Master of Operational Research (MOR)

# Python Programming Practical List

1. Write a program to enter name and display as “Hello, Name”.
2. Write a menu driven program to enter two numbers and print the arithmetic operations like

a. + b. – c. \* d. / e. // f. %.

1. To compute the roots of a quadratic equation.
2. Write a menu driven Program to reverse the entered numbers and print the sum of digits entered.
3. Write a menu driven Program to enter the number and print whether the number is
   1. odd or even
   2. prime.
4. Program to find maximum out of entered 3 numbers
5. Write a program to display ASCII code of a character and vice versa.
6. Write a Program to check if the entered number is Armstrong or not.
7. Write a Program to find factorial of the entered number using recursion.
8. Write a Program to enter the number of terms and to print the Fibonacci Series.
9. Write a Program to enter the numbers and to print greatest number using loop.
10. Write a Program to enter the string and to check if it’s palindrome or not using loop.
11. Write a Program to enter the 5 subjects numbers and print the grades A/B/C/D/E.
12. Write a program in python language to display the given pattern:



1. Write a python function sin(x,n) to calculate the value of sin(x) using its Taylor series expansion up to n terms.



1. Write a Program to determine EOQ using various inventory models.
2. Write a Program to determine different characteristics using various Queuing models
3. Write a Program to implement Inheritance. Create a class Employee inherit two classes Manager and Clerk from Employee.
4. Program to fit Poisson distribution on a given data.
5. Write a program to implement linear regression using python.
6. Write a program to perform read and write operation with .csv file.
7. Write a Program to enter multiple values-based data in multiple columns/rows and show that data in python using DataFrames and pandas.
8. WAP in python to perform various statistical measures using pandas.
9. Write a program to plot a bar chart in python to display the result of a school for five consecutive years.
10. Write a program in python to plot a graph for the function y = x2
11. Write a program in python to plot a pie chart on consumption of water in daily life.

# Note: -

It is advised that students should install various libraries (NumPy, Pandas, Mathplotlib, etc.) in their system.

**Program1**

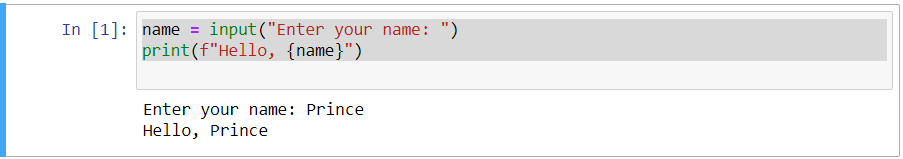
**Aim Write a program to enter name and display as “Hello, Name”.**

**Code**

name = input("Enter your name: ")

print(f"Hello, {name}")

**Output**



**……**

**End of Program1**

**Program2**

**Aim Write a menu driven program to enter two numbers and print the arithmetic operations like**

**a. + b. – c. \* d. / e. // f. %.**

**Code**

num1 = float(input("Enter first number: "))

num2 = float(input("Enter second number: "))

print("Select operation:")

print("a. + b. - c. \* d. / e. // f. %")

choice = input("Enter choice: ")

if choice == 'a':

print("Sum:", num1 + num2)

elif choice == 'b':

print("Difference:", num1 - num2)

elif choice == 'c':

print("Product:", num1 \* num2)

elif choice == 'd':

print("Division:", num1 / num2)

elif choice == 'e':

print("Floor Division:", num1 // num2)

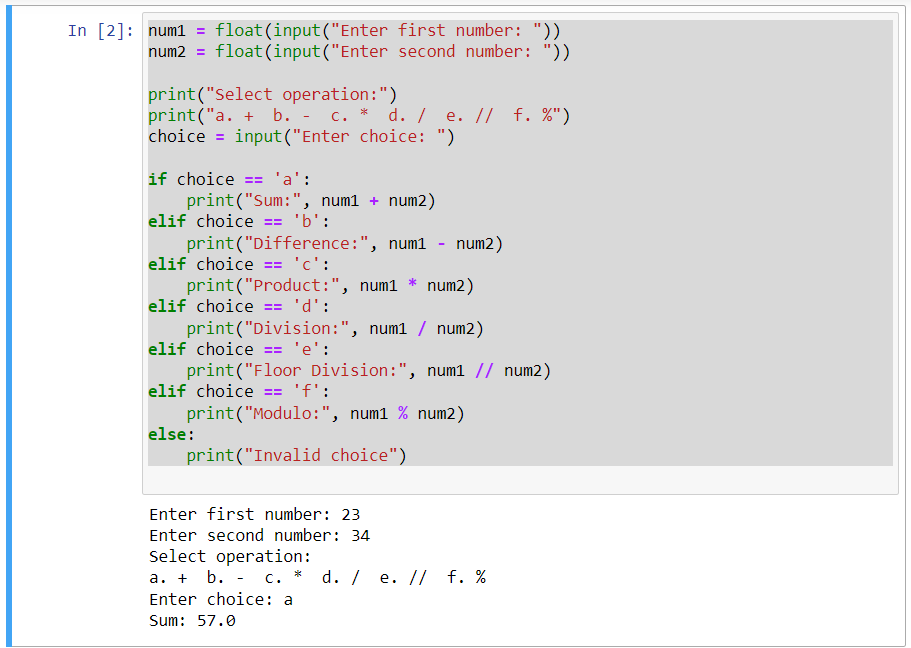
elif choice == 'f':

print("Modulo:", num1 % num2)

else:

print("Invalid choice")

**Output**



**……**

**End of Program2**

**Program3**

**Aim To compute the roots of a quadratic equation.**

**Code**

import cmath

a = float(input("Enter coefficient a: "))

b = float(input("Enter coefficient b: "))

c = float(input("Enter coefficient c: "))

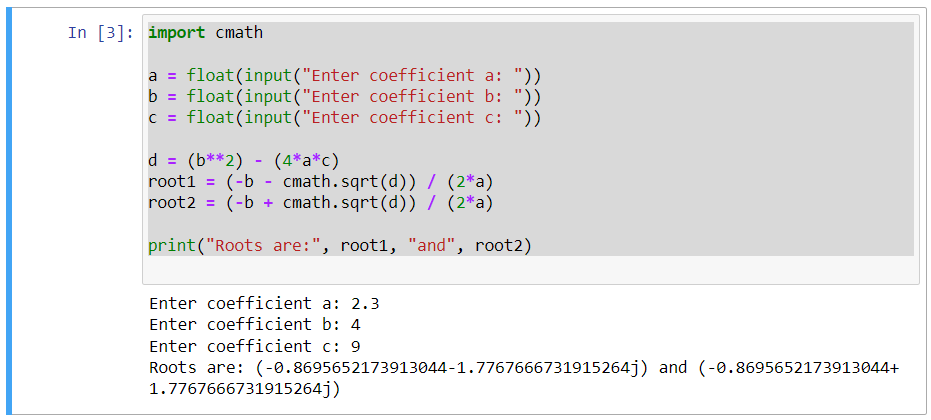
d = (b\*\*2) - (4\*a\*c)

root1 = (-b - cmath.sqrt(d)) / (2\*a)

root2 = (-b + cmath.sqrt(d)) / (2\*a)

print("Roots are:", root1, "and", root2)

**Output**



**……**

**End of Program3**

**Program4**

**Aim Write a menu driven Program to reverse the entered numbers and print the sum of digits entered.**

**Code**

num = int(input("Enter a number: "))

reverse\_num = 0

sum\_of\_digits = 0

while num > 0:

digit = num % 10

reverse\_num = reverse\_num \* 10 + digit

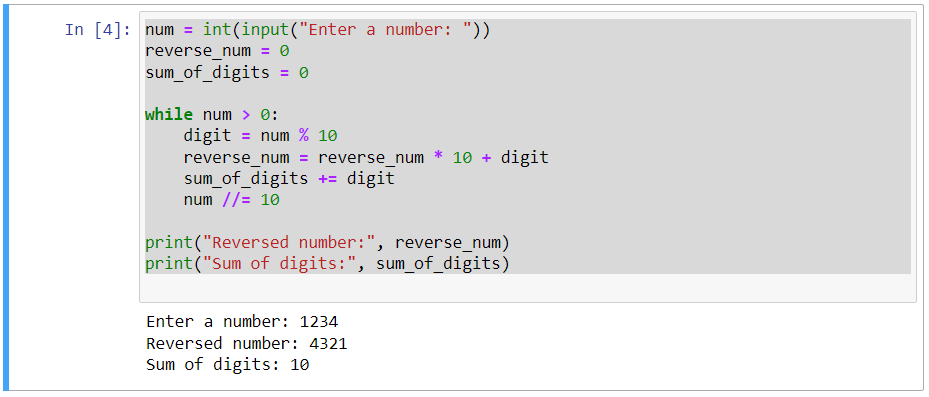
sum\_of\_digits += digit

num //= 10

print("Reversed number:", reverse\_num)

print("Sum of digits:", sum\_of\_digits)

**Output**



**……**

**End of Program4**

**Program5**

**Aim Write a menu driven Program to enter the number and print whether the number is**

* 1. **odd or even**
  2. **prime.**

**Code**

num = int(input("Enter a number: "))

# Check for odd/even

if num % 2 == 0:

print("Number is even")

else:

print("Number is odd")

# Check for prime

is\_prime = True

if num > 1:

for i in range(2, int(num\*\*0.5) + 1):

if num % i == 0:

is\_prime = False

break

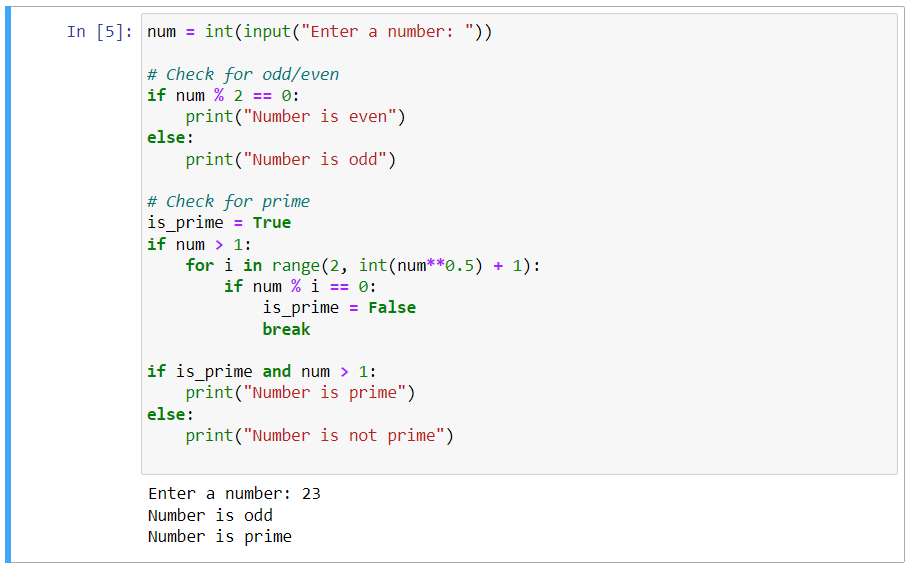
if is\_prime and num > 1:

print("Number is prime")

else:

print("Number is not prime")

**Output**



**……**

**End of Program5**

**Program6**

**Aim Program to find maximum out of entered 3 numbers**

**Code**

num1 = float(input("Enter first number: "))

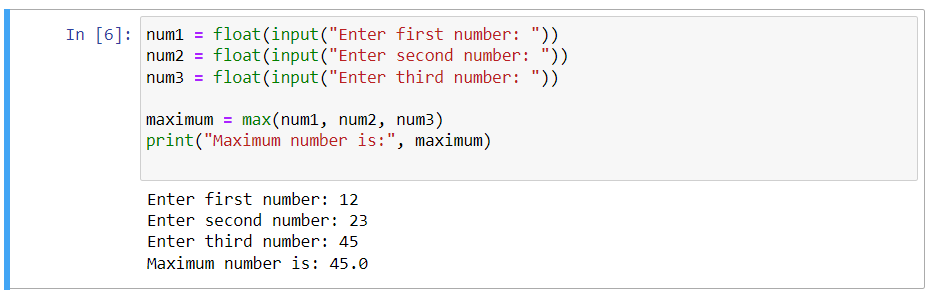
num2 = float(input("Enter second number: "))

num3 = float(input("Enter third number: "))

maximum = max(num1, num2, num3)

print("Maximum number is:", maximum)

**Output**



**……**

**End of Program6**

**Program7**

**AimWrite a program to display ASCII code of a character and vice versa.**

**Code**

# Character to ASCII

char = input("Enter a character: ")

ascii\_value = ord(char)

print(f"The ASCII value of {char} is {ascii\_value}")

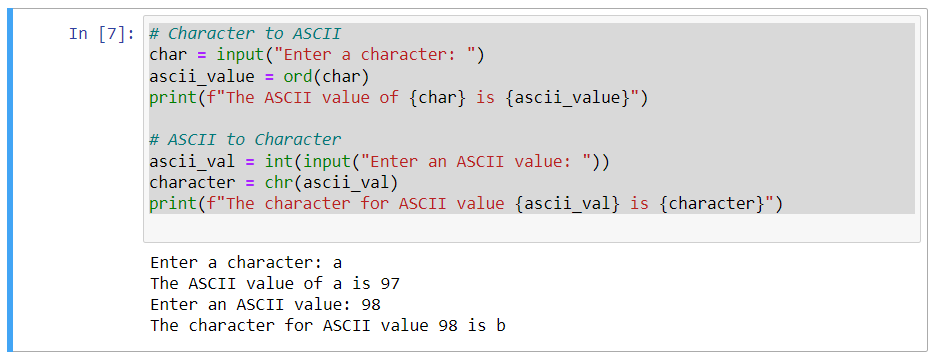
# ASCII to Character

ascii\_val = int(input("Enter an ASCII value: "))

character = chr(ascii\_val)

print(f"The character for ASCII value {ascii\_val} is {character}")

**Output**



**……**

**End of Program7**

**Program8**

**AimWrite a Program to check if the entered number is Armstrong or not.**

**Code**

num = int(input("Enter a number: "))

order = len(str(num))

sum = 0

temp = num

while temp > 0:

digit = temp % 10

sum += digit \*\* order

temp //= 10

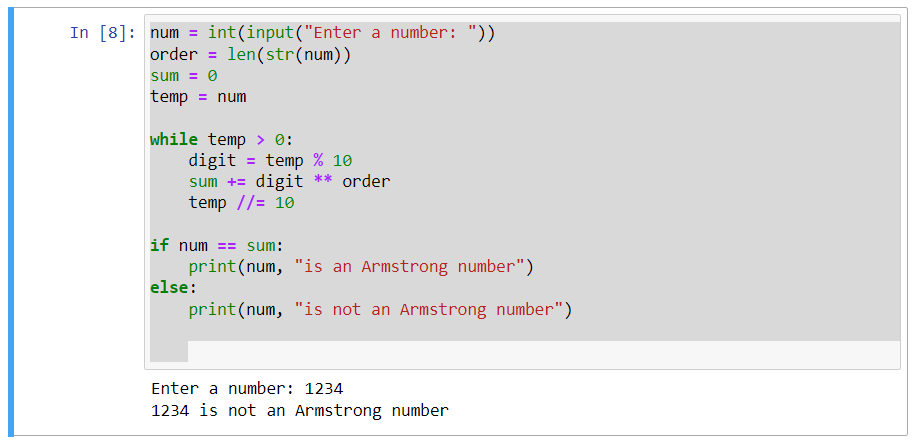
if num == sum:

print(num, "is an Armstrong number")

else:

print(num, "is not an Armstrong number")

**Output**



**……**

**End of Program8**

**Program9**

**Aim Write a Program to find factorial of the entered number using recursion.**

**Code**

def factorial(n):

if n == 0 or n == 1:

return 1

else:

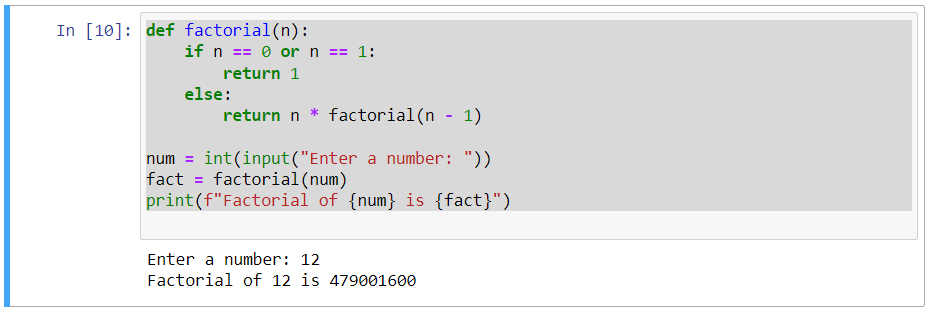
return n \* factorial(n - 1)

num = int(input("Enter a number: "))

fact = factorial(num)

print(f"Factorial of {num} is {fact}")

**Output**



**……**

**End of Program9**

**Program10**

**Aim Write a Program to enter the number of terms and to print the Fibonacci Series.**

**Code**

def fibonacci(n):

series = []

a, b = 0, 1

for \_ in range(n):

series.append(a)

a, b = b, a + b

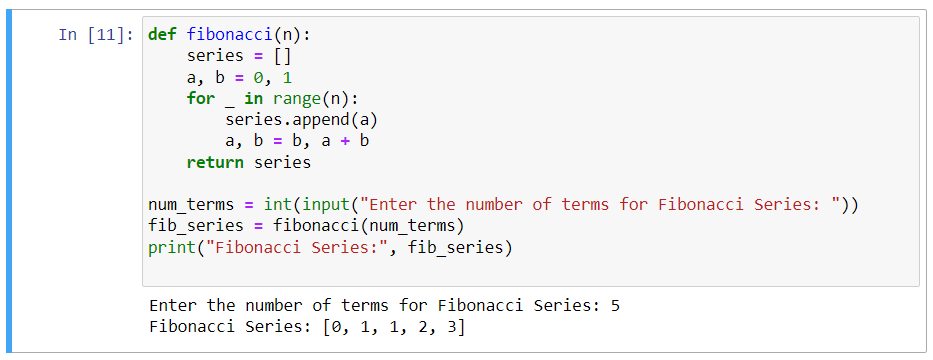
return series

num\_terms = int(input("Enter the number of terms for Fibonacci Series: "))

fib\_series = fibonacci(num\_terms)

print("Fibonacci Series:", fib\_series)

**Output**



**……**

**End of Program10**

**Program11**

**AimWrite a Program to enter the numbers and to print greatest number using loop.**

**Code**

count = int(input("How many numbers do you want to compare? "))

numbers = []

for i in range(count):

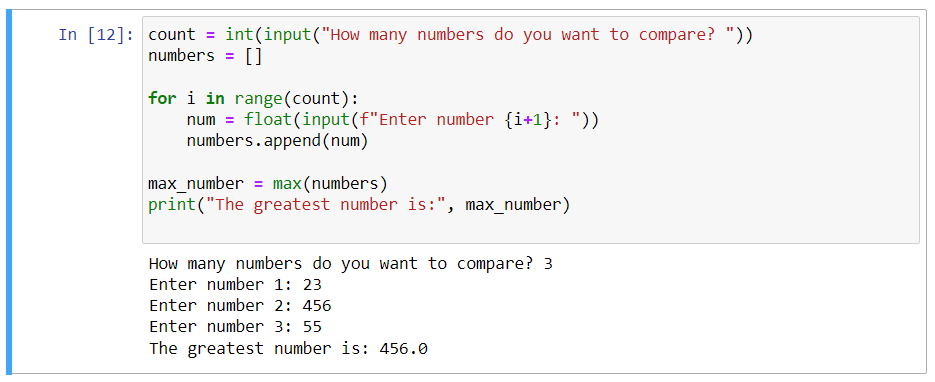
num = float(input(f"Enter number {i+1}: "))

numbers.append(num)

max\_number = max(numbers)

print("The greatest number is:", max\_number)

**Output**



**……**

**End of Program11**

**Program12**

**Aim Write a Program to enter the string and to check if it’s palindrome or not using loop.**

**Code**

string = input("Enter a string: ")

reversed\_string = string[::-1]

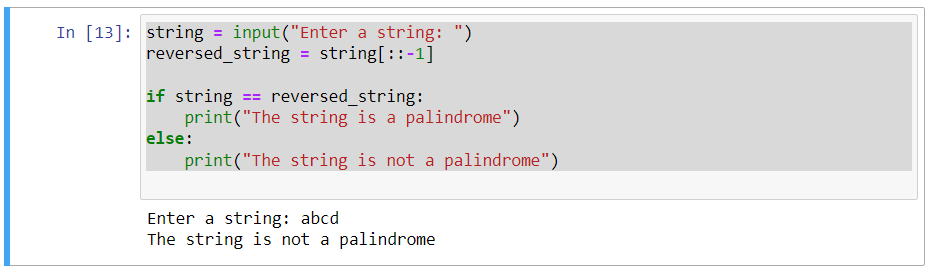
if string == reversed\_string:

print("The string is a palindrome")

else:

print("The string is not a palindrome")

**Output**



**……**

**End of Program12**

**Program13**

**Aim Write a Program to enter the 5 subjects numbers and print the grades A/B/C/D/E.**

**Code**

subjects = []

total\_marks = 0

for i in range(5):

subject = float(input(f"Enter marks for subject {i+1}: "))

subjects.append(subject)

total\_marks += subject

average\_marks = total\_marks / 5

if average\_marks >= 90:

print("Grade: A")

elif average\_marks >= 80:

print("Grade: B")

elif average\_marks >= 70:

print("Grade: C")

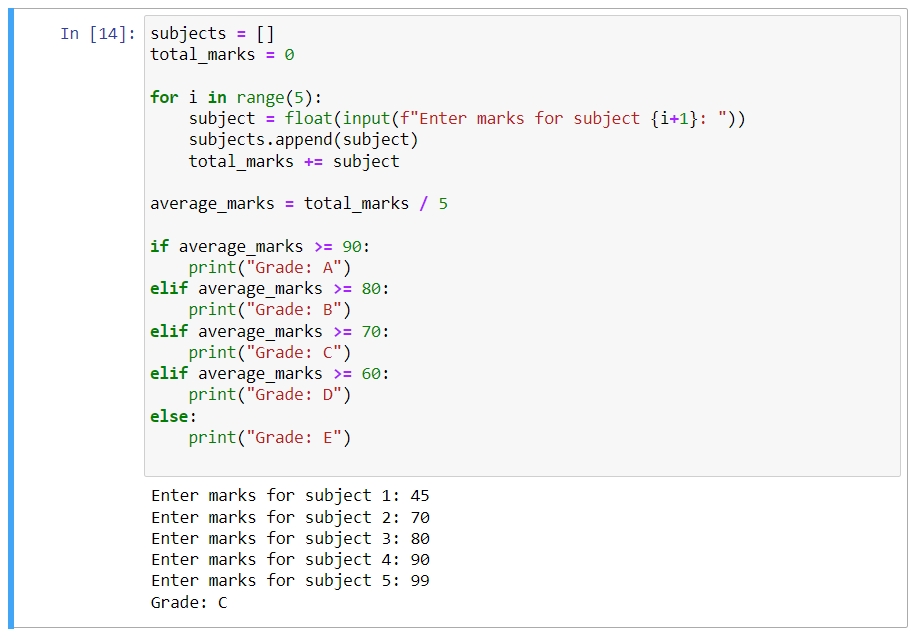
elif average\_marks >= 60:

print("Grade: D")

else:

print("Grade: E")

**Output**



**……**

**End of Program13**

**Program14**

**Aim Write a program in python language to display the given pattern:**



**Code**

rows = 5 # Number of rows in the pattern

for i in range(rows, 0, -1):

for j in range(1, i):

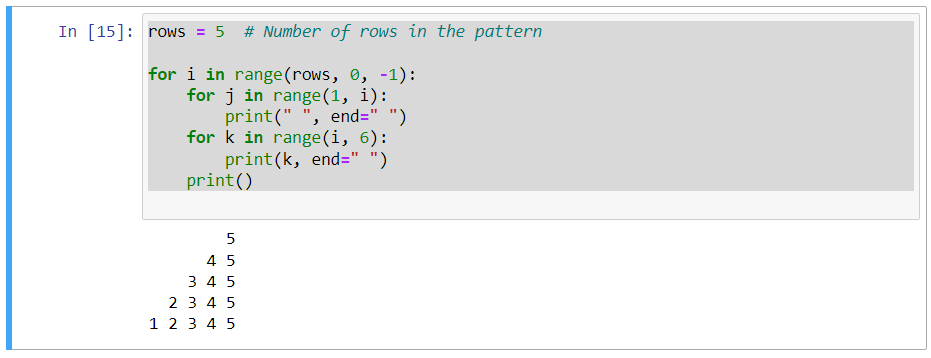
print(" ", end=" ")

for k in range(i, 6):

print(k, end=" ")

print()

**Output**



**……**

**End of Program14**

**Program15**

**Aim** Write a python function sin(x,n) to calculate the value of sin(x) using its Taylor series expansion up to n terms.



**Code**

import math

def sin(x, n):

x = math.radians(x) # Convert degrees to radians for math.sin()

result = 0

for i in range(n):

coef = (-1) \*\* i

num = x \*\* ((2 \* i) + 1)

denom = math.factorial((2 \* i) + 1)

result += (coef \* num) / denom

return result

# Example usage:

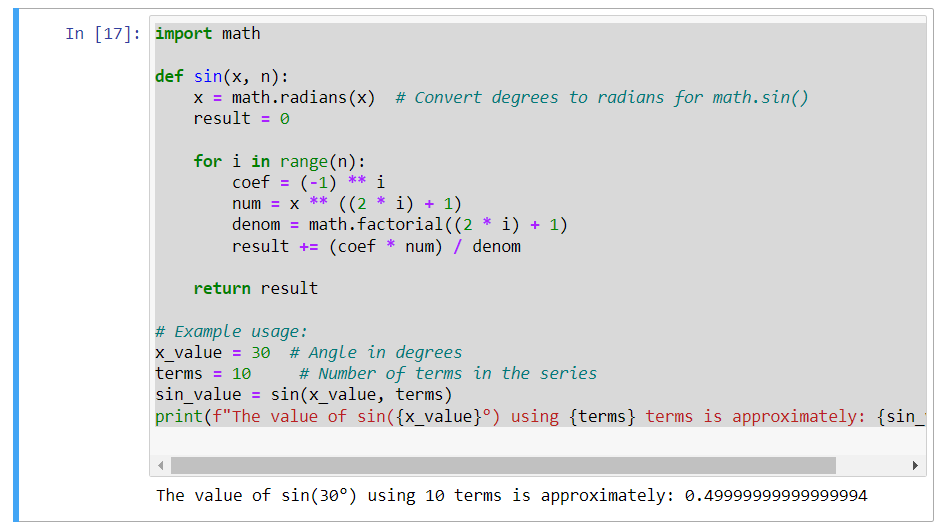
x\_value = 30 # Angle in degrees

terms = 10 # Number of terms in the series

sin\_value = sin(x\_value, terms)

print(f"The value of sin({x\_value}°) using {terms} terms is approximately: {sin\_value}")

**Output**



**……**

**End of Program15**

**Program16**

**Aim Write a Program to determine EOQ using various inventory models**

**Code**

import math

def calculate\_eoq(demand, ordering\_cost, holding\_cost):

# Calculate EOQ

eoq = math.sqrt((2 \* demand \* ordering\_cost) / holding\_cost)

return eoq

# Example usage

demand\_per\_year = 1000 # Demand per year

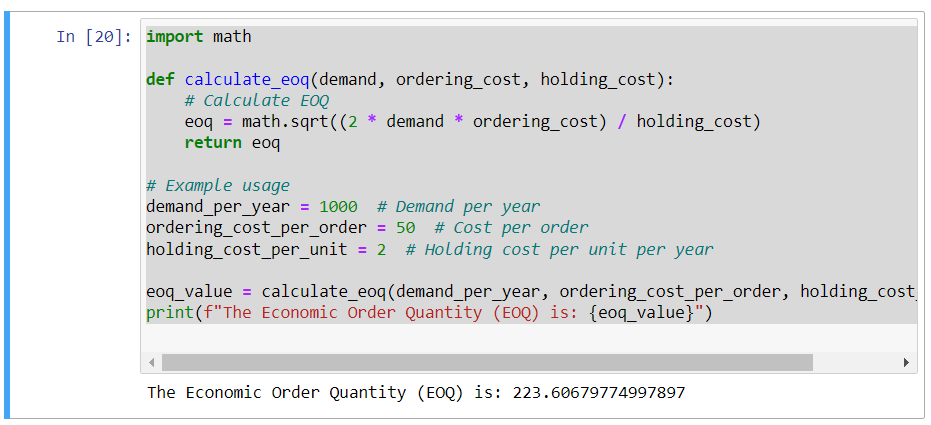
ordering\_cost\_per\_order = 50 # Cost per order

holding\_cost\_per\_unit = 2 # Holding cost per unit per year

eoq\_value = calculate\_eoq(demand\_per\_year, ordering\_cost\_per\_order, holding\_cost\_per\_unit)

print(f"The Economic Order Quantity (EOQ) is: {eoq\_value}")

**Output**



**……**

**End of Program16**

**Program17**

**Aim Write a Program to determine different characteristics using various Queuing models**

**Code**

import numpy as np

def mm1\_queue(lambd, mu):

rho = lambd / mu # Traffic intensity

avg\_waiting\_time = rho / (mu - lambd) if mu > lambd else np.inf

server\_utilization = rho if mu > lambd else 1.0

return avg\_waiting\_time, server\_utilization

# Example parameters

arrival\_rate = 4 # Arrival rate (λ)

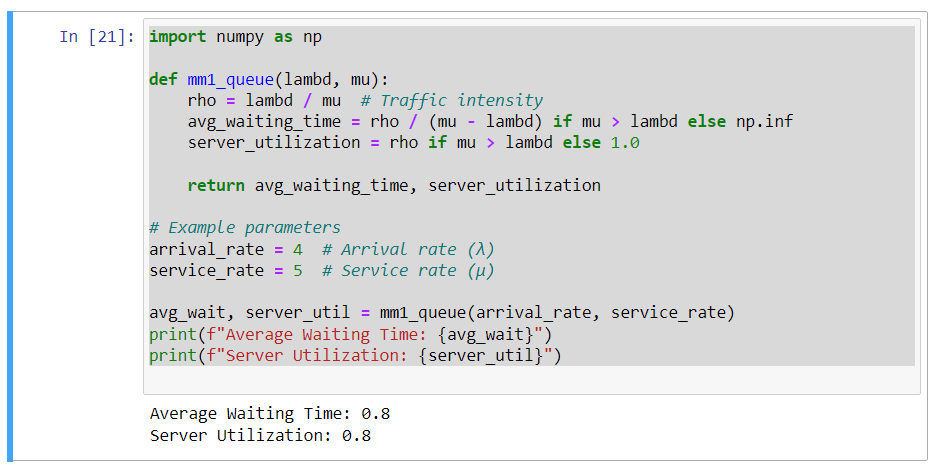
service\_rate = 5 # Service rate (μ)

avg\_wait, server\_util = mm1\_queue(arrival\_rate, service\_rate)

print(f"Average Waiting Time: {avg\_wait}")

print(f"Server Utilization: {server\_util}")

**Output**



**……**

**End of Program17**

**Program18**

**Aim Write a Program to implement Inheritance. Create a class Employee inherit two classes Manager and Clerk from Employee**

**Code**

class Employee:

def \_\_init\_\_(self, name, emp\_id):

self.name = name

self.emp\_id = emp\_id

def display\_info(self):

print(f"Name: {self.name}")

print(f"Employee ID: {self.emp\_id}")

class Manager(Employee):

def \_\_init\_\_(self, name, emp\_id, department):

super().\_\_init\_\_(name, emp\_id)

self.department = department

def display\_info(self):

super().display\_info()

print(f"Department: {self.department}")

class Clerk(Employee):

def \_\_init\_\_(self, name, emp\_id, office):

super().\_\_init\_\_(name, emp\_id)

self.office = office

def display\_info(self):

super().display\_info()

print(f"Office: {self.office}")

# Creating instances of Manager and Clerk

manager = Manager("John Doe", 1001, "Sales")

clerk = Clerk("Alice Smith", 2001, "Reception")

# Displaying information using inherited methods

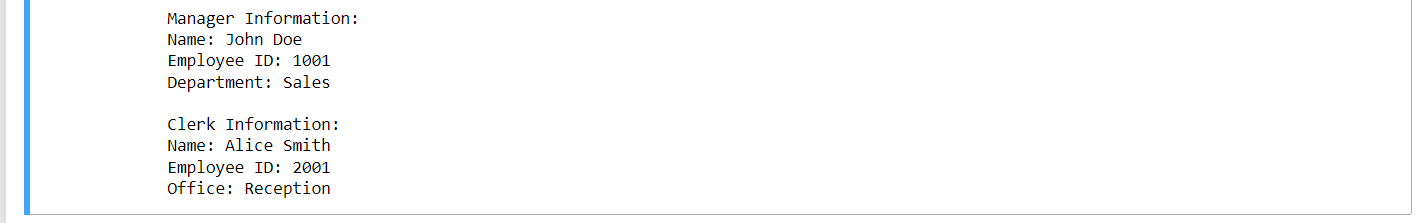
print("Manager Information:")

manager.display\_info()

print("\nClerk Information:")

clerk.display\_info()

**Output**



**……**

**End of Program18**

**Program19**

**Aim Program to fit Poisson distribution on a given data**

**Code**

import numpy as np

from scipy.optimize import minimize

import matplotlib.pyplot as plt

# Sample data

data = np.array([1, 2, 2, 3, 3, 3, 4, 4, 4, 4])

# Define the negative log-likelihood function for Poisson distribution

def neg\_log\_likelihood(lambda\_param, data):

return -np.sum(np.log(poisson.pmf(data, lambda\_param)))

# Initial guess for lambda

initial\_lambda = np.mean(data)

# Minimize negative log-likelihood to estimate lambda

result = minimize(neg\_log\_likelihood, initial\_lambda, args=(data,), method='Nelder-Mead')

estimated\_lambda = result.x[0]

# Generate Poisson distribution using the estimated lambda

x\_values = np.arange(0, np.max(data) + 1)

pmf\_values = poisson.pmf(x\_values, estimated\_lambda)

# Plot the original data and the fitted Poisson distribution

plt.hist(data, bins=np.arange(0, np.max(data) + 2), density=True, alpha=0.5, label='Data Histogram')

plt.plot(x\_values, pmf\_values, 'r', marker='o', label='Fitted Poisson Distribution')

plt.xlabel('Values')

plt.ylabel('Probability')

plt.title('Fitting Poisson Distribution to Data')

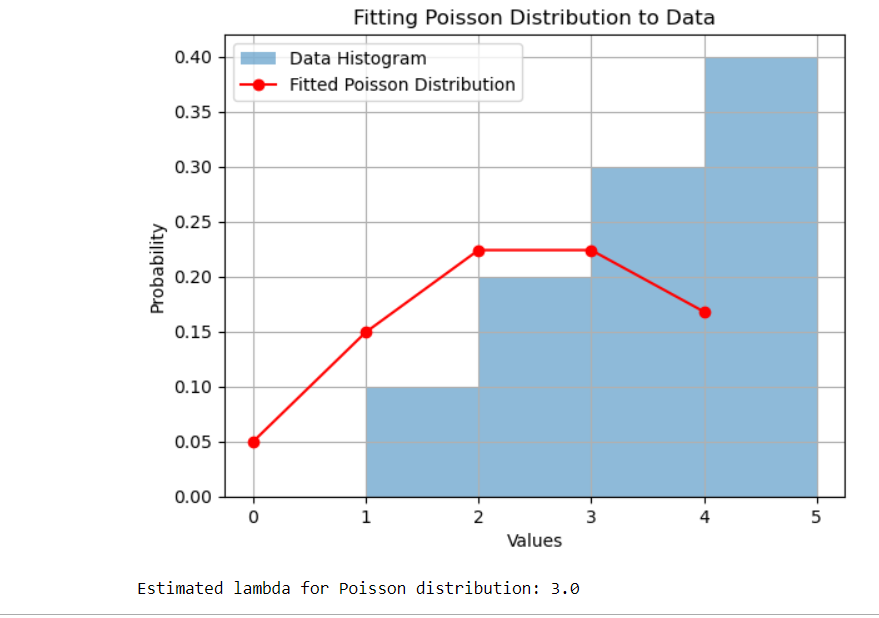
plt.legend()

plt.grid(True)

plt.show()

print("Estimated lambda for Poisson distribution:", estimated\_lambda)

**Output**



**……**

**End of Program19**

**Program20**

**Aim . Write a program to implement linear regression using python**

**Code**

import numpy as np

from sklearn.linear\_model import LinearRegression

import matplotlib.pyplot as plt

# Sample data

X = np.array([1, 2, 3, 4, 5]).reshape(-1, 1) # Independent variable (feature)

y = np.array([2, 4, 5, 4, 5]) # Dependent variable (target)

# Create a linear regression model

model = LinearRegression()

# Fit the model to the data

model.fit(X, y)

# Predictions

y\_pred = model.predict(X)

# Plotting the original data and the regression line

plt.scatter(X, y, label='Original data')

plt.plot(X, y\_pred, color='red', label='Fitted line')

plt.xlabel('X')

plt.ylabel('y')

plt.title('Linear Regression')

plt.legend()

plt.grid(True)

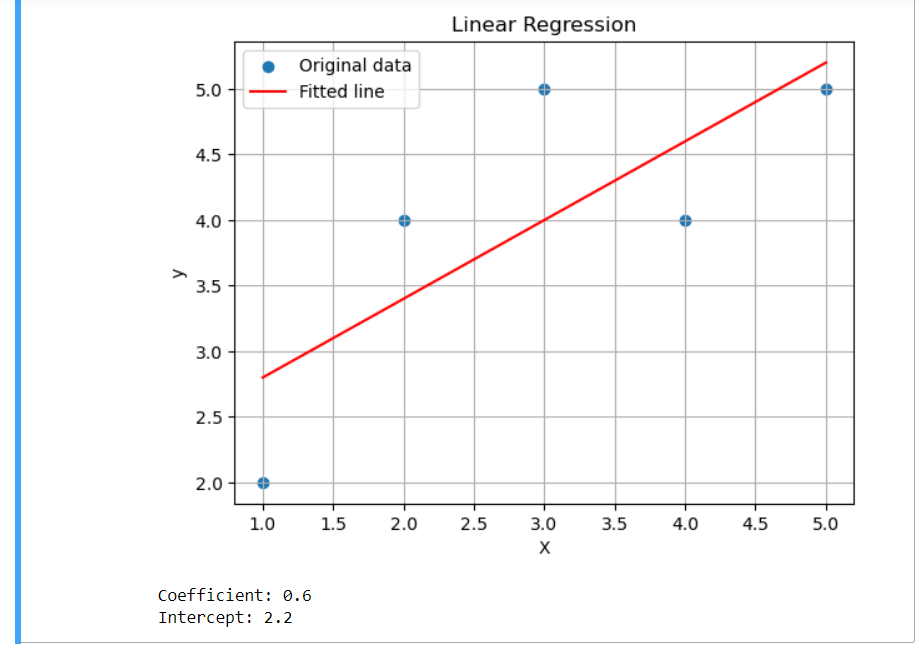
plt.show()

# Coefficients and Intercept

print("Coefficient:", model.coef\_[0])

print("Intercept:", model.intercept\_)

**Output**



**……**

**End of Program20**

**Program21**

**Aim Write a program to perform read and write operation with .csv file.**

**Code**

from sklearn.datasets import load\_iris

import pandas as pd

# Load the Iris dataset

iris = load\_iris()

iris\_data = pd.DataFrame(data=iris.data, columns=iris.feature\_names)

# Adding target column (species) to the dataframe

iris\_data['species'] = iris.target

# Write the Iris data to temp.csv file

iris\_data.to\_csv('temp.csv', index=False)

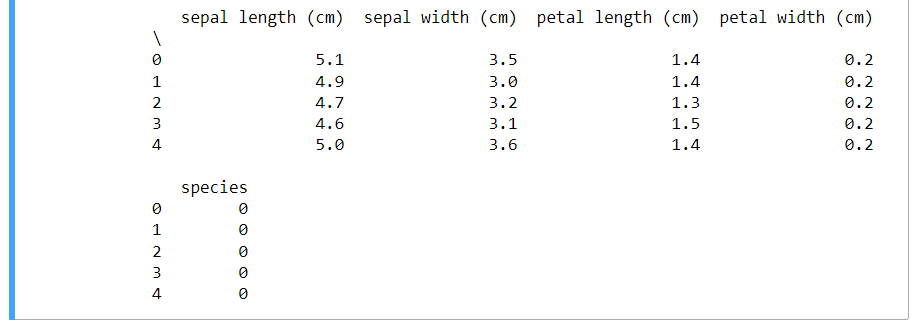
# Read the data from temp.csv

read\_data = pd.read\_csv('temp.csv')

# Displaying the read data

print(read\_data.head()) # Displaying the first few rows

**Output**



**……**

**End of Program21**

**Program22**

**Aim Write a Program to enter multiple values-based data in multiple columns/rows and show that data in python using DataFrames and pandas**

**Code**

import pandas as pd

# Initialize empty lists for each column

names = []

ages = []

grades = []

# Number of entries

num\_entries = 3

# Input data for multiple columns

for i in range(num\_entries):

name = input(f"Enter name {i + 1}: ")

age = int(input(f"Enter age {i + 1}: "))

grade = float(input(f"Enter grade {i + 1}: "))

# Append data to respective lists

names.append(name)

ages.append(age)

grades.append(grade)

# Create a DataFrame from the entered data

data = {

'Name': names,

'Age': ages,

'Grade': grades

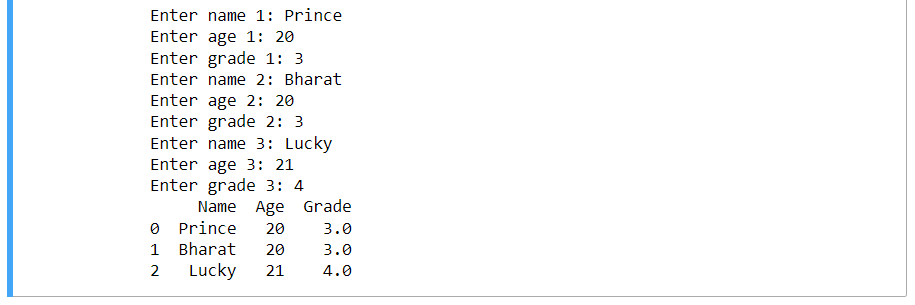
}

df = pd.DataFrame(data)

# Display the DataFrame

print(df)

**Output**



**……**

**End of Program22**

**Program23**

**Aim WAP in python to perform various statistical measures using pandas.**

**Code**

import pandas as pd

import numpy as np

# Create a sample DataFrame

data = {

'A': np.random.randint(0, 100, 10),

'B': np.random.normal(50, 10, 10),

'C': np.random.uniform(0, 1, 10)

}

df = pd.DataFrame(data)

# Displaying the DataFrame

print("DataFrame:")

print(df)

print()

# Statistical measures using Pandas

print("Statistical Measures:")

print("Mean:")

print(df.mean()) # Mean of each column

print("\nMedian:")

print(df.median()) # Median of each column

print("\nStandard Deviation:")

print(df.std()) # Standard deviation of each column

print("\nMinimum Values:")

print(df.min()) # Minimum value of each column

print("\nMaximum Values:")

print(df.max()) # Maximum value of each column

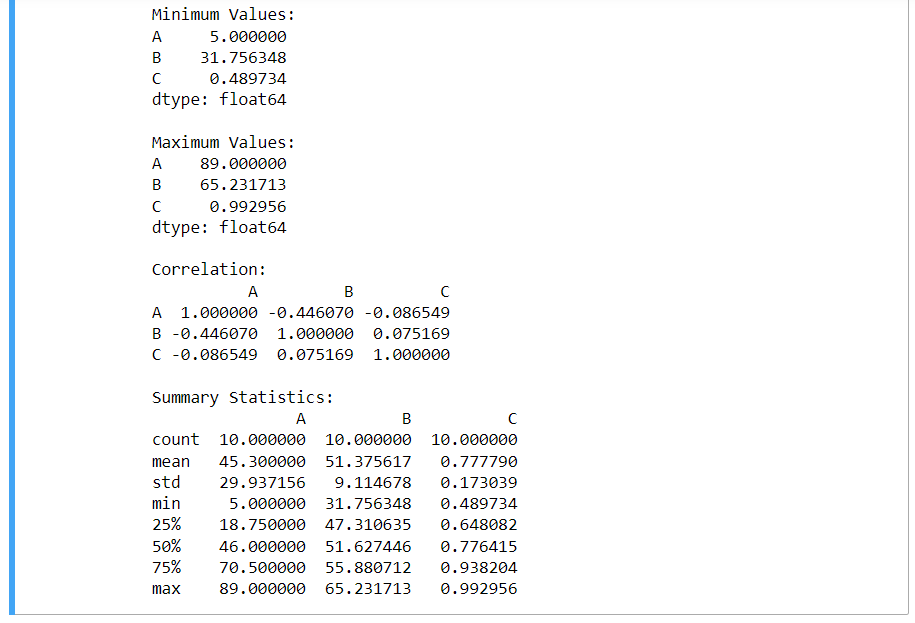
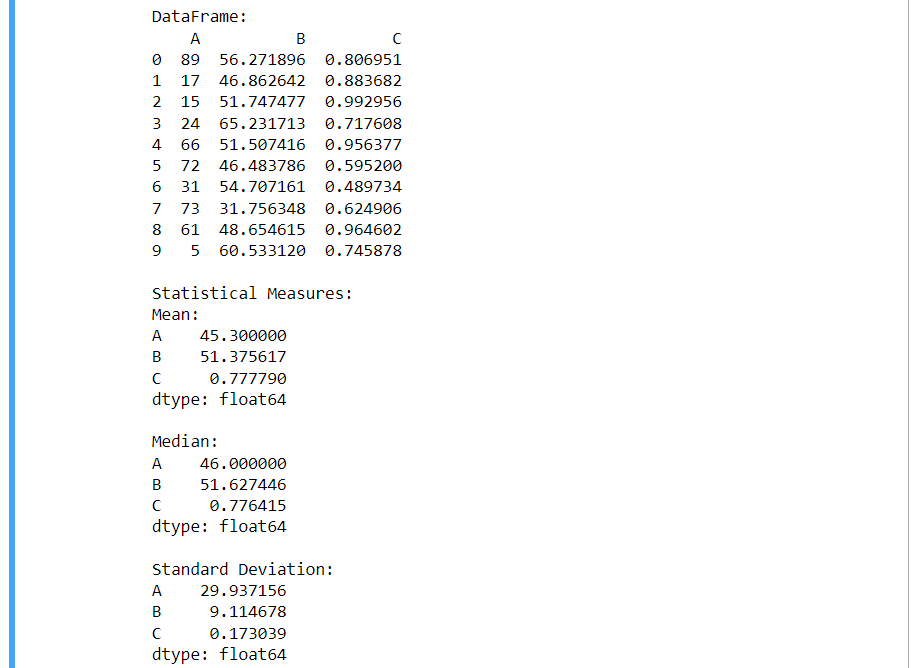
print("\nCorrelation:")

print(df.corr()) # Correlation matrix

print("\nSummary Statistics:")

print(df.describe()) # Summary statistics for numerical columns

**Output**



**……**

**End of Program23**

**Program24**

**Aim Write a program to plot a bar chart in python to display the result of a school for five consecutive years**

**Code**

import matplotlib.pyplot as plt

# Years and corresponding results

years = ['Year 1', 'Year 2', 'Year 3', 'Year 4', 'Year 5']

results = [85, 92, 88, 78, 90] # Sample results for each year

# Creating the bar chart

plt.figure(figsize=(8, 6)) # Set the figure size

plt.bar(years, results, color='skyblue')

# Adding labels and title

plt.xlabel('Years')

plt.ylabel('Results')

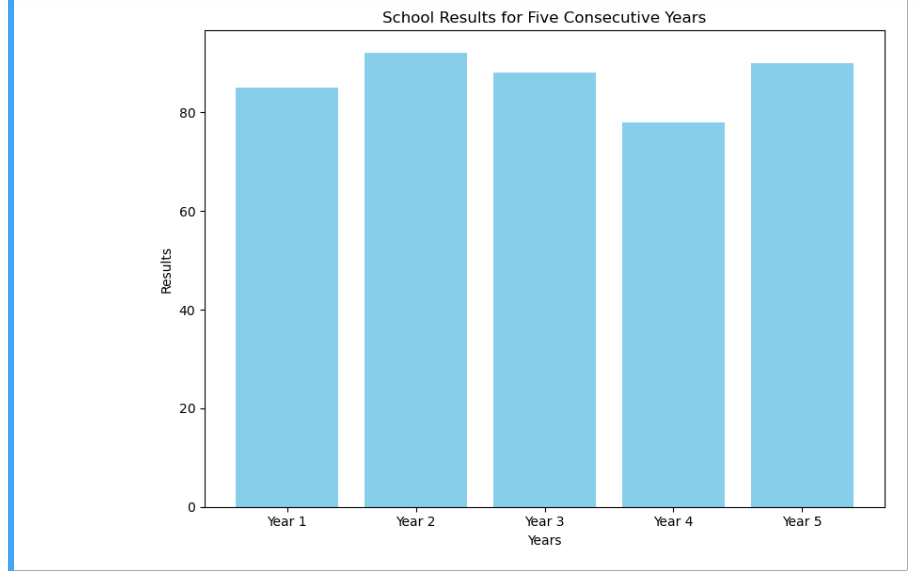
plt.title('School Results for Five Consecutive Years')

# Display the plot

plt.tight\_layout() # Adjust layout to prevent cropping

plt.show()

**Output**



**……**

**End of Program24**

**Program25**

**Aim Write a program in python to plot a graph for the function y = x^2**

**Code**

import matplotlib.pyplot as plt

import numpy as np

# Generating x values

x = np.linspace(-10, 10, 100) # Creating 100 points from -10 to 10

# Calculating y values (y = x^2)

y = x\*\*2

# Plotting the graph

plt.figure(figsize=(8, 6))

plt.plot(x, y, label='y = x^2', color='blue')

# Adding labels and title

plt.xlabel('x')

plt.ylabel('y')

plt.title('Graph of y = x^2')

plt.legend()

# Display the plot

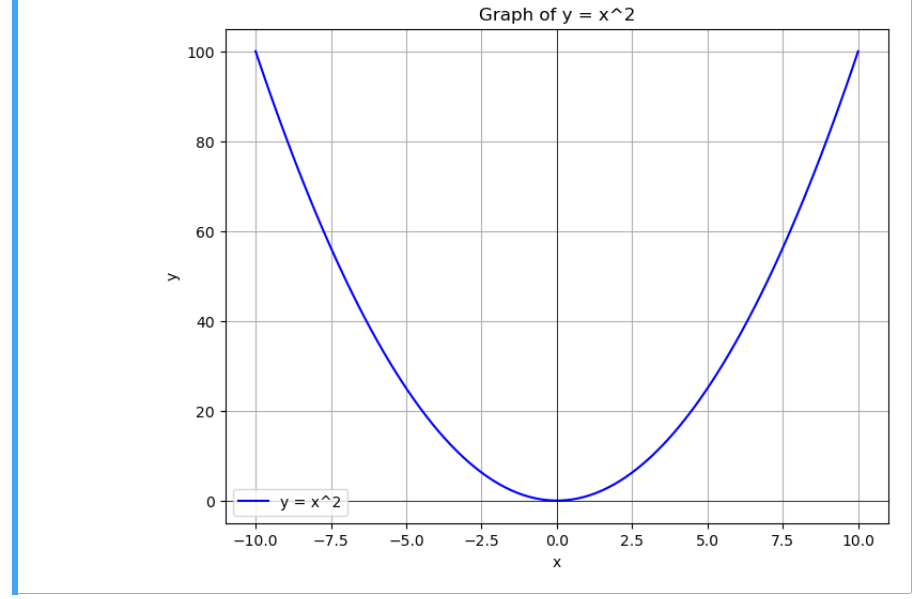
plt.grid(True)

plt.axhline(0, color='black',linewidth=0.5)

plt.axvline(0, color='black',linewidth=0.5)

plt.show()

**Output**



**……**

**End of Program25**

**Program26**

**Aim Write a program in python to plot a pie chart on consumption of water in daily life**

**Code**

import matplotlib.pyplot as plt

# Data for water consumption categories

categories = ['Drinking', 'Cooking', 'Bathing', 'Cleaning', 'Other']

consumption = [40, 10, 25, 20, 5] # Percentage of water consumption for each category

# Explode the 'Drinking' category

explode = (0.1, 0, 0, 0, 0) # Explode the 1st slice (Drinking) to highlight it

# Creating the pie chart

plt.figure(figsize=(8, 6))

plt.pie(consumption, labels=categories, explode=explode, autopct='%1.1f%%', startangle=140)

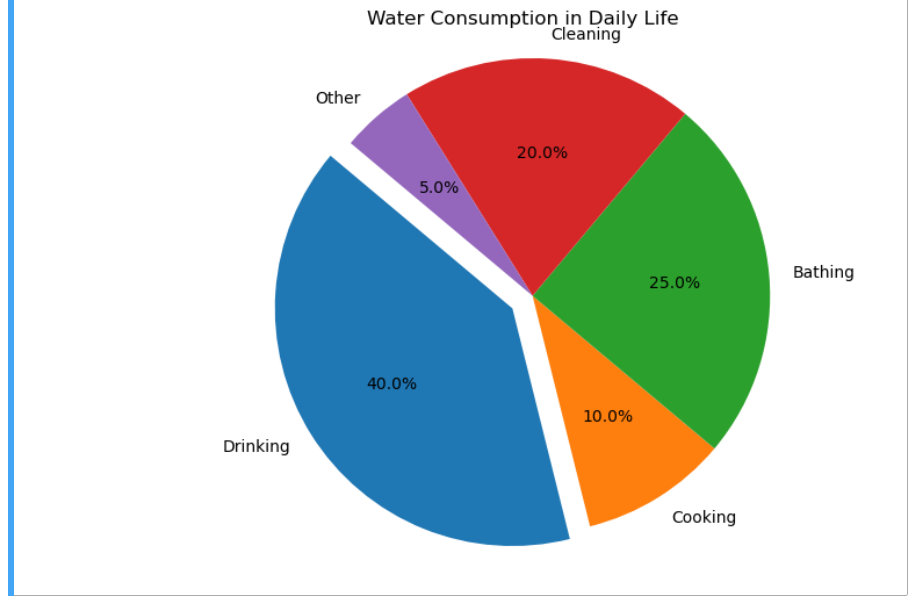
plt.title('Water Consumption in Daily Life')

# Display the plot

plt.axis('equal') # Equal aspect ratio ensures that pie is drawn as a circle.

plt.show()

**Output**



**……**

**End of Program26**