|  |  |
| --- | --- |
| https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcQdAoy8HAYiGwgqBMeq0EtfxKBc5i9totR_W7yOXHEosxAvJ3u6k6r7MVNRo_M3U1QAJMc&usqp=CAU | **E.G.S. Pillay Engineering College**  An Autonomous Institution Affiliated to Anna University, Chennai | Approved by AICTE, New Delhi  Accredited by NAAC with A++ Grade | Accredited by NBA T1 (B.E. – CSE, B.E. – ECE & B.Tech – IT)  Old Nagore Road, Thethi, Nagore Village, Nagapattinam – 611002, Tamil Nadu, India |

**Department Of Artificial Intelligence And Data Science**

**2301GE201 - PROGRAMMING USING PYTHON**

**LAB MANUAL**

**Experiment 1: Familiarization of Python 3 Working Environment & Basic Programs**

**A: Hello World**

print("Hello, World!")

**Output:**

Hello, World!

**B: Simple Addition**

a = 10

b = 20

sum = a + b

print("Sum:", sum)

**Output:**

Sum: 30

**C : Taking User Input**

name = input("Enter your name: ")

print("Hello", name)

**Output (example):**

Enter your name: Alice

Hello Alice

**D : Basic Calculator**

x = int(input("Enter first number: "))

y = int(input("Enter second number: "))

print("Sum:", x + y)

print("Difference:", x - y)

print("Product:", x \* y)

print("Quotient:", x / y)

**Output (example):**

Enter first number: 8

Enter second number: 2

Sum: 10

Difference: 6

Product: 16

Quotient: 4.0

**E : Area of Circle**

**Program:**

radius = float(input("Enter radius: "))

area = 3.14159 \* radius \* radius

print("Area of Circle:", area)

**Output (example):**

Enter radius: 5

Area of Circle: 78.53975

**Experiment 2: Implementation of Control Flow Statements: Numerical & Logical Problems**

**A: Check if Number is Even or Odd**

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

if num % 2 == 0:

print("Even number")

else:

print("Odd number")

**Output (example):**

Enter a number: 7

Odd number

**B: Find the Greatest of Three Numbers**

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

b = int(input("Enter second number: "))

c = int(input("Enter third number: "))

if a >= b and a >= c:

print("Greatest:", a)

elif b >= a and b >= c:

print("Greatest:", b)

else:

print("Greatest:", c)

**Output (example):**

Enter first number: 12

Enter second number: 45

Enter third number: 27

Greatest: 45

**C: Check Leap Year**

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

if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):

print(year, "is a leap year")

else:

print(year, "is not a leap year")

**Output (example):**

Enter a year: 2024

2024 is a leap year

**D: Number is Positive, Negative, or Zero**

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

if num > 0:

print("Positive number")

elif num == 0:

print("Zero")

else:

print("Negative number")

**Output (example):**

Enter a number: -3

Negative number

**E: Simple Grading System**

marks = int(input("Enter your marks: "))

if marks >= 90:

grade = 'A'

elif marks >= 80:

grade = 'B'

elif marks >= 70:

grade = 'C'

elif marks >= 60:

grade = 'D'

else:

grade = 'F'

print("Grade:", grade)

**Output (example):**

Enter your marks: 85

Grade: B

**Experiment 3: Problem Solving Using Functions**

**A: Find Factorial of a Number Using Function**

def factorial(n):

result = 1

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

result \*= i

return result

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

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

**Output (example):**

Enter a number: 5

Factorial: 120

**B: Check Prime Number Using Function**

def is\_prime(n):

if n <= 1:

return False

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

if n % i == 0:

return False

return True

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 (example):**

Enter a number: 11

11 is a prime number

**C: Find GCD (Greatest Common Divisor) Using Function**

def gcd(a, b):

while b:

a, b = b, a % b

return a

x = int(input("Enter first number: "))

y = int(input("Enter second number: "))

print("GCD:", gcd(x, y))

**Output (example):**

Enter first number: 60

Enter second number: 48

GCD: 12

**D: Fibonacci Series Using Function**

def fibonacci(n):

a, b = 0, 1

for \_ in range(n):

print(a, end=' ')

a, b = b, a + b

terms = int(input("Enter number of terms: "))

fibonacci(terms)

**Output (example):**

Enter number of terms: 6

0 1 1 2 3 5

**E: Sum of Digits Using Function**

def sum\_of\_digits(n):

total = 0

while n > 0:

total += n % 10

n //= 10

return total

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

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

**Output (example):**

Enter a number: 1234

Sum of digits: 10

**Experiment 4: Implementation of Lists as Stacks & Queues**

**A: Stack Implementation Using List**

stack = []

# Push elements

stack.append(10)

stack.append(20)

stack.append(30)

print("Stack after pushes:", stack)

# Pop element

popped\_element = stack.pop()

print("Popped element:", popped\_element)

print("Stack after pop:", stack)

**Output:**

Stack after pushes: [10, 20, 30]

Popped element: 30

Stack after pop: [10, 20]

**B: Queue Implementation Using List**

queue = []

# Enqueue elements

queue.append(10)

queue.append(20)

queue.append(30)

print("Queue after enqueues:", queue)

# Dequeue element

dequeued\_element = queue.pop(0)

print("Dequeued element:", dequeued\_element)

print("Queue after dequeue:", queue)

**Output:**

Queue after enqueues: [10, 20, 30]

Dequeued element: 10

Queue after dequeue: [20, 30]

**C: Stack with User Input and Menu**

stack = []

while True:

print("\n1. Push\n2. Pop\n3. Display\n4. Exit")

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

if choice == 1:

item = int(input("Enter item to push: "))

stack.append(item)

elif choice == 2:

if not stack:

print("Stack is empty")

else:

print("Popped:", stack.pop())

elif choice == 3:

print("Stack:", stack)

elif choice == 4:

break

else:

print("Invalid choice")

**Output (example):**

1. Push

2. Pop

3. Display

4. Exit

Enter choice: 1

Enter item to push: 50

Enter choice: 3

Stack: [50]

**D: Queue with User Input and Menu**

queue = []

while True:

print("\n1. Enqueue\n2. Dequeue\n3. Display\n4. Exit")

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

if choice == 1:

item = int(input("Enter item to enqueue: "))

queue.append(item)

elif choice == 2:

if not queue:

print("Queue is empty")

else:

print("Dequeued:", queue.pop(0))

elif choice == 3:

print("Queue:", queue)

elif choice == 4:

break

else:

print("Invalid choice")

**Output (example):**

1. Enqueue

2. Dequeue

3. Display

4. Exit

Enter choice: 1

Enter item to enqueue: 25

Enter choice: 3

Queue: [25]

**Experiment 5: Implementation of Tuples, Sequences, Sets & Dictionaries**

**A: Working with Tuples**

# Creating a tuple

my\_tuple = (10, 20, 30, 40)

# Accessing elements

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

# Slicing

print("Slice:", my\_tuple[1:3])

# Length

print("Length of tuple:", len(my\_tuple))

**Output:**

First element: 10

Slice: (20, 30)

Length of tuple: 4

**B: Working with Sequences (List Example)**

# List as sequence

my\_list = [1, 2, 3, 4, 5]

# Indexing

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

# Slicing

print("Slice:", my\_list[2:])

# Reversing

print("Reversed list:", my\_list[::-1])

**Output:**

Second element: 2

Slice: [3, 4, 5]

Reversed list: [5, 4, 3, 2, 1]

**C: Working with Sets**

# Creating sets

set1 = {1, 2, 3, 4}

set2 = {3, 4, 5, 6}

# Union

print("Union:", set1 | set2)

# Intersection

print("Intersection:", set1 & set2)

# Difference

print("Difference (set1 - set2):", set1 - set2)

**Output:**

Union: {1, 2, 3, 4, 5, 6}

Intersection: {3, 4}

Difference (set1 - set2): {1, 2}

**D: Working with Dictionaries**

# Creating a dictionary

student = {

"name": "Alice",

"age": 21,

"course": "Python"

}

# Accessing values

print("Name:", student["name"])

# Adding a new key-value pair

student["grade"] = "A"

# Iterating

for key, value in student.items():

print(key, ":", value)

**Output:**

Name: Alice

name : Alice

age : 21

course : Python

grade : A

**E: Frequency Count using Dictionary**

text = input("Enter a word: ")

freq = {}

for char in text:

if char in freq:

freq[char] += 1

else:

freq[char] = 1

print("Character Frequency:")

for char, count in freq.items():

print(char, ":", count)

**Output (example):**

Enter a word: hello

Character Frequency:

h : 1

e : 1

l : 2

o : 1

**Experiment 6: Programs Using Classes, Objects and Inheritance**

**A: Basic Class and Object**

class Student:

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

self.name = name

self.age = age

def display(self):

print("Name:", self.name)

print("Age:", self.age)

# Creating object

s1 = Student("Alice", 20)

s1.display()

**Output:**

Name: Alice

Age: 20

**B : Class with Methods and Constructor**

class Rectangle:

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

self.length = length

self.breadth = breadth

def area(self):

return self.length \* self.breadth

# Create object

r = Rectangle(10, 5)

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

**Output:**

Area of rectangle: 50

**C : Single Inheritance**

class Person:

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

self.name = name

def show(self):

print("Name:", self.name)

class Student(Person):

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

super().\_\_init\_\_(name)

self.course = course

def show\_course(self):

print("Course:", self.course)

# Create object of derived class

s = Student("Bob", "Python")

s.show()

s.show\_course()

**Output:**

Name: Bob

Course: Python

**D : Multiple Inheritance**

class Father:

def skills(self):

print("Gardening, Cooking")

class Mother:

def skills(self):

print("Art, Craft")

class Child(Father, Mother):

def skills(self):

Father.skills(self)

Mother.skills(self)

print("Dancing, Singing")

# Create object

c = Child()

c.skills()

**Output:**

Gardening, Cooking

Art, Craft

Dancing, Singing

**E: Multilevel Inheritance**

class Animal:

def sound(self):

print("Animals make sound")

class Dog(Animal):

def bark(self):

print("Dog barks")

class Puppy(Dog):

def weep(self):

print("Puppy weeps")

# Create object

p = Puppy()

p.sound()

p.bark()

p.weep()

**Output:**

Animals make sound

Dog barks

Puppy weeps

## ****Experiment 7: Exploring Input and Output Operations – Reading and Writing Files****

### ****Aim:****

To learn how to read from and write to files using Python.

### ****Theory:****

File handling in Python allows us to store data permanently in a file. Python provides built-in functions to perform basic operations on files like:

* Creating and opening a file
* Writing to a file
* Reading from a file
* Appending data
* Closing the file

File modes:

* 'r' – Read (default mode)
* 'w' – Write (creates new or overwrites existing file)
* 'a' – Append (adds data at the end)
* 'r+' – Read and write
* 'with open() – Ensures automatic file closing (best practice)

### ****Programs:****

### ****A: Writing to a File****

file = open("example.txt", "w")

file.write("This is the first line.\n")

file.write("This is the second line.")

file.close()

print("File written successfully.")

**Output:**

File written successfully.

### ****B: Reading from a File****

file = open("example.txt", "r")

content = file.read()

print("File Content:\n", content)

file.close()

**Output (example):**

File Content:

This is the first line.

This is the second line.

### ****C: Reading File Line by Line****

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

print("Reading line by line:")

for line in file:

print(line.strip())

**Output:**

Reading line by line:

This is the first line.

This is the second line.

### ****D: Appending to a File****

with open("example.txt", "a") as file:

file.write("\nThis line is added later.")

print("Data appended successfully.")

**Output:**

Data appended successfully.

### ****E: Writing and Reading Numbers****

# Writing numbers to a file

with open("numbers.txt", "w") as file:

for i in range(1, 6):

file.write(str(i) + "\n")

# Reading the numbers

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

print("Numbers in file:")

for line in file:

print(line.strip())

**Output:**

Numbers in file:

1

2

3

4

5

# ****Experiment 8: Exploring Modules & Packages****

### ****Aim:****

To learn how to use Python modules and packages by importing built-in and custom modules, and organizing code in packages.

### ****A: Using a Built-in Module (****math****)****

import math

print("Square root of 16:", math.sqrt(16))

print("Value of pi:", math.pi)

**Output:**

Square root of 16: 4.0

Value of pi: 3.141592653589793

### ****B: Creating and Using a Custom Module****

Create a file named mymodule.py:

def greet(name):

print("Hello,", name)

Then, in a separate file:

import mymodule

mymodule.greet("Alice")

**Output:**

Hello, Alice

### ****C: Using Packages****

Create this folder structure:

mypackage/

\_\_init\_\_.py

mod1.py

mod2.py

mod1.py:

def hello():

print("Hello from mod1!")

mod2.py:

def info():

print("Info from mod2!")

In main program:

from mypackage import mod1, mod2

mod1.hello()

mod2.info()

**Output:**

Hello from mod1!

Info from mod2!

# ****Experiment 9: Implementation of Errors and Exceptions****

### ****Aim:****

To understand and implement error handling in Python using exceptions.

### ****A: Handling ZeroDivisionError****

try:

a = 10

b = 0

c = a / b

except ZeroDivisionError:

print("Error: Division by zero is not allowed.")

**Output:**

Error: Division by zero is not allowed.

### ****B: Handling Multiple Exceptions****

try:

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

y = 10 / x

except ZeroDivisionError:

print("Error: Cannot divide by zero.")

except ValueError:

print("Error: Invalid input; please enter an integer.")

**Output (example):**

Enter a number: 0

Error: Cannot divide by zero.

Or

Enter a number: abc

Error: Invalid input; please enter an integer.

### ****C: Using else and finally****

try:

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

if num < 0:

raise ValueError("Negative number entered!")

except ValueError as ve:

print("Error:", ve)

else:

print("You entered:", num)

finally:

print("Execution completed.")

**Output (example):**

Enter a positive number: -5

Error: Negative number entered!

Execution completed.

Or

Enter a positive number: 10

You entered: 10

Execution completed.

### ****Result:****

Successfully implemented error and exception handling using try, except, else, and finally blocks to manage runtime errors gracefully.

# ****Experiment 10: Working with Python Standard Library Functions****

### ****Aim:****

To explore and use various standard library functions provided by Python for common tasks.

### ****Procedure:****

1. Import Python standard library modules like math, random, and datetime.
2. Use functions from these modules to perform mathematical calculations, generate random numbers, and handle dates and times.
3. Write programs that demonstrate the usage of at least three different standard library modules.
4. Run the programs and observe the output to understand how built-in functions simplify coding.

### ****A: Using**** math ****module****

import math

print("Square root of 49:", math.sqrt(49))

print("Factorial of 5:", math.factorial(5))

print("Value of e:", math.e)

**Output:**

Square root of 49: 7.0

Factorial of 5: 120

Value of e: 2.718281828459045

### ****B: Using**** random ****module****

import random

print("Random number between 1 and 10:", random.randint(1, 10))

print("Random choice from list:", random.choice(['apple', 'banana', 'cherry']))

**Output:**

Random number between 1 and 10: 7

Random choice from list: banana

### ****C: Using**** datetime ****module****

import datetime

now = datetime.datetime.now()

print("Current date and time:", now)

print("Year:", now.year)

print("Month:", now.month)

**Output:**

Current date and time: 2025-05-20 14:30:45.123456

Year: 2025

Month: 5

### ****Result:****

Successfully used Python standard library functions to perform mathematical operations, generate random values, and handle date and time data.

## ****Experiment 11: Creating Virtual Environments and Managing Packages with pip****

### ****Aim:****

To learn how to create a Python virtual environment and manage packages using pip.

### ****Apparatus/Software Required:****

* Python 3.x installed on your system
* Command prompt (Windows) / Terminal (Linux or Mac)

### ****Theory:****

Virtual environments help in isolating project-specific dependencies, avoiding conflicts between projects using different package versions.  
pip is the package manager for Python, used to install and manage additional libraries.

### ****Procedure:****

1. **Open the command prompt/terminal.**
2. **Navigate to your project directory:**
3. cd path\_to\_your\_project\_directory
4. **Create a virtual environment:**
5. python -m venv myenv
   * This creates a folder named myenv with an isolated Python setup.
6. **Activate the virtual environment:**
   * On **Windows**:
   * myenv\Scripts\activate
   * On **Linux/Mac**:
   * source myenv/bin/activate
   * The prompt will change to show the environment is activated, e.g., (myenv).
7. **Install packages using pip inside the virtual environment:**
8. pip install requests
9. **Verify the package installation:**
10. pip show requests
11. **Write and run a Python program using the installed package.**
12. **Deactivate the virtual environment when done:**
13. deactivate

### ****Sample Program:****

import requests

response = requests.get("https://api.github.com")

print("Status Code:", response.status\_code)

### ****Expected Output:****

Status Code: 200

### ****Result:****

The virtual environment was successfully created and activated. The package requests was installed using pip. The sample program executed correctly, demonstrating the use of an external package in an isolated environment.

# ****Experiment 12: Installing & Exploring NumPy, Pandas & Matplotlib for Real World Problems****

### ****Aim:****

To install and explore the basic features of NumPy, Pandas, and Matplotlib libraries used for solving real-world data problems.

**Procedure:**

**Step 1: Open Command Prompt / Terminal**

* **Windows:**
  + Press Win + R, type cmd, and press Enter.
  + Alternatively, search for "Command Prompt" in the Start menu.
* **Linux/Mac:**
  + Open the Terminal application from your system’s application menu.

**Step 2: Verify Python and pip Installation**

* Check if Python is installed by typing:
* python --version

or

python3 --version

* + You should see the version of Python installed. If not installed, download it from [python.org](https://www.python.org/downloads/).
* Check if pip is installed (pip is the Python package manager):
* pip --version

or

pip3 --version

* + If pip is missing, you need to install it first or reinstall Python ensuring the “Add Python to PATH” option is selected.

**Step 3: Install NumPy, Pandas, and Matplotlib Libraries**

* In the command prompt/terminal, run the following command to install all three libraries at once:
* pip install numpy pandas matplotlib
  + If you have multiple Python versions, you may need to use pip3 instead of pip.
* Wait while pip downloads and installs the packages. You will see messages about downloading and installing dependencies.
* After completion, you can verify by typing:
* pip list

and checking if numpy, pandas, and matplotlib are listed.

**Step 4: Open Your Python Environment**

* You can use any Python IDE or text editor (e.g., IDLE, VS Code, PyCharm, Jupyter Notebook), or just run Python interactively by typing:
* python

or

python3

**Step 5: Import Libraries and Test Basic Functionality**

* Type the following in your Python interpreter or script:
* import numpy as np
* import pandas as pd
* import matplotlib.pyplot as plt
* If no errors occur, the libraries are installed correctly.

**Step 6: Write and Run Sample Program for NumPy**

* Create a NumPy array and perform some basic operations:
* arr = np.array([1, 2, 3, 4, 5])
* print("Array:", arr)
* print("Mean:", np.mean(arr))
* print("Standard Deviation:", np.std(arr))
* This will display the array and calculate statistical measures.

**Step 7: Write and Run Sample Program for Pandas**

* Create a simple DataFrame and analyze the data:
* data = {
* 'Name': ['Alice', 'Bob', 'Charlie'],
* 'Age': [25, 30, 35],
* 'City': ['New York', 'Los Angeles', 'Chicago']
* }
* df = pd.DataFrame(data)
* print(df)
* print("\nAverage Age:", df['Age'].mean())
* This shows tabular data and calculates the average age.

**Step 8: Write and Run Sample Program for Matplotlib**

* Plot a simple line graph to visualize data:
* x = [1, 2, 3, 4, 5]
* y = [10, 15, 13, 17, 20]
* plt.plot(x, y)
* plt.title('Sample Line Plot')
* plt.xlabel('X-axis')
* plt.ylabel('Y-axis')
* plt.show()
* A graph window should open displaying the line plot.

**Step 9: Save Your Script (Optional)**

* If using an editor, save your Python script with .py extension, e.g., test\_libraries.py.
* Run the script from the command prompt/terminal by navigating to its folder and typing:
* python test\_libraries.py

**Step 10: Troubleshooting**

* If you get an error importing a module, ensure the library is installed in the correct Python environment.
* For multiple Python versions, use virtual environments to manage packages separately.
* Use pip install --upgrade pip to update pip if installations fail.

### ****Result:****

Libraries installed and basic programs ran successfully.

**ADDITIONAL PROGRAM:**

# 1. Even or Odd

def even\_odd(num):

if num % 2 == 0:

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

else:

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

even\_odd(10)

even\_odd(7)

# 2. Positive, Negative, or Zero

def pos\_neg\_zero(num):

if num > 0:

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

elif num < 0:

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

else:

print("Zero.")

pos\_neg\_zero(5)

pos\_neg\_zero(-3)

pos\_neg\_zero(0)

# 3. Maximum of Two Numbers

def max\_two(a, b):

if a > b:

print(f"{a} is greater.")

else:

print(f"{b} is greater or equal.")

max\_two(12, 8)

max\_two(5, 5)

# 4. Absolute Value

def absolute\_val(num):

if num < 0:

return -num

else:

return num

print(absolute\_val(-7))

print(absolute\_val(9))

# 5. Vowel or Consonant

def vowel\_consonant(char):

vowels = "aeiouAEIOU"

if char in vowels:

print(f"{char} is a vowel.")

else:

print(f"{char} is a consonant.")

vowel\_consonant('a')

vowel\_consonant('b')

# 6. Leap Year

def leap\_year(year):

if (year % 4 == 0 and year % 100 != 0) or year % 400 == 0:

print(f"{year} is a leap year.")

else:

print(f"{year} is not a leap year.")

leap\_year(2024)

leap\_year(1900)

# 7. Simple Calculator (if-else)

def calculator(num1, num2, operator):

if operator == '+':

print(num1 + num2)

elif operator == '-':

print(num1 - num2)

elif operator == '\*':

print(num1 \* num2)

elif operator == '/':

if num2 != 0:

print(num1 / num2)

else:

print("Cannot divide by zero.")

else:

print("Invalid operator.")

calculator(5, 2, '+')

calculator(10, 0, '/')

# 8. Grading System

def grade(marks):

if marks >= 90:

print("A")

elif marks >= 80:

print("B")

elif marks >= 70:

print("C")

elif marks >= 60:

print("D")

else:

print("F")

grade(85)

# 9. Check Divisibility

def divisible(num1, num2):

if num1 % num2 == 0:

print(f"{num1} is divisible by {num2}.")

else:

print(f"{num1} is not divisible by {num2}.")

divisible(10, 5)

# 10. Smallest of Three Numbers

def smallest\_three(a, b, c):

if a <= b and a <= c:

print(f"{a} is the smallest.")

elif b <= a and b <= c:

print(f"{b} is the smallest.")

else:

print(f"{c} is the smallest.")

smallest\_three(8, 3, 12)

# 11. While Loop Counting

i = 1

while i <= 10:

print(i)

i += 1

# 12. Sum of Natural Numbers (while)

def sum\_natural(n):

total = 0

i = 1

while i <= n:

total += i

i += 1

return total

print(sum\_natural(5))

# 13. Factorial (while)

def factorial\_while(n):

fact = 1

while n > 0:

fact \*= n

n -= 1

return fact

print(factorial\_while(5))

# 14. Reverse a Number (while)

def reverse\_num(num):

reversed\_num = 0

while num > 0:

digit = num % 10

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

num //= 10

return reversed\_num

print(reverse\_num(1234))

# 15. Palindrome Check (while)

def palindrome\_check(num):

original\_num = num

reversed\_num = 0

while num > 0:

digit = num % 10

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

num //= 10

if original\_num == reversed\_num:

return True

else:

return False

print(palindrome\_check(121))

print(palindrome\_check(123))

# 16. Prime Number Check (if, while)

def is\_prime(num):

if num <= 1:

return False

i = 2

while i \* i <= num:

if num % i == 0:

return False

i += 1

return True

print(is\_prime(17))

print(is\_prime(10))

# 17. Fibonacci Series (while)

def fibonacci(limit):

a, b = 0, 1

while a <= limit:

print(a, end=" ")

a, b = b, a + b

fibonacci(50)

print()

# 18. Multiplication Table (while)

def multiplication\_table(num):

i = 1

while i <= 10:

print(f"{num} x {i} = {num \* i}")

i += 1

multiplication\_table(7)

# 19. Count Digits (while)

def count\_digits(num):

count = 0

while num != 0:

num //= 10

count += 1

return count

print(count\_digits(12345))

# 20. Sum of Digits (while)

def sum\_digits(num):

total = 0

while num != 0:

digit = num % 10

total += digit

num //= 10

return total

print(sum\_digits(123))

# 21. Function to Add Two Numbers

def add(a, b):

return a + b

print(add(3, 5))

# 22. Function to Find Max

def find\_max(a, b):

return max(a, b)

print(find\_max(10, 6))

# 23. Function to Check Even/Odd

def is\_even(num):

return num % 2 == 0

print(is\_even(8))

# 24. Function to Calculate Area of a Circle

import math

def circle\_area(radius):

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

print(circle\_area(5))

# 25. Function to Calculate Area of a Rectangle

def rectangle\_area(length, width):

return length \* width

print(rectangle\_area(4, 6))

# 26. Function to Check Prime

def is\_prime\_func(num):

if num <= 1:

return False

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

if num % i == 0:

return False

return True

print(is\_prime\_func(13))

# 27. Function to Calculate Factorial

def factorial\_func(n):

if n == 0:

return 1

else:

result = 1

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

result \*= i

return result

print(factorial\_func(6))