1. **Write a program to check whether the given number is prime or not?**

def is\_prime(number):

if number < 2:

return False

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

if number % i == 0:

return False

return True

# Test the function

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

if is\_prime(num):

print(num, "is a prime number.")

else:

print(num, "is not a prime number.")

**OUTPUT:-**

Enter a number: 57

57 is not a prime number.

1. **Write a program to print all perfect nos given range?**

def is\_perfect\_number(number):

sum\_of\_divisors = 0

for i in range(1, number):

if number % i == 0:

sum\_of\_divisors += i

return sum\_of\_divisors == number

# Test the function

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

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

print("Perfect numbers between", start, "and", end, ":")

for num in range(start, end+1):

if is\_perfect\_number(num):

print(num)

**OUTPUT:-**

Enter the starting number: 20

Enter the ending number: 57

Perfect numbers between 20 and 57 :

28

1. **Write a program to print sum of digit of any number.**

def calculate\_digit\_sum(number):

sum\_of\_digits = 0

while number > 0:

digit = number % 10

sum\_of\_digits += digit

number //= 10

return sum\_of\_digits

# Test the function

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

digit\_sum = calculate\_digit\_sum(number)

print("Sum of digits:", digit\_sum)

**OUTPUT:-**

Enter a number: 10

Sum of digits: 1

* 1. **Write a program to create tuple of name of 5 fruits and print them using two ways**

1. **Directly print all the item**

Ans.

fruits = ('apple', 'banana', 'orange', 'grape', 'mango')

# Method 1: Directly printing all the items

print("Method 1:")

print(fruits)

# Method 2: Printing each item using a loop

print("Method 2:")

for fruit in fruits:

print(fruit)

**OUTPUT:-**

Method 1:

('apple', 'banana', 'orange', 'grape', 'mango')

Method 2:

apple

banana

orange

grape

mango

1. **Write a Python function to multiply all the numbers in a list.**

def multiply\_list(numbers):

result = 1

for num in numbers:

result \*= num

return result

# Test the function

nums = [2, 3, 4, 5]

product = multiply\_list(nums)

print("Product:", product)

**OUTPUT:-**

Product: 120

1. **Write a Python program to reverse a string.**

def reverse\_string(string):

return string[::-1]

# Test the function

input\_string = input("Enter a string: ")

reversed\_string = reverse\_string(input\_string)

print("Reversed string:", reversed\_string)

**OUTPUT:-**

Enter a string: python

Reversed string: nohtyp

1. **Write a program to find out sum of 2 numbers 3 numbers 4 numbers and 5 numbers using default parameter passing.**

def calculate\_sum(a, b, c=None, d=None, e=None):

if c is None: # Only 2 numbers

return a + b

elif d is None: # 3 numbers

return a + b + c

elif e is None: # 4 numbers

return a + b + c + d

else: # 5 numbers

return a + b + c + d + e

# Testing the function

print("Sum of 2 numbers:", calculate\_sum(2, 3))

print("Sum of 3 numbers:", calculate\_sum(2, 3, 4))

print("Sum of 4 numbers:", calculate\_sum(2, 3, 4, 5))

print("Sum of 5 numbers:", calculate\_sum(2, 3, 4, 5, 6))

**OUTPUT:-**

Sum of 2 numbers: 5

Sum of 3 numbers: 9

Sum of 4 numbers: 14

Sum of 5 numbers: 20

1. **Write a program to find out multiplication of 2 numbers 3 numbers 4 numbers and 5 numbers using default parameter passing.**

def multiply\_numbers(a, b, c=1, d=1, e=1):

return a \* b \* c \* d \* e

# Test the function with different numbers of arguments

print("Multiplication of 2 numbers:", multiply\_numbers(2, 3))

print("Multiplication of 3 numbers:", multiply\_numbers(2, 3, 4))

print("Multiplication of 4 numbers:", multiply\_numbers(2, 3, 4, 5))

print("Multiplication of 5 numbers:", multiply\_numbers(2, 3, 4, 5, 6))

**OUTPUT:-**

Multiplication of 2 numbers: 6

Multiplication of 3 numbers: 24

Multiplication of 4 numbers: 120

Multiplication of 5 numbers: 720

**9) Write a program to find out area circumference of circle using lambda function.**

# Define lambda functions for area and circumference

calculate\_area = lambda radius: 3.14159 \* radius\*\*2

calculate\_circumference = lambda radius: 2 \* 3.14159 \* radius

# Test the lambda functions

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

area = calculate\_area(radius)

circumference = calculate\_circumference(radius)

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

print("Circumference of the circle:", circumference)

**OUTPUT:-**

Enter the radius of the circle: 5

Area of the circle: 78.53975

Circumference of the circle: 31.4159

**10) Write a program to pass variable length arguments (dictionary and tuple to print email.**

def print\_email(\*args, \*\*kwargs):

# Extracting dictionary and tuple from args

if len(args) > 0:

email\_dict = args[0]

else:

email\_dict = {}

# Extracting dictionary and tuple from kwargs

if 'email\_tuple' in kwargs:

email\_tuple = kwargs['email\_tuple']

else:

email\_tuple = ()

# Printing email components

print("To:", email\_dict.get('to', ''))

print("CC:", email\_dict.get('cc', ''))

print("BCC:", email\_dict.get('bcc', ''))

print("Subject:", email\_dict.get('subject', ''))

print("Body:", email\_dict.get('body', ''))

for attachment in email\_tuple:

print("Attachment:", attachment)

# Test the function

email\_info = {

'to': 'recipient@example.com',

'cc': 'cc@example.com',

'subject': 'Hello, World!',

'body': 'This is the body of the email.'

}

attachments = ('document1.pdf', 'image.jpg', 'spreadsheet.xlsx')

print\_email(email\_info, email\_tuple=attachments)

**OUTPUT:-**

To: recipient@example.com

CC: cc@example.com

BCC:

Subject: Hello, World!

Body: This is the body of the email.

Attachment: document1.pdf

Attachment: image.jpg

Attachment: spreadsheet.xlsx

**11)Write a program to create class dog to store Breed Age and Color attribute and display it.**

class Dog:

def \_\_init\_\_(self, breed, age, color):

self.breed = breed

self.age = age

self.color = color

def display\_info(self):

print("Breed:", self.breed)

print("Age:", self.age)

print("Color:", self.color)

# Create an instance of the Dog class

my\_dog = Dog("Labrador Retriever", 3, "Golden")

# Display the dog's information

my\_dog.display\_info()

**OUTPUT:-**

Breed: Labrador Retriever

Age: 3

Color: Golden

**12)** **Write a program to create class person to store name,age,sex of the person use constructors to initialize objects.**

class Person:

def \_\_init\_\_(self, name, age, sex):

self.name = name

self.age = age

self.sex = sex

def display\_info(self):

print("Name:", self.name)

print("Age:", self.age)

print("Sex:", self.sex)

# Create instances of the Person class

person1 = Person("John Doe", 25, "Male")

person2 = Person("Jane Smith", 30, "Female")

# Display the information of the persons

person1.display\_info()

print() # Print an empty line for separation

person2.display\_info()

**OUTPUT:-**

Name: John Doe

Age: 25

Sex: Male

Name: Jane Smith

Age: 30

Sex: Female

**13) Q1. Write a program to find following pattern from the string “Sky is only Limit of the Programmer in Year 2023”**

**All capital alphabets**

**All digits**

**All small alphabets**

**All lower case between “m-“ to “s”**

import re

def find\_pattern(text):

# Pattern for all capital alphabets

capital\_pattern = re.findall(r'[A-Z]', text)

print("All capital alphabets:", capital\_pattern)

# Pattern for all digits

digit\_pattern = re.findall(r'\d', text)

print("All digits:", digit\_pattern)

# Pattern for all small alphabets

small\_pattern = re.findall(r'[a-z]', text)

print("All small alphabets:", small\_pattern)

# Pattern for all lowercase letters between 'm-' and 's'

lowercase\_pattern = re.findall(r'm-([a-rst-z])', text)

print("All lowercase between 'm-' and 's':", lowercase\_pattern)

# Test the function

text = "Sky is only Limit of the Programmer in Year 2023"

find\_pattern(text)

**OUTPUT:-**

All capital alphabets: ['S', 'L', 'P', 'Y']

All digits: ['2', '0', '2', '3']

All small alphabets: ['k', 'y', 'i', 's', 'o', 'n', 'l', 'y', 'i', 't', 'o', 'f', 't', 'h', 'e', 'r', 'o', 'g', 'r', 'a', 'm', 'm', 'e', 'r', 'i', 'n', 'e', 'a', 'r']

All lowercase between 'm-' and 's': ['o', 'n', 'l', 'y', 'i', 't', 'o', 'f']

**14) Write a program to handle name error exception.**

def handle\_name\_error():

try:

# Code that may raise a NameError

print(x) # Assuming 'x' is not defined

except NameError:

print("A NameError occurred: 'x' is not defined.")

else:

print("No exception occurred.")

finally:

print("The 'handle\_name\_error' function has completed.")

# Test the function

handle\_name\_error()

**OUTPUT:-**

A NameError occurred: 'x' is not defined.

The 'handle\_name\_error' function has completed.

**15)Write a program to read any text file and print report of total characters as follows**

**File report of a.txt**

**Character Types Total Count**

**Capital letters 22**

**Small letters 123**

**Digits 34**

**Special Characters 20**

def generate\_file\_report(file\_name):

# Initialize counters

capital\_count = 0

small\_count = 0

digit\_count = 0

special\_count = 0

try:

# Open the file in read mode

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

# Read the contents of the file

content = file.read()

# Iterate over each character in the content

for char in content:

if char.isupper(): # Check for capital letters

capital\_count += 1

elif char.islower(): # Check for small letters

small\_count += 1

elif char.isdigit(): # Check for digits

digit\_count += 1

else: # Check for special characters

special\_count += 1

# Print the report

print(f"File report of {file\_name}")

print("{:<20} {:<15}".format("Character Types", "Total Count"))

print("{:<20} {:<15}".format("Capital letters", capital\_count))

print("{:<20} {:<15}".format("Small letters", small\_count))

print("{:<20} {:<15}".format("Digits", digit\_count))

print("{:<20} {:<15}".format("Special Characters", special\_count))

except FileNotFoundError:

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

except IOError:

print("An error occurred while reading the file.")

# Test the function

file\_name = "a.txt"

generate\_file\_report(file\_name)

**OUTPUT:-**

File report of a.txt

Character Types Total Count

Capital letters 2

Small letters 21

Digits 5

Special Characters 5

**16. ) Write a program to read any text file and convert all small case letters to capital letters**

def convert\_to\_uppercase(file\_name):

try:

# Open the file in read mode

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

# Read the contents of the file

content = file.read()

# Convert all small letters to uppercase

modified\_content = content.upper()

# Open the file in write mode

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

# Write the modified content back to the file

file.write(modified\_content)

print(f"The file '{file\_name}' has been successfully updated.")

except FileNotFoundError:

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

except IOError:

print("An error occurred while reading or writing the file.")

# Test the function

file\_name = "example.txt" # Replace with the actual name of the text file

convert\_to\_uppercase(file\_name)

**OUTPUT:-**

HELLO, WORLD!

THIS IS AN EXAMPLE FILE.

**17) Write a program to read any text file and convert all capital case letters to small letters**

def convert\_to\_lowercase(file\_name):

try:

# Open the file in read mode

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

# Read the contents of the file

content = file.read()

# Convert all capital letters to lowercase

modified\_content = content.lower()

# Open the file in write mode

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

# Write the modified content back to the file

file.write(modified\_content)

print(f"The file '{file\_name}' has been successfully updated.")

except FileNotFoundError:

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

except IOError:

print("An error occurred while reading or writing the file.")

# Test the function

file\_name = "example.txt" # Replace with the actual name of the text file

convert\_to\_lowercase(file\_name)

**OUTPUT:-**

hello, world!

this is an example file.

**18) Write a Python program to create "employee" collection with fields (ID, name, address, phone, email, dept) in MongoDB. Perform the following operations-**

**i. Insert minimum 5 documents in employee collection.**

**ii. Display all employees in "Accounts" department.**

**iii. Delete employee with ID - 210345.**

**iv. Update phone with new phone for employee Id- 123**

from pymongo import MongoClient

# Establish connection to MongoDB

client = MongoClient("mongodb://localhost:27017/")

# Create or access the "employee" database

db = client["employee"]

# Create or access the "employees" collection

employees = db["employees"]

# i. Insert minimum 5 documents in the "employees" collection

employees\_data = [

{

"ID": 123,

"name": "John Doe",

"address": "123 Main St",

"phone": "555-1234",

"email": "johndoe@example.com",

"dept": "Accounts"

},

{

"ID": 210345,

"name": "Jane Smith",

"address": "456 Elm St",

"phone": "555-5678",

"email": "janesmith@example.com",

"dept": "IT"

},

{

"ID": 6789,

"name": "Michael Johnson",

"address": "789 Oak St",

"phone": "555-9876",

"email": "michaeljohnson@example.com",

"dept": "Accounts"

},

{

"ID": 54321,

"name": "Emily Davis",

"address": "321 Pine St",

"phone": "555-4321",

"email": "emilydavis@example.com",

"dept": "HR"

},

{

"ID": 98765,

"name": "David Wilson",

"address": "987 Maple St",

"phone": "555-8765",

"email": "davidwilson@example.com",

"dept": "Accounts"

}

]

employees.insert\_many(employees\_data)

print("Documents inserted successfully.")

# ii. Display all employees in the "Accounts" department

accounts\_employees = employees.find({"dept": "Accounts"})

print("Employees in the Accounts department:")

for employee in accounts\_employees:

print(employee)

# iii. Delete employee with ID - 210345

employees.delete\_one({"ID": 210345})

print("Employee with ID 210345 deleted successfully.")

# iv. Update phone with new phone for employee ID - 123

new\_phone = "555-9999"

employees.update\_one({"ID": 123}, {"$set": {"phone": new\_phone}})

print("Phone updated for employee with ID 123.")

# Close the MongoDB connection

client.close()

**19) Write a Python program to create "movie" collection with fields (movie\_id, movie\_name, movie\_type, movie\_year, movie\_lead\_actor, movie\_director) in MongoDB. Perform the following operations-**

**i. Insert minimum 5 documents in movie collection.**

**ii. Display the list of all movies released in year 2020.**

**iii. Update the movie \_type to "Comedy" wherever movie\_lead\_actor is "Govinda".**

**iv. Display the list of movies in descending order of movie\_year.**

**v. Delete movie having movie\_director as "David Dhavan".**

from pymongo import MongoClient

# Establish connection to MongoDB

client = MongoClient("mongodb://localhost:27017/")

# Create or access the "movie" database

db = client["movie"]

# Create or access the "movies" collection

movies = db["movies"]

# i. Insert minimum 5 documents in the "movies" collection

movies\_data = [

{

"movie\_id": 1,

"movie\_name": "The Avengers",

"movie\_type": "Action",

"movie\_year": 2012,

"movie\_lead\_actor": "Robert Downey Jr.",

"movie\_director": "Joss Whedon"

},

{

"movie\_id": 2,

"movie\_name": "Inception",

"movie\_type": "Sci-Fi",

"movie\_year": 2010,

"movie\_lead\_actor": "Leonardo DiCaprio",

"movie\_director": "Christopher Nolan"

},

{

"movie\_id": 3,

"movie\_name": "Dilwale Dulhania Le Jayenge",

"movie\_type": "Romance",

"movie\_year": 1995,

"movie\_lead\_actor": "Shah Rukh Khan",

"movie\_director": "Aditya Chopra"

},

{

"movie\_id": 4,

"movie\_name": "The Dark Knight",

"movie\_type": "Action",

"movie\_year": 2008,

"movie\_lead\_actor": "Christian Bale",

"movie\_director": "Christopher Nolan"

},

{

"movie\_id": 5,

"movie\_name": "Andaz Apna Apna",

"movie\_type": "Comedy",

"movie\_year": 1994,

"movie\_lead\_actor": "Aamir Khan",

"movie\_director": "Rajkumar Santoshi"

}

]

movies.insert\_many(movies\_data)

print("Documents inserted successfully.")

# ii. Display the list of all movies released in year 2020

movies\_2020 = movies.find({"movie\_year": 2020})

print("Movies released in the year 2020:")

for movie in movies\_2020:

print(movie)

# iii. Update the movie\_type to "Comedy" wherever movie\_lead\_actor is "Govinda"

movies.update\_many({"movie\_lead\_actor": "Govinda"}, {"$set": {"movie\_type": "Comedy"}})

print("Movie type updated for movies with lead actor Govinda.")

# iv. Display the list of movies in descending order of movie\_year

movies\_descending\_order = movies.find().sort("movie\_year", -1)

print("Movies in descending order of movie\_year:")

for movie in movies\_descending\_order:

print(movie)

# v. Delete movie having movie\_director as "David Dhavan"

movies.delete\_one({"movie\_director": "David Dhavan"})

print("Movie with movie\_director David Dhavan deleted.")

# Close the MongoDB connection

client.close()

**20 ) Write a Python program to create "Library" database. In Library database create 2 collections such as "Book" with fields (book\_id, title, author, publisher, price, copies) and "Transaction" with fields (trans\_id, trans\_type, trans\_date, stud\_id, book\_id). Perform the following operations -**

**i. Insert minimum 5 documents in Book & Transaction collection.**

**ii. Display the list of books in Library in ascedning order of Title.**

**iii. Display the trasaction details for the transactions done on "18-06-2023".**

**iv. Update the copies of books to 10 for the subject of Java.**

**v. Delete the trasactions done in year 2020.**

from pymongo import MongoClient

from datetime import datetime

# Establish connection to MongoDB

client = MongoClient("mongodb://localhost:27017/")

# Create or access the "Library" database

db = client["Library"]

# Create or access the "Book" collection

books = db["Book"]

# Create or access the "Transaction" collection

transactions = db["Transaction"]

# i. Insert minimum 5 documents in "Book" and "Transaction" collections

books\_data = [

{

"book\_id": 1,

"title": "Python Crash Course",

"author": "Eric Matthes",

"publisher": "No Starch Press",

"price": 25.99,

"copies": 5

},

{

"book\_id": 2,

"title": "Java Programming",

"author": "Herbert Schildt",

"publisher": "McGraw-Hill Education",

"price": 30.99,

"copies": 3

},

{

"book\_id": 3,

"title": "The Alchemist",

"author": "Paulo Coelho",

"publisher": "HarperOne",

"price": 15.99,

"copies": 10

},

{

"book\_id": 4,

"title": "The Great Gatsby",

"author": "F. Scott Fitzgerald",

"publisher": "Scribner",

"price": 12.99,

"copies": 8

},

{

"book\_id": 5,

"title": "To Kill a Mockingbird",

"author": "Harper Lee",

"publisher": "J. B. Lippincott & Co.",

"price": 11.99,

"copies": 6

}

]

books.insert\_many(books\_data)

print("Documents inserted into the 'Book' collection.")

transactions\_data = [

{

"trans\_id": 1,

"trans\_type": "Borrow",

"trans\_date": datetime(2023, 6, 18),

"stud\_id": 1,

"book\_id": 1

},

{

"trans\_id": 2,

"trans\_type": "Return",

"trans\_date": datetime(2023, 6, 18),

"stud\_id": 2,

"book\_id": 3

},

{

"trans\_id": 3,

"trans\_type": "Borrow",

"trans\_date": datetime(2022, 12, 10),

"stud\_id": 3,

"book\_id": 4

},

{

"trans\_id": 4,

"trans\_type": "Borrow",

"trans\_date": datetime(2023, 6, 18),

"stud\_id": 2,

"book\_id": 2

},

{

"trans\_id": 5,

"trans\_type": "Return",

"trans\_date": datetime(2021, 7, 25),

"stud\_id": 1,

"book\_id": 5

}

]

transactions.insert\_many(transactions\_data)

print("Documents inserted into the 'Transaction' collection.")

# ii. Display the list of books in Library in ascending order of Title

books\_list = books.find().sort("title", 1)

print("Books in the Library (ascending order of Title):")

for book in books\_list:

print(book)

# iii. Display the transaction details for the transactions done on "18-06-2023"

transaction\_date = datetime(2023, 6, 18)

transactions\_on\_date = transactions.find({"trans\_date": transaction\_date})

print("Transactions done on 18-06-2023:")

for transaction in transactions\_on\_date:

print(transaction)

# iv. Update the copies of books to 10 for the subject of Java

books.update\_many({"title": "Java Programming"}, {"$set": {"copies": 10}})

print("Copies updated for books with the subject of Java.")

# v. Delete the transactions done in year 2020

transactions.delete\_many({"trans\_date": {"$gte": datetime(2020, 1, 1), "$lt": datetime(2021, 1, 1)}})

print("Transactions done in year 2020 deleted.")

# Close the MongoDB connection

client.close()

**21) Write a menu driven (add subtract, multiply and division, exit) program to handle two 1 D array**

def add\_arrays(arr1, arr2):

result = []

for i in range(len(arr1)):

result.append(arr1[i] + arr2[i])

return result

def subtract\_arrays(arr1, arr2):

result = []

for i in range(len(arr1)):

result.append(arr1[i] - arr2[i])

return result

def multiply\_arrays(arr1, arr2):

result = []

for i in range(len(arr1)):

result.append(arr1[i] \* arr2[i])

return result

def divide\_arrays(arr1, arr2):

result = []

for i in range(len(arr1)):

# Check for division by zero

if arr2[i] == 0:

result.append(float('inf')) # Infinity

else:

result.append(arr1[i] / arr2[i])

return result

def print\_array(arr):

print("[", end="")

for i in range(len(arr)):

if i == len(arr) - 1:

print(arr[i], end="")

else:

print(arr[i], end=", ")

print("]")

# Menu-driven program

while True:

print("Menu:")

print("1. Add arrays")

print("2. Subtract arrays")

print("3. Multiply arrays")

print("4. Divide arrays")

print("5. Exit")

choice = input("Enter your choice (1-5): ")

if choice == "1":

arr1 = input("Enter the elements of the first array separated by commas: ").split(",")

arr2 = input("Enter the elements of the second array separated by commas: ").split(",")

arr1 = [int(x) for x in arr1]

arr2 = [int(x) for x in arr2]

result = add\_arrays(arr1, arr2)

print("Result:")

print\_array(result)

elif choice == "2":

arr1 = input("Enter the elements of the first array separated by commas: ").split(",")

arr2 = input("Enter the elements of the second array separated by commas: ").split(",")

arr1 = [int(x) for x in arr1]

arr2 = [int(x) for x in arr2]

result = subtract\_arrays(arr1, arr2)

print("Result:")

print\_array(result)

elif choice == "3":

arr1 = input("Enter the elements of the first array separated by commas: ").split(",")

arr2 = input("Enter the elements of the second array separated by commas: ").split(",")

arr1 = [int(x) for x in arr1]

arr2 = [int(x) for x in arr2]

result = multiply\_arrays(arr1, arr2)

print("Result:")

print\_array(result)

elif choice == "4":

arr1 = input("Enter the elements of the first array separated by commas: ").split(",")

arr2 = input("Enter the elements of the second array separated by commas: ").split(",")

arr1 = [int(x) for x in arr1]

arr2 = [int(x) for x in arr2]

result = divide\_arrays(arr1, arr2)

print("Result:")

print\_array(result)

elif choice == "5":

print("Exiting the program.")

break

else:

print("Invalid choice. Please try again.")

**OUTPUT:-**

Menu:

1. Add arrays

2. Subtract arrays

3. Multiply arrays

4. Divide arrays

5. Exit

Enter your choice (1-5): 1

Enter the elements of the first array separated by commas: 1, 2, 3

Enter the elements of the second array separated by commas: 4, 5, 6

Result:

[5, 7, 9]

Menu:

1. Add arrays

2. Subtract arrays

3. Multiply arrays

4. Divide arrays

5. Exit

Enter your choice (1-5): 2

Enter the elements of the first array separated by commas: 7, 5, 9

Enter the elements of the second array separated by commas: 3, 2, 1

Result:

[4, 3, 8]