S#	Activity Title
1	Calculator – Net Run Rate (NRR)
2	Character Classification Program
3	Find the Largest among 3 numbers
4	Generate an Electricity bill for a customer based on units consumed
5	Check if a Given number is an Armstrong Number
6	Print Multiplication table of a given number
7	Generate a Number Pattern with consecutive integers
8	Find the Maximum Mark from a List of Student's Marks
9	Check if a Given String is a Palindrome
10	Count the Frequency of Elements in a given List
11	Check whether given number is Prime or not
12	Create a 2D array using NumPy
13	Converting a Python List to a NumPy Array
14	Creating 4x5 matrix from 11 to 30 using NumPy
15	Creating a Dataframe to store interview data of Candidates
16	Creating a Player Dataframe displaying players with scores above 1000
17	Representing mobile game ratings using a Bar Chart
18	Calculating Variance and Standard Deviation
19	Menu-Driven Program to Calculate Mean, Mode, and Median
20	Plotting Sales Data of Clothes Store on a Line Chart
21	Sales Stats-Monthly Sales of a Salesman

1. Calculator – Net Run Rate (NRR)

CODE:

```
#Input runs scored and runs conceded
runs_scored = int(input("Enter total runs scored: "))
overs_batted = float(input("Enter total overs batted: "))
runs_conceded = int(input("Enter total runs conceded: "))
overs_bowled = float(input("Enter total overs bowled: "))
# Calculate net run rate
net_run_rate = (runs_scored / overs_batted) - (runs_conceded / overs_bowled)
# Display the result
print("Net Run Rate:", net_run_rate)
```

OUPUT:

```
Enter total runs scored:  220
Enter total overs batted:  20
Enter total runs conceded:  215
Enter total overs bowled:  18.5
Net Run Rate: -0.621621621621621
```

2. Character Classification Program

CODE:

```
# Input the character to check
ch=input("Enter Any Character:")
# Checking whether it is upperletter or lowerletter or digit or a special character
if ch.isupper():
    print(ch, " is an upper case letter")
elif ch.islower():
    print(ch, " is a lower case letter")
elif ch.isdigit():
    print(ch, " is a digit")
elif ch.isspace():
    print(ch, " is a space")
else:
    print(ch," is a special character")
```

OUPUT:

```
Enter Any Character: #
#  is a special character
```

3.Find the Largest among 3 numbers

CODE:

```
# Find the Greatest among 3 numbers
n1=int(input("Enter the Number1:"))
n2=int(input("Enter the Number2:"))
n3=int(input("Enter the Number3:"))
if n1>n2 and n1>n3:
    print(n1, " is greater")
elif n2>n1 and n2>n3:
    print(n2, " is greater")
elif n3>n1 and n3>n2:
    print(n3, " is greater")
else:
    print("All are same")
```

OUPUT:

```
Enter the Number1: 5
Enter the Number2: 7
Enter the Number3: 3
7  is greater
```

4. Generate an Electricity bill for a customer based on units consumed

CODE:

```
cno = int(input("Enter Consumer Number: "))
pc = float(input("Enter Power Consumed (in units): "))
if pc <= 0:
    print("Invalid Power Consumed Units! Power consumption must be greater than 0.")
else:
    if pc <= 100:
        bill_amt = pc * 1
    elif pc <= 300:
        bill_amt = 100 + (pc - 100) * 1.25
    elif pc < 500:
        bill_amt = 350 + (pc - 300) * 1.50
    else:
        bill_amt = 650 + (pc - 500) * 1.75
    print("~" * 60)
    print("\t\tABC Power Company Ltd.")
    print("~" * 60)
    print(f"Consumer Number: {cno}")
    print(f"Consumed Units: {pc}")
    print("-----------------------------")
    print(f"Bill Amount: {bill_amt:.2f}")
```

OUPUT:

```
Enter Consumer Number:  1010
Enter Power Consumed (in units):  200
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
                ABC Power Company Ltd.
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Consumer Number: 1010
Consumed Units: 200.0
-----------------------------
Bill Amount: 225.00
```

5. Check if a Given number is an Armstrong Number

CODE:

```
number = int(input("Enter a number to check: "))
num_str = str(number)
num_digits = len(num_str)
sum_of_powers = 0
for digit in num_str:
 sum_of_powers += int(digit) ** num_digits
if sum_of_powers == number:
 print(f"{number} is an Armstrong number")
else:
 print(f"{number} is not an Armstrong number")
```

OUPUT:

```
Enter a number to check:  370
370 is an Armstrong number
```

6. Write a program to print a multiplication table of a given number

CODE:

```
n=int(input("Enter number to print multiplication table:"))
#Take for loop for multiple
i=1
while (i<=10):
 print(n," x ", i, " = ", n*i )
 i=i+1
```

OUPUT:

```
Enter number to print multiplication table: 5
5  x  1  =  5
5  x  2  =  10
5  x  3  =  15
5  x  4  =  20
5  x  5  =  25
5  x  6  =  30
5  x  7  =  35
5  x  8  =  40
5  x  9  =  45
5  x  10  =  50
```

7. Generate a Number Pattern with consecutive integers

CODE:

```
n=int(input("Enter n:"))
k=1
for i in range(1,n+1):
 for j in range(1,i+1):
     print(k,end=" ")
     k=k+1
 print()
```

OUPUT:

```
Enter n: 5
1
2 3
4 5 6
7 8 9 10
11 12 13 14 15
```

8. Find the Maximum Mark from a List of Student's Marks

CODE:

```
n=int(input("Enter no. of subjects:"))
l=[]
for i in range(n):
 m=int(input("Enter marks:"))
 l.append(m)
print("Maximum marks scored:",max(l))
```

OUPUT:

```
Enter no. of subjects: 2
Enter marks: 60
Enter marks: 70
Maximum marks scored: 70
```

9. Check if the given String is Palindrome

CODE:

```
string = input("Enter a string to check: ")
string = string.replace(" ", "").lower()
reversed_string = string[::-1]
if string == reversed_string:
 print(string, "- is a palindrome.")
else:
 print(string, "- is a pliandrom.")
```

OUPUT:

```
Enter a string to check:  malayalam
malayalam - is a palindrome.
```

10. Count the Frequency of Elements in a given List

CODE:

```
l = []
n=int(input("Enter the no. of elements:"))
for i in range(n):
 val=int(input("Enter value "+str(i+1)+":"))
 l.append(val)
#Decalring a dictionary object to store the data
f = {}
for i in l:
 if (i in f):
     f[i] += 1
 else:
     f[i] = 1
for i, j in f.items():
 print(i, "->", j)
```

OUPUT:

```
Enter the no. of elements: 3
Enter value 1: 1
Enter value 2: 1
Enter value 3: 2
1 -> 2
2 -> 1
```

11. Check whether given number is Prime or not

CODE:

```
num = int(input("Enter Number:"))
flag = False
if num == 0 or num == 1:
    print(num, "is not a prime number")
elif num > 1:
 for i in range(2, num):
     if (num % i) == 0:
         flag = True
     break
 if flag:
     print(num, "is not a prime number")
 else:
     print(num, "is a prime number")
```

OUPUT:

```
Enter Number: 7
7 is a prime number
```

12. Create a 2D array using NumPy

CODE:

```
import numpy as np
array_2d = np.array([[1, 2, 3],
 [4, 5, 6],
 [7, 8, 9]])
print("The 2D array is:")
print(array_2d)
```

OUPUT:

```
The 2D array is:
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

13. Converting a Python list to a NumPy array

CODE:

```
import numpy as np
python_list = [1, 2, 3, 4, 5]
numpy_array = np.array(python_list)
print("The NumPy array is:")
print(numpy_array)
```

OUPUT:

```
The NumPy array is:
[1 2 3 4 5]
```

14. Creating a 4x5 matrix from 11 to 30 using NumPy

CODE:

```
import numpy as np
numbers = np.arange(11, 30 + 1)
matrix_4x5 = numbers.reshape(4, 5)
print("The 4x5 matrix is:")
print(matrix_4x5)
```

OUPUT:

```
The 4x5 matrix is:
[[11 12 13 14 15]
 [16 17 18 19 20]
 [21 22 23 24 25]
 [26 27 28 29 30]]
```

15. Creating a Dataframe to store interview data of Candidates

CODE:

```
import pandas as pd
data = {
 'name': ['Alice', 'Bob', 'Charlie', 'David', 'Eve'],
 'score': [85, 72, 90, 65, 78],
 'attempts': [1, 2, 1, 3, 2],
 'qualify': ['Yes', 'No', 'Yes', 'No', 'Yes']
}
index_labels = ['C001', 'C002', 'C003', 'C004', 'C005']
df = pd.DataFrame(data, index=index_labels)
print("Data of candidates who appeared in interviews:")
print(df)
```

OUPUT:

```
Data of candidates who appeared in interviews:
         name  score  attempts qualify
C001    Alice     85         1     Yes
C002      Bob     72         2      No
C003  Charlie     90         1     Yes
C004    David     65         3      No
C005      Eve     78         2     Yes
```

16. Creating a Player Dataframe displaying players with scores above 1000

CODE:

```
import pandas as pd
data = {
 'team': ['Team A', 'Team B', 'Team A', 'Team C', 'Team B'],
 'no. of matches': [50, 120, 75, 100, 85],
 'runs': [1200, 950, 1800, 800, 1500],
 'average': [24.0, 20.8, 32.0, 18.5, 28.0]
}
index_labels = ['Player1', 'Player2', 'Player3', 'Player4', 'Player5']
player = pd.DataFrame(data, index=index_labels)
filtered_players = player[player['runs'] > 1000]
print("Players with runs greater than 1000:")
print(filtered_players)
```

OUPUT:

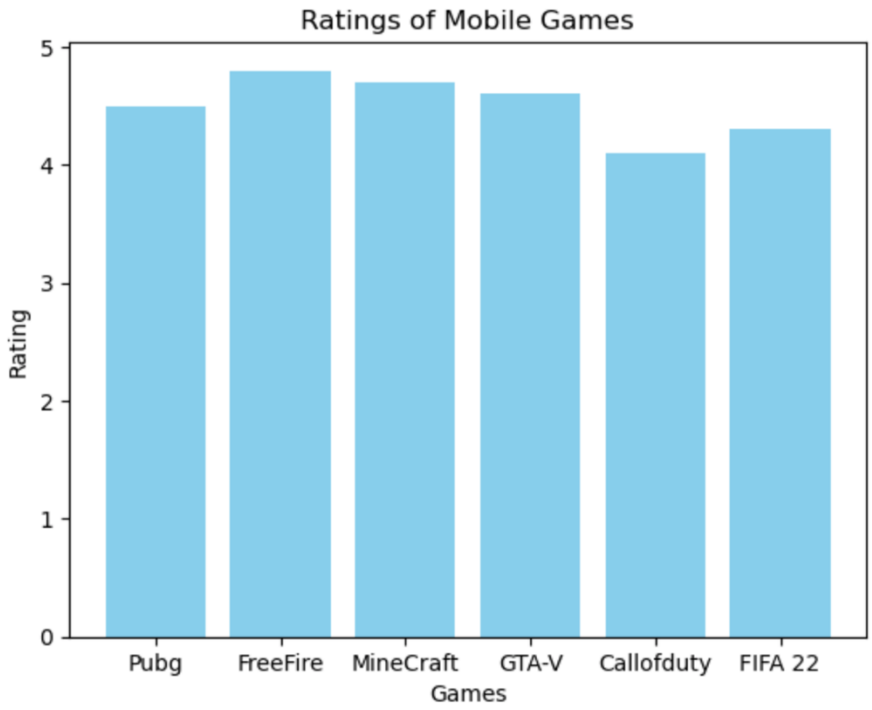
```
Players with runs greater than 1000:
           team  no. of matches  runs  average
Player1  Team A              50  1200     24.0
Player3  Team A              75  1800     32.0
Player5  Team B              85  1500     28.0
```

17. Write a program to represent the data on the ratings of mobile games on bar chart. The sample data is given as: Games: Pubg, FreeFire,MineCraft, GTA-V, Callofduty, FIFA 22. Rating: 4.5,4.8,4.7,4.6,4.1,4.3.

CODE:

```
import matplotlib.pyplot as plt
games = ['Pubg', 'FreeFire', 'MineCraft', 'GTA-V', 'Callofduty', 'FIFA 22']
ratings = [4.5, 4.8, 4.7, 4.6, 4.1, 4.3]
plt.bar(games, ratings, color='skyblue')
plt.title('Ratings of Mobile Games')
plt.xlabel('Games')
plt.ylabel('Rating')
plt.show()
```

OUPUT:



18. Write a program to calculate variance and standard deviation for the given data: [33,44,55,67,54,22,33,44,56,78,21,31,43,90,21,33,44,55,87]

CODE:

```
import numpy as np
data = [33, 44, 55, 67, 54, 22, 33, 44, 56, 78, 21, 31, 43, 90, 21, 33, 44, 55, 87]
variance = np.var(data)
std_deviation = np.std(data)
print("Variance:", variance)
print("Standard Deviation:", std_deviation)
```

OUPUT:

```
Variance: 416.5761772853186
Standard Deviation: 20.410197874722297
```

19. Write a menu-driven program to calculate the mean, mode and median for the given data: [5,6,1,3,4,5,6,2,7,8,6,5,4,6,5,1,2,3,4]

CODE:

```
import statistics
data = [5, 6, 1, 3, 4, 5, 6, 2, 7, 8, 6, 5, 4, 6, 5, 1, 2, 3, 4]
while True:
    print("\nMenu:")
    print("1. Calculate Mean")
    print("2. Calculate Mode")
    print("3. Calculate Median")
    print("4. Exit")

    try:
        choice = int(input("Enter your choice: "))

        if choice == 1:
            mean = sum(data) / len(data)
            print(f"Mean: {mean}")
        elif choice == 2:
            try:
                mode = statistics.mode(data)
                print(f"Mode: {mode}")
            except statistics.StatisticsError:
                print("Error: No unique mode, multiple modes exist.")
        elif choice == 3:
            median = statistics.median(data)
            print(f"Median: {median}")
        elif choice == 4:
            print("Exiting the program...")
            break  # Exit the loop
        else:
            print("Invalid choice. Please try again.")
    except ValueError:
        print("Invalid input! Please enter a valid integer choice.")
```

OUPUT:

```
Menu:
1. Calculate Mean
2. Calculate Mode
3. Calculate Median
4. Exit
Enter your choice:  1
Mean: 4.368421052631579
```

20. Plotting Sales Data of Cloth Store on a Line Chart

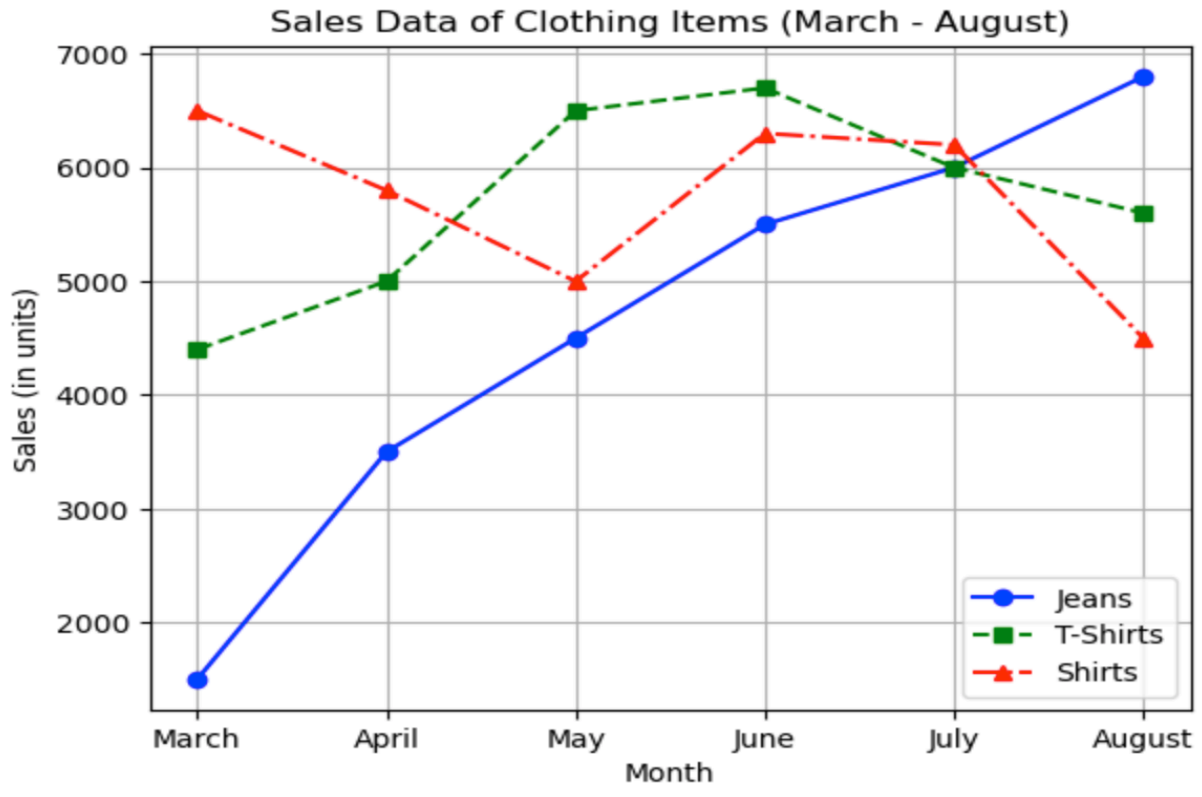
Month	Jeans	T-Shirts	Shirts
March	1500	4400	6500
April	3500	4500	5000
May	6500	5500	5800
June	6700	6000	6300
July	6000	5600	6200
August	6800	6300	4500

CODE:

```
import matplotlib.pyplot as plt
months = ['March', 'April', 'May', 'June', 'July', 'August']
jeans_sales = [1500, 3500, 4500, 5500, 6000, 6800]
t_shirts_sales = [4400, 5000, 6500, 6700, 6000, 5600]
shirts_sales = [6500, 5800, 5000, 6300, 6200, 4500]

plt.plot(months, jeans_sales, label='Jeans', marker='o', linestyle='-', color='blue')
plt.plot(months, t_shirts_sales, label='T-Shirts', marker='s', linestyle='--', color='green')
plt.plot(months, shirts_sales, label='Shirts', marker='^', linestyle='-.', color='red')
plt.title('Sales Data of Clothing Items (March - August)')
plt.xlabel('Month')
plt.ylabel('Sales (in units)')
plt.grid(True)
plt.legend()
plt.show()
```

OUPUT:



21. Observe the given data for monthly sales of one of the salesmen for 6 months. Plot them on the line chart.

Month	January	February	March	April	May	June
Sales	2500	2100	1700	3500	3000	3800

Apply the following customization to the chart:

- Give the title for the chart – “Sales Stats”
- Use the “Month” label for X-Axis and “Sales” for Y-Axis
- Display legends
- Use dashed lines with the width 5 point
- Use red color for the line
- Use dot marker with blue edge color and black fill color

CODE:

```
import matplotlib.pyplot as plt
months = ['January', 'February', 'March', 'April', 'May', 'June']
sales = [2500, 2100, 1700, 3500, 3000, 3800]
plt.plot(months, sales, label='Sales', linestyle='--', color='red', linewidth=5, marker='o',
markersize=8, markeredgecolor='blue', markerfacecolor='black')
plt.title('Sales Stats')
plt.xlabel('Month')
plt.ylabel('Sales')
plt.legend()
plt.show()
```

OUPUT: