Roll No. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Exam No. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Dr D Y Patil Educational Enterprises Charitable Trust’s

**DR D Y PATIL SCHOOL OF MCA**

Dr. D.Y. Patil Knowledge City, Charholi (Bk.), Via Lohegaon, Pune – 412105



**CERTIFICATE**

This is to certify that

Mr./ Miss **Hrishikesh Dasharath Parab**, Of Class **MCA – First year (Sem- II) Div. C** Roll No. **340** Has completed all the practical work in the subject **Python Programming (IT21L)** satisfactorily in the Department of **MCA** as prescribed by University of Pune, in the academic year 2023 – 2024.

University Seat No. \_\_\_\_\_\_\_\_\_\_\_\_\_

Staff In-charge Head of the Department Director

|  |  |
| --- | --- |
|  | Dr D Y Patil Educational Enterprises Charitable Trust’s  **DR D Y PATIL SCHOOL OF MCA**  Dr. D.Y. Patil Knowledge City, Charholi (Bk.), Via Lohegaon, Pune – 412105 |

**INDEX**

Name : **Hrishikesh Dasharath Parab** Div : **C** Roll No. **340**

Class : **MCA I year (SEM II)** Subject : **Python Programming**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr. No.** | **Title of Program** | **Page no** | **Remark** |
| 1. | Write a Python program to check whether the no is even and odd. |  |  |
| 2. | Write a Python program to display Greater no between 3 inputs. |  |  |
| 3. | Write a Python program to print prime no between n and m. |  |  |
| 4. | Write a Python program to display Fibonacci series. |  |  |
| 5. | Write a Python program factorial of no. |  |  |
| 6. | Write a Python script to generate and print a dictionary that contains a number (between 1 and n) in the form  (x, x\*x). |  |  |
|  |
| Sample Dictionary ( n = 5) : |
| Expected Output : {1: 1, 2: 4, 3: 9, 4: 16, 5: 25} |
| 7. | Write a Python script to concatenate following dictionaries to create a new one. |  |  |
| dic1={1:10, 2:20} |
| dic2={3:30, 4:40} |
| dic3={5:50,6:60} |
| Expected Result : {1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60} |
|  |
| 8. | Write a Python program to perform star pattern. |  |  |
| \* |
| \*\* |
| \*\*\* |
| \*\*\*\* |
| 9. | Write a Python program to perform star Pattern. |  |  |
| \*\*\*\* |
| \*\*\* |
| \*\* |
| \* |
| 10. | Write a Python program to construct the following pattern, using a nested loop number. |  |  |
| 687954231 |
| 87954231 |
| 7954231 |
| 954231 |
| 54231 |
| 4231 |
| 231 |
| 31 |
| 1 |
| 11. | Write a Python program to get a string from a given string where all occurrences of its first char have been changed to '$', except the first char itself |  |  |
|  |
| Sample String : 'restart' |
| Expected Result : 'resta$t' |
| 12. | Write a Python program to remove the characters which have odd index values of a given string |  |  |
| 13. | Write a Python program to get a string made of the first 2 and the last 2 chars from a given a string. If the string length is less than 2, return instead of the empty string |  |  |
| Sample String : “Welcome to DYPSOMCA” |
| Expected Result : WTDYP |
| Sample String : 'We' |
| Expected Result : 'WeWe' |
| Sample String : ' w' |
| Expected Result : Empty String |
| 14. | Write a Python class to implement pow(x, n). |  |  |
| 15. | Write a Python class named Circle constructed by a radius and two methods which will compute the area and the perimeter of a circle. |  |  |
| 16. | Write a Python program to get the factorial of a non-negative integer using recursion. |  |  |
| 17. | Write a Python program to read first n lines of a file |  |  |
| 18. | Write a Python program to read a file line by line store it into a variable. |  |  |
| 19. | Write a Python program to combine each line from first file with the corresponding line in second file. |  |  |
| 20. | Write a program in which you need to calculate size of rectangle, square, circle and Triangle.(Separate Package need to be created for the same) |  |  |
| 21. | Write a program for Decorator and Generator. |  |  |
| 22. | Write a program to validate URL using regular expression. |  |  |
| 23. | Write a program to validate email using regular expression. |  |  |
| 24. | Write a program to validate Password using regular expression. |  |  |
| 25. | Write a program that ask for an integer number. Accepted number should be in between 1 to 10 and break the loop if not then generate the exception and print an error message. |  |  |
| 26. | Write a Python Program to perform synchronized Multi-threading. |  |  |
| 27. | Write a Program for Performing basic CRUD operations with MongoDB and python. |  |  |
| 28. | Write a Program Create Numpy Array with Random Values – numpy.random.rand() |  |  |
| 29. | Write a Program Numpy array and Reverse the Array. |  |  |
| 30. | Write a Programs for series and data frames. |  |  |
| 31. | Write a Program for data visualization using Matplotlib. |  |  |

1. **Write a Python program to check whether the no is even and odd.**

def check\_even\_odd(num):

if num % 2 == 0:

print(f"{num} is even.")

else:

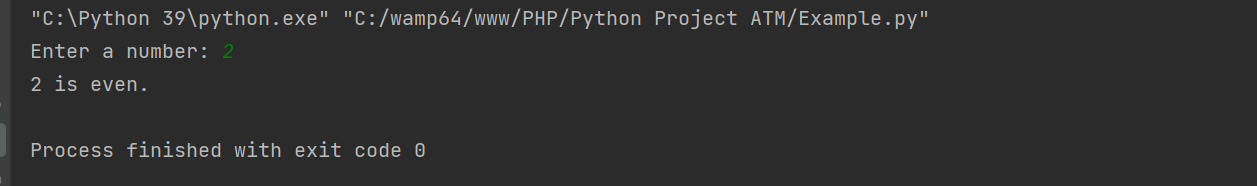
print(f"{num} is odd.")

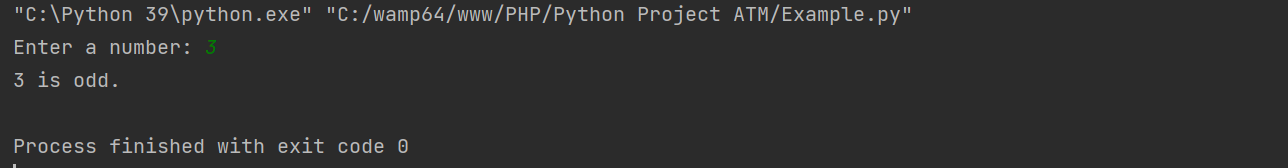
# Example usage

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

check\_even\_odd(num)

**OUTPUT:**





1. **Write a Python program to display Greater no between 3 inputs.**

def find\_greatest(num1, num2, num3):

greatest = num1

if num2 > greatest:

greatest = num2

if num3 > greatest:

greatest = num3

return greatest

# Input from the user

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

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

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

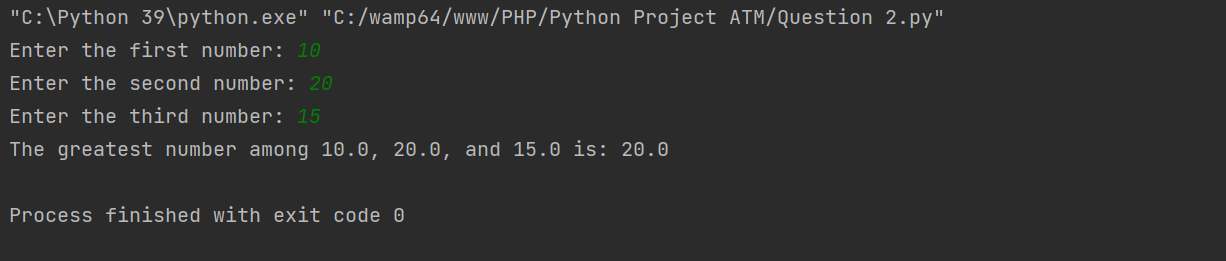
# Finding the greatest number

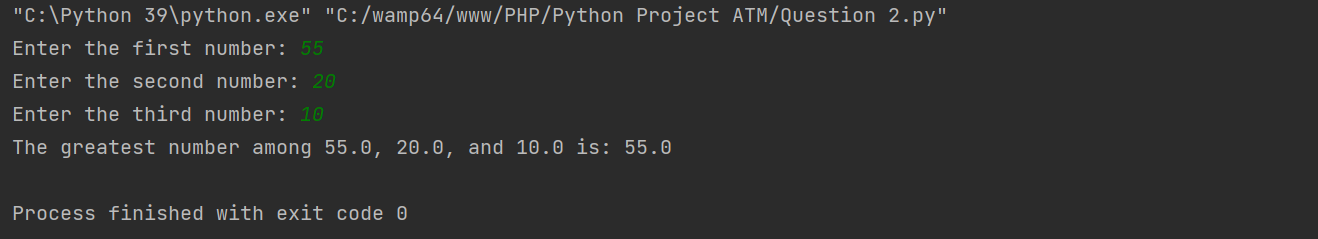
result = find\_greatest(num1, num2, num3)

# Displaying the result

print(f"The greatest number among {num1}, {num2}, and {num3} is: {result}")

**OUTPUT:**



****

1. **Write a Python program to print prime no between n and m.**

def is\_prime(num):

if num <= 1:

return False

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

if num % i == 0:

return False

return True

def print\_prime\_numbers(n, m):

print(f"Prime numbers between {n} and {m}:")

for num in range(n, m + 1):

if is\_prime(num):

print(num, end=" ")

# Input from the user

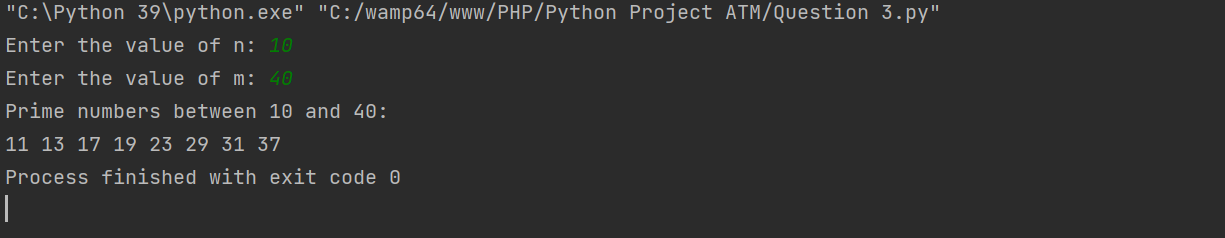
n = int(input("Enter the value of n: "))

m = int(input("Enter the value of m: "))

# Displaying prime numbers between n and m

print\_prime\_numbers(n, m)

**OUTPUT:**

****

1. **Write a Python program to display Fibonacci series.**

def fibonacci(n):

fib\_series = [0, 1] # Initialize the Fibonacci series with the first two terms

if n <= 0:

print("Please enter a positive integer.")

elif n == 1:

print("Fibonacci series up to", n, "term:", fib\_series[:1])

else:

while len(fib\_series) < n:

next\_term = fib\_series[-1] + fib\_series[-2]

fib\_series.append(next\_term)

print("Fibonacci series up to", n, "terms:", fib\_series)

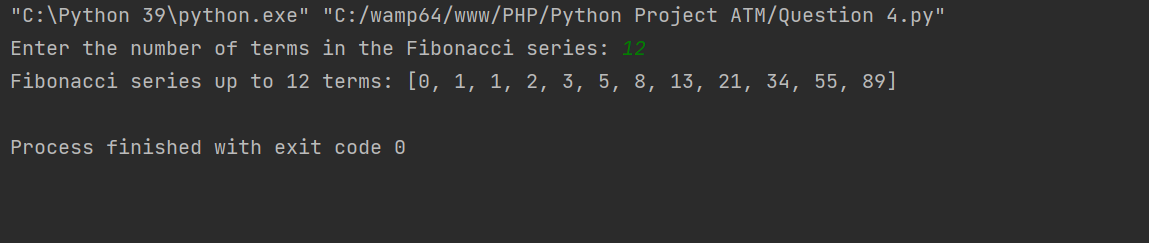
# Input from the user

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

# Displaying the Fibonacci series

fibonacci(num\_terms)

**OUTPUT:**



1. **Write a Python program factorial of no.**

def factorial(n):

if n < 0:

return "Factorial is not defined for negative numbers."

elif n == 0:

return 1

else:

result = 1

for i in range(1, n + 1):

result \*= i

return result

# Input from the user

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

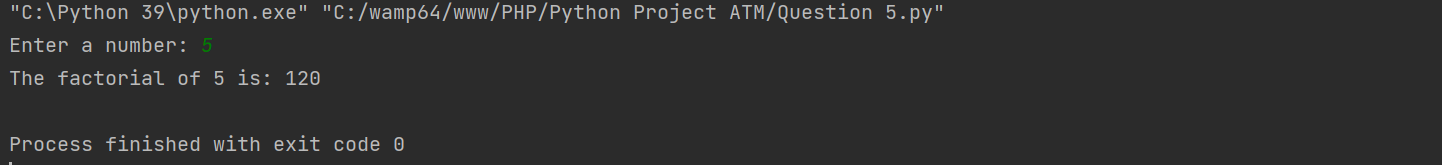
# Calculating factorial

fact = factorial(num)

# Displaying the result

print(f"The factorial of {num} is: {fact}")

**OUTPUT:**

****

1. **Write a Python script to generate and print a dictionary that contains a number (between 1 and n) in the form (x, x\*x).**

def generate\_square\_dictionary(n):

square\_dict = {}

for x in range(1, n + 1):

square\_dict[x] = x \* x

return square\_dict

# Input from the user

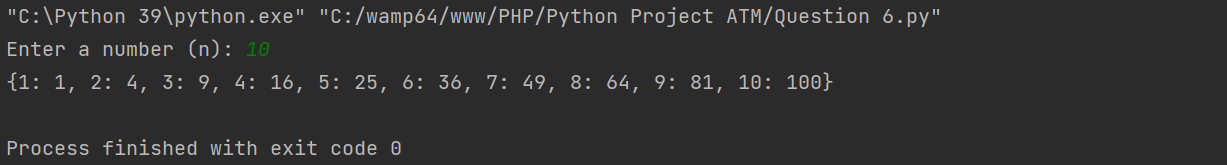
n = int(input("Enter a number (n): "))

# Generate the dictionary

square\_dict = generate\_square\_dictionary(n)

print(square\_dict)

**OUTPUT:**



1. **Write a Python script to concatenate following dictionaries to create a new one.**

**dic1={1:10, 2:20}**

**dic2={3:30, 4:40}**

**dic3={5:50,6:60}**

**Expected Result : {1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}**

# Given dictionaries

dic1 = {1: 10, 2: 20}

dic2 = {3: 30, 4: 40}

dic3 = {5: 50, 6: 60}

# Concatenating dictionaries

result\_dict = {}

result\_dict.update(dic1)

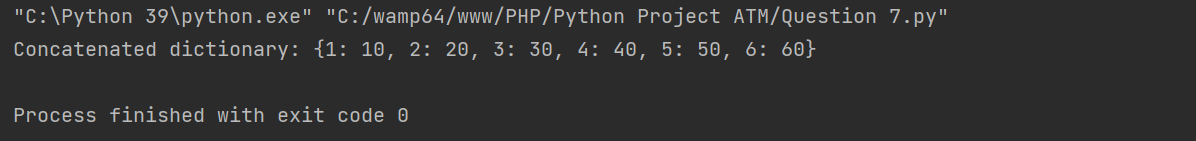
result\_dict.update(dic2)

result\_dict.update(dic3)

# Printing the result

print("Concatenated dictionary:", result\_dict)

**OUTPUT:**



1. **Write a Python program to perform star pattern.**

**\***

**\*\***

**\*\*\***

**\*\*\*\***

def print\_star\_pattern(rows):

for i in range(1, rows + 1):

for j in range(i):

print("\*", end="")

print()

# Input from the user

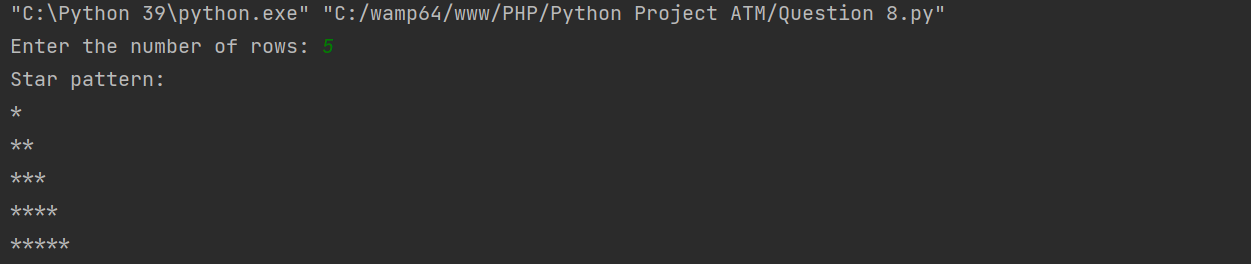
rows = int(input("Enter the number of rows: "))

# Print the star pattern

print("Star pattern:")

print\_star\_pattern(rows)

**OUTPUT:**



1. **Write a Python program to perform star Pattern.**

**\*\*\*\***

**\*\*\***

**\*\***

**\***

def print\_star\_pattern(rows):

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

for j in range(i):

print("\*", end="")

print()

# Input from the user

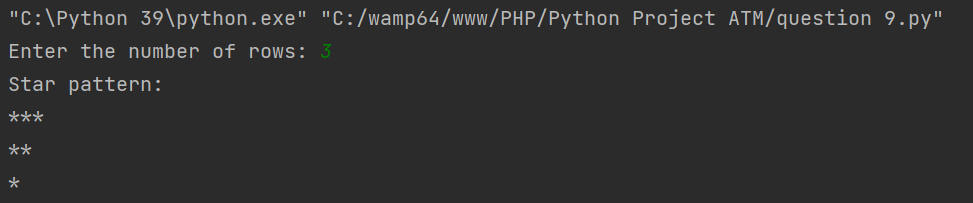
rows = int(input("Enter the number of rows: "))

# Print the star pattern

print("Star pattern:")

print\_star\_pattern(rows)

**OUTPUT:**



1. **Write a Python program to construct the following pattern, using a nested loop number.**

**687954231**

**87954231**

**7954231**

**954231**

**54231**

**4231**

**231**

**31**

**1**

def print\_pattern(n):

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

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

print(j, end="")

print()

# Input from the user

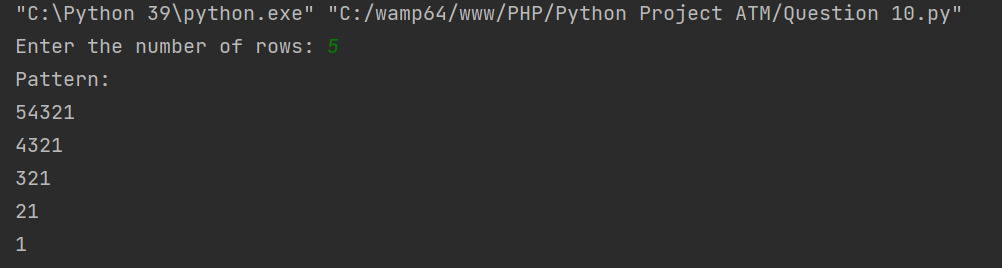
rows = int(input("Enter the number of rows: "))

# Print the pattern

print("Pattern:")

print\_pattern(rows)

**OUTPUT:**



1. **Write a Python program to get a string from a given string where all occurrences of its first char have been changed to '$', except the first char itself**

**Sample String : 'restart'**

**Expected Result : 'resta$t'**

def change\_char(string):

first\_char = string[0]

modified\_string = first\_char

for char in string[1:]:

if char == first\_char:

modified\_string += '$'

else:

modified\_string += char

return modified\_string

# Sample string

sample\_string = 'restart'

# Get the modified string

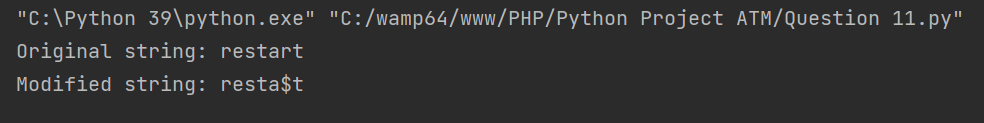
result = change\_char(sample\_string)

# Print the result

print("Original string:", sample\_string)

print("Modified string:", result)

**OUTPUT:**



1. **Write a Python program to remove the characters which have odd index values of a given string**

def remove\_odd\_index\_chars(string):

result = ""

for i in range(len(string)):

if i % 2 == 0:

result += string[i]

return result

# Sample string

sample\_string = 'abcdefgh'

# Remove characters with odd index values

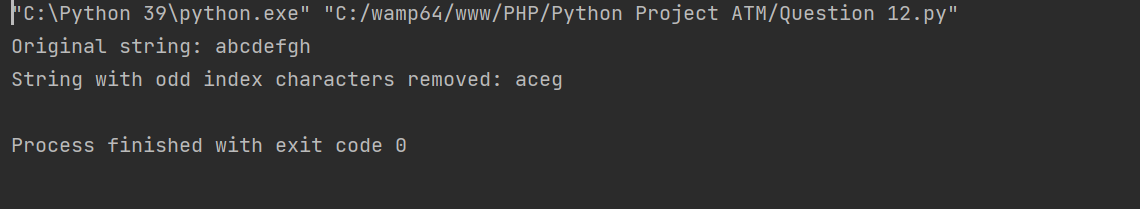
result = remove\_odd\_index\_chars(sample\_string)

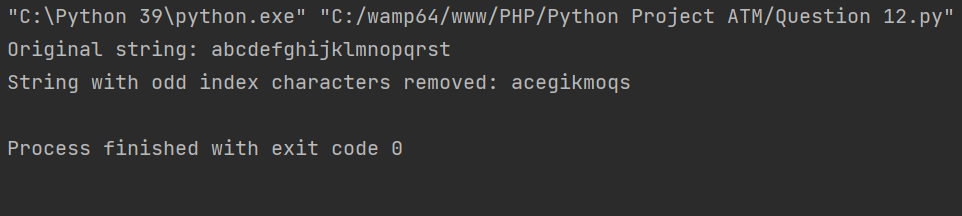
# Print the result

print("Original string:", sample\_string)

print("String with odd index characters removed:", result)

**OUTPUT:**

****

****

1. **Write a Python program to get a string made of the first 2 and the last 2 chars from a given a string. If the string length is less than 2, return instead of the empty string**

**Sample String : “Welcome to DYPSOMCA”**

**Expected Result : WTDYP**

**Sample String : 'We'**

**Expected Result : 'WeWe'**

**Sample String : ' w'**

**Expected Result : Empty String**

def get\_first\_last\_chars(string):

if len(string) < 2:

return ""

first\_two\_chars = string[:2]

last\_two\_chars = string[-2:]

return first\_two\_chars + last\_two\_chars

# Sample strings

sample\_strings = ["Welcome to DYPSOMCA", "We", " w"]

# Iterate through sample strings

for sample\_string in sample\_strings:

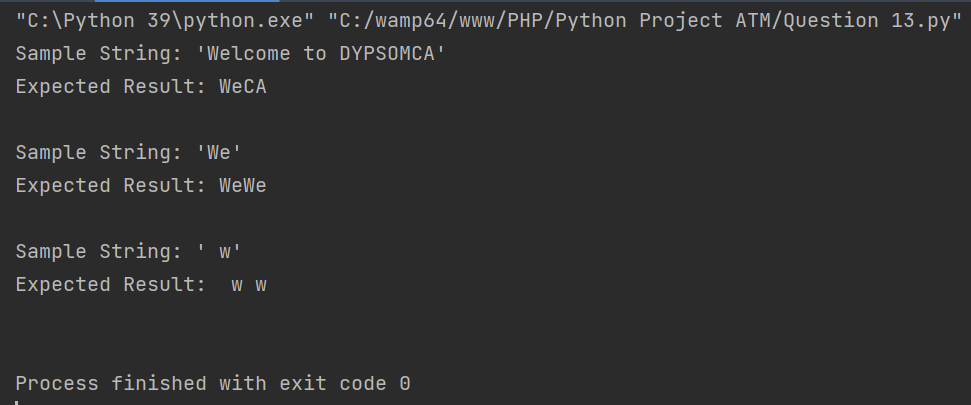
result = get\_first\_last\_chars(sample\_string)

print(f"Sample String: '{sample\_string}'")

print("Expected Result:", result)

print()

**OUTPUT:**

****

1. **Write a Python class to implement pow(x, n).**

class PowerCalculator:

def calculate\_power(self, x, n):

if n == 0:

return 1

elif n < 0:

return 1 / self.calculate\_power(x, -n)

elif n % 2 == 0:

return self.calculate\_power(x\*x, n/2)

else:

return x \* self.calculate\_power(x\*x, (n-1)/2)

# Create an instance of the PowerCalculator class

calculator = PowerCalculator()

# Test the calculate\_power method

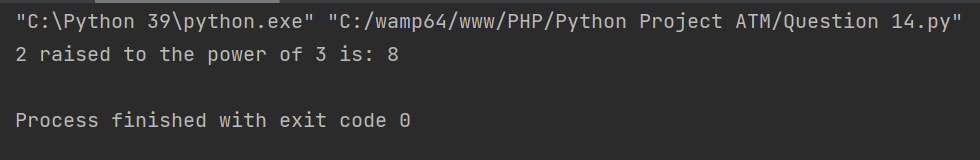
x = 2

n = 3

result = calculator.calculate\_power(x, n)

print(f"{x} raised to the power of {n} is:", result)

**OUTPUT:**

****

1. **Write a Python class named Circle constructed by a radius and two methods which will compute the area and the perimeter of a circle.**

import math

class Circle:

def \_\_init\_\_(self, radius):

self.radius = radius

def area(self):

return math.pi \* self.radius\*\*2

def perimeter(self):

return 2 \* math.pi \* self.radius

# Create an instance of the Circle class

radius = float(input("Enter the radius of the circle: "))

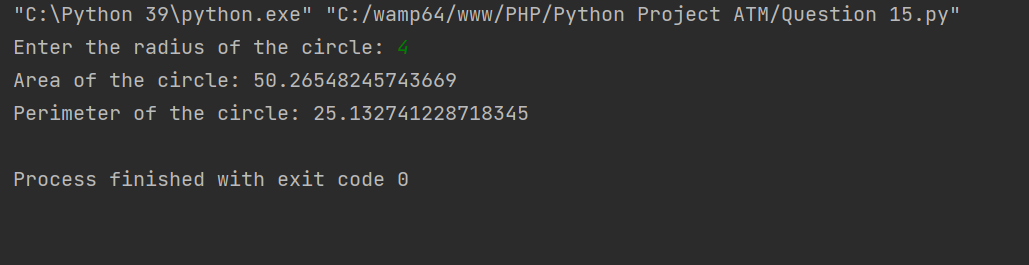
circle = Circle(radius)

# Calculate and display the area and perimeter of the circle

print("Area of the circle:", circle.area())

print("Perimeter of the circle:", circle.perimeter())

**OUTPUT:**



1. **Write a Python program to get the factorial of a non-negative integer using recursion.**

def factorial(n):

if n == 0:

return 1

else:

return n \* factorial(n - 1)

# Input from the user

num = int(input("Enter a non-negative integer: "))

# Calculate and display the factorial

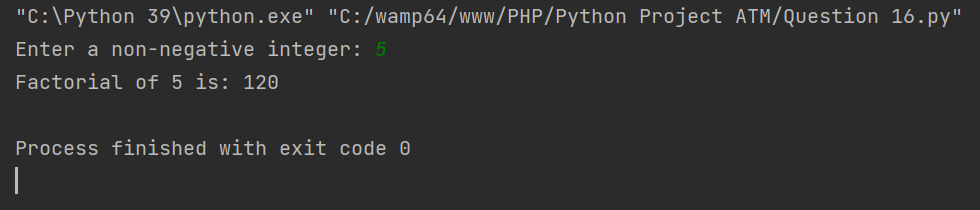
if num < 0:

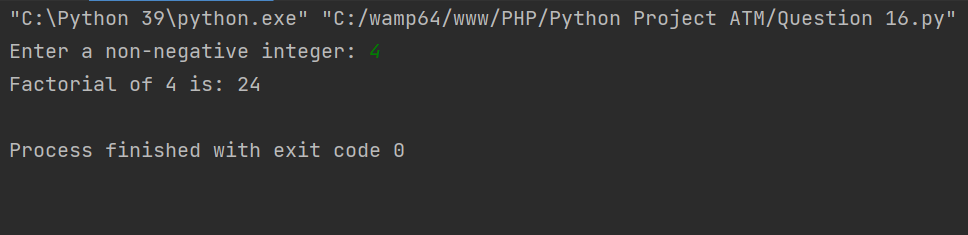
print("Factorial is not defined for negative numbers.")

else:

print("Factorial of", num, "is:", factorial(num))

**OUTPUT:**

****

****

1. **Write a Python program to read first n lines of a file**

def read\_first\_n\_lines(file\_name, n):

try:

with open(file\_name, 'r') as file:

lines = file.readlines()

for line in lines[:n]:

print(line.strip())

except FileNotFoundError:

print("File not found!")

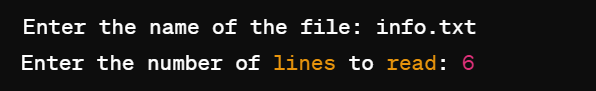
if \_\_name\_\_ == "\_\_main\_\_":

file\_name = input("Enter the name of the file: ")

n = int(input("Enter the number of lines to read: "))

read\_first\_n\_lines(file\_name, n)

**OUTPUT:**





1. **Write a Python program to read a file line by line store it into a variable.**

def read\_file\_into\_variable(file\_name):

try:

with open(file\_name, 'r') as file:

content = file.read()

return content

except FileNotFoundError:

print("File not found!")

if \_\_name\_\_ == "\_\_main\_\_":

file\_name = input("Enter the name of the file: ")

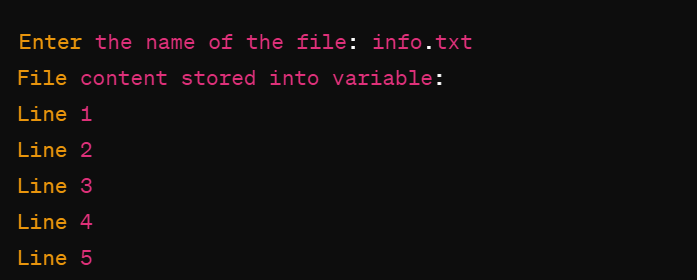
file\_content = read\_file\_into\_variable(file\_name)

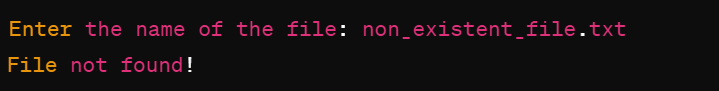
if file\_content:

print("File content stored into variable:")

print(file\_content)

**OUTPUT:**





1. **Write a Python program to combine each line from first file with the corresponding line in second file.**

def combine\_files(file1, file2, output\_file):

try:

with open(file1, 'r') as f1, open(file2, 'r') as f2, open(output\_file, 'w') as out\_file:

for line1, line2 in zip(f1, f2):

combined\_line = line1.strip() + ' ' + line2.strip() + '\n'

out\_file.write(combined\_line)

print("Files combined successfully!")

except FileNotFoundError:

print("One or more files not found!")

if \_\_name\_\_ == "\_\_main\_\_":

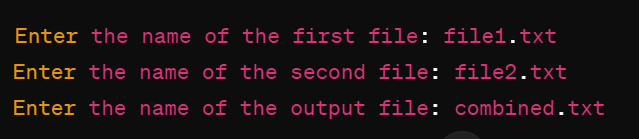
file1 = input("Enter the name of the first file: ")

file2 = input("Enter the name of the second file: ")

output\_file = input("Enter the name of the output file: ")

combine\_files(file1, file2, output\_file)

**OUTPUT:**



1. **Write a program in which you need to calculate size of rectangle, square, circle and Triangle.(Separate Package need to be created for the same)**

Here's the directory structure:

shapes/

\_\_init\_\_.py

rectangle.py

square.py

circle.py

triangle.py

main.py

**rectangle.py**

class Rectangle:

def \_\_init\_\_(self, length, width):

self.length = length

self.width = width

def area(self):

return self.length \* self.width

**square.py**

class Square:

def \_\_init\_\_(self, side):

self.side = side

def area(self):

return self.side \* self.side

**circle.py**

import math

class Circle:

def \_\_init\_\_(self, radius):

self.radius = radius

def area(self):

return math.pi \* self.radius \* self.radius

**triangle.py**

class Triangle:

def \_\_init\_\_(self, base, height):

self.base = base

self.height = height

def area(self):

return 0.5 \* self.base \* self.height

**main.py**

from shapes.rectangle import Rectangle

from shapes.square import Square

from shapes.circle import Circle

from shapes.triangle import Triangle

# Example usage

if \_\_name\_\_ == "\_\_main\_\_":

rectangle = Rectangle(5, 4)

square = Square(5)

circle = Circle(5)

triangle = Triangle(4, 3)

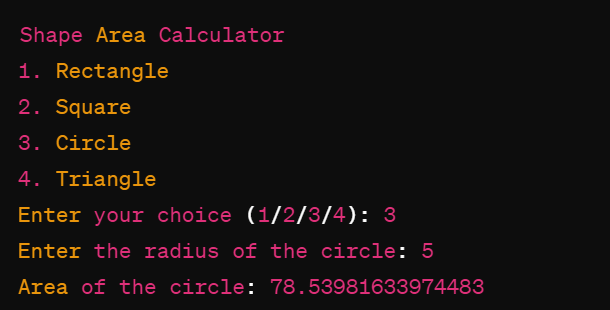
print("Area of Rectangle:", rectangle.area())

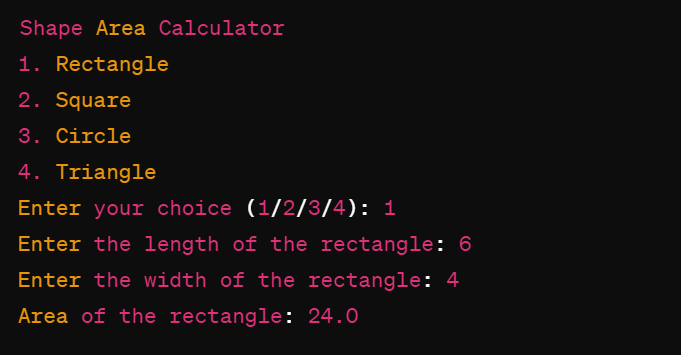
print("Area of Square:", square.area())

print("Area of Circle:", circle.area())

print("Area of Triangle:", triangle.area())

**OUTPUT:**





1. **Write a program for Decorator and Generator.**

**Decorator Example:**

*A decorator is a design pattern in Python that allows a user to add new functionality to an existing object without modifying its structure. Here's a simple example:*

# Decorator function

def my\_decorator(func):

def wrapper():

print("Something is happening before the function is called.")

func()

print("Something is happening after the function is called.")

return wrapper

# Function to be decorated

def say\_hello():

print("Hello!")

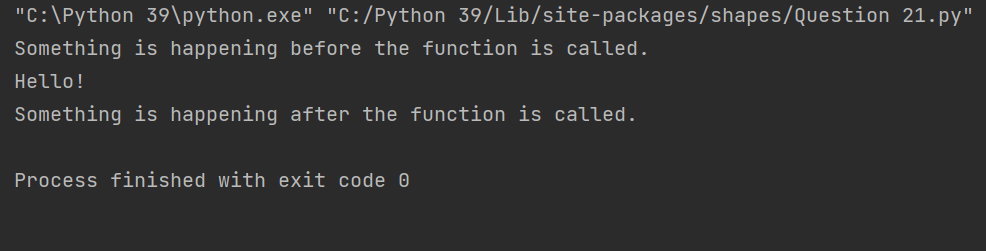
# Decorating the function

say\_hello = my\_decorator(say\_hello)

# Calling the decorated function

say\_hello()

**OUTPUT:**



**Generator Example:**

*A generator is a special type of iterator in Python that allows you to generate a sequence of values lazily, without needing to store the entire sequence in memory. Here's an example:*

# Generator function

def square\_numbers(nums):

for num in nums:

yield num \* num

# Using the generator function

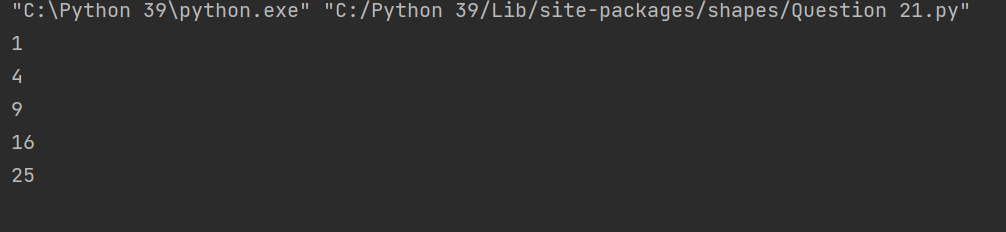
my\_nums = square\_numbers([1, 2, 3, 4, 5])

# Iterating over the generator

for num in my\_nums:

print(num)

**OUTPUT:**



1. **Write a program to validate URL using regular expression.**

import re

def validate\_url(url):

# Regular expression pattern for URL validation

pattern = re.compile(

r'^(?:http|https)://' # Scheme (http or https)

r'(?:[A-Z0-9-]+\.)+[A-Z]{2,6}' # Domain

r'(?:/?|[/?]\S+)$', re.IGNORECASE)

# Check if the URL matches the pattern

if re.match(pattern, url):

return True

else:

return False

# Test the function with some URLs

urls = [

"https://www.example.com",

"http://subdomain.example.com/page.html",

"ftp://example.com", # Invalid URL

"www.example.com", # Invalid URL (missing scheme)

"https://example" # Invalid URL (missing domain extension)

]

for url in urls:

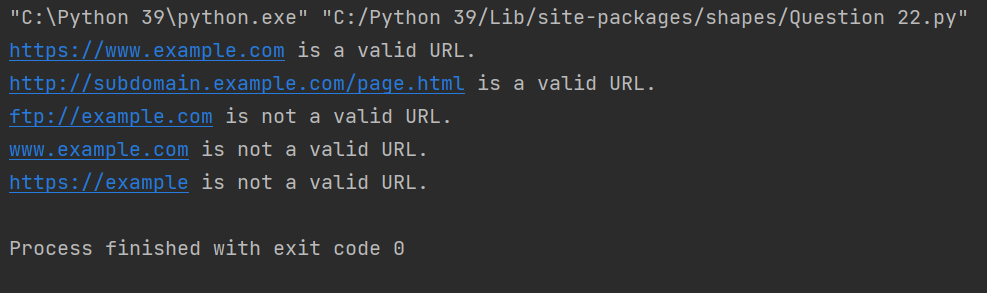
if validate\_url(url):

print(f"{url} is a valid URL.")

else:

print(f"{url} is not a valid URL.")

**OUTPUT:**



1. **Write a program to validate email using regular expression.**

import re

def validate\_email(email):

# Regular expression pattern for email validation

pattern = re.compile(r'^[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$')

# Check if the email matches the pattern

if re.match(pattern, email):

return True

else:

return False

# Test the function with some email addresses

emails = [

"user@example.com",

"user123@example.co.uk",

"user.name@example.domain.com",

"user123", # Invalid email

"user@example", # Invalid email

"user@.com" # Invalid email

]

for email in emails:

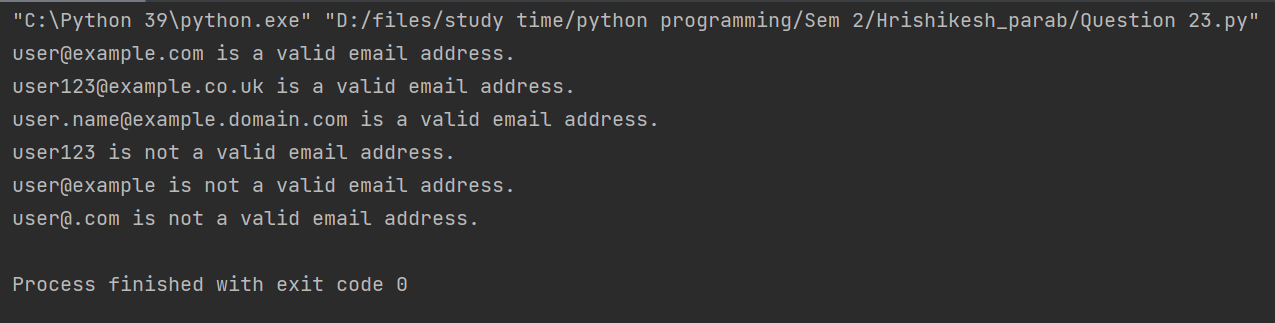
if validate\_email(email):

print(f"{email} is a valid email address.")

else:

print(f"{email} is not a valid email address.")

**OUTPUT:**



1. **Write a program to validate Password using regular expression.**

import re

def validate\_password(password):

# Regular expression pattern for password validation

pattern = re.compile(r'^(?=.\*[a-z])(?=.\*[A-Z])(?=.\*\d)(?=.\*[@$!%\*?&])[A-Za-z\d@$!%\*?&]{8,}$')

# Check if the password matches the pattern

if re.match(pattern, password):

return True

else:

return False

# Test the function with some passwords

passwords = [

"Password@123",

"abcDEF456!",

"12345678", # Invalid password (missing uppercase, lowercase, and special characters)

"Abcdefgh", # Invalid password (missing digits and special characters)

"P@ssw0rd", # Invalid password (missing uppercase character)

"PASSWORD123!", # Invalid password (missing lowercase character)

"p@ssword123" # Invalid password (missing uppercase character and special character)

]

for password in passwords:

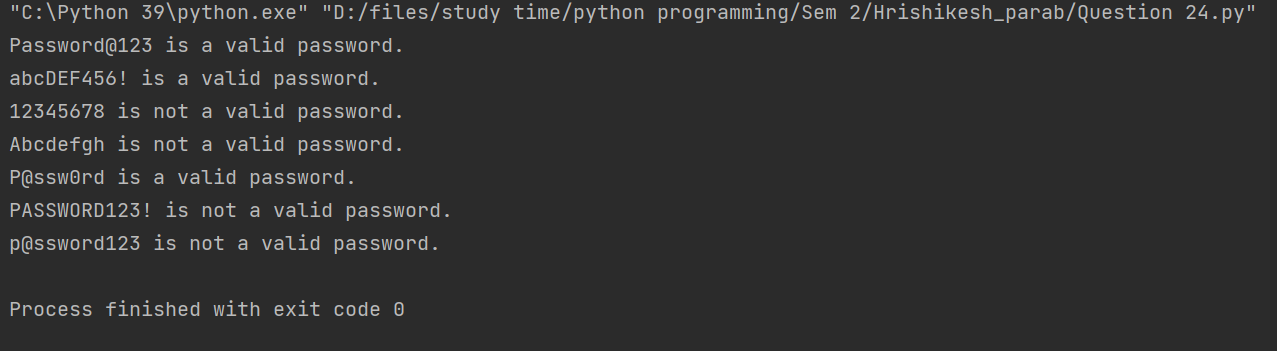
if validate\_password(password):

print(f"{password} is a valid password.")

else:

print(f"{password} is not a valid password.")

**OUTPUT:**



1. **Write a program that ask for an integer number. Accepted number should be in between 1 to 10 and break the loop if not then generate the exception and print an error message.**

while True:

try:

number = int(input("Enter an integer number between 1 and 10: "))

if 1 <= number <= 10:

print("Valid number entered:", number)

break

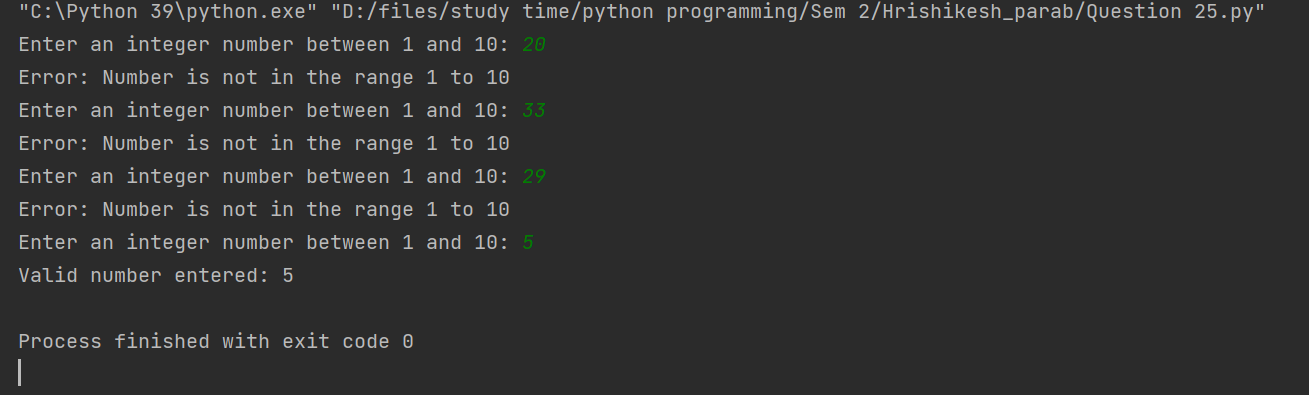
else:

raise ValueError("Number is not in the range 1 to 10")

except ValueError as e:

print("Error:", e)

**OUTPUT:**



1. **Write a Python Program to perform synchronized Multi-threading.**

import threading

# Shared resource

shared\_resource = 0

# Lock for synchronization

lock = threading.Lock()

# Function to increment the shared resource

def increment():

global shared\_resource

for \_ in range(1000000):

with lock: # Acquire the lock

shared\_resource += 1

print("Increment finished")

# Function to decrement the shared resource

def decrement():

global shared\_resource

for \_ in range(1000000):

with lock: # Acquire the lock

shared\_resource -= 1

print("Decrement finished")

# Create threads

thread1 = threading.Thread(target=increment)

thread2 = threading.Thread(target=decrement)

# Start threads

thread1.start()

thread2.start()

# Wait for threads to finish

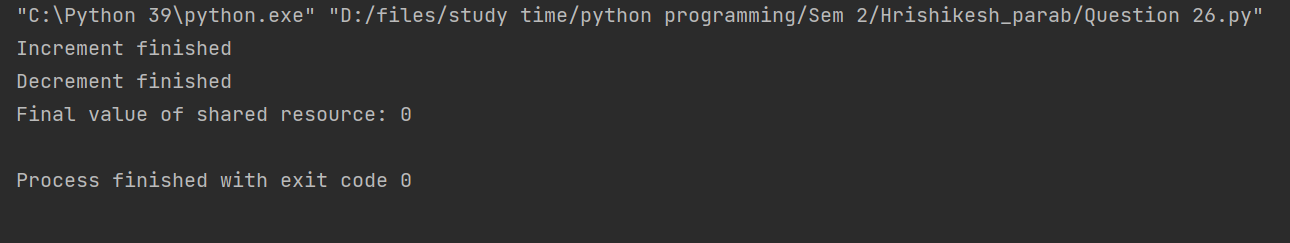
thread1.join()

thread2.join()

# Print the final value of the shared resource

print("Final value of shared resource:", shared\_resource)

**OUTPUT:**



1. **Write a Program for Performing basic CRUD operations with MongoDB and python.**

from pymongo import MongoClient

# Connect to MongoDB

client = MongoClient('mongodb://192.168.32.130:27017/')

db = client['test\_database']

collection = db['test\_collection']

# Create operation

def create\_document(data):

result = collection.insert\_one(data)

print(f"Document inserted with id: {result.inserted\_id}")

# Read operation

def read\_documents():

documents = collection.find()

for doc in documents:

print(doc)

# Update operation

def update\_document(query, new\_data):

result = collection.update\_one(query, new\_data)

print(f"{result.modified\_count} document(s) updated.")

# Delete operation

def delete\_document(query):

result = collection.delete\_one(query)

print(f"{result.deleted\_count} document(s) deleted.")

if \_\_name\_\_ == "\_\_main\_\_":

# Create

data = {"name": "John", "age": 30, "city": "New York"}

create\_document(data)

# Read

print("Documents in the collection:")

read\_documents()

# Update

query = {"name": "John"}

new\_data = {"$set": {"age": 35}}

update\_document(query, new\_data)

# Read after update

print("Documents in the collection after update:")

read\_documents()

# Delete

delete\_query = {"name": "John"}

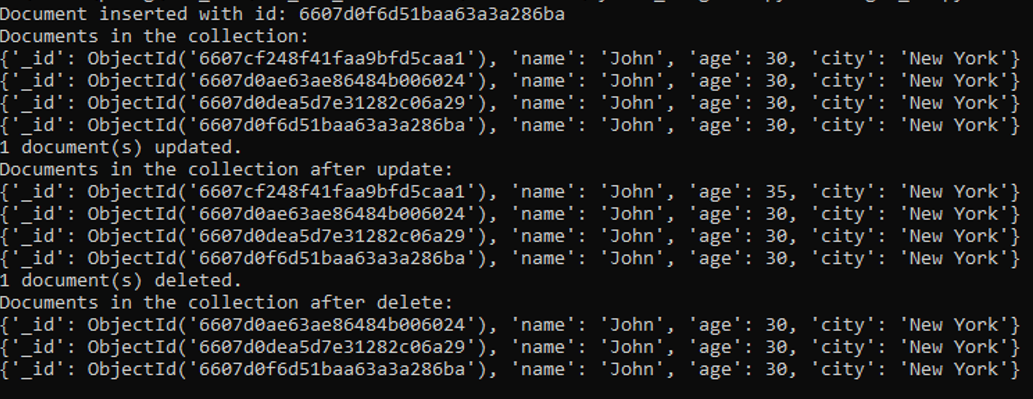
delete\_document(delete\_query)

# Read after delete

print("Documents in the collection after delete:")

read\_documents()

**OUTPUT:**



1. **Write a Program Create Numpy Array with Random Values – numpy.random.rand()**

import numpy as np

def create\_random\_array(shape):

# Generate a NumPy array with random values

random\_array = np.random.rand(\*shape)

return random\_array

# Define the shape of the array

shape = (3, 4) # Example shape: 3 rows, 4 columns

# Create a random NumPy array with the specified shape

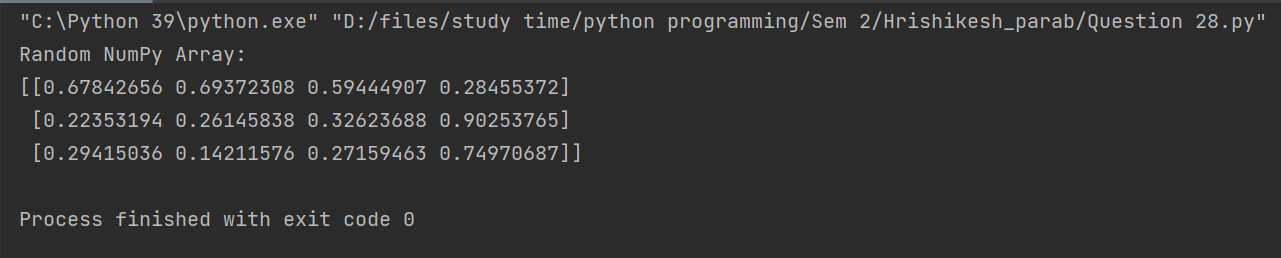
random\_array = create\_random\_array(shape)

# Print the random array

print("Random NumPy Array:")

print(random\_array)

**OUTPUT:**



1. **Write a Program Numpy array and Reverse the Array.**

import numpy as np

# Create a NumPy array

my\_array = np.array([1, 2, 3, 4, 5])

# Reverse the array

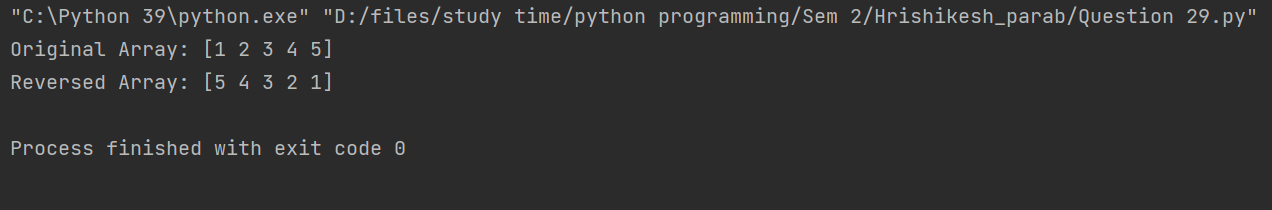
reversed\_array = np.flip(my\_array)

# Print the original and reversed arrays

print("Original Array:", my\_array)

print("Reversed Array:", reversed\_array)

**OUTPUT:**



1. **Write a Programs for series and data frames.**

import pandas as pd

# Create a Pandas Series

data = [1, 2, 3, 4, 5]

index = ['a', 'b', 'c', 'd', 'e']

series = pd.Series(data, index=index)

# Print the Series

print("Pandas Series:")

print(series)

# Create a Pandas DataFrame

data = {'Name': ['Alice', 'Bob', 'Charlie', 'David'],

'Age': [25, 30, 35, 40],

'City': ['New York', 'Los Angeles', 'Chicago', 'Houston']}

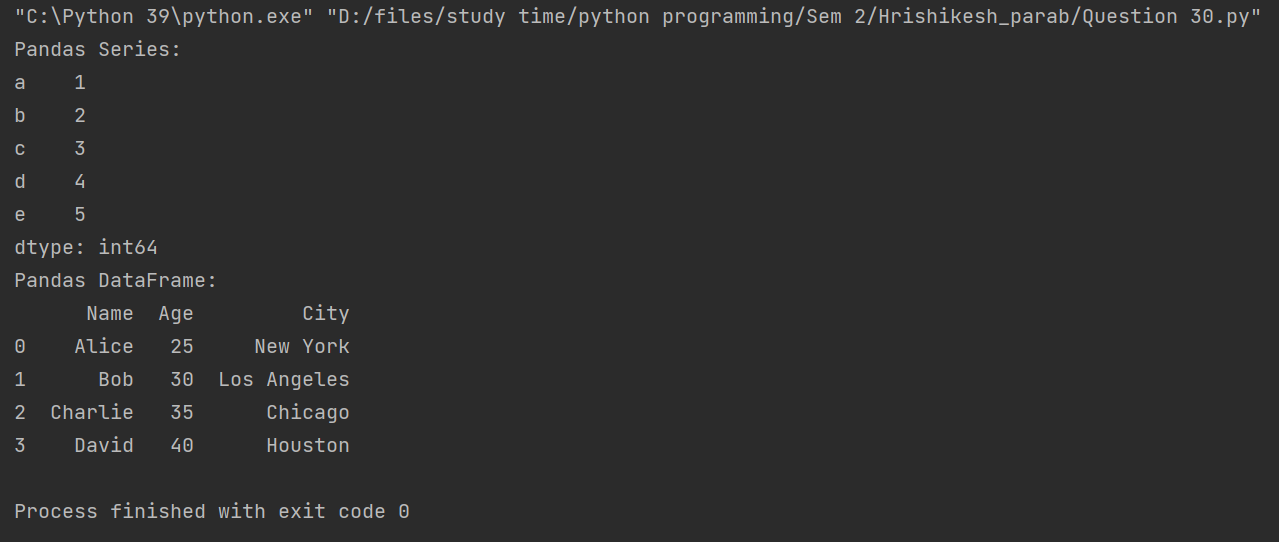
df = pd.DataFrame(data)

# Print the DataFrame

print("Pandas DataFrame:")

print(df)

**OUTPUT:**



1. **Write a Program for data visualization using Matplotlib.**

import matplotlib.pyplot as plt

# Sample data

x = [1, 2, 3, 4, 5]

y = [2, 3, 5, 7, 11]

# Create a line plot

plt.plot(x, y)

# Add title and labels

plt.title('Line Plot Example')

plt.xlabel('X-axis')

plt.ylabel('Y-axis')

# Show the plot

plt.show()

**OUTPUT:**

