with Data science
2. NumPy: Arithmetic Operations on Arrays
3. Generate Pseudo Random numbers using various methods in NumPy
4. Perform Linear search, binary search using NumPy arrays.
5. Loading and extracting data from different Data frames
6. Pandas: Program to deal with missing data by reading data from a file.
7. Implement data wrangling functions on raw data
8. Matplotlib: Visualize data by plotting a scatter plot.
9. Program to visualize data using pie and bar graphs.
10. Implement programs on Date and Time Data Types

G Pulla Reddy Engineering College (Autonomous), Kurnool
Electronics and Communication Engineering Department
DATA SCIENCE LAB (DSL(P))
Lab Component (Scheme-20)

Python Environment setup to work with Data science

GPREC/ECE/DSL LAB/EXPT 01

DATE: 12/07/2023

1. Python Environment setup to work with Datascience

The Anaconda distribution simplifies the installation process by including Python, Spyder, and other packages and tools in one installation file. It contains the core Python language, as well as all of the essential libraries including NumPy, Pandas, SciPy, Matplotlib, and IPython. By using the graphical installer, downloading Python is as easy as downloading any computer program

Installing Anaconda

Step 1:

Go to <http://continuum.io/downloads>

Step 2:

Scroll down to find your operating system and click on Python 3.9 to download the graphical installer.

Step 3:

For windows, see <http://windows.microsoft.com/en-us/windows/32-bitand-64-bit-windows> to find out whether your computer has a 32-bit or 64-bit version of windows. Click on the FAQ “How can I tell if my computer is running a 32-bit or 64-bit version of windows?”, and follow the instructions to find out. If your computer is running a 32-bit version of Windows, click on “Windows 32-bit Python 3.4 Graphical Installer” under OTHER INSTALLERS, instead of the Windows 64-bit installer, and follow the same directions to install it.

Step 4:

Save the file to your computer.

G Pulla Reddy Engineering College (Autonomous), Kurnool
Electronics and Communication Engineering Department
DATA SCIENCE LAB (DSL(P))
Lab Component (Scheme-20)

Step 5:

Double click on the downloaded file to open it

Step 6:

Follow the on-screen installation instructions, leaving options as they are currently set. This finishes the installation process.

Step 7:

Next, check for any updates using Conda. Conda is one of the extras that is installed through the distribution Anaconda. It handles things like updates, set-up, and package installation through a command line interface.

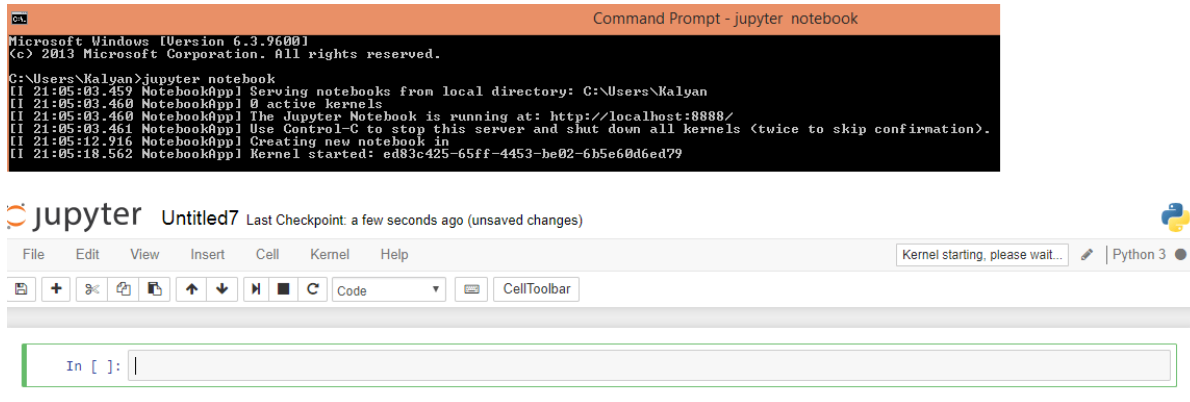
WINDOWS:

- a. Open Anaconda Command Prompt. Start typing Anaconda Command Prompt into the search box in the start menu, and it will show up.
- b. Type `conda update conda` at the command prompt, typing `y` for Yes and then pressing enter when it asks if you want to proceed. Your installation may identify different packages that need updated
- c. After that completes, type `conda update anaconda` at the command prompt. If it prompts you to proceed with installation or updating, type `y` for Yes and press enter
- d. After that, type `conda install seaborn` at the command prompt, then type `y` after the Proceed `([y]/n)?` line
- e. After that completes, you can then close the command prompt window.

Getting to Know Jupyter Notebook

After installing, you will get a launcher containing a number of programs. The most important one is the iPython notebook, which is also called Jupyter notebook. Once you launch the notebook, the terminal is opened and a notebook is opened in your browser.

G Pulla Reddy Engineering College (Autonomous), Kurnool
Electronics and Communication Engineering Department
DATA SCIENCE LAB (DSL(P))
Lab Component (Scheme-20)



Python is a general purpose language and is often used for things other than data analysis and data science. What makes Python extremely useful for working with data?

There are libraries that give users the necessary functionality when crunching data. Below are the major Python libraries that are used for working with data. You should take some time to familiarize yourself with the basic purposes of these packages.

Numpy and Scipy

NumPy stands for Numerical Python. The most powerful feature of NumPy is n-dimensional array. This library also contains basic linear algebra functions, Fourier transforms, advanced random number capabilities and tools for integration with other low level languages like Fortran, C and C++. SciPy stands for Scientific Python. It is built on NumPy. Scipy is one of the most useful library for variety of high level science and engineering modules like discrete Fourier transform, Linear Algebra, Optimization and Sparse matrices.

Pandas

Pandas for structured data operations and manipulations. It is extensively used for data munging and preparation. Pandas were added relatively recently to Python and have been instrumental in boosting Python's usage in data scientist community.

Matplotlib

Matplotlib for plotting vast variety of graphs, starting from histograms to line plots to heat plots.

G Pulla Reddy Engineering College (Autonomous), Kurnool
Electronics and Communication Engineering Department
DATA SCIENCE LAB (DSL(P))
Lab Component (Scheme-20)

Scikit-learn

Scikit Learn for machine learning. Built on NumPy, SciPy and matplotlib, this library contains a lot of efficient tools for machine learning and statistical modelling including classification, regression, clustering and dimensional reduction.

StatsModels:

Statsmodels for statistical modelling. It is a Python module that allows users to explore data, estimate statistical models, and perform statistical tests. An extensive list of descriptive statistics, statistical tests, plotting functions, and result statistics are available for different types of data and each estimator.

Seaborn

Seaborn for statistical data visualization. It is a library for making attractive and informative statistical graphics in Python. It is based on matplotlib. Seaborn aims to make visualization a central part of exploring and understanding data.

G Pulla Reddy Engineering College (Autonomous), Kurnool
Electronics and Communication Engineering Department
DATA SCIENCE LAB (DSL(P))
Lab Component (Scheme-20)

NumPy: Arithmetic Operations on Arrays

GPREC/ECE/DSL LAB/EXPT 02

DATE: 12/07/2023

Perform the following arithmetic operations on matrices.

- **Addition, Subtraction, Multiplication, Division, Transpose, Inverse, Power**

Perform the following operations on matrices

- **Basic Indexing**
- **Indexing using slicing operator**
- **Indexing 2D arrays**
- **Indexing 3D arrays**
- **Advanced indexing using integer arrays**
- **Advanced indexing using Boolean conditions**
- **Fancy indexing**

```
import numpy as np
arr = np.array([[3,4],[5,6]])
print(arr)
```

```
import numpy as np
arr = np.array([[1,2,3],[4,5,6]])
print(arr)
```

```
import numpy as np
arr = np.random.randint(20,size=(3,4))
print(arr)
```

```
import numpy as np
arr = np.arange(start=1,stop=13).reshape(3,4)
print(arr)
```

```
import numpy as np
arr = np.arange(36).reshape(6,6)
print(arr)
```

G Pulla Reddy Engineering College (Autonomous), Kurnool
Electronics and Communication Engineering Department
DATA SCIENCE LAB (DSL(P))
Lab Component (Scheme-20)

```
import numpy as np
arr1 = np.arange(36).reshape(6,6)
arr2 = np.random.randint(100,size=(6,6))
print(arr1)
print(arr2)
```

```
import numpy as np
arr1 = np.arange(36).reshape(6,6)
arr2 = np.random.randint(100,size=(6,6))
print("arr1 = ",arr1)
print("arr2 = ",arr2)
print("sum = ",arr1+arr2)
print("Difference = ",arr1-arr2)
print("Multiplication = ",arr1*arr2)
```

Perform the following arithmetic operations on matrices.

Addition, Subtraction, Multiplication, Division, Transpose, Inverse, Power

```
import numpy as np
arr1 = np.arange(36).reshape(6,6)
arr2 = np.random.randint(100,size=(6,6))
print("arr1 = ",arr1)
print("arr2 = ",arr2)
print("sum = ",arr1+arr2)
print("Difference = ",arr1-arr2)
print("Multiplication = ",arr1*arr2)
print("Vector Multiplication = ",np.dot(arr1,arr2))
print("Transverse of arr1 = ",np.transpose(arr1))
print("inverse of arr1 = ",np.linalg.inv(arr1))
print("power of arr1 = ",np.linalg.matrix_power(arr1,2))
print("power of arr2 = ",arr2**2)
```

G Pulla Reddy Engineering College (Autonomous), Kurnool
Electronics and Communication Engineering Department
DATA SCIENCE LAB (DSL(P))
Lab Component (Scheme-20)

```
import numpy as np
arr = np.arange(0,150,10)
print(arr)
print(arr[1])
print(arr[2:3])
print(arr[3:8:2])
print(arr[4::3])
```

```
import numpy as np
arr = np.array([])
for i in range(12):
    arr = np.append(arr,input())
arr = arr.reshape(3,4)
print(arr)
```

```
import numpy as np
arr = np.array([])
arr = np.append(arr,[[1,2,3,4]])
print(arr)
```

```
import numpy as np
arr = np.array([])
arr = np.append(arr,[[1,2,3,4]])
print(arr)
print(arr.reshape(2,2))
```


G Pulla Reddy Engineering College (Autonomous), Kurnool
Electronics and Communication Engineering Department
DATA SCIENCE LAB (DSL(P))
Lab Component (Scheme-20)

```
import numpy as np
arr = np.array([])
for i in range(12):
    arr = np.append(arr,[1,2,3,4])
arr = arr.reshape(12,4)
print(arr)
```

```
import numpy as np
arr = np.random.randint(100,size=16)
print(arr)
print(arr[2:10:3])
arr = arr.reshape(4,4)
print(arr)
print(arr[3][2])
print(arr[3,:])
print(arr[:,2])
print(arr[1:3,2])
print(arr[3,0:2])
print(arr[0:2,1:3])
arr = arr.reshape(2,2,4)
print(arr)
print(arr[1])
print(arr[0,1,:])
print(arr[1,.,2])
print(arr[1,0,3])
```

```
import numpy as np
arr=np.random.randint(20,size=(10))
print(arr)
index_arr_1 = np.array([3,5,8])
```

G Pulla Reddy Engineering College (Autonomous), Kurnool
Electronics and Communication Engineering Department
DATA SCIENCE LAB (DSL(P))
Lab Component (Scheme-20)

```
print(arr[index_arr_1])  
print(arr[arr>5])
```

```
arr = np.arange(10)  
print(arr)  
index_arr_1 = np.array([3,5,9])  
print('The 3rd, 5th and 9th elements of the array are \n', arr[index_arr_1])
```

```
import numpy as np  
arr= np.empty((8,4))  
for i in range(8):  
    arr[i] = i  
print(arr)  
arr[[4,3,0,5]]  
arr[[1, 5, 7, 2], [0, 3, 1, 2]]  
arr[[1, 5, 7, 2]][:, [0, 3, 1, 2]]  
arr[np.ix_([1, 5, 7, 2], [0, 3, 1, 2])]
```

G Pulla Reddy Engineering College (Autonomous), Kurnool
Electronics and Communication Engineering Department
DATA SCIENCE LAB (DSL(P))
Lab Component (Scheme-20)

Perform Linear Search, Binary Search using Numpy Arrays

GPREC/ECE/DSL LAB /EXPT 04

DATE: 12/07/2023

Perform Linear search, binary search using NumPy arrays.

For the given numpy array perform the following operation.

- **Find the indexes where the value is 4**
- **Find the indexes where the values are even**
- **Find the indexes where the values are odd**
- **Find the indexes where the value 7 should be inserted**
- **Find the indexes where the value 7 should be inserted, starting from the right**
- **Find the indexes where the values 2, 4, and 6 should be inserted**

Find the indexes where the value is 4:

```
import numpy as np
arr = np.array([1, 2, 3, 4, 5, 4, 4])
x = np.where(arr == 4)
print(x)
```

output:

```
(array([3, 5, 6]),)
```

Example:

Find the indexes where the values are even:

```
import numpy as np
arr = np.array([1, 2, 3, 4, 5, 6, 7, 8])
x = np.where(arr%2 == 0)
print(x)
```

output:

```
(array([1, 3, 5, 7]),)
```

Example:

Find the indexes where the values are odd:

```
import numpy as np
arr = np.array([1, 2, 3, 4, 5, 6, 7, 8])
```

G Pulla Reddy Engineering College (Autonomous), Kurnool
Electronics and Communication Engineering Department
DATA SCIENCE LAB (DSL(P))
Lab Component (Scheme-20)

```
x = np.where(arr%2 == 1)
print(x)
```

output:

```
(array([0, 2, 4, 6]),)
```

Find the indexes where the value 7 should be inserted:

```
import numpy as np
arr = np.array([6, 7, 8, 9])
x = np.searchsorted(arr, 7)
print(x)
```

output:

```
1
```

Example:

Find the indexes where the value 7 should be inserted, starting from the right:

```
import numpy as np
arr = np.array([6, 7, 8, 9])
x = np.searchsorted(arr, 7, side='right')
print(x)
```

output:

```
2
```

To search for more than one value, use an array with the specified values.

Example:

Find the indexes where the values 2, 4, and 6 should be inserted:

```
import numpy as np
arr = np.array([1, 3, 5, 7])
x = np.searchsorted(arr, [2, 4, 6])
print(x)
```

output:

```
[1 2 3]
```

G Pulla Reddy Engineering College (Autonomous), Kurnool
Electronics and Communication Engineering Department
DATA SCIENCE LAB (DSL(P))
Lab Component (Scheme-20)

Generate Pseudo Random numbers Using Methods in Numpy

GPREC/ECE/DSL LAB /EXPT 03

DATE: 12/07/2023

Generate Pseudo Random numbers using various methods in NumPy

Perform the following operation.

- **Generate a random integer from 0 to 100**
- **Generate a random float from 0 to 1**
- **Generate a 1-D array containing 5 random integers from 0 to 100**
- **Generate a 2-D array with 3 rows, each row containing 5 random integers from 0 to 100**
- **Generate a 1-D array containing 5 random floats**
- **Generate a 2-D array with 3 rows, each row containing 5 random numbers**
- **Return one of the values in an array using choice function.**
- **Generate a 2-D array that consists of the values in the array parameter (3, 5, 7, and 9)**

Generate a random integer from 0 to 100:

```
from numpy import random
x = random.randint(100)
print(x)
```

Output:

12

Generate a random float from 0 to 1:

```
from numpy import random
x = random.rand()
print(x)
```

Output:

0.4039443604901367

Generate a 1-D array containing 5 random integers from 0 to 100:

```
from numpy import random
x=random.randint(100, size=(5))
```

G Pulla Reddy Engineering College (Autonomous), Kurnool
Electronics and Communication Engineering Department
DATA SCIENCE LAB (DSL(P))
Lab Component (Scheme-20)

```
print(x)
```

Output:

```
[15 2 78 69 47]
```

Generate a 2-D array with 3 rows, each row containing 5 random integers from 0 to 100:

```
from numpy import random
x = random.randint(100, size=(3, 5))
print(x)
```

Output:

```
[[80 54 19 74 65]
 [26 60 69 34 25]
 [50 16 53 84 90]]
```

Example

Generate a 1-D array containing 5 random floats:

```
from numpy import random
x = random.rand(5)
print(x)
```

Output:

```
[0.0717149 0.1610171 0.7472575 0.5405383 0.6102538]
```

Generate a 2-D array with 3 rows, each row containing 5 random numbers:

```
from numpy import random
x = random.rand(3, 5)
print(x)
```

Output:

```
[[0.03379952 0.78263517 0.9834899 0.47851523 0.02948659]
 [0.36284007 0.10740884 0.58485016 0.20708396 0.00969559]
 [0.88232193 0.86068608 0.75548749 0.61233486 0.06325663]]
```

Return one of the values in an array:

G Pulla Reddy Engineering College (Autonomous), Kurnool
Electronics and Communication Engineering Department
DATA SCIENCE LAB (DSL(P))
Lab Component (Scheme-20)

```
from numpy import random
x = random.choice([3, 5, 7, 9])
print(x)
```

Output:

3

The choice() method also allows you to return an array of values.
Add a size parameter to specify the shape of the array.

Generate a 2-D array that consists of the values in the array parameter (3, 5, 7, and 9):

```
from numpy import random
x = random.choice([3, 5, 7, 9], size=(3, 5))
print(x)
```

Output:

```
[[9 3 5 5 7]
 [7 5 3 3 9]
 [7 5 9 9 7]]
```

G Pulla Reddy Engineering College (Autonomous), Kurnool
Electronics and Communication Engineering Department
DATA SCIENCE LAB (DSL(P))
Lab Component (Scheme-20)

Loading and extracting data from different Dataframes

GPREC/ECE/DSL LAB /EXPT 05

DATE: 12/07/2023

```
import pandas as pd
```

```
# Create a data frame from a dictionary
```

```
data = {'Name': ['John', 'Emily', 'Sam', 'Sophia'],  
        'Age': [25, 30, 18, 21],  
        'City': ['New York', 'London', 'Paris', 'Tokyo']}  
df1 = pd.DataFrame(data)
```

```
# Create a data frame from a list of lists
```

```
data = [['Apple', 1.2, 100],  
        ['Banana', 0.5, 150],  
        ['Orange', 0.8, 75],  
        ['Mango', 1.5, 50]]  
columns = ['Fruit', 'Weight', 'Price']  
df2 = pd.DataFrame(data, columns=columns)
```

```
# Extract specific columns from dataframe
```

```
column1 = df1['column_name1'] # Extract a single column from df1  
columns2 = df2[['column_name2', 'column_name3']] # Extract multiple columns from df2
```

```
# Extract specific rows from dataframe
```

```
rows1 = df1.loc[df1['column_name'] == 'value'] # Extract rows from df1 based on a condition  
rows2 = df2.iloc[2:5] # Extract rows 2 to 4 from df2
```

```
# Merge or join dataframes based on a common key
```


G Pulla Reddy Engineering College (Autonomous), Kurnool
Electronics and Communication Engineering Department
DATA SCIENCE LAB (DSL(P))
Lab Component (Scheme-20)

```
merged_df = pd.merge(df1, df2, on='common_column') # Merge df1 and df2 based on a common column
```

```
# Concatenate dataframes vertically or horizontally
```

```
concatenated_df = pd.concat([df1, df2], axis=0) # Concatenate df1 and df2 vertically
```

```
concatenated_df2 = pd.concat([df1, df2], axis=1) # Concatenate df1 and df2 horizontally
```

```
# Group data by a column and perform aggregate functions
```

```
grouped_df = df1.groupby('column_name').sum() # Group df1 by column_name and sum the values
```

```
# Perform other operations on dataframes such as sorting, filtering, etc.
```

```
sorted_df = df1.sort_values('column_name') # Sort df1 based on column_name
```

```
filtered_df = df1[df1['column_name'] > 10] # Filter df1 based on a condition
```

```
# Access values using row and column indices
```

```
value = df1.iloc[0, 2] # Access value in the first row and third column of df1
```

G Pulla Reddy Engineering College (Autonomous), Kurnool
Electronics and Communication Engineering Department
DATA SCIENCE LAB (DSL(P))
Lab Component (Scheme-20)

Pandas: Program to deal with missing data by reading data from a file

GPREC/ECE/DSL LAB /EXPT 06

DATE: 12/07/2023

```
>>> df = pd.DataFrame([[np.nan, 2, np.nan, 0],  
...                    [3, 4, np.nan, 1],  
...                    [np.nan, np.nan, np.nan, np.nan],  
...                    [np.nan, 3, np.nan, 4]],  
...                    columns=list("ABCD"))  
>>> df
```

```
   A  B  C  D  
0 NaN 2.0 NaN 0.0  
1 3.0 4.0 NaN 1.0  
2 NaN NaN NaN NaN  
3 NaN 3.0 NaN 4.0
```

```
>>> df.fillna(0)
```

```
   A  B  C  D  
0 0.0 2.0 0.0 0.0  
1 3.0 4.0 0.0 1.0  
2 0.0 0.0 0.0 0.0  
3 0.0 3.0 0.0 4.0
```

```
>>> df.fillna(method="ffill")
```

```
   A  B  C  D  
0 NaN 2.0 NaN 0.0  
1 3.0 4.0 NaN 1.0  
2 3.0 4.0 NaN 1.0  
3 3.0 3.0 NaN 4.0
```

```
>>> values = {"A": 0, "B": 1, "C": 2, "D": 3}
```

```
>>> df.fillna(value=values)
```

```
   A  B  C  D  
0 0.0 2.0 2.0 0.0  
1 3.0 4.0 2.0 1.0  
2 0.0 1.0 2.0 3.0  
3 0.0 3.0 2.0 4.0
```

G Pulla Reddy Engineering College (Autonomous), Kurnool
Electronics and Communication Engineering Department
DATA SCIENCE LAB (DSL(P))
Lab Component (Scheme-20)

```
import csv

def handle_missing_data(data):
    # Iterate over each row
    for row in data:
        # Iterate over each value in the row
        for key, value in row.items():
            if value == "":
                # Handle missing data (you can customize this part)
                row[key] = None

# Function to read data from CSV file
def read_csv_file(file_path):
    data = []
    with open(file_path, 'r') as file:
        reader = csv.DictReader(file)
        for row in reader:
            data.append(row)
    return data

# Example usage
file_path = 'data.csv'
data = read_csv_file(file_path)
handle_missing_data(data)

# Print the processed data
for row in data:
    print(row)
```

G Pulla Reddy Engineering College (Autonomous), Kurnool
Electronics and Communication Engineering Department
DATA SCIENCE LAB (DSL(P))
Lab Component (Scheme-20)

Implement data wrangling functions on raw data

GPREC/ECE/DSL LAB /EXPT 07

DATE: 12/07/2023

Data Merging - Combining DataFrames:

```
import pandas as pd
```

```
# Sample data for two datasets
```

```
data1 = {  
    'ID': [1, 2, 3],  
    'Name': ['Alice', 'Bob', 'Charlie']  
}
```

```
data2 = {  
    'ID': [2, 3, 4],  
    'City': ['London', 'Paris', 'Berlin']  
}
```

```
df1 = pd.DataFrame(data1)  
df2 = pd.DataFrame(data2)
```

```
# Merging the two DataFrames based on 'ID'  
merged_df = pd.merge(df1, df2, on='ID', how='inner')
```

```
print(merged_df)
```

Data Aggregation - Grouping and Summarizing Data:

```
import pandas as pd
```

```
# Sample data for sales transactions
```

G Pulla Reddy Engineering College (Autonomous), Kurnool
Electronics and Communication Engineering Department
DATA SCIENCE LAB (DSL(P))
Lab Component (Scheme-20)

```
data = {  
    'Product': ['A', 'B', 'A', 'B', 'A', 'B'],  
    'Amount': [100, 150, 200, 120, 180, 90]  
}  
  
df = pd.DataFrame(data)  
  
# Grouping by 'Product' and calculating total sales  
grouped_df = df.groupby('Product')['Amount'].sum()  
  
print(grouped_df)
```

G Pulla Reddy Engineering College (Autonomous), Kurnool
Electronics and Communication Engineering Department
DATA SCIENCE LAB (DSL(P))
Lab Component (Scheme-20)

Matplotlib: Visualize data by plotting a scatter plot.

GPREC/ECE/DSL LAB /EXPT 08

DATE: 12/07/2023

```
import matplotlib.pyplot as plt
```

```
x=[5, 7, 8, 7, 2, 17, 2, 9, 4, 11, 12, 9, 6]
```

```
y =[99, 86, 87, 88, 100, 86, 103, 87, 94, 78, 77, 85, 86]
```

```
plt.scatter(x, y, c ="blue")
```

```
# To show the plot
```

```
plt.show()
```

12.b.

```
import matplotlib.pyplot as plt
```

```
# dataset-1
```

```
x1 = [89, 43, 36, 36, 95, 10, 66, 34, 38, 20]
```

```
y1 = [21, 46, 3, 35, 67, 95, 53, 72, 58, 10]
```

```
# dataset2
```

```
x2 = [26, 29, 48, 64, 6, 5, 36, 66, 72, 40]
```

```
y2 = [26, 34, 90, 33, 38, 20, 56, 2, 47, 15]
```

```
plt.scatter(x1, y1, c ="pink",linewidths = 2,marker ="s",edgecolor  
="green",s = 50)
```

```
plt.scatter(x2, y2, c ="yellow",linewidths = 2,marker ="^",edgecolor  
="red",s = 200)
```

G Pulla Reddy Engineering College (Autonomous), Kurnool
Electronics and Communication Engineering Department
DATA SCIENCE LAB (DSL(P))
Lab Component (Scheme-20)

```
plt.xlabel("&quot;X-axis&quot;")
```

```
plt.ylabel("&quot;Y-axis&quot;")
```

```
plt.show()
```

G Pulla Reddy Engineering College (Autonomous), Kurnool
Electronics and Communication Engineering Department
DATA SCIENCE LAB (DSL(P))
Lab Component (Scheme-20)

Program to visualize data using pie and bar graphs.

GPREC/ECE/DSL LAB /EXPT 09

DATE: 12/07/2023

Pie Graph:

```
import matplotlib.pyplot as plt
import numpy as np
```

```
plt.style.use('_mpl-gallery-nogrid')
```

```
# make data
```

```
x = [1, 2, 3, 4]
```

```
colors = plt.get_cmap('Blues')(np.linspace(0.2, 0.7, len(x)))
```

```
# plot
```

```
fig, ax = plt.subplots()
```

```
ax.pie(x, colors=colors, radius=3, center=(4, 4),
      wedgeprops={"linewidth": 1, "edgecolor": "white"}, frame=True)
```

```
ax.set(xlim=(0, 8), xticks=np.arange(1, 8),
      ylim=(0, 8), yticks=np.arange(1, 8))
```

```
plt.show()
```

Bar Graphs:

```
import matplotlib.pyplot as plt
```


G Pulla Reddy Engineering College (Autonomous), Kurnool
Electronics and Communication Engineering Department
DATA SCIENCE LAB (DSL(P))
Lab Component (Scheme-20)

```
import numpy as np
plt.style.use('_mpl-gallery')

# make data:
x = 0.5 + np.arange(8)
y = [4.8, 5.5, 3.5, 4.6, 6.5, 6.6, 2.6, 3.0]

# plot
fig, ax = plt.subplots()

ax.bar(x, y, width=1, edgecolor="white", linewidth=0.7)

ax.set(xlim=(0, 8), xticks=np.arange(1, 8),
       ylim=(0, 8), yticks=np.arange(1, 8))

plt.show()
```

G Pulla Reddy Engineering College (Autonomous), Kurnool
Electronics and Communication Engineering Department
DATA SCIENCE LAB (DSL(P))
Lab Component (Scheme-20)

Implement programs on Date and Time Data Types

GPREC/ECE/DSL LAB /EXPT 10

DATE: 12/07/2023

```
from datetime import datetime, date, time
```

```
# Current date and time
```

```
now = datetime.now()
```

```
print(now)
```

```
# Specific date and time
```

```
dt = datetime(2023, 7, 18, 15, 30)
```

```
print(dt)
```

```
# Extracting date and time components
```

```
print(dt.year)
```

```
print(dt.month)
```

```
print(dt.day)
```

```
print(dt.hour)
```

```
print(dt.minute)
```

```
print(dt.second)
```

```
# Formatting datetime as string
```

```
formatted = dt.strftime('%Y-%m-%d %H:%M:%S')
```

```
print(formatted)
```

```
# Parsing string to datetime
```

```
parsed = datetime.strptime('2023-07-18 15:30:00', '%Y-%m-%d %H:%M:%S')
```

G Pulla Reddy Engineering College (Autonomous), Kurnool
Electronics and Communication Engineering Department
DATA SCIENCE LAB (DSL(P))
Lab Component (Scheme-20)

```
print(parsed)
```

```
import datetime
```

```
current_datetime = datetime.datetime.now()  
print(current_datetime)
```

```
specific_datetime = datetime.datetime(2023, 7, 18, 12, 30, 0)  
print(specific_datetime)
```

```
date_string = "2023-07-18"  
parsed_date = datetime.datetime.strptime(date_string, "%Y-%m-%d")  
print(parsed_date)
```

```
formatted_date = specific_datetime.strftime("%Y-%m-%d %H:%M:%S")  
print(formatted_date)
```