08.03.2024

1))take any one list and perform all list python operations

# Example list

my\_list = [5, 12, 7, 3, 9, 15, 8, 10]

# 1. Access elements by index

print("1. Access elements by index:")

print("First element:", my\_list[0])

print("Last element:", my\_list[-1])

print("Third to fifth elements:", my\_list[2:5])

print()

# 2. Update elements by index

print("2. Update elements by index:")

my\_list[1] = 20

print("Updated list:", my\_list)

print()

# 3. Add elements to the end of the list

print("3. Add elements to the end of the list:")

my\_list.append(25)

print("List after appending 25:", my\_list)

print()

# 4. Insert element at a specific position

print("4. Insert element at a specific position:")

my\_list.insert(3, 30)

print("List after inserting 30 at index 3:", my\_list)

print()

# 5. Remove element by value

print("5. Remove element by value:")

my\_list.remove(9)

print("List after removing 9:", my\_list)

print()

# 6. Remove element by index

print("6. Remove element by index:")

removed\_element = my\_list.pop(4)

print("List after removing element at index 4:", my\_list)

print("Removed element:", removed\_element)

print()

# 7. Find index of an element

print("7. Find index of an element:")

index\_of\_8 = my\_list.index(8)

print("Index of element 8:", index\_of\_8)

print()

# 8. Check if an element is in the list

print("8. Check if an element is in the list:")

is\_15\_present = 15 in my\_list

print("Is 15 present in the list?", is\_15\_present)

print()

# 9. Length of the list

print("9. Length of the list:")

list\_length = len(my\_list)

print("Length of the list:", list\_length)

print()

# 10. Sorting the list

print("10. Sorting the list:")

my\_list.sort()

print("Sorted list:", my\_list)

print()

# 11. Reversing the list

print("11. Reversing the list:")

my\_list.reverse()

print("Reversed list:", my\_list)

print()

# 12. Clearing the entire list

print("12. Clearing the entire list:")

my\_list.clear()

print("List after clearing:", my\_list)

---------------------------------------------

output::

1. Access elements by index:

First element: 5

Last element: 10

Third to fifth elements: [7, 3, 9]

2. Update elements by index:

Updated list: [5, 20, 7, 3, 9, 15, 8, 10]

3. Add elements to the end of the list:

List after appending 25: [5, 20, 7, 3, 9, 15, 8, 10, 25]

4. Insert element at a specific position:

List after inserting 30 at index 3: [5, 20, 7, 30, 3, 9, 15, 8, 10, 25]

5. Remove element by value:

List after removing 9: [5, 20, 7, 30, 3, 15, 8, 10, 25]

6. Remove element by index:

List after removing element at index 4: [5, 20, 7, 30, 15, 8, 10, 25]

Removed element: 3

7. Find index of an element:

Index of element 8: 5

8. Check if an element is in the list:

Is 15 present in the list? True

9. Length of the list:

Length of the list: 8

10. Sorting the list:

Sorted list: [5, 7, 8, 10, 15, 20, 25, 30]

11. Reversing the list:

Reversed list: [30, 25, 20, 15, 10, 8, 7, 5]

12. Clearing the entire list:

List after clearing: []

Process finished with exit code 0

=====================================================================================================

2)))take any two sets and perform all set operations

# Example sets

set1 = {2, 4, 6, 8, 10}

set2 = {5, 8, 10, 12, 15}

# 1. Union of sets

print("1. Union of sets:")

union\_set = set1.union(set2)

print("Union set:", union\_set)

print()

# 2. Intersection of sets

print("2. Intersection of sets:")

intersection\_set = set1.intersection(set2)

print("Intersection set:", intersection\_set)

print()

# 3. Difference between sets

print("3. Difference between sets:")

difference\_set1 = set1.difference(set2)

difference\_set2 = set2.difference(set1)

print("Difference set (set1 - set2):", difference\_set1)

print("Difference set (set2 - set1):", difference\_set2)

print()

# 4. Symmetric difference between sets

print("4. Symmetric difference between sets:")

symmetric\_difference\_set = set1.symmetric\_difference(set2)

print("Symmetric difference set:", symmetric\_difference\_set)

print()

# 5. Check if one set is subset of another

print("5. Check if one set is subset of another:")

is\_subset = set1.issubset(set2)

print("Is set1 a subset of set2?", is\_subset)

print()

# 6. Check if sets are disjoint

print("6. Check if sets are disjoint:")

are\_disjoint = set1.isdisjoint(set2)

print("Are set1 and set2 disjoint?", are\_disjoint)

print()

# 7. Adding elements to a set

print("7. Adding elements to a set:")

set1.add(12)

set2.update({6, 18, 20})

print("Updated set1:", set1)

print("Updated set2:", set2)

print()

# 8. Removing elements from a set

print("8. Removing elements from a set:")

set1.remove(8)

set2.discard(15)

print("Set1 after removing 8:", set1)

print("Set2 after discarding 15:", set2)

print()

# 9. Clearing the entire set

print("9. Clearing the entire set:")

set1.clear()

set2.clear()

print("Set1 after clearing:", set1)

print("Set2 after clearing:", set2)

---------------------------------------------------

output::::

1. Union of sets:

Union set: {2, 4, 5, 6, 8, 10, 12, 15}

2. Intersection of sets:

Intersection set: {8, 10}

3. Difference between sets:

Difference set (set1 - set2): {2, 4, 6}

Difference set (set2 - set1): {12, 5, 15}

4. Symmetric difference between sets:

Symmetric difference set: {2, 4, 5, 6, 12, 15}

5. Check if one set is subset of another:

Is set1 a subset of set2? False

6. Check if sets are disjoint:

Are set1 and set2 disjoint? False

7. Adding elements to a set:

Updated set1: {2, 4, 6, 8, 10, 12}

Updated set2: {18, 20, 5, 6, 8, 10, 12, 15}

8. Removing elements from a set:

Set1 after removing 8: {2, 4, 6, 10, 12}

Set2 after discarding 15: {18, 20, 5, 6, 8, 10, 12}

9. Clearing the entire set:

Set1 after clearing: set()

Set2 after clearing: set()

Process finished with exit code 0

============================================================================

3))perform and explore dictionary operations

# Example dictionary

my\_dict = {'name': 'John', 'age': 25, 'city': 'New York', 'gender': 'Male'}

# 1. Accessing values by key

print("1. Accessing values by key:")

print("Name:", my\_dict['name'])

print("Age:", my\_dict['age'])

print()

# 2. Updating values by key

print("2. Updating values by key:")

my\_dict['age'] = 26

my\_dict['city'] = 'San Francisco'

print("Updated dictionary:", my\_dict)

print()

# 3. Adding a new key-value pair

print("3. Adding a new key-value pair:")

my\_dict['occupation'] = 'Software Engineer'

print("Dictionary after adding occupation:", my\_dict)

print()

# 4. Removing a key-value pair

print("4. Removing a key-value pair:")

removed\_age = my\_dict.pop('age')

print("Dictionary after removing age:", my\_dict)

print("Removed age:", removed\_age)

print()

# 5. Getting keys and values

print("5. Getting keys and values:")

keys = my\_dict.keys()

values = my\_dict.values()

print("Keys:", keys)

print("Values:", values)

print()

# 6. Check if key is present in the dictionary

print("6. Check if key is present in the dictionary:")

is\_gender\_present = 'gender' in my\_dict

print("Is 'gender' present in the dictionary?", is\_gender\_present)

print()

# 7. Check if value is present in the dictionary

print("7. Check if value is present in the dictionary:")

is\_age\_present = 26 in my\_dict.values()

print("Is age 26 present in the dictionary?", is\_age\_present)

print()

# 8. Copying a dictionary

print("8. Copying a dictionary:")

copied\_dict = my\_dict.copy()

print("Copied dictionary:", copied\_dict)

print()

# 9. Clearing the entire dictionary

print("9. Clearing the entire dictionary:")

my\_dict.clear()

print("Dictionary after clearing:", my\_dict)

-------------------------------------------------

output::

1. Accessing values by key:

Name: John

Age: 25

2. Updating values by key:

Updated dictionary: {'name': 'John', 'age': 26, 'city': 'San Francisco', 'gender': 'Male'}

3. Adding a new key-value pair:

Dictionary after adding occupation: {'name': 'John', 'age': 26, 'city': 'San Francisco', 'gender': 'Male', 'occupation': 'Software Engineer'}

4. Removing a key-value pair:

Dictionary after removing age: {'name': 'John', 'city': 'San Francisco', 'gender': 'Male', 'occupation': 'Software Engineer'}

Removed age: 26

5. Getting keys and values:

Keys: dict\_keys(['name', 'city', 'gender', 'occupation'])

Values: dict\_values(['John', 'San Francisco', 'Male', 'Software Engineer'])

6. Check if key is present in the dictionary:

Is 'gender' present in the dictionary? True

7. Check if value is present in the dictionary:

Is age 26 present in the dictionary? False

8. Copying a dictionary:

Copied dictionary: {'name': 'John', 'city': 'San Francisco', 'gender': 'Male', 'occupation': 'Software Engineer'}

9. Clearing the entire dictionary:

Dictionary after clearing: {}

Process finished with exit code 0

================================================================================================================================

4))type casting( converting one data type to another)

# Example 1: Integer to float

integer\_value = 42

float\_value = float(integer\_value)

print(f"Example 1: Integer to float - Original: {integer\_value}, Casted: {float\_value}")

# Example 2: Float to integer

float\_number = 3.14

integer\_number = int(float\_number)

print(f"Example 2: Float to integer - Original: {float\_number}, Casted: {integer\_number}")

# Example 3: Integer to string

number = 123

string\_number = str(number)

print(f"Example 3: Integer to string - Original: {number}, Casted: '{string\_number}'")

# Example 4: String to integer

string\_digit = "456"

integer\_digit = int(string\_digit)

print(f"Example 4: String to integer - Original: '{string\_digit}', Casted: {integer\_digit}")

# Example 5: String to float

string\_float = "7.89"

float\_number\_from\_string = float(string\_float)

print(f"Example 5: String to float - Original: '{string\_float}', Casted: {float\_number\_from\_string}")

--------------------------------------------

output::

Example 1: Integer to float - Original: 42, Casted: 42.0

Example 2: Float to integer - Original: 3.14, Casted: 3

Example 3: Integer to string - Original: 123, Casted: '123'

Example 4: String to integer - Original: '456', Casted: 456

Example 5: String to float - Original: '7.89', Casted: 7.89

Process finished with exit code 0

===============================================================================================================

5)))explore operator precedence

>>Parentheses (): Parentheses have the highest precedence and can be used to force the order of evaluation within an expression.

>>Exponentiation \*\*: Exponentiation has the second-highest precedence.

>>Positive +, Negative -, Bitwise NOT ~: These unary operators have the next level of precedence.

>>Multiplication \*, Division /, Floor Division //, Modulus %: These arithmetic operators have the same level of precedence and are evaluated from left to right.

>>Addition +, Subtraction -: These arithmetic operators have the same level of precedence and are evaluated from left to right.

>>Bitwise Shifts <<, >>: Bitwise shift operators have the same level of precedence and are evaluated from left to right.

>>Bitwise AND &: Bitwise AND has a lower precedence than shifts.

>>Bitwise XOR ^: Bitwise XOR has a lower precedence than bitwise AND.

>>Bitwise OR |: Bitwise OR has a lower precedence than bitwise XOR.

>>Comparison operators ==, !=, <, >, <=, >=, in, not in, is, is not: These comparison operators have the next level of precedence.

>>Logical NOT not: Logical NOT has a lower precedence than comparison operators.

>>Logical AND and: Logical AND has a lower precedence than logical NOT.

>>Logical OR or: Logical OR has the lowest precedence among the logical operators.

>>Conditional Expression if-else: The conditional expression (ternary operator) x if condition else y has the lowest precedence

==============================

# Example: Operator Precedence

result = 5 + 2 \* 3 \*\* 2 / 4

print("Result:", result)

output::

Result: 9.5

12.03.2024

1))write a python programme which prints even numberfrom range 1 to 100

for x in range(1,100):

if x % 2 ==0:

print(x,end=",")...........................displays even numbers

2))write a python programme which prints alternate/random numbers from the range 1 to 500

for x in range(1,500,15):

print(x).............>dispalys numbers with difference 15

3)))write a python programme that takes a number between 1 to 12 as input and prints the corresponding month name

a =int(input("enter month number:"))

a -= 1

month =["jan", "feb", "mar", "apr", "may", "june", "july", "aug", "sep", "oct", "nov", "dec"]

print(month[a])

output::

Enter month number: 8

Aug

======================================================================

4))write a python programme that takes three numbers as an input and printout the largest number among them

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

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

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

largestnum = max(num1, num2, num3)

print("The highest number is:", largestnum)

---------------------------

output::

Enter the first number: 56

Enter the second number: 76

Enter the third number: 43

The highest number is: 76

=====================================================================================================================

5)))write a python programme that takes a password as input and checks its strength based on following critiria:weak(less than 6 characters),

medium(6 to 10 characters),strong(more than 10 characters with a mix of upper case,lower case,digits and special characters)

import string  
password = input("enter your password:")  
  
print(string.ascii\_letters)  
if len(password) <= 6:  
 print("weak")  
  
elif 6 < len(password) <= 8:  
 print("medium")  
  
elif len(password) >= 8 and \  
 any(c.islower() for c in password) and \  
 any(c.isupper() for c in password) and \  
 any(c.isdigit() for c in password) and \  
 any(c in string.punctuation for c in password):  
 print("strong")

output::

Enter your password: annu

Weak

Enter your password: anusha8578

Medium

Enter your password: Anusha@2024

Strong

================================================================================================================================

13.03.2024

1)write own examples for each and every function

def add(a,b):

return a+b

c = add(4,6)

print("the sum of given numbers", + c)

----------------------------------------------------------------------

recursive function::

def factorial(n):

if n == 0:

return 1

else:

return n \* factorial(n - 1)

num = 6

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

output::

Factorial of num: 720

----------------------------------------------------------------------------------------------

2)write example for \*args and \*\*kwargs

def add(\*args):

sum = 0

for num in args:

sum += num

return sum

print("sum:", add(34, 43, 56, 2))

print("sum:", add(65, 4, 12, 1, 5, 3, 6))

output::

sum: 135

sum: 96

=========================================================

def exmkw(\*\*kwargs):

for key, value in kwargs.items():

print(f"{key}: {value}")

# Example usage

exmkw(name="John", age=25, city="New York")

exmkw(title="Python Developer", experience="3 years", language="Python")

output::

name: John

age: 25

city: New York

title: Python Developer

experience: 3 years

language: Python

==========================================================================================

Built-in Functions:

print("Hello, World!")

result = len([1, 2, 3, 4, 5])

User-Defined Functions:

def add\_numbers(a, b):

return a + b

result = add\_numbers(3, 5)

print(result)

# Output: 8

Lambda Functions:

multiply = lambda x, y: x \* y

result = multiply(4, 6)

print(result) # Output: 24

division= lambda x, y: x % y

result = division(2, 8)

print(result) ::::output::2

Methods:

my\_list = [1, 2, 3]

my\_list.append(4) # append() is a method of the list class

print(my\_list) # Output: [1, 2, 3, 4]

MODULES::

1..>>create a python module named random\_generator.py taht contains a function to generate a list of random numbers.

generate a list of 10 random numbers within a specified range entered by the user..

IN MODULE FILE

::

def random(start, end, diff):

sequence = []

for num in range(start, end, diff):

sequence.append(num)

return sequence

start = int(input("Enter the starting number: "))

end = int(input("Enter the ending number: "))

diff = int(input("Enter the desired difference: "))

result = random(start, end, diff)

print("Generated sequence:", result)

IN MAIN FILE:

import random\_generator()

random\_generator.random()

OUTPUT::

Enter the starting number: 5

Enter the ending number: 50

Enter the desired difference: 5

Generated sequence: [5, 10, 15, 20, 25, 30, 35, 40, 45]

=================================================================================================================================

2..>>>create a python module named calculator.py that contains functions for addition,substraction,multiplication and division.

and import that module and perform arthimatic operations on two numbers entered by the user

IN MODULE FILE:

def addition(a, b):

return a + b

def subtraction(a, b):

return a - b

def multiplication(a, b):

return a \* b

def division(a, b):

return a / b

IN MAIN FILE:

import calculator

def main():

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

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

print("Sum:", calculator.addition(a, b))

print("Difference:", calculator.subtraction(a, b))

print("Product:", calculator.multiplication(a, b))

print("Division is:", calculator.division(a,b))

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

main()

OUTPUT::

Enter the first number: 25

Enter the second number: 5

Sum: 30.0

Difference: 20.0

Product: 125.0

Division is: 5.0

=================================================================================================================================

FILE HANDLING

::

3...>>>Create a python programm to read the contents of text file named input.text and display them on the screen

def main():

with open("input.txt", "r") as file:

contents = file.read()

print(contents)

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

main()

----------------------------------------------------------------------

with open("input.txt", "r") as file:

contents = file.read()

print(contents)

=================================================================================================================================

4..>>Create a python programm to take user input and write it to a new text file named output.text

file\_name = "output.txt"

user\_input = input("Enter text to write to the file: ")

with open(file\_name, "w") as file:

file.write(user\_input)

print(f"Data has been written to '{file\_name}' successfully.")

=================================================================================================================================

5..>>Create a python programm to copy the contents of one text file named source.text to another text file named destination.text

def main():

source\_file = "source.txt"

destination\_file = "destination.txt"

with open(source\_file, "r") as source:

contents = source.read()

with open(destination\_file, "w") as destination:

destination.write(contents)

print(f"Contents of '{source\_file}' copied to '{destination\_file}' successfully.")

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

main()

=================================================================================================================================

6...>>>>Create a python programm to read data from a csv file named data.csv and display it in tabular format

import csv

def display\_csv\_data(file\_name):

try:

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

csv\_reader = csv.reader(file)

data = list(csv\_reader)

# Calculate the maximum width of each column

column\_widths = [max(len(item) for item in column) for column in zip(\*data)]

# Display the data in tabular format

for row in data:

formatted\_row = " | ".join(item.ljust(width) for item, width in zip(row, column\_widths))

print(formatted\_row)

except FileNotFoundError:

print(f"Error: File '{file\_name}' not found.")

except Exception as e:

print(f"An error occurred: {e}")

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

file\_name = "data.csv"

display\_csv\_data(file\_name)

=================================================================================================================================

7...>>>Create a python programm to take user input and write it to a new row in a csv file named data.csv

import csv

# Sample data to write to the CSV file

data = [

["Name", "Age", "City"],

["John", 30, "New York"],

["Alice", 25, "Los Angeles"],

["Bob", 35, "Chicago"]

]

# Specify the file name

file\_name = "output.csv"

# Open the file in write mode ('w') and newline='' to avoid extra newline characters

with open(file\_name, 'w', newline='') as file:

writer = csv.writer(file)

# Write the data to the CSV file row by row

for row in data:

writer.writerow(row)

print("Data written successfully to", file\_name)

OUTPUT::

Data written successfully to output.csv

-----------------------------------------------------------------------------

import csv

data =[

['name', 'place', 'branch'],

['anusha', 'ongole', 'ece'],

['kavya', 'kukatpally', 'cse'],

['sravani', 'kothapeta', 'auronatical']

]

csv\_file\_path = 'indata.csv'

with open(csv\_file\_path, 'w', newline='') as csv.file:

writer = csv.writer(csv.file)

writer.writerows(data)

print("csv file created succesfully")

-------------------------------------------------

TO CHECK THE DATA IN THE FILE

import csv

# Specify the file name

file\_name = "output.csv"

# Open the file in read mode ('r')

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

reader = csv.reader(file)

# Iterate over each row in the CSV file and print it

for row in reader:

print(row)

OUTPUT::

['Name', 'Age', 'City']

['John', '30', 'New York']

['Alice', '25', 'Los Angeles']

['Bob', '35', 'Chicago']

=================================================================================================================================

8...>>>Create a python programm to read data from a json file named jason.csv and display them on the screen

import json

def read\_and\_display\_json(json\_file):

try:

with open(json\_file, 'r') as file:

data = json.load(file)

print("Data from JSON file:")

print(data)

except FileNotFoundError:

print(f"Error: File '{json\_file}' not found.")

except Exception as e:

print(f"An error occurred while reading JSON file: {e}")

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

json\_file = "json.json"

read\_and\_display\_json(json\_file)

=================================================================================================================================

9...>>>Create a python programm to read data from a json file yaml and display them on the screen

pip install pyyaml

import json

import yaml

def read\_and\_display\_json(json\_file):

try:

with open(json\_file, 'r') as file:

data = json.load(file)

print("Data from JSON file:")

print(data)

except FileNotFoundError:

print(f"Error: File '{json\_file}' not found.")

except Exception as e:

print(f"An error occurred while reading JSON file: {e}")

def read\_and\_display\_yaml(yaml\_file):

try:

with open(yaml\_file, 'r') as file:

data = yaml.safe\_load(file)

print("\nData from YAML file:")

print(data)

except FileNotFoundError:

print(f"Error: File '{yaml\_file}' not found.")

except Exception as e:

print(f"An error occurred while reading YAML file: {e}")

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

json\_file = "data.json"

yaml\_file = "data.yaml"

read\_and\_display\_json(json\_file)

read\_and\_display\_yaml(yaml\_file)

=================================================================================================================================

in data.py file::

import csv

with open('data.csv', 'r') as csv\_file:

csv\_reader = csv.reader((csv\_file))

print(csv\_reader)

output::

<\_csv.reader object at 0x000001EFA537F0A0>

-------------------------------------------------------------

import csv

with open('data.csv', 'r') as csv\_file:

csv\_reader = csv.reader((csv\_file))

print(csv\_reader)

for line in csv\_reader:

print(line)

output::

<\_csv.reader object at 0x000002851E05F0A0>

['import csv']

[]

[" with open('names.csv'", " 'r') as csv\_file:"]

[' csv\_reader = csv.reader(csv.file)']

[' print(csv\_reader)']

1))Implement a heirarchy of shapes using classes. Create a superclass called shapes and subclasses for different

shapes such as circle,rectangular and triangle. Include metods to calculate area and perimeter for each.

import math

class Shape():

def area(self):

pass

def perimeter(self):

pass

class Circle(Shape):

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

class Rectangle(Shape):

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

self.width = width

self.height = height

def area(self):

return self.width \* self.height

def perimeter(self):

return 2 \* (self.width + self.height)

class Triangle(Shape):

def \_\_init\_\_(self,side1,side2,side3):

self.side1 = side1

self.side2 = side2

self.side3 = side3

def area(self):

s = (self.side1 + self.side2 + self.side3) / 2

return math.sqrt(s \* (s - self.side1) \* (s - self.side2) \*(s - self.side3))

def perimeter(self):

return self.side1 + self.side2 + self.side3

circle = Circle(8)

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

print("Circle Perimeter:", circle.perimeter())

rectangle = Rectangle(7, 9)

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

print("Rectangle Perimeter:", rectangle.perimeter())

triangle = Triangle(15, 13, 12)

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

print("Triangle Perimeter:", triangle.perimeter()

OUTPUT::

Circle Area: 201.06192982974676

Circle Perimeter: 50.26548245743669

Rectangle Area: 63

Rectangle Perimeter: 126

Triangle Area: 74.83314773547883

Triangle Perimeter: 40

========================================================================================================================

2))) Design a python program to manage stiudent recordsl. Implement a class for student, course and grade.

Include methods for enrolling the sudents in courses, recording the grades and generating transcripts.

class Student():

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

self.student\_id = student\_id

self.name = name

self.courses = {}

def enroll(self, course):

self.courses[course.course\_id] = course

def \_\_str\_\_(self):

return f"Student ID: {self.student\_id}, Name: {self.name}"

class Course:

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

self.course\_id = course\_id

self.name = name

self.students = {}

def enroll\_student(self, student):

self.students[student.student\_id] = student

def \_\_str\_\_(self):

return f"Course ID: {self.course\_id}, Name: {self.name}"

class Grade:

def \_\_init\_\_(self, student\_id, course\_id, grade):

self.student\_id = student\_id

self.course\_id = course\_id

self.grade = grade

def \_\_str\_\_(self):

return f"Student ID: {self.student\_id}, Course ID: {self.course\_id}, Grade: {self.grade}"

class Transcript:

@staticmethod

def generate(student):

transcript = f"Transcript for {student.name} (Student ID: {student.student\_id}):\n"

for course\_id, course in student.courses.items():

transcript += f"\n{course}\n"

transcript += "Grades:\n"

for grade in course.students[student.student\_id].grades:

transcript += f"{grade}\n"

return transcript

# Example Usage:

student1 = Student("S001", "Alice")

student2 = Student("S002", "Bob")

course1 = Course("C001", "Mathematics")

course2 = Course("C002", "Physics")

course1.enroll\_student(student1)

course2.enroll\_student(student1)

course2.enroll\_student(student2)

student1.enroll(course1)

student1.enroll(course2)

student2.enroll(course2)

# Record grades

student1.courses["C001"].students["S001"].grades = [Grade("S001", "C001", "A"),

Grade("S001", "C002", "B")]

student2.courses["C002"].students["S002"].grades = [Grade("S002", "C002", "B")]

# Generate transcripts

print(Transcript.generate(student1))

print(Transcript.generate(student2))

1.Create a python class MathOperations with static methods basic mathematical operations such as addition,

substraction, multiplication and division. Also, include a class variable PI for the value of PI

import math

class Mathoperations:

PI = 3.414

@staticmethod

def add(x,y):

print(x+y)

@staticmethod

def sub(x,y):

print(x-y)

@staticmethod

def mul(x,y):

print(x\*y)

@staticmethod

def div(x,y):

print(x/y)

obj=Mathoperations()

obj.add(2,5)

obj.sub(5,3)

obj.mul(2,2)

obj.div(10,2)

print(Mathoperations.PI)

.............................................................................................................

2)Implement a python class FileUtility with static methods for reading and writing files. Include

methods for reading text files, writing to text files and copying files

import shutil

class FileUtility:

@staticmethod

def read\_text\_file(file\_path):

try:

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

return file.read()

except FileNotFoundError:

print(f"Error: File '{file\_path}' not found.")

return None

except Exception as e:

print(f"Error occurred while reading file '{file\_path}': {e}")

return None

@staticmethod

def write\_text\_file(file\_path, content):

try:

with open(file\_path, 'w') as file:

file.write(content)

print(f"Text content successfully written to '{file\_path}'.")

except Exception as e:

print(f"Error occurred while writing to file '{file\_path}': {e}")

@staticmethod

def copy\_file(source\_path, destination\_path):

try:

shutil.copyfile(source\_path, destination\_path)

print(f"File copied from '{source\_path}' to '{destination\_path}'.")

except FileNotFoundError:

print(f"Error: Source file '{source\_path}' not found.")

except Exception as e:

print(f"Error occurred while copying file from '{source\_path}' to '{destination\_path}': {e}")

# Example usage:

# Reading a text file

content = FileUtility.read\_text\_file("example.txt")

if content:

print("Content of example.txt:")

print(content)

# Writing to a text file

FileUtility.write\_text\_file("output.txt", "This is a test content.")

# Copying a file

FileUtility.copy\_file("example.txt", "example\_copy.txt")

...............................................................................................................

Task:

write a python program which hits the API and check the response

import requests

def check\_api\_response(api\_url):

try:

response = requests.get(api\_url)

if response.status\_code == 404:

print("API is reachable. Response content:")

print(response.json()) # Assuming the response is JSON

else:

print(f"Failed to hit the API. Status code: {response.status\_code}")

except request.RequestsException as e:

print(f"Error occurred while sending request: {e}")

# Example usage:

api\_url = "https://reqres.in/api/users/23" # Example API URL

check\_api\_response(api\_url)

output:

API is reachable. Response content:

{}

.....................................................................

Task:

4) Design python class GeometryHelper with static methods for calculating the area and perimeter of

common geometry shapes such as circles, rectangles, and tyriangles. Include class variablew for default values

such as the value of PI and standard lenghts.

import math

class GeometryHelper:

PI = math.pi # Class variable for the value of PI

STANDARD\_LENGTH = 1 # Class variable for standard lengths

@staticmethod

def circle\_area(r):

print(GeometryHelper.PI \* r \*\* 2)

@staticmethod

def circle\_perimeter(r):

print(2 \* GeometryHelper.PI \* r)

@staticmethod

def rectangle\_area(l,b):

print(l\*b)

@staticmethod

def rectangle\_perimeter(l, b):

print(2 \* (l + b))

@staticmethod

def triangle\_area(b, h):

print(0.5 \* b \* h)

@staticmethod

def triangle\_perimeter(a,b,c):

print(a+b+c)

obj=GeometryHelper()

obj.circle\_area(5)

obj.circle\_perimeter(5)

obj.rectangle\_area(5,6)

obj.rectangle\_perimeter(5,6)

obj.triangle\_area(1,2)

obj.triangle\_perimeter(1,2,3)

OUTPUT:

78.53981633974483

31.41592653589793

30

22

1.0

6

..................................................................................

5)write a program which explains methodoveriding and method overloading

methodoveriding

class A:

def display(self):

print("this is A")

class B(A):

def display(self):

print("this is B")

super().display()

obj=B()

obj.display()

output:

this is B

this is A

method overloading

class Main:

def sum(self,a=2,b=2):

print(a+b)

def sum(self,a,b,c):

print(a+b+c)

obj=Main()

obj.sum(1,2,3)

output:

6

......................................................................

>>>UPGRADE PIP IN YOUR SYSTEM TO LATEST VERSION

pip3 install pip output:24.0 ---> upgraded version

.................................................................................

9)CREATE A VIRTUAL ENVIRONMENT (VENV) IN YOUR SYSTEM

pip install virtualenv

..............................................................................

3)design a python class ConfigParser with static methods for parsing configuration files in different formats

( e.g json , yaml). Include class variables for default configuration file path and formats.

................................................................................

4)CREATE A PACKAGE FOLDER WITH TWO MODULE FILES CONTAINING 4 FUNCTIONS AND IMPORT THE

PACKAGE WITH TWO SPECIFIC FUNCTIONS(MODULE1.PY,MODULE2.PY)

1)) Write a python program which makes ssh connection to Linux VM and runs any one command

import paramiko

def ssh\_command(hostname, port, username, password, command):

try:

# Create SSH client instance

ssh\_client = paramiko.SSHClient()

# Automatically add host keys

ssh\_client.set\_missing\_host\_key\_policy(paramiko.AutoAddPolicy())

# Connect to the SSH server

ssh\_client.connect(hostname=hostname, port=port, username=username, password=password)

# Execute the command

stdin, stdout, stderr = ssh\_client.exec\_command(command)

# Read the output

output = stdout.read().decode()

error = stderr.read().decode()

# Print output or error

if output:

print("Command output:")

print(output)

if error:

print("Command error:")

print(error)

except Exception as e:

print(f"An error occurred: {e}")

finally:

# Close the SSH connection

ssh\_client.close()

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

# SSH connection details

hostname = "your\_host\_name\_or\_ip"

port = 22 # default SSH port

username = "your\_username"

password = "your\_password"

# Command to execute

command = "ls -l" # Example command, replace with your own

# Execute the SSH command

ssh\_command(hostname, port, username, password, command)

------------------------------------------------------------

import paramiko

def run\_ssh\_command(hostname, username, password, command):

# Create an SSH client instance

client = paramiko.SSHClient()

# Automatically add the server's host key

client.set\_missing\_host\_key\_policy(paramiko.AutoAddPolicy())

try:

# Connect to the server

client.connect(hostname, username=username, password=password)

# Run the command

stdin, stdout, stderr = client.exec\_command(command)

# Read the output

output = stdout.read().decode("utf-8")

# Print the output

print(output)

except Exception as e:

print(f"Error: {e}")

finally:

# Close the SSH connection

client.close()

# Example usage

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

hostname ="192.168.1.60"

username = "kavyakavya"

password = "Kavya@2001"

command = ("ls")# Example command

run\_ssh\_command(hostname, username, password, command)

==================================================================================================================================

2))Write a program which explains the special methods in python like \_\_add\_\_ \_\_sub\_\_ \_\_mul\_\_ and rest of the mathematical methods

class Vector:

def \_\_init\_\_(self, x, y):

self.x = x

self.y = y

def \_\_repr\_\_(self):

return f"Vector({self.x}, {self.y})"

def \_\_add\_\_(self, other):

return Vector(self.x + other.x, self.y + other.y)

def \_\_sub\_\_(self, other):

return Vector(self.x - other.x, self.y - other.y)

def \_\_mul\_\_(self, scalar):

return Vector(self.x \* scalar, self.y \* scalar)

def \_\_truediv\_\_(self, scalar):

return Vector(self.x / scalar, self.y / scalar)

def \_\_floordiv\_\_(self, scalar):

return Vector(self.x // scalar, self.y // scalar)

def \_\_mod\_\_(self, scalar):

return Vector(self.x % scalar, self.y % scalar)

def \_\_pow\_\_(self, exponent):

return Vector(self.x \*\* exponent, self.y \*\* exponent)

def \_\_abs\_\_(self):

return (self.x \*\* 2 + self.y \*\* 2) \*\* 0.5

# Creating Vector objects

v1 = Vector(3, 4)

v2 = Vector(1, 2)

# Addition

print("Addition:", v1 + v2)

# Subtraction

print("Subtraction:", v1 - v2)

# Multiplication

print("Multiplication:", v1 \* 2)

# True division

print("True Division:", v1 / 2)

# Floor division

print("Floor Division:", v1 // 2)

# Modulus

print("Modulus:", v1 % 2)

# Exponentiation

print("Exponentiation:", v1 \*\* 2)

# Absolute value

print("Absolute Value of v1:", abs(v1))

==================================================================================================================================

3))Write a python program for fibonacci series using GENERATORS

def fibonacci():

a, b = 0, 1

while True:

yield a

a, b = b, a + b

# Create a generator object

fib\_gen = fibonacci()

# Generate the Fibonacci sequence up to a certain limit

limit = 10

for \_ in range(limit):

print(next(fib\_gen))

==================================================================================================================================

4))Write a file reader class that reads lines from a file and support iteration using the \_\_iter\_\_ and \_\_next\_\_ methods

class FileReader:

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

self.filename = filename

self.file = open(filename, 'r')

def \_\_iter\_\_(self):

return self

def \_\_next\_\_(self):

line = self.file.readline()

if not line:

self.file.close()

raise StopIteration

return line.rstrip('\n')

# Example usage:

file\_reader = FileReader('example.txt')

for line in file\_reader:

print(line)

==================================================================================================================================

5))implement a custom list class that supports addition of two lists using the \_\_add\_\_ method

class CustomList:

def \_\_init\_\_(self, elements=None):

if elements is None:

self.elements = []

else:

self.elements = elements

def \_\_add\_\_(self, other):

if len(self.elements) != len(other.elements):

raise ValueError("Lists must have the same length for addition.")

result = [x + y for x, y in zip(self.elements, other.elements)]

return CustomList(result)

def \_\_repr\_\_(self):

return repr(self.elements)

# Example usage:

list1 = CustomList([1, 2, 3])

list2 = CustomList([4, 5, 6])

result = list1 + list2

print(result) # Output will be: [5, 7, 9]

==================================================================================================================================

implement the nested functionality of the functions(any example)

passing function as an argument to another function(any example)

implementing decorator for logging or authentication

1)Implement a python class FileUtility with static methods for reading and writing files. Include

methods for reading text files, writing to text files and copying files

import shutil

class FileUtility:

@staticmethod

def read\_text\_file(file\_path):

"""

Read the contents of a text file.

Parameters:

file\_path (str): The path to the text file.

Returns:

str: The contents of the text file.

"""

try:

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

return file.read()

except FileNotFoundError:

print(f"File '{file\_path}' not found.")

return None

except Exception as e:

print(f"Error reading file '{file\_path}': {e}")

return None

@staticmethod

def write\_text\_file(file\_path, content):

"""

Write content to a text file.

Parameters:

file\_path (str): The path to the text file.

content (str): The content to write to the file.

Returns:

bool: True if successful, False otherwise.

"""

try:

with open(file\_path, 'w') as file:

file.write(content)

return True

except Exception as e:

print(f"Error writing to file '{file\_path}': {e}")

return False

@staticmethod

def copy\_file(source\_file, destination\_file):

"""

Copy a file from source to destination.

Parameters:

source\_file (str): The path to the source file.

destination\_file (str): The path to the destination file.

Returns:

bool: True if successful, False otherwise.

"""

try:

shutil.copyfile(source\_file, destination\_file)

return True

except FileNotFoundError:

print(f"File '{source\_file}' not found.")

return False

except Exception as e:

print(f"Error copying file '{source\_file}' to '{destination\_file}': {e}")

return False

# Example usage:

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

# Read a text file

file\_content = FileUtility.read\_text\_file("example.txt")

if file\_content:

print("File content:")

print(file\_content)

# Write to a text file

content\_to\_write = "This is some example content to write to a file."

write\_success = FileUtility.write\_text\_file("new\_file.txt", content\_to\_write)

if write\_success:

print("Content successfully written to file.")

# Copy a file

copy\_success = FileUtility.copy\_file("example.txt", "example\_copy.txt")

if copy\_success:

print("File copied successfully.")

-------------------------------------------------------------------------------------------------

=====================================================================================================================

2)Create a python class MathOperations with static methods basic mathematical operations such as addition,

substraction, multiplication and division. Also, include a class variable PI for the value of PI

class MathOperations:

PI = 3.14159 # Class variable for PI value

@staticmethod

def add(x, y):

"""Performs addition."""

return x + y

@staticmethod

def subtract(x, y):

"""Performs subtraction."""

return x - y

@staticmethod

def multiply(x, y):

"""Performs multiplication."""

return x \* y

@staticmethod

def divide(x, y):

"""

Performs division.

Raises:

ValueError: If division by zero is attempted.

"""

if y == 0:

raise ValueError("Cannot divide by zero")

return x / y

# Example usage:

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

print("PI:", MathOperations.PI)

print("Addition:", MathOperations.add(5, 3))

print("Subtraction:", MathOperations.subtract(5, 3))

print("Multiplication:", MathOperations.multiply(5, 3))

try:

print("Division:", MathOperations.divide(5, 3))

print("Division by zero:", MathOperations.divide(5, 0)) # This will raise an exception

except ValueError as e:

print("Error:", e)

--------------------------------------

output::

PI: 3.14159

Addition: 8

Subtraction: 2

Multiplication: 15

Division: 1.6666666666666667

Error: Cannot divide by zero

-------------------------------------------------------

import math

class MathOperations:

PI = 3.414

@staticmethod

def add(a,b):

print(a + b)

@staticmethod

def sub(a,b):

print(a - b)

@staticmethod

def mul(a,b):

print(a \* b)

@staticmethod

def div(a,b):

if(b == 0):

print("denominator should not be zero")

print(a / b)

obj = MathOperations()

num1 = int(input("enter first number:"))

num2 = int(input("enter second number:"))

obj.add(num1,num2)

obj.sub(num1,num2)

obj.mul(num1,num2)

obj.div(num1,num2)

print(MathOperations.PI)

output::

enter first number:25

enter second number:5

30

20

125

5.0

3.414

=======================================================================================================================

3)design a python class ConfigParser with static methods for parsing configuration files in different formats

( e.g json , yaml). Include class variables for default configuration file path and formats

import json

import yaml

class ConfigParser:

DEFAULT\_FILE\_PATH = "config.json" # Default configuration file path

SUPPORTED\_FORMATS = ["json", "yaml"] # Supported formats

@staticmethod

def parse\_json(file\_path):

"""

Parse a JSON configuration file.

Parameters:

file\_path (str): The path to the JSON file.

Returns:

dict: The parsed configuration.

"""

try:

with open(file\_path, "r") as file:

return json.load(file)

except FileNotFoundError:

print(f"File '{file\_path}' not found.")

return None

except json.JSONDecodeError as e:

print(f"Error decoding JSON file '{file\_path}': {e}")

return None

except Exception as e:

print(f"Error parsing JSON file '{file\_path}': {e}")

return None

@staticmethod

def parse\_yaml(file\_path):

"""

Parse a YAML configuration file.

Parameters:

file\_path (str): The path to the YAML file.

Returns:

dict: The parsed configuration.

"""

try:

with open(file\_path, "r") as file:

return yaml.safe\_load(file)

except FileNotFoundError:

print(f"File '{file\_path}' not found.")

return None

except yaml.YAMLError as e:

print(f"Error parsing YAML file '{file\_path}': {e}")

return None

except Exception as e:

print(f"Error parsing YAML file '{file\_path}': {e}")

return None

# Example usage:

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

# Example using JSON format

json\_config = ConfigParser.parse\_json("config.json")

if json\_config:

print("JSON Configuration:", json\_config)

# Example using YAML format

yaml\_config = ConfigParser.parse\_yaml("config.yaml")

if yaml\_config:

print("YAML Configuration:", yaml\_config)

=======================================================================================================================

4))] Design python class GeometryHelper with static methods for calculating the area and perimeter of

common geometry shapes such as circles, rectangles, and tyriangles. Include class variablew for default values

such as the value of PI and standard lenghts.

import math

class GeoMetryHelper:

PI = math.pi

DEFAULT\_RADIUS = 1.0

DEFAULT\_LENGTH\_RECTANGLE = 2.0

DEFAULT\_WIDTH\_RECTANGLE = 1.0

DEFAULT\_SIDE\_TRIANGLE\_1 = 3.0

DEFAULT\_SIDE\_TRIANGLE\_2 = 4.0

DEFAULT\_SIDE\_TRIANGLE\_3 = 5.0

@staticmethod

def circle\_area(radius):

return GeoMetryHelper.PI \* radius \*\* 2

@staticmethod

def rectangle\_perimeter(length, width):

return 2 \* (length + width)

@staticmethod

def rectangle\_area(length, width):

return length \* width

@staticmethod

def triangle\_area(a, b, c):

s = (a + b + c) / 2

return math.sqrt(s \* (s - a) \* (s - b) \* (s - c))

@staticmethod

def triangle\_perimeter(a, b, c):

return a + b + c

obj = GeoMetryHelper()

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

length = float(input("enter the length of rectangle:"))

width = float(input("enter the width of rectangle:"))

side1 = float(input("enter the first side:"))

side2 = float(input("enter the second side:"))

side3 = float(input("enter the third side:"))

print("area of the circle:", obj.circle\_area(radius))

print("area of the rectangle:", obj.rectangle\_area(length, width))

print("perimeter of the rectangle:", obj.rectangle\_perimeter(length, width))

print("area of the triangle:", obj.triangle\_area(side1, side2, side3))

print("perimeter of the triangle:", obj.triangle\_perimeter(side1, side2, side3))

output::

enter the radius of circle:>? 6

enter the length of rectangle:>? 9

enter the width of rectangle:>? 5

enter the first side:>? 9

enter the second side:>? 12

enter the third side:>? 15

area of the circle: 113.09733552923255

area of the rectangle: 45.0

perimeter of the rectangle: 28.0

area of the triangle: 54.0

perimeter of the triangle: 36.0

=======================================================================================================================

5))write a python program which hitys the API and check the response

import requests

def check\_api\_response(api\_url):

try:

response = requests.get(api\_url)

# Check if the response status code is 200 (OK)

if response.status\_code == 200:

print("API is reachable. Response content:")

print(response.json()) # Assuming the response is in JSON format

else:

print(f"Failed to get response from the API. Status code: {response.status\_code}")

except requests.RequestException as e:

print(f"Error occurred while requesting the API: {e}")

# Example usage:

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

# Example API URL

api\_url = "https://jsonplaceholder.typicode.com/posts/1" # This is just a sample API, replace with your actual API URL

check\_api\_response(api\_url)

=======================================================================================================================

6))write a program which explains methodoveriding and method overloading

class Animal:

def make\_sound(self):

print("Generic animal sound")

class Dog(Animal):

def make\_sound(self):

print("Woof!")

class Cat(Animal):

def make\_sound(self):

print("Meow!")

class MathOperations:

def add(self, x, y):

return x + y

def add(self, x, y, z):

return x + y + z

# Method Overriding Example

print("Method Overriding Example:")

animal = Animal()

dog = Dog()

cat = Cat()

animal.make\_sound() # Output: Generic animal sound

dog.make\_sound() # Output: Woof!

cat.make\_sound() # Output: Meow!

# Method Overloading Example

print("\nMethod Overloading Example:")

math = MathOperations()

# This will call the second add method

print("Sum of 2 + 3 + 4 =", math.add(2, 3, 4))

# This will raise a TypeError as the first add method is overridden

# print("Sum of 2 + 3 =", math.add(2, 3))

=======================================================================================================================

7)CREATE A PACKAGE FOLDER WITH TWO MODULE FILES CONTAINING 4 FUNCTIONS AND IMPORT THE

PACKAGE WITH TWO SPECIFIC FUNCTIONS(MODULE1.PY,MODULE2.PY)

my\_package/

│

├── \_\_init\_\_.py

├── module1.py

└── module2.py

&&module1.py:

def func1():

print("Function 1 from module 1")

def func2():

print("Function 2 from module 1")

&&module2.py:

def func3():

print("Function 1 from module 2")

def func4():

print("Function 2 from module 2")

# Importing specific functions from module1 and module2

from my\_package.module1 import func1, func2

from my\_package.module2 import func3, func4

# Using functions from module1

print("Using functions from module1:")

func1() # Output: Function 1 from module 1

func2() # Output: Function 2 from module 1

# Using functions from module2

print("\nUsing functions from module2:")

func3() # Output: Function 1 from module 2

func4() # Output: Function 2 from module 2

=======================================================================================================================

8)UPGRADE PIP IN YOUR SYSTEM TO LATEST VERSION

=======================================================================================================================

9)CREATE A VIRTUAL ENVIRONMENT (VENV) IN YOUR SYSTEM

..> Create a folder named virtual environment in the system folder of projects

..> Open the command prompt in the same folder

..>>write a command >> python -m venv myenv

=======================================================================================================================

PARAMIKO

SSH-PARAMIKO

21/03/2024

PRACTICE SERIES AND DATA FRAMES IN PANDAS

PRACTICE TO CREATE TABLE WITH MIN 10 ENTRIES AND EXPLORE METHODS

PRACTICE::

FILE HANDLING

FUNCTIONS with exception handling  
DATA TYPES

OOPS

DATA TYPES IN PYTHON::

INTEGER

FLOAT

STRING

BOOLEAN

LIST

SET

TUPLE

DICTIONARY

PRIME NIMBER::

def prime(n):  
 if n == 1 or n % 2 == 0:  
 return False  
 if n in [2,3]:  
 return True  
 r = 3  
 while r \* r <= n:  
 if n % r == 0:  
 return False  
 r += 1  
 return True  
  
num = int(input("enter the number:"))  
print(prime(num))

WRITE INTO CSV FILE::

import csv  
with open("emp.csv", "w") as f:  
 f = csv.writer(f)  
 f.writerow(["eno", "ename", "esal", "eaddr"])  
 while True:  
 eno = int(input("enter employee number:"))  
 ename = input("enter employee name:")  
 esal = int(input("enter employee salary:"))  
 eaddr = input("enter employee address:")  
 f.writerow([eno,ename,esal,eaddr])  
 option = input("do you want to continue:[yes/no]:")  
 if option.lower() == "no":  
 break  
 print("records successfully added")

X

Task:

write a python program which hits the API and check the response

import requests

def check\_api\_response(api\_url):

try:

response = requests.get(api\_url)

if response.status\_code == 404:

print("API is reachable. Response content:")

print(response.json()) # Assuming the response is JSON

else:

print(f"Failed to hit the API. Status code: {response.status\_code}")

except request.RequestsException as e:

print(f"Error occurred while sending request: {e}")

# Example usage:

api\_url = "https://reqres.in/api/users/23" # Example API URL

check\_api\_response(api\_url)

output:

API is reachable. Response content:

{}

STRING PALINDROME::

def is\_string\_palindrome(s):

"""

Function to check if a string is a palindrome.

"""

original\_string = s

string\_copy = s[:] # Make a copy of the original string

reversed\_string = string\_copy[::-1]

return original\_string == reversed\_string

# Example usage:

result = is\_string\_palindrome("radar")

print("Is 'radar' a palindrome?", result)

NUMBER PALINDROME::

def is\_number\_palindrome(num):

"""

Function to check if a number is a palindrome.

"""

original\_num = num

num\_copy = num # Make a copy of the original number

reversed\_num = 0

while num\_copy > 0:

digit = num\_copy % 10

reversed\_num = reversed\_num \* 10 + digit

num\_copy //= 10

return original\_num == reversed\_num

# Example usage:

result = is\_number\_palindrome(12321)

print("Is 12321 a palindrome?", result)

(OR)

def is\_palindrome(s):

"""

Function to check if a string is a palindrome.

"""

s = s.lower() # Convert to lowercase for case-insensitive comparison

return s == s[::-1]

# Example usage:

result = is\_palindrome("Madam")

print("Is 'Madam' a palindrome?", result)

PRIME NUMBER():

def is\_prime(num):

"""

Function to check if a number is prime.

"""

if num <= 1:

return False

elif num <= 3:

return True

elif num % 2 == 0 or num % 3 == 0:

return False

i = 5

while i \* i <= num:

if num % i == 0 or num % (i + 2) == 0:

return False

i += 6

return True

# Example usage:

result = is\_prime(17)

print("Is 17 a prime number?", result)

FIBONACCI SERIES()::

def fibonacci(n):

"""

Function to generate the Fibonacci sequence up to the nth term.

"""

fib\_sequence = [0, 1]

while len(fib\_sequence) < n:

fib\_sequence.append(fib\_sequence[-1] + fib\_sequence[-2])

return fib\_sequence[:n]

# Example usage:

result = fibonacci(10)

print("First 10 terms of the Fibonacci sequence:", result)

FACTORIAL()::

def factorial(n):

"""

Function to calculate the factorial of a non-negative integer.

"""

if n == 0:

return 1

else:

return n \* factorial(n - 1)

# Example usage:

result = factorial(5)

print("Factorial of 5:", result)