A Laboratory Manual of

Data Visualization with Python

Table of Contents

Data Visualization with Python (BCS358D)

Data Visualization with Python		Semester	III
Course Code	**BCS358D**	CIE Marks	50
Teaching Hours/Week (L:T:P: S)	0: 0: 2: 0	SEE Marks	50
Credits	01	Exam Hours	100
Examination type (SEE)	Practical		

Course objectives:

- CLO 1. Demonstrate the use of IDLE or PyCharm IDE to create Python Applications
- CLO 2. Using Python programming language to develop programs for solving real-world problems
- CLO 3. Implementation of Matplotlib for drawing different Plots
- CLO 4. Demonstrate working with Seaborn, Bokeh.
- CLO 5. Working with Plotly for 3D, Time Series and Maps.

	Experiments
Sl. No.	***PART A – List of problems for which student should develop program and execute in theLaboratory***
1	a) Write a python program to find the best of two test average marks out of three test's marks accepted from the user. b) Develop a Python program to check whether a given number is palindrome or not andalso count the number of occurrences of each digit in the input number. Datatypes: https://www.youtube.com/watch?v=gCCVsvgR2KU Operators: https://www.youtube.com/watch?v=v5MR5JnKcZI Flow Control: https://www.youtube.com/watch?v=PqFKRqpHrjwFor loop: https://www.youtube.com/watch?v=0ZvaDa8eT5s While loop: https://www.youtube.com/watch?v=HZARImviDxg Exceptions: https://www.youtube.com/watch?v=6SPDvPK38tw
2	a) Defined as a function F as Fn = Fn-1 + Fn-2. Write a Python program which accepts a value for N (where N >0) as input and pass this value to the function. Display suitable error message if the condition for input value is not followed. b) Develop a python program to convert binary to decimal, octal to hexadecimal using functions. Functions:https://www.youtube.com/watch?v=BVfCWuca9nw Arguments:https://www.youtube.com/watch?v=ijXMGpoMkhQ Return value: https://www.youtube.com/watch?v=nuNXiEDnM44
3	a) Write a Python program that accepts a sentence and find the number of words, digits, uppercase letters and lowercase letters. b) Write a Python program to find the string similarity between two given strings **Sample Output:** Original string: Python Exercises Python Exercises Similarity between two said strings: **Sample Output:** Original string: Python Exercises Python Exercise Similarity between two said strings:1.0 0.967741935483871 Strings: https://www.youtube.com/watch?v=lSItwlnF0eU String functions: https://www.youtube.com/watch?v=9a3CxJyTq00

4	a) Write a Python program to Demonstrate how to Draw a Bar Plot using Matplotlib. b) Write a Python program to Demonstrate how to Draw a Scatter Plot using Matplotlib. https://www.youtube.com/watch?v=RRHQ6Fs1b8w&list=PLjVLYmrlmjGcC0B_FP3bkJ-JIPkV5GuZR&index=3 https://www.youtube.com/watch?v=7ABCuhWO9II&list=PLjVLYmrlmjGcC0B_FP3bkJ-JIPkV5GuZR&index=4
5	a) Write a Python program to Demonstrate how to Draw a Histogram Plot using Matplotlib. b) Write a Python program to Demonstrate how to Draw a Pie Chart using Matplotlib. https://www.youtube.com/watch?v=Qk7caotaQUQ&list=PLjVLYmrlmjGcC0B_FP3bkJ-JIPkV5GuZR&index=6 https://www.youtube.com/watch?v=PSji21jUNO0&list=PLjVLYmrlmjGcC0B_FP3bkJ-JIPkV5GuZR&index=7
6	a) Write a Python program to illustrate Linear Plotting using Matplotlib. b) Write a Python program to illustrate liner plotting with line formatting using Matplotlib. https://www.youtube.com/watch?v=UO98lJQ3QGI&list=PL-osiE80TeTvipOqomVEeZ1HRrcEvtZB_
7	Write a Python program which explains uses of customizing seaborn plots with Aesthetic functions. https://www.youtube.com/watch?v=6GUZXDef2U0
8	Write a Python program to explain working with bokeh line graph using Annotations and Legends. a) Write a Python program for plotting different types of plots using Bokeh. https://www.youtube.com/watch?v=HDvxYoRadcA
9	Write a Python program to draw 3D Plots using Plotly Libraries. https://www.youtube.com/watch?v=cCck7hCanpw&list=PLE50-dh6JzC4onX-qkv9H3HtPbBVA8M94&index=4
10	a) Write a Python program to draw Time Series using Plotly Libraries. b) Write a Python program for creating Maps using Plotly Libraries. https://www.youtube.com/watch?v=xnJ2TNrGYik&list=PLE50-dh6JzC4onX-qkv9H3HtPbBVA8M94&index=5 https://www.youtube.com/watch?v=D35m2CdMhVs&list=PLE50-dh6JzC4onX- qkv9H3HtPbBVA8M94&index=6

Python (Full Course): https://www.youtube.com/watch?v=_uQrJ0TkZlc

Pedagogy	For the above experiments the following pedagogy can be considered. Problem based learning, Active learning, MOOC, Chalk &Talk

Course outcomes (Course Skill Set):
At the end of the course the student will be able to:

CO 1. Demonstrate the use of IDLE or PyCharm IDE to create Python Applications
CO 2. Use Python programming constructs to develop programs for solving real-world problems
CO 3. Use Matplotlib for drawing different Plots
CO 4. Demonstrate working with Seaborn, Bokeh for visualization.
CO 5. Use Plotly for drawing Time Series and Maps.

Assessment Details (both CIE and SEE)

The weightage of Continuous Internal Evaluation (CIE) is 50% and for Semester End Exam (SEE) is 50%. The minimum passing mark for the CIE is 40% of the maximum marks (20 marks out of 50) and for the SEE minimum passing mark is 35% of the maximum marks (18 out of 50 marks). A student shall be deemed to have satisfied the academic requirements and earned the credits allotted to each subject/ course if the student secures a minimum of 40% (40 marks out of 100) in the sum total of the CIE (Continuous Internal Evaluation) and SEE (Semester End Examination) taken together

Continuous Internal Evaluation (CIE):

CIE marks for the practical course are **50 Marks**.
The split-up of CIE marks for record/ journal and test are in the ratio **60:40**.

- Each experiment is to be evaluated for conduction with an observation sheet and record write-up. Rubrics for the evaluation of the journal/write-up for hardware/software experiments are designed by the faculty who is handling the laboratory session and are made known to students at the beginning of the practical session.
- Record should contain all the specified experiments in the syllabus and each experiment write-up will be evaluated for 10 marks.
- Total marks scored by the students are scaled down to **30 marks** (60% of maximum marks).
- Weightage to be given for neatness and submission of record/write-up on time.
- Department shall conduct a test of 100 marks after the completion of all the experiments listed in the syllabus.
- In a test, test write-up, conduction of experiment, acceptable result, and procedural knowledge will carry a weightage of 60% and the rest 40% for viva-voce.
- The suitable rubrics can be designed to evaluate each student’s performance and learning ability.
- The marks scored shall be scaled down to **20 marks** (40% of the maximum marks).

The Sum of scaled-down marks scored in the report write-up/journal and marks of a test is thetotal CIE marks scored by the student.

Semester End Evaluation (SEE):

SEE marks for the practical course are 50 Marks.

SEE shall be conducted jointly by the two examiners of the same institute, examiners are appointed by the Head of the Institute.

- The examination schedule and names of examiners are informed to the university beforethe conduction of the examination. These practical examinations are to be conducted between the schedule mentioned in the academic calendar of the University.All

  laboratory experiments are to be included for practical examination.

  (Rubrics) Breakup of marks and the instructions printed on the cover page of the answerscript to be strictly adhered to by the examiners. **OR** based on the course requirement evaluation rubrics shall be decided jointly by examiners.

- Students can pick one question (experiment) from the questions lot prepared by theexaminers jointly.

- Evaluation of test write-up/ conduction procedure and result/viva will be conductedjointly by examiners.

- General rubrics suggested for SEE are mentioned here, writeup-20%, Conductionprocedure and result in -60%, Viva-voce 20% of maximum marks. SEE for practical shall be evaluated for 100 marks and scored marks shall be scaled down to 50 marks (however, based on course type, rubrics shall be decided by the examiners)

- Change of experiment is allowed only once and 15% of Marks allotted to the procedurepart are to be made zero.

The minimum duration of SEE is 02 hours

- **Weightage of marks for PART A is 80% and for PART B is 20%. General rubrics suggested to be followed for part A and part B.**
- **Change of experiment is allowed only once and Marks allotted to the procedure part to be made zero (Not allowed for Part B).**
- **The duration of SEE is 03 hours**

Rubrics suggested in Annexure-II of Regulation book

Textbooks:

1. **Al Sweigart, "Automate the Boring Stuff with Python",1stEdition, No Starch Press, 2015. (Available under CC-BY-NC-SA license at https://automatetheboringstuff.com/)**
2. **Reema Thareja "Python Programming Using Problem Solving Approach" Oxford University Press.**
3. **Allen B. Downey, "Think Python: How to Think Like a Computer Scientist", 2nd Edition, Green Tea Press, 2015. (Available under CC-BY-NC license at http://greenteapress.com/thinkpython2/thinkpython2.pdf)**

- **Jake VanderPlas "Python Data Science Handbook" 1st Edition, O'REILLY.**

Sample Lab Programs

1. Write a Python program to calculate the sum, product, division, multiplication, and modular division of two numbers entered by the user.

```
# Input the two numbers from the user
num1 = float(input("Enter the first number: "))
num2 = float(input("Enter the second number: "))
# Calculate the sum of the two numbers
sum_result = num1 + num2

# Calculate the product of the two numbers
product_result = num1 * num2

# Calculate the division of the two numbers
division_result = num1 / num2

# Calculate the multiplication of the two numbers
multiplication_result = num1 * num2

# Calculate the modulus (remainder) of the two numbers
modulus_result = num1 % num2

# Print the results
print("Sum:", sum_result)
print("Product:", product_result)
print("Division:", division_result)
print("Multiplication:", multiplication_result)
print("Modulus:", modulus_result)
```

2. Write a Python program to check if a given number is even or odd.

```
number = int(input("Enter a number: "))
if number % 2 == 0:
    print(number, "is even.")
else:
    print(number, "is odd.")
```

3. Write a Python program to convert temperature in Celsius to Fahrenheit.

```
celsius = float(input("Enter temperature in Celsius: "))
fahrenheit = (celsius * 9/5) + 32
print("Temperature in Fahrenheit:", fahrenheit)
```

4. Write a Python program to calculate the factorial of a given number.

```
number = int(input("Enter a number: "))
factorial = 1
if number < 0:
    print("Factorial cannot be calculated for negative numbers.")
elif number == 0:
    print("The factorial of 0 is 1.")
else:
    for i in range(1, number + 1):
        factorial *= i
print("The factorial of", number, "is", factorial)
```

5. Write a Python program to find the area of a triangle given its base and height.

```
base = float(input("Enter the base of the triangle: "))
height = float(input("Enter the height of the triangle: "))
area = (base * height) / 2
print("The area of the triangle is:", area)
```

6. Write a Python Program to find the simple interest.

```
# Input the principal amount, interest rate, and time period from the user
principal = float(input("Enter the principal amount: "))
```

```
rate = float(input("Enter the interest rate: "))
time = float(input("Enter the time period (in years): "))

# Calculate the simple interest
interest = (principal * rate * time) / 100

# Print the result
print("The simple interest is:", interest)
```

7. Write a python program to swap two numbers using all 3 approaches.

Approach 1 : Using a temporary variable

```
# Input the two numbers from the user
num1 = float(input("Enter the first number: "))
num2 = float(input("Enter the second number: "))

# Swap the numbers using a temporary variable
temp = num1
num1 = num2
num2 = temp

# Print the swapped numbers
print("After swapping:")
print("First number:", num1)
print("Second number:", num2)
```

Approach 2: Using arithmetic operations.

```
# Input the two numbers from the user
num1 = float(input("Enter the first number: "))
num2 = float(input("Enter the second number: "))

# Swap the numbers using arithmetic operations
num1 = num1 + num2
num2 = num1 - num2
num1 = num1 - num2
```

```
# Print the swapped numbers
print("After swapping:")
print("First number:", num1)
print("Second number:", num2)
```

Approach 3: Using multiple assignment

```
# Input the two numbers from the user
num1 = float(input("Enter the first number: "))
num2 = float(input("Enter the second number: "))

# Swap the numbers using multiple assignment
num1, num2 = num2, num1

# Print the swapped numbers
print("After swapping:")
print("First number:", num1)
print("Second number:", num2)
```

8. Write a program to find the sum of n natural numbers.

Approach 1:

```
# Input the value of n from the user
n = int(input("Enter a positive integer: "))

# Calculate the sum of the first n natural numbers
sum_natural = (n * (n + 1)) // 2

# Print the result
print("The sum of the first", n, "natural numbers is:", sum_natural)
```

Approach 2:

```
# Input the value of n from the user
n = int(input("Enter a positive integer: "))

# Initialize variables
sum_natural_numbers = 0
count = 1
```

```
# Calculate the sum of the first n natural numbers using a while loop
while count <= n:
    sum_natural_numbers += count
    count += 1

# Print the result
print("The sum of the first", n, "natural numbers is:", sum_natural_numbers)
```

Approach 3:

```
# Input the value of n from the user
n = int(input("Enter a positive integer: "))

# Initialize variable
sum_natural_numbers = 0

# Calculate the sum of the first n natural numbers using a for loop
for i in range(1, n + 1):
    sum_natural_numbers += i

# Print the result
print("The sum of the first", n, "natural numbers is:", sum_natural_numbers)
```

Laboratory Program 1:

1) **A)** Write a python program to find the best of two test average marks out of three test’s marks accepted from the user.

Source Code:

```
mark1 = int(input("Enter the marks in the first test:"))
mark2 = int(input("Enter the marks in second test: "))
mark3 = int(input("Enter the marks in third test: "))

if (mark1 > mark2):
    if (mark2  >  mark3):
        total_marks = mark1 + mark2
    else:
        total_marks = mark1 + mark3
elif (mark1 > mark3):
    total_marks = mark1 + mark2
else:
    total_marks = mark2 + mark3

Average_marks = total_marks / 2
print("The average of the best two test marks is:", Average_marks)
```

Sample Output:

```
Enter the marks in the first test:18
Enter the marks in second test: 19
Enter the marks in third test: 24
The average of the best two test marks is: 21.5
```

1) **B)** Develop a Python program to check whether a given number is palindrome or not and also count the number of occurrences of each digit in the input number.

Source Code:

```
num = int(input("Enter a number: "))
temp1 = num
reverse  = 0
while temp1 > 0:
    remainder = temp1 % 10
    reverse = (reverse * 10) + remainder
    temp1 = temp1 // 10
if num == reverse:
  print('The Entered Number %d is Palindrome'%(num))
else:
  print("The Entered Number %d  is Not a Palindrome"%(num))
print("Digit\tFrequency")
for i in range(0,10):
    count=0
    temp2=num
    while temp2>0:
        digit=temp2%10
        if digit==i:
            count=count+1
        temp2=temp2//10
    if count>0:
print(i,"\t",count)
```

Sample Output:

```
Enter a number: 3663
The Entered Number 3663 is Palindrome
Digit     Frequency
3    2
6    2
```

Laboratory Program 2:

2)A) Defined as a function F as Fn = Fn-1 + Fn-2. Write a Python program which accepts a value for N (where N >0) as input and pass this value to the function. Display suitable error message if the condition for input value is not followed.

Source Code:

```
# Function for nth Fibonacci number
def Fibonacci(n):
   if n == 1:
      return 0
   elif n==2:
      return 1
   else:
      return Fibonacci(n-1) + Fibonacci(n-2)

count = 1
N = int(input("Enter a value of N : "))
if N<=0:
   print("Invalid Input: Enter a value of N (>0)")
else:
   print("Fibonacci sequence : ")
   while count <= N:
     print(Fibonacci(count),end=' ')
    count = count + 1
```

Sample Output:

```
Enter a value of N: 10

Fibonacci  sequence:

0 1 1 2 3 5 8 13 21 34
```

2)B) Develop a python program to convert binary to decimal, octal to hexadecimal using functions.

Source Code:

```
def binary_to_decimal(binary):
    decimal = 0
    power = 0
    while binary != 0:
        last_digit = binary % 10
        decimal += last_digit * (2 ** power)
        binary //= 10
        power += 1
    return decimal

def octal_to_hexadecimal(octal):
    decimal = 0
    power = 0
    while octal > 0:
        digit = octal % 10
        decimal += digit * (8 ** power)
        octal //= 10
        power += 1
    conversion_table = {10: 'A', 11: 'B', 12: 'C', 13: 'D', 14: 'E', 15: 'F'}
    hexadecimal = ""
    while (decimal > 0):
        remainder = decimal % 16
        if remainder >= 10:
            hexadecimal = conversion_table[remainder] + hexadecimal
        else:
            hexadecimal = str(remainder) + hexadecimal
        decimal = decimal // 16
    return hexadecimal
binary = int(input("Enter a binary number: "))
decimal = binary_to_decimal(binary)
print("Decimal equivalent:", decimal)
```

```
octal = int(input("Enter an octal number: "))
hexadecimal = octal_to_hexadecimal(octal)
print("Hexadecimal equivalent:", hexadecimal)
```

Sample Output:

```
Enter a binary number: 100

Decimal equivalent: 4

Enter an octal number: 7302

Hexadecimal equivalent: EC2
```

Laboratory Program 3:

3)A) Write a Python program that accepts a sentence and find the number of words, digits, uppercase letters and lowercase letters.

Source Code:

```
sentence = input("Enter a sentence: ")
(words, digits, upper, lower) = (0, 0, 0, 0)
l_w = sentence.split()
words = len(l_w)
for ch in sentence:
    if ch.isdigit():
        digits = digits + 1
    elif ch.isupper():
        upper = upper + 1
    elif ch.islower():
        lower = lower + 1

print ("No of Words: ", words)
print ("No of Digits: ", digits)
print ("No of Uppercase letters: ", upper)
print ("No of Lowercase letters: ", lower)
```

Sample Output:

```
Enter a sentence: SDMIT Ujire 574240
No of Words:  3
No of Digits:  6
No of Uppercase letters: 6
No of Lowercase letters:  4
```

3)B) Write a Python program to find the string similarity between two given strings

Source Code:

```
def similarity(str1, str2):
    len1 = len(str1)
    len2 = len(str2)
    max_len = max(len1, len2)
    common_chars = 0
    for i in range(max_len):
        if i < len1 and i < len2 and str1[i] == str2[i]:
            common_chars += 1
    return common_chars / max_len
str1 = input("Enter the first string: ")
str2 = input("Enter the second string: ")

print("Original string1:\n", str1)
print("Second string:\n",str2)

print("Similarity between two said strings:")
print(similarity(str1, str2))
```

Sample Output:

```
Enter the first string: Python Exercises
Enter the second string: Python Exercises
Original string1:
 Python Exercises
Second string:
 Python Exercises
Similarity between two said strings:
1.0

Enter the first string: Python Exercises
Enter the second string: Python Exercise
Original string1:
 Python Exercises
Second string:
 Python Exercise
Similarity between two said strings:
0.9375
```

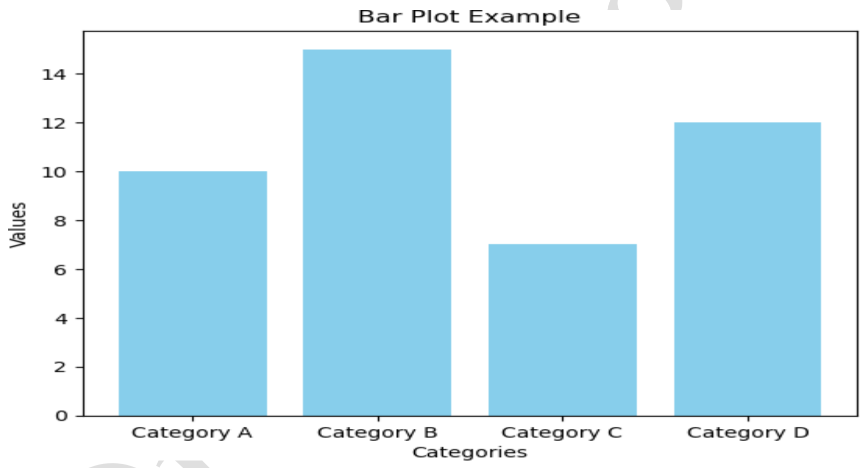
Laboratory Program 4:

4) A) Write a Python program to Demonstrate how to Draw a Bar Plot using Matplotlib.

Source Code:

```
import matplotlib.pyplot as plt
# Sample data
categories = ['Category A', 'Category B', 'Category C', 'Category D']
values = [10, 15, 7, 12]

# Create a bar plot
plt.bar(categories, values, color='skyblue')

# Adding labels and title
plt.xlabel('Categories')
plt.ylabel('Values')
plt.title('Bar Plot Example')

# Display the plot
plt.show()
```

Sample Output:



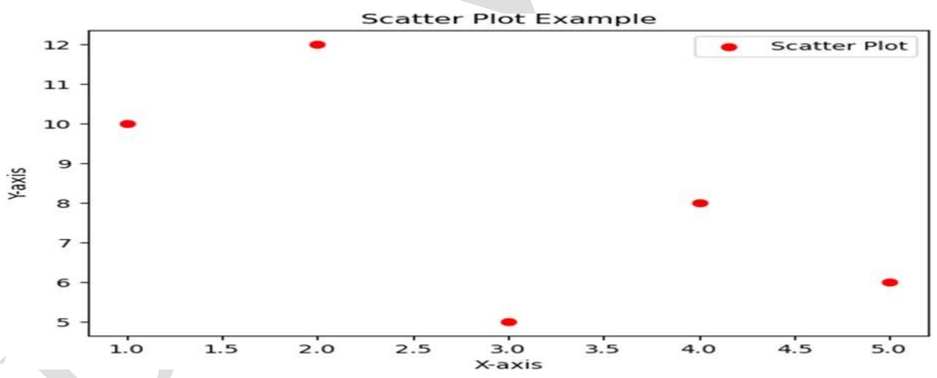
Note: In this example, we import Matplotlib, create a list of categories and their corresponding values, and then use the plt.bar() function to create a bar plot. We set the color of the bars to 'skyblue' for illustration. You can change the color or other plot properties as needed.
Finally, we add labels and a title to the plot and display it using plt.show(). When you run this script, it will generate a simple bar plot with the provided data.
Make sure you have Matplotlib installed and replace the sample data with your own data for a customized bar plot

4) B) Write a Python program to Demonstrate how to Draw a Scatter Plot using Matplotlib.

Source Code:

```
import matplotlib.pyplot as plt
# Sample data
x = [1, 2, 3, 4, 5]
y = [10, 12, 5, 8, 6]

# Create a scatter plot
plt.scatter(x, y, label='Scatter Plot', color='red', marker='o')

# Adding labels and title
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Scatter Plot Example')

# Display the legend (if multiple data sets are used)
plt.legend()

# Display the plot
plt.show()
```

Sample Output:



Note: In this example, we import Matplotlib, create two lists, x and y, representing the data points, and then use the plt.scatter() function to create a scatter plot. We set the color of the markers to 'red' and the marker style to 'o' (circle) for illustration. You can change the color, marker style, or other plot properties as needed. Finally, we add labels, a title, and a legend (if you have multiple data sets) to the plot and display it using plt.show(). When you run this script, it will generate a simple scatter plot with the provided data.

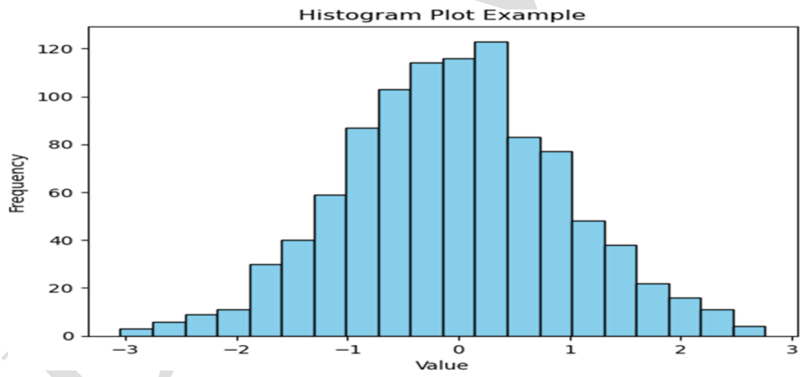
Laboratory Program 5:

5)A) Write a Python program to Demonstrate how to Draw a Histogram Plot using Matplotlib.

Source Code:

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

# Generate random data for demonstration
np.random.seed(0)
data = np.random.randn(1000)  # Generate 1000 random data points

# Create a histogram
plt.hist(data, bins=20, color='skyblue', edgecolor='black')

# Adding labels and title
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.title('Histogram Plot Example')

# Display the plot
plt.show()
```

Sample Output:



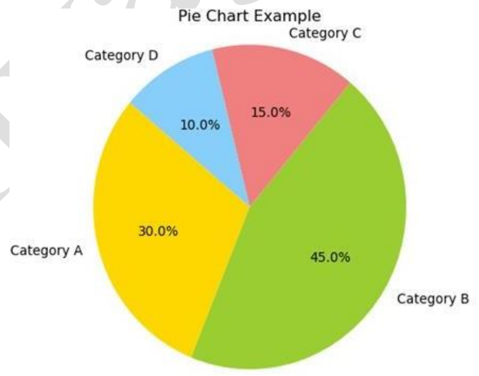
Note: In this example, we import Matplotlib and NumPy, generate random data using NumPy's random.randn() function, and then use the plt.hist() function to create a histogram plot. We set the number of bins to 20 for illustration. You can change the number of bins, color, edgecolor, or other plot properties as needed.
Finally, we add labels and a title to the plot and display it using plt.show(). When you run this script, it will generate a simple histogram plot with the provided data.

5)B) Write a Python program to Demonstrate how to Draw a Pie Chart using Matplotlib.

Source Code:

```
import matplotlib.pyplot as plt

# Sample data
labels = ['Category A', 'Category B', 'Category C', 'Category D']
sizes = [30, 45, 15, 10]
colors = ['gold', 'yellowgreen', 'lightcoral', 'lightskyblue']

# Create a pie chart
plt.pie(sizes, labels=labels, colors=colors, autopct='%1.1f%%', startangle=140)

# Adding a title
plt.title('Pie Chart Example')

# Display the plot
plt.axis('equal')  # Equal aspect ratio ensures that the pie is drawn as a circle.
plt.show()
```

Sample Output:



Note: In this example, we import Matplotlib and create three lists: labels (for category labels), sizes (for the sizes or proportions of each category), and colors (for the colors of each segment of the pie chart). We use the plt.pie() function to create the pie chart, and the autopct parameter is used to display the percentage labels on each slice. The startangle parameter rotates the chart to start at the specified angle.
Finally, we add a title, ensure the plot has an equal aspect ratio (to make it a circle), and display the pie chart using plt.show(). When you run this script, it will generate a simple pie chart with the provided data.

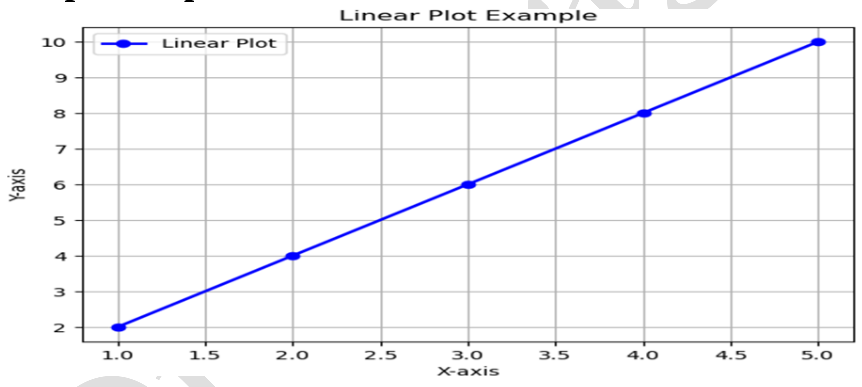
Laboratory Program 6:

6) A) Write a Python program to illustrate Linear Plotting using Matplotlib.

Source Code:

```
import matplotlib.pyplot as plt
# Sample data
x = [1, 2, 3, 4, 5]
y = [2, 4, 6, 8, 10]
# Create a line plot
plt.plot(x, y, label='Linear Plot', color='blue', marker='o', linestyle='-', linewidth=2)
# Adding labels and title
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Linear Plot Example')

# Display the legend
plt.legend()

# Display the plot
plt.grid(True)
plt.show()
```

Sample Output:



Note: In this example, we import Matplotlib, create two lists, x and y, representing data points, and then use the plt.plot() function to create a line plot. We set the color of the line to 'blue', the marker style to 'o' (circle), the linestyle to '-', and the linewidth to 2 for illustration. You can change these properties as needed.

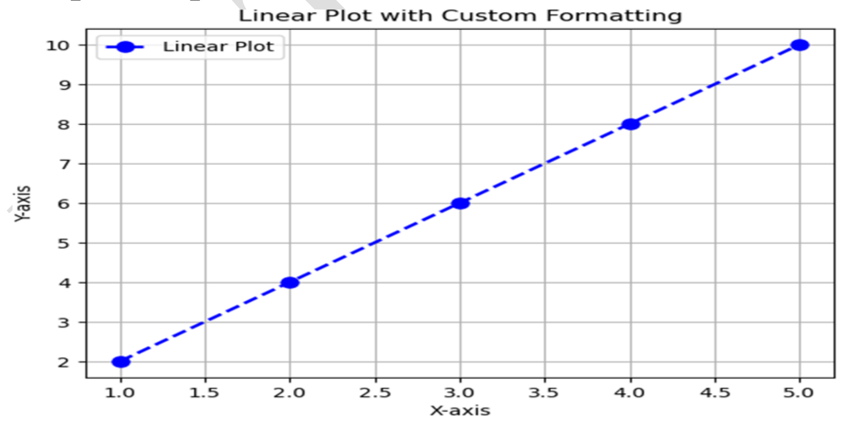
Finally, we add labels, a title, a legend (if you have multiple data sets), enable grid lines, and display the plot using plt.show(). When you run this script, it will generate a simple linear plot with the provided data.

6)B) Write a Python program to illustrate liner plotting with line formatting using Matplotlib.

Source Code:

```
import matplotlib.pyplot as plt

# Sample data
x = [1, 2, 3, 4, 5]
y = [2, 4, 6, 8, 10]

# Create a line plot with custom formatting
plt.plot(x, y, label='Linear Plot', color='blue', marker='o', linestyle='--', linewidth=2, markersize=8)

# Adding labels and title
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Linear Plot with Custom Formatting')

# Display the legend
plt.legend()

# Display the plot
plt.grid(True)
plt.show()
```

Sample Output:



Note: In this example, we use the plt.plot() function to create a line plot with custom line formatting. Here's a breakdown of the custom formatting options used:

color='blue': Sets the color of the line to blue.
linestyle='--': Sets the line style to dashed.
linewidth=2: Sets the line width to 2.
marker='o': Sets the marker style to a circle.
markersize=8: Sets the marker size to 8.

You can adjust these formatting options according to your preferences. Finally, we add labels, a title, a legend (if needed), enable grid lines, and display the plot using plt.show(). When you run this script, it will generate a linear plot with the specified

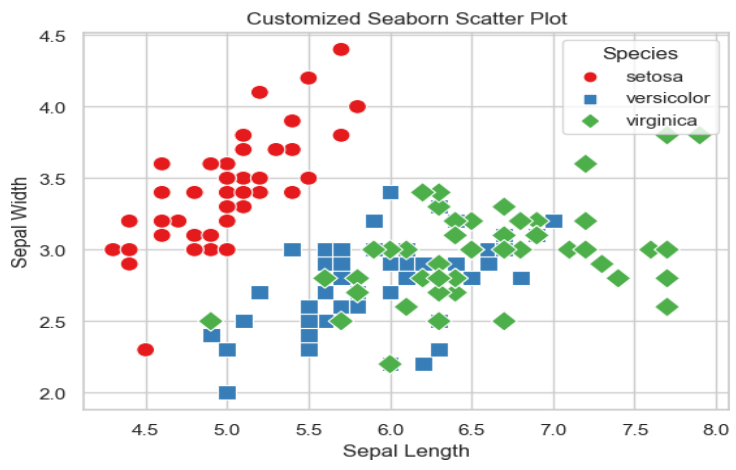
Laboratory Program 7:

7) Write a Python program which explains uses of customizing seaborn plots with Aesthetic functions.

Source Code:

```
import seaborn as sns
import matplotlib.pyplot as plt

# Load a sample dataset (Iris dataset)
iris = sns.load_dataset("iris")

# Create a scatter plot using Seaborn
sns.set(style="whitegrid")  # Set the style for the plot

# Customize the plot aesthetics
sns.scatterplot(x="sepal_length", y="sepal_width", data=iris, hue="species", palette="Set1",
style="species", s=100, markers=["o", "s", "D"])

# Adding labels and a title
plt.xlabel("Sepal Length")
plt.ylabel("Sepal Width")
plt.title("Customized Seaborn Scatter Plot")

# Display the legend
plt.legend(title="Species")

# Display the plot
plt.show()
```

Sample Output:



Note: In this example, we use the Iris dataset, and we create a scatter plot with customized aesthetics using Seaborn. Here's an explanation of the aesthetic functions used:

sns.set(style="whitegrid"): Sets the style of the plot. Seaborn provides several built-in styles like "whitegrid," "darkgrid," etc.

sns.scatterplot(): Creates a scatter plot. We customize it with the following options:

x and y: The data columns for the x and y axes.
hue: Colors the points based on the "species" column.
palette: Specifies the color palette to use.
style: Distinguishes data points based on the "species" column.
s: Sets the marker size.
markers: Specifies marker styles for each "species."
plt.xlabel(), plt.ylabel(), plt.title(): Adds labels and a title to the plot.

plt.legend(): Displays a legend with a title.

When you run this script, it will generate a customized Seaborn scatter plot using the Iris dataset. You can adjust the customization options and use different datasets for your specific needs. Seaborn provides many other aesthetic functions to further customize your plots, including options for color palettes, fonts, and more.

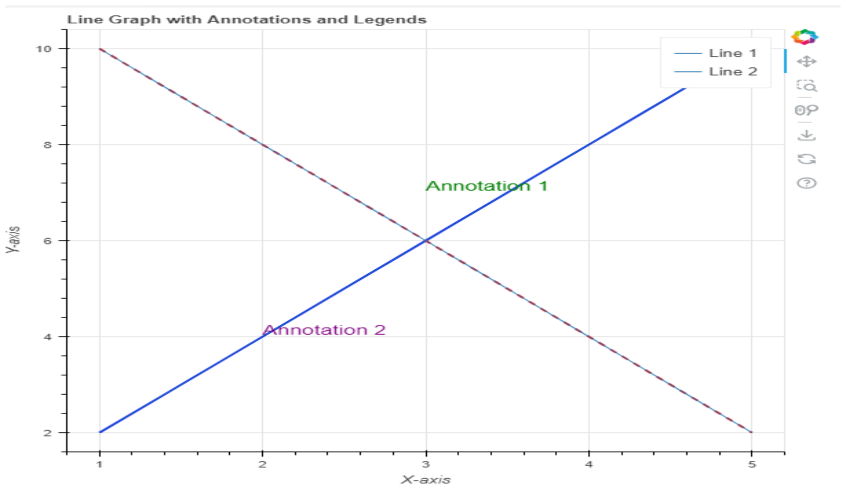
Laboratory Program 8:

8) Write a Python program to explain working with bokeh line graph using Annotations and Legends.

Source Code:

```
from bokeh.plotting import figure, show, output_file
from bokeh.models import Label, Legend

# Sample data
x = [1, 2, 3, 4, 5]
y1 = [2, 4, 6, 8, 10]
y2 = [10, 8, 6, 4, 2]

# Create a new plot
p = figure(title='Line Graph with Annotations and Legends', x_axis_label='X-axis',
y_axis_label='Y-axis')

# Add lines to the plot
p.line(x, y1, legend_label='Line 1', line_color='blue', line_width=2)
p.line(x, y2, legend_label='Line 2', line_color='red', line_width=2, line_dash='dashed')

# Add annotations
annotation1 = Label(x=3, y=7, text='Annotation 1', text_font_size='12pt', text_color='green')
annotation2 = Label(x=2, y=4, text='Annotation 2', text_font_size='12pt',
text_color='purple')

p.add_layout(annotation1)
p.add_layout(annotation2)

# Add a legend
legend = Legend(items=[('Line 1', [p.line(x, y1)]), ('Line 2', [p.line(x, y2)])])
p.add_layout(legend)

# Specify the output file
output_file('line_graph_with_annotations_and_legends.html')

# Show the plot
show(p)
```

Sample Output:



Note: In this example:

We import the necessary modules from Bokeh.
We define sample data for x and y values.
We create a new plot using figure.
We add two lines to the plot using line.
We add annotations using Label.
We create a legend using Legend.
The Legend allows us to create a legend by specifying items with labels and corresponding renderers.

Finally, we specify the output file and display the plot using show.

Run this script, and it will generate an interactive line graph with annotations and legends, saved in an HTML file. You can open the HTML file in a web browser to interact with the plot.

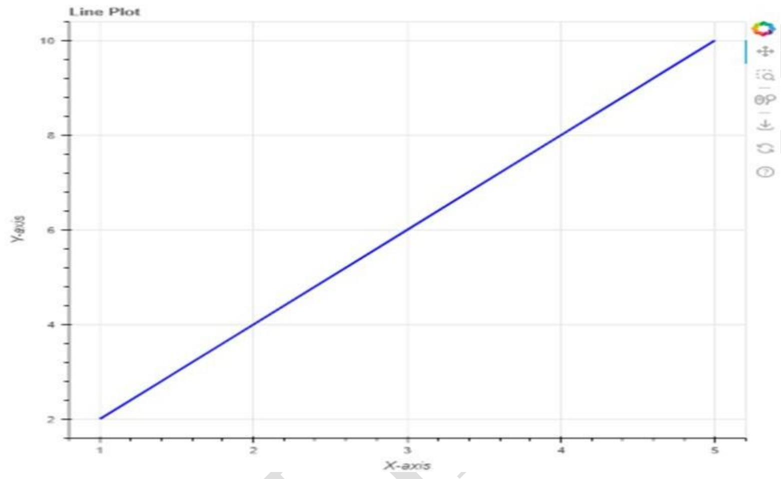
8)A) Write a Python program for plotting different types of plots using Bokeh.

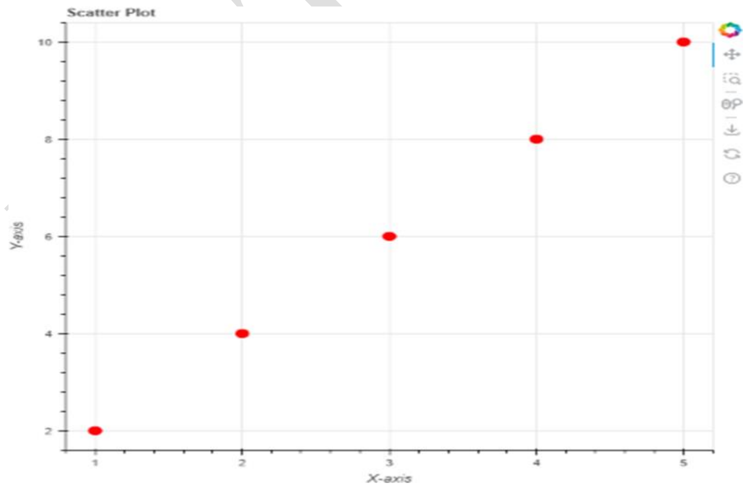
Source Code:

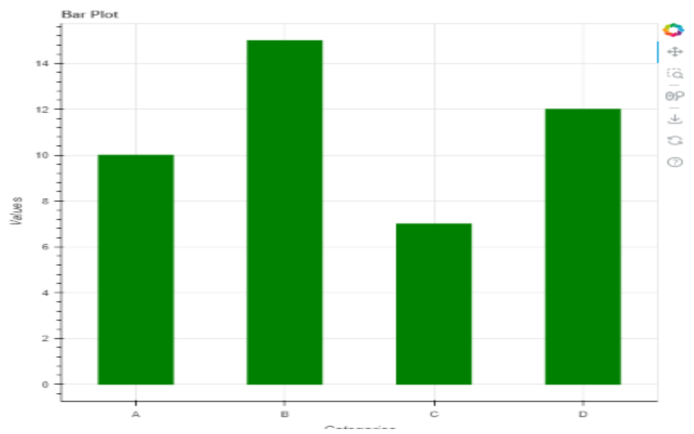
```
from bokeh.plotting import figure, show
from bokeh.models import ColumnDataSource
from bokeh.io import output_notebook
from math import pi
import pandas as pd
from bokeh.transform import cumsum

# Output to Jupyter Notebook
output_notebook()

# Sample data
x = [1, 2, 3, 4, 5]
y = [2, 4, 6, 8, 10]

# Line Plot
line_plot = figure(title="Line Plot", x_axis_label="X-axis", y_axis_label="Y-axis")
line_plot.line(x, y, line_color="blue", line_width=2)
show(line_plot)

# Scatter Plot
scatter_plot = figure(title="Scatter Plot", x_axis_label="X-axis", y_axis_label="Y-axis")
scatter_plot.circle(x, y, size=10, color="red")
show(scatter_plot)

# Bar Plot
data = {'Categories': ['A', 'B', 'C', 'D'], 'Values': [10, 15, 7, 12]}
source = ColumnDataSource(data=data)
bar_plot = figure(x_range=data['Categories'], title="Bar Plot", x_axis_label="Categories",
y_axis_label="Values")
bar_plot.vbar(x='Categories', top='Values', source=source, width=0.5, color="green")
show(bar_plot)
```

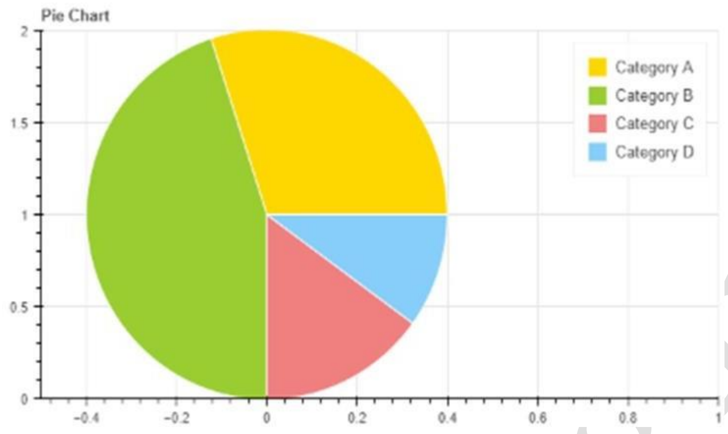
```
# Pie Chart
data = {'Categories': ['Category A', 'Category B', 'Category C', 'Category D'], 'Values': [30,
45, 15, 10]}
df = pd.DataFrame(data)
df['angle'] = df['Values']/df['Values'].sum() * 2*pi
df['color'] = ["gold", "yellowgreen", "lightcoral", "lightskyblue"]
pie_chart = figure(height=350, title="Pie Chart", toolbar_location=None,
tools="hover", tooltips="@Categories: @Values", x_range=(-0.5, 1.0))

pie_chart.wedge(x=0, y=1, radius=0.4, start_angle=cumsum('angle', include_zero=True),
end_angle=cumsum('angle'), line_color="white", fill_color='color',
legend_field='Categories', source=df)

show(pie_chart)
```

Sample Output:






Bar Plot
Values
0
2
4
6
8
10
12
14
A
B
C
D
Categories


Pie Chart
Category A
Category B
Category C
Category D
0
0.5
1
1.5
2
−0.4
−0.2
0
0.2
0.4
0.6
0.8
1

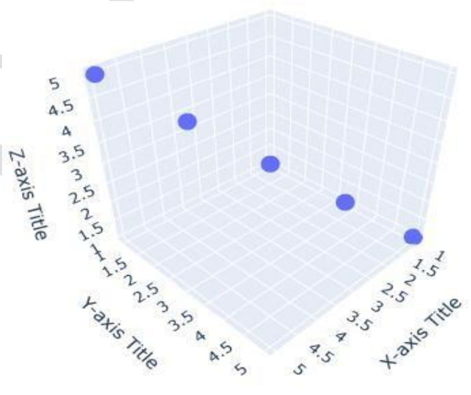
Laboratory Program 9:

Write a Python program to draw 3D Plots using Plotly Libraries.

Source Code:

```
import plotly.express as px
# Sample data
import pandas as pd
data = pd.DataFrame({'X': [1, 2, 3, 4, 5],
                     'Y': [5, 4, 3, 2, 1],
                     'Z': [1, 2, 3, 4, 5]})

# Create a 3D scatter plot
fig = px.scatter_3d(data, x='X', y='Y', z='Z')

# Customize the plot
fig.update_layout(
    scene=dict(
        xaxis_title='X-axis Title',
        yaxis_title='Y-axis Title',
        zaxis_title='Z-axis Title'
    )
)
# Show the plot
fig.show()
```

Sample Output:



Note: In this example, we first create some sample data as a Pandas DataFrame, and then use Plotly Express to create a 3D scatter plot. You can customize the plot further by updating the layout, axis titles, and other properties.
Make sure to replace the sample data and axis titles with your own data and labels as needed. You can also create various types of 3D plots (e.g., surface plots, line plots) using Plotly, with slight modifications to the code and by using different functions provided by the library.

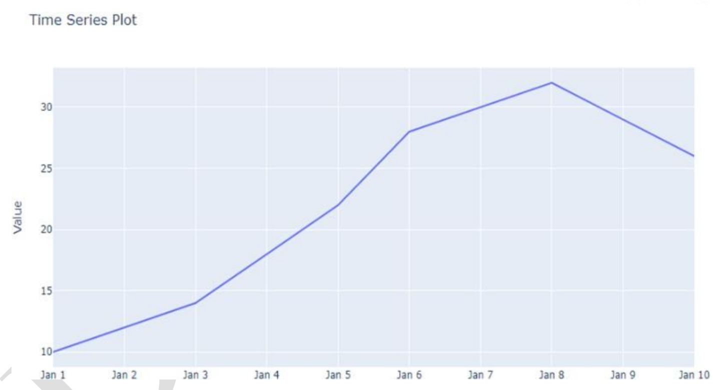
Laboratory Program 10:

10)A) Write a Python program to draw Time Series using Plotly Libraries.

Source Code:

```
import plotly.express as px
import pandas as pd
# Sample time series data
data = pd.DataFrame({
    'Date': pd.date_range(start='2023-01-01', periods=10, freq='D'),
    'Value': [10, 12, 14, 18, 22, 28, 30, 32, 29, 26]
})
# Create a time series plot
fig = px.line(data, x='Date', y='Value', title='Time Series Plot')
# Customize the plot
fig.update_xaxes(title_text='Date')
fig.update_yaxes(title_text='Value')

# Show the plot
fig.show()
```

Sample Output:



Note: In this example, we generate some sample time series data with dates and values. We use Plotly Express to create a line plot representing the time series. You can customize the plot by updating the axis titles and other layout properties.

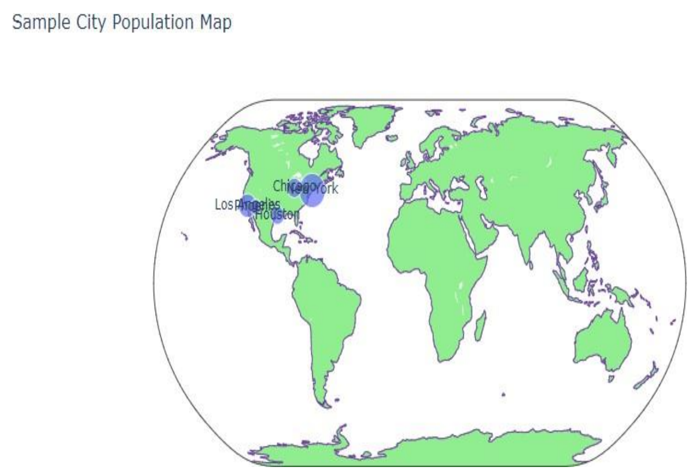
Make sure to replace the sample data with your own time series data. You can also add multiple lines, markers, and other features to your time series plot using Plotly's features and functions.

10) B) Write a Python program for creating Maps using Plotly Libraries.

Source Code:

```
import plotly.express as px

# Sample data
import pandas as pd
data = pd.DataFrame({
    'City': ['New York', 'Los Angeles', 'Chicago', 'Houston', 'Phoenix'],
    'Lat': [40.7128, 34.0522, 41.8781, 29.7604, 33.4484],
    'Lon': [-74.0060, -118.2437, -87.6298, -95.3698, -112.0740],
    'Population': [8398748, 3990456, 2716000, 2320255, 1680992]
})

# Create a map
fig = px.scatter_geo(data, lat='Lat', lon='Lon', text='City', size='Population',
                     projection="natural earth", title='Sample City Population Map')

# Customize the map
fig.update_geos(
    showcoastlines=True,
    coastlinecolor="RebeccaPurple",
    showland=True,
    landcolor="LightGreen",
)

# Show the map
fig.show()
```

Sample Output:



References

1. https://www.python.org/
2. https://www.anaconda.com/download
3. https://www.youtube.com/watch?v=gCCVsvgR2KU
4. https://www.youtube.com/watch?v=v5MR5JnKcZI
5. https://www.youtube.com/watch?v=PqFKRqpHrjw
6. https://www.youtube.com/watch?v=0ZvaDa8eT5s
7. https://www.youtube.com/watch?v=HZARImviDxg
8. https://www.youtube.com/watch?v=6SPDvPK38tw
9. https://www.youtube.com/watch?v=BVfCWuca9nw
10. https://www.youtube.com/watch?v=ijXMGpoMkhQ
11. https://www.youtube.com/watch?v=nuNXiEDnM44
12. https://www.youtube.com/watch?v=lSItwlnF0eU
13. https://www.youtube.com/watch?v=9a3CxJyTq00
14. https://www.youtube.com/watch?v=RRHQ6Fs1b8w&list=PLjVLYmrlmjGcC0B_FP3bkJ-JIPkV5GuZR&index=4
15. https://www.youtube.com/watch?v=7ABCuhWO9II&list=PLjVLYmrlmjGcC0B_FP3bkJ-JIPkV5GuZR&index=5
16. https://www.youtube.com/watch?v=Qk7caotaQUQ
17. https://www.youtube.com/watch?v=Qk7caotaQUQ&list=PLjVLYmrlmjGcC0B_FP3bkJJIPkV5GuZR&index=6
18. https://www.youtube.com/watch?v=PSji21jUNO0&list=PLjVLYmrlmjGcC0B_FP3bkJJIPkV5GuZR&index=7
19. https://www.youtube.com/watch?v=UO98lJQ3QGI&list=PL-osiE80TeTvipOqomVEeZ1HRrcEvtZB_
20. https://www.youtube.com/watch?v=6GUZXDef2U0
21. https://www.youtube.com/watch?v=HDvxYoRadcA
22. https://www.youtube.com/watch?v=cCck7hCanpw&list=PLE50-dh6JzC4onXqkv9H3HtPbBVA8M94&index=4
23. https://www.youtube.com/watch?v=xnJ2TNrGYik&list=PLE50-dh6JzC4onXqkv9H3HtPbBVA8M94&index=5
24. https://www.youtube.com/watch?v=D35m2CdMhVs&list=PLE50-dh6JzC4onXqkv9H3HtPbBVA8M94&index=6

Viva Questions

Lab 1a: Write a python program to find the best of two test average marks out of three test's marks accepted from the user.

1. How does your program accept user input for the three test marks?
2. What steps does your program take to calculate the average marks for each test?
3. How does your program determine the best of the two test averages?
4. Could you explain the use of any conditional statements or loops in your program?
5. How would your program handle invalid inputs, such as non-numeric characters or negative values for test scores?

Lab 1b: Develop a Python program to check whether a given number is palindrome or not and also count the number of occurrences of each digit in the input number.

1. How does your program determine whether the given number is a palindrome or not?
2. Could you explain the approach your program uses to count the occurrences of each digit in the input number?
3. How does your program handle potential errors or invalid inputs, such as non-numeric characters or negative values?
4. Does your program utilize any data structures or specific techniques to count the occurrences of digits efficiently?
5. What optimizations, if any, have you implemented to enhance the efficiency of your program, especially in counting the occurrences of digits?

Lab 2a: Defined as a function F as Fn = Fn-1 + Fn-2. Write a Python program which accepts a value for N (where N >0) as input and pass this value to the function. Display suitable error message if the condition for input value is not followed.

1. Could you explain the concept behind the function F, where Fn = Fn-1 + Fn-2?
2. How does your Python program verify and handle the condition for the input value (N) to ensure it's greater than zero?
3. Could you explain the process within your program that passes the user-provided value (N) to the defined function F?
4. Does your program utilize any iterative or recursive methods to calculate the function Fn = Fn-1 + Fn-2?
5. How does your program ensure the display of an appropriate error message when the input value does not satisfy the condition N > 0?

Lab 2b: **Develop a python program to convert binary to decimal, octal to hexadecimal using functions.**

1. Can you explain the specific functions you've created for converting binary to decimal and octal to hexadecimal?
2. How does your program validate the input received from the user to ensure it represents a valid binary or octal number before performing the conversion?
3. Could you describe the algorithms or mathematical operations used within the functions to perform the conversions accurately?
4. How does your program manage potential errors or unexpected inputs during the conversion process?
5. Do your functions handle large binary numbers or octal values effectively, or are there limitations to the size of the numbers they can process?

Lab 3a: **Write a Python program that accepts a sentence and find the number of words, digits, uppercase letters and lowercase letters**

1. How does your program determine the number of words in the provided sentence?
2. Could you explain the process your program follows to identify and count the number of digits within the sentence?
3. What approach does your program take to calculate the count of uppercase letters in the sentence?
4. How does your program differentiate between lowercase and uppercase letters while counting their occurrences?
5. Does your program account for potential edge cases or unexpected inputs, such as special characters or punctuation within the sentence?

Lab 3b: **Write a Python program to find the string similarity between two given strings**

1. What specific metric or approach does your program use to calculate the similarity between two strings?
2. Can you explain how your program handles variations in string lengths when computing their similarity?
3. Could you discuss the types of string similarity measurements or algorithms your program is utilizing?
4. How does your program account for cases where the strings have similar content but different structures or order of words?
5. Does your program consider the impact of case sensitivity or different character types (such as Unicode characters) in determining string similarity?

Lab 4a: Write a Python program to Demonstrate how to Draw a Bar Plot using Matplotlib.

1. How does your program utilize Matplotlib to create a bar plot in Python?
2. Could you explain the data representation used in your program for creating the bar plot?
3. What customization options or attributes does your program employ to enhance the visualization of the bar plot (e.g., colors, labels, axes formatting)?
4. Does your program allow for multiple bars, grouped bars, or stacked bars in the plot? If so, how does it achieve this functionality?
5. How does your program handle potential errors or unexpected data types while creating the bar plot?

Lab 4b: Write a Python program to Demonstrate how to Draw a Scatter Plot using Matplotlib.

1. How does your program use Matplotlib to create a scatter plot in Python?
2. What type of data structure does your program utilize for generating the scatter plot?
3. Could you explain the customization options applied in your program to enhance the visual aspects of the scatter plot (e.g., marker styles, colors, labels, axis formatting)?
4. Does your program handle additional dimensions in the scatter plot, such as size or color of markers to represent another variable? If so, how is this accomplished?
5. How does your program address potential issues or unexpected data types when creating the scatter plot?

Lab 5a: Write a Python program to Demonstrate how to Draw a Histogram Plot using Matplotlib.

1. How does your program utilize Matplotlib to create a histogram plot in Python?
2. What type of data representation or structure does your program use to create the histogram?
3. Could you explain the customization options implemented in your program to enhance the visual aspects of the histogram (e.g., bins, colors, labels, axes formatting)?
4. Does your program provide flexibility in setting the number of bins or handling different bin sizes in the histogram plot? If so, how is this managed?
5. How does your program manage potential errors or unexpected data types while creating the histogram plot?

Lab 5b: **Write a Python program to Demonstrate how to Draw a Pie Chart using Matplotlib.**

1. How does your program utilize Matplotlib to create a pie chart in Python?

2. What type of data representation does your program employ for generating the pie chart?
3. Could you explain the customization options your program uses to enhance the appearance of the pie chart (e.g., colors, labels, explode, shadow effects)?
4. Does your program handle the labeling of slices in the pie chart, especially when there are many slices or small portions? If so, how is this managed?
5. How does your program address potential errors or unexpected data types while creating the pie chart?

Lab 6a: **Write a Python program to illustrate Linear Plotting using Matplotlib.**

1. How does your program use Matplotlib to create a linear plot in Python?
2. Could you explain the data representation and format used within your program to illustrate the linear plot?
3. What customization options or attributes does your program utilize to enhance the appearance of the linear plot (e.g., line styles, colors, labels, axes formatting)?
4. Does your program handle multiple lines on the same plot or incorporate markers along the plotted line? If so, how is this implemented?
5. How does your program manage potential errors or unexpected data types while creating the linear plot?

Lab 6b: **Write a Python program to illustrate liner plotting with line formatting using Matplotlib**

1. How does your program use Matplotlib to perform linear plotting with line formatting in Python?
2. What line formatting options or attributes does your program employ to enhance the appearance of the plotted lines (e.g., line styles, colors, thickness)?
3. Could you explain how your program differentiates and illustrates various lines using different formatting in the same plot?
4. Does your program allow for the inclusion of markers or points along the plotted lines, and how are these incorporated with the line formatting?
5. Does your program allow for the inclusion of markers or points along the plotted lines, and how are these incorporated with the line formatting?

Lab 7: **Write a Python program which explains uses of customizing seaborn plots with Aesthetic functions**

1. How does your program utilize Seaborn's aesthetic functions to customize plots in Python?

2. Could you explain the types of customizations applied using Seaborn's aesthetic functions (e.g., changing color palettes, modifying plot styles, adjusting plot elements)?
3. What are the advantages of using Seaborn's aesthetic functions over standard plotting libraries for plot customization?
4. Does your program demonstrate the application of aesthetic functions in multiple types of plots (e.g., bar plots, line plots, scatter plots)? If so, how is this achieved?
5. How does your program handle potential errors or unexpected inputs when using Seaborn's aesthetic functions for plot customization?

Lab 8: Write a Python program for plotting different types of plots using Bokeh

1. How does your program utilize Bokeh to create different types of plots in Python?
2. Which types of plots are demonstrated in your program using Bokeh?
3. Could you explain the approach your program uses to handle and format the various plot types (e.g., line plots, scatter plots, bar plots) in Bokeh?
4. Does your program showcase interactivity features or tools available in Bokeh for the created plots? If so, how are these features implemented?
5. How does your program handle potential errors or unexpected data inputs when creating different types of plots using Bokeh?

Lab 9: Write a Python program to draw 3D Plots using Plotly Libraries

1. How does your program use the Plotly library to create 3D plots in Python?
2. Which types of 3D plots are demonstrated in your program using Plotly (e.g., 3D surface plots, scatter plots, contour plots)?
3. Could you explain the approach your program takes to handle and format the different types of 3D plots in Plotly?
4. Does your program showcase any interactive or customizable features available in Plotly for the created 3D plots? If so, how are these features implemented?
5. How does your program handle potential errors or unexpected data inputs when creating various 3D plots using Plotly?

Lab 10a: Write a Python program to draw Time Series using Plotly Libraries.

1. How does your program utilize Plotly to create Time Series plots in Python?
2. Which specific features of Plotly does your program use to handle time-based data in creating Time Series plots?
3. Could you explain the process your program follows to format and represent Time Series data within the Plotly environment?
4. Does your program offer any interactive elements for the Time Series plots, such as zooming, panning, or tooltips? If so, how are these implemented?

5. How does your program manage potential errors or unexpected data formats when creating Time Series plots using Plotly?

Lab 10b: Write a Python program for creating Maps using Plotly Libraries

1. How does your program utilize Plotly libraries to create maps in Python?
2. Which types of maps are demonstrated in your program using Plotly (e.g., scatter plots on maps, choropleth maps, geographic heatmaps)?
3. Could you explain the data representation and source used within your program to create maps in Plotly?
4. Does your program showcase any interactive or customizable features available in Plotly for the created maps (e.g., zoom, hover information, custom styling)? If so, how are these features implemented?
5. How does your program handle potential errors or unexpected data inputs when creating different types of maps using Plotly?

Note : All Lab Programs from 4 to 10 can be written using files as input.such programs will be updated as earliest