**Lab 1. Basic Python Program (Hello World)**

print("Hello, World! Welcome to Python Programming")

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

**output:** Hello, World! Welcome to Python Programming

**----------------------------------------------------------------------------------------------------**

**Lab 2. Data Types & Operators**

a = 10  # Integer

b = 3.14  # Float

c = "Python"  # String

d = [1, 2, 3]  # List

e = (4, 5, 6)  # Tuple

f = {"name": "Mohan", "age": 25}  # Dictionary

g = a + b  # Arithmetic Operator

h = a > 5  # Comparison Operator

print(type(a), type(b), type(c))

print("Addition:", g, "Comparison:", h)

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

**output:**

<class 'int'> <class 'float'> <class 'str'>

Addition: 13.14 Comparison: True

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

**Lab 3. Conditional Statements (if-else)**

num = 15

if num % 2 == 0:

print("Even Number")

else:

print("Odd Number")

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

**output:** Odd Number

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

year = int(input("enter the year to check whether it is a leap or not"))

if year%4==0 and year%100 != 0 and year%400!=0 :

print("Leap year")

else:

print("it is not a leap year")

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

OUTPUT

enter the year to check whether it is a leap or not2024

Leap year

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

**WAP to Guess a number**

var\_num = 23

guess\_num = int(input('Enter the number\n'))

if guess\_num == var\_num:

print('you are astrologer')

print('congratulations')

elif var\_num < guess\_num <= 50:

print('you are moving away from the actuals')

print('please think towards it')

elif 0 < guess\_num < var\_num:

print('you are pessimistic thiking')

print('think positively')

else :

print('you are too much away')

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

OUTPUT

Enter the number

60

you are too much away

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

**Lab 4. Loops (For and While)**

# For Loop

for i in range(3):

    print("Iteration:", i)

# While Loop

count = 1

while count <= 3:

    print("While Loop Count:", count)

    count += 1

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

OUTPUT

Iteration: 0

Iteration: 1

Iteration: 2

While Loop Count: 1

While Loop Count: 2

While Loop Count: 3

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

# Program to find the sum of numbers in a list

numbers = [2, 4, 6, 8, 10]

sum\_numbers = 0

for num in numbers:

    sum\_numbers += num

print("Sum:", sum\_numbers)

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

OUTPUT

Sum: 30

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

**Lab 5. Functions in Python**

def greet(name="Guest"):

print("Hello,", name)

greet("Mohan")

greet()

**output:**   
Hello, Mohan

Hello, Guest

def my\_func():

    print('welcome to the functions')

    print('start learning function to simplify the programs')

my\_func()

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

OUTPUT

welcome to the functions

start learning function to simplify the programs

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

1. **WAP to pass argument to the function and print the same**

def my\_func\_arg(message):

print('the argument passed in function', message)

# function with arguments

my\_func\_arg('Mallikarjuna')

my\_func\_arg('Trainer from UPL')

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

OUTPUT

the argument passed in function Mallikarjuna

the argument passed in function Trainer from UPL

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

1. **WAP to function with two arguments and print the same**

def my\_func\_two\_arg(firstname, secondname):

print('the argument passed in function')

print('Name is :', firstname, secondname)

# function with two arguments

my\_func\_two\_arg('Mallikarjuna', 'Mysuru')

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

OUTPUT

the argument passed in function

Name is : Mallikarjuna Mysuru

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

1. **WAP to pass variable number of arguments and print the same**

def my\_func\_varargs(\*args):

print('the argument', args)

print('the argument passed in function', args[3])

# function with variable number of arguments

my\_func\_varargs('Mallikarjuna', 'Mysuru', '9739201041', 'Trainer', 'Software Engineer')

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

OUTPUT

the argument ('Mallikarjuna', 'Mysuru', '9739201041', 'Trainer', 'Software Engineer')

the argument passed in function Trainer

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

1. **WAP to computational arithmetic operations using function**

def add(x, y):

return x + y

def subtract(x, y):

return x - y

def multiply(x, y):

return x \* y

def divide(x, y):

if y != 0:

return x / y

else:

return "Cannot divide by zero"

def calculate(operation, x, y):

operations = {

'add': add,

'subtract': subtract,

'multiply': multiply,

'divide': divide,

}

# Get the function based on the operation or use a default function

selected\_operation = operations.get(operation, lambda x, y: "Invalid operation")

# Perform the calculation

result = selected\_operation(x, y)

return result

# Example usage:

result\_add = calculate('add', 5, 3)

print("Addition:", result\_add)

result\_multiply = calculate('multiply', 4, 6)

print("Multiplication:", result\_multiply)

result\_invalid = calculate('power', 2, 3)

print("Invalid Operation:", result\_invalid)

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

OUTPUT

Addition: 8

Multiplication: 24

Invalid Operation: Invalid operation

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

**Lab 6. Modules & Importing Libraries**

import math

print("Square Root of 25:", math.sqrt(25))

**output:** Square Root of 25: 5.0

**Lab 7. Class and Object in Python**

class Car:

def \_\_init\_\_(self, brand, color):

self.brand = brand

self.color = color

def display(self):

print(f"Car Brand: {self.brand}, Color: {self.color}")

car1 = Car("Toyota", "Red")

car1.display()

**output:** Car Brand: Toyota, Color: Red

1. **WAP create Employee and Display information (class and objects)**

class Employee:

name = ""

design = ""

sal= 0.0

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

self.name = name

self.design = design

self.sal = salar

def display(self):

print("Employee Details:;")

print(self.name ," :: " , self.design , " :: " ,self.sal)

Employee("Mallikarjuna", "Software", 1200000).display()

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

OUTPUT:

Employee Details:;

Mallikarjuna :: Software :: 1200000

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

1. **WAP to build the calculator (class and objects)**

class Calculator:

firstNumber=0.0

secondNumber=0.0

def \_\_init\_\_(self,fn,sn):

self.firstNumber=fn

self.secondNumber=sn

def add(self):

return self.firstNumber + self.secondNumber

def sub(self):

return self.firstNumber - self.secondNumber

def mul(self):

return self.firstNumber \* self.secondNumber

def div(self):

return self.firstNumber % self.secondNumber

fn = float(input("enter the first number"))

sn = float(input("enter the second number"))

cal= Calculator(fn,sn)

print("Add::",cal.add())

print("Sub::",cal.sub())

print("Mul::",cal.mul())

print("Div::",cal.div())

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

OUTPUT:

enter the first number100

enter the second number22

Add:: 122.0

Sub:: 78.0

Mul:: 2200.0

Div:: 12.0

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

**Lab 8. Encapsulation**

class BankAccount:

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

self.\_\_balance = balance

def deposit(self, amount):

self.\_\_balance += amount

def get\_balance(self):

return self.\_\_balance

account = BankAccount(1000)

account.deposit(500)

print("Bank Balance:", account.get\_balance())

**output:** Bank Balance: 1500

**Lab 9. Inheritance**

class Animal:

def sound(self):

print("Animals make sounds")

class Dog(Animal):

def sound(self):

print("Dogs bark")

dog = Dog()

dog.sound()

**output:** Dogs bark

**Lab 10. Polymorphism**

class Bird:

def fly(self):

print("Birds can fly")

class Penguin(Bird):

def fly(self):

print("Penguins cannot fly")

animals = [Bird(), Penguin()]

for animal in animals:

animal.fly()

**output:** Birds can fly

Penguins cannot fly

**Lab 11. Abstraction**

from abc import ABC, abstractmethod

class Employee(ABC):

@abstractmethod

def get\_salary(self):

pass

class Developer(Employee):

def get\_salary(self):

return 50000

emp = Developer()

print("Developer Salary:", emp.get\_salary())

**output:** Developer Salary: 50000

**Lab 12. File Handling (Read & Write)**

with open("sample.txt", "w") as f:

f.write("This is a sample file.")

with open("sample.txt", "r") as f:

print("File Content:", f.read())

**output:** File Content: This is a sample file.

1. **WAP to read file from path**

def read\_file(file\_path):

try:

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

content = file.read()

print("File content:")

print(content)

except FileNotFoundError:

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

# Example usage

read\_file("E:/temp/example.txt")

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

OUTPUT:

File content:

def read\_file(file\_path):

try:

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

content = file.read()

print("File content:")

print(content)

except FileNotFoundError:

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

# Example usage

read\_file("example.txt")

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

1. **WAP to demonstrate write to file**

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

try:

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

file.write(content)

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

except IOError as e:

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

# Example usage

write\_to\_file("./example.txt", "Hello, World!")

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

OUTPUT:

Content successfully written to './example.txt'.

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

1. **WAP to demonstrate Appending to File**

def append\_to\_file(file\_path, content):

try:

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

file.write(content)

print(f"Content successfully appended to '{file\_path}'.")

except IOError as e:

print(f"Error appending to file: {e}")

# Example usage

append\_to\_file("./example.txt", "\nAppended content.")

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

OUTPUT:

Content successfully appended to './example.txt'.

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

**Lab 13. Exception Handling**

try:

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

print("Result:", 10 / num)

except ZeroDivisionError:

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

except ValueError:

print("Error: Invalid input")

**output:** Error: Division by zero is not allowed

**Lab 14. Pandas Data Handling**

import pandas as pd

data = {'Name': ['Alice', 'Bob', 'Charlie'], 'Age': [25, 30, 35], 'City': ['New York', 'London', 'Paris']}

df = pd.DataFrame(data)

print(df.head())

**output:**

**Name Age City**

0 Alice 25 New York

1 Bob 30 London

2 Charlie 35 Paris

Example on numpy and pandas

import pandas as pd

import numpy as np

# Creating a DataFrame from a dictionary

dict\_data = {

'Name': ['Virat Kohli', 'Rohit Sharma', 'MS Dhoni'],

'Age': [35, 36, 42],

'City': ['Delhi', 'Mumbai', 'Ranchi']

}

df\_dict = pd.DataFrame(dict\_data)

print("DataFrame from Dictionary:")

print(df\_dict)

print()

# Creating a DataFrame from a list

list\_data = [

['Virat Kohli', 35, 'Delhi'],

['Rohit Sharma', 36, 'Mumbai'],

['MS Dhoni', 42, 'Ranchi']

]

columns = ['Name', 'Age', 'City']

df\_list = pd.DataFrame(list\_data, columns=columns)

print("DataFrame from List:")

print(df\_list)

print()

# Creating a DataFrame from a set

set\_data = {

'Virat Kohli', 35, 'Delhi',

'Rohit Sharma', 36, 'Mumbai',

'MS Dhoni', 42, 'Ranchi'

}

# Convert the set to a list and reshape it into a 3x3 matrix (3 rows, 3 columns)

list\_data = list(set\_data)

num\_columns = 3

num\_rows = len(list\_data) // num\_columns

reshaped\_list = [list\_data[i \* num\_columns:(i + 1) \* num\_columns] for i in range(num\_rows)]

# Define column names

columns = ['Name', 'Age', 'City']

# Create the DataFrame

df\_set = pd.DataFrame(reshaped\_list, columns=columns)

# Display the resulting DataFrame

print("DataFrame from Set:")

print(df\_set)

# Creating a DataFrame from a NumPy ndarray

ndarray\_data = np.array([

['Virat Kohli', 35, 'Delhi'],

['Rohit Sharma', 36, 'Mumbai'],

['MS Dhoni', 42, 'Ranchi']

])

# Define column names

columns = ['Name', 'Age', 'City']

# Create the DataFrame

df\_ndarray = pd.DataFrame(ndarray\_data, columns=columns)

# Display the resulting DataFrame

print("DataFrame from NumPy ndarray:")

print(df\_ndarray)

# Creating a DataFrame from an Iterable (tuple)

iterable\_data = (

('Virat Kohli', 35, 'Delhi'),

('Rohit Sharma', 36, 'Mumbai'),

('MS Dhoni', 42, 'Ranchi')

)

# Define column names

columns = ['Name', 'Age', 'City']

# Create the DataFrame

df\_iterable = pd.DataFrame(iterable\_data, columns=columns)

# Display the resulting DataFrame

print("DataFrame from Iterable:")

print(df\_iterable)

# Creating a DataFrame (original)

data = {

'Name': ['Virat Kohli', 'Rohit Sharma', 'MS Dhoni'],

'Age': [35, 36, 42],

'City': ['Delhi', 'Mumbai', 'Ranchi']

}

original\_df = pd.DataFrame(data)

# Creating a DataFrame from an existing DataFrame (copying)

copied\_df = original\_df.copy()

# Display the original and copied DataFrames

print("Original DataFrame:")

print(original\_df)

print("\nDataFrame created by copying another DataFrame:")

print(copied\_df)

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

**OUTPUT:**

DataFrame from Dictionary:

Name Age City

0 Virat Kohli 35 Delhi

1 Rohit Sharma 36 Mumbai

2 MS Dhoni 42 Ranchi

DataFrame from List:

Name Age City

0 Virat Kohli 35 Delhi

1 Rohit Sharma 36 Mumbai

2 MS Dhoni 42 Ranchi

DataFrame from Set:

Name Age City

0 35 36 Rohit Sharma

1 Virat Kohli Delhi 42

2 Mumbai Ranchi MS Dhoni

DataFrame from NumPy ndarray:

Name Age City

0 Virat Kohli 35 Delhi

1 Rohit Sharma 36 Mumbai

2 MS Dhoni 42 Ranchi

DataFrame from Iterable:

Name Age City

0 Virat Kohli 35 Delhi

1 Rohit Sharma 36 Mumbai

2 MS Dhoni 42 Ranchi

Original DataFrame:

Name Age City

0 Virat Kohli 35 Delhi

1 Rohit Sharma 36 Mumbai

2 MS Dhoni 42 Ranchi

DataFrame created by copying another DataFrame:

Name Age City

0 Virat Kohli 35 Delhi

1 Rohit Sharma 36 Mumbai

2 MS Dhoni 42 Ranchi

**---------------------------------------------------------------------------------------------**

**Lab 15. NumPy Basics**

import numpy as np

arr = np.array([1, 2, 3, 4, 5])

print("NumPy Array:", arr)

print("Mean of array:", np.mean(arr))

**output:** NumPy Array: [1 2 3 4 5]

Mean of array: 3.0

import numpy as np

# Create an array of numbers

numbers\_array = np.array([2, 4, 6, 8, 10])

# Calculate the mean, median, and standard deviation

mean\_value = np.mean(numbers\_array)

median\_value = np.median(numbers\_array)

std\_dev\_value = np.std(numbers\_array)

print("Original Array:")

print(numbers\_array)

print("\nMean:", mean\_value)

print("Median:", median\_value)

print("Standard Deviation:", std\_dev\_value)

# Normalize the array (subtract the mean and divide by the standard deviation)

normalized\_array = (numbers\_array - mean\_value) / std\_dev\_value

print("\nNormalized Array:")

print(normalized\_array)

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

OUTPUT

Original Array:

[ 2 4 6 8 10]

Mean: 6.0

Median: 6.0

Standard Deviation: 2.8284271247461903

Normalized Array:

[-1.41421356 -0.70710678 0. 0.70710678 1.41421356]

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

**Create two matrices and perform matrix multiplication.**

**Calculate the determinant of a 3x3 matrix.**

**Solve a system of linear equations using NumPy**

import numpy as np

# Create two matrices and perform matrix multiplication

matrix\_a = np.array([[1, 2], [3, 4]])

matrix\_b = np.array([[5, 6], [7, 8]])

# Matrix Multiplication

matrix\_multiplication\_result = np.dot(matrix\_a, matrix\_b)

print("Matrix Multiplication Result:")

print(matrix\_multiplication\_result)

# Calculate the determinant of a 3x3 matrix

matrix\_3x3 = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

determinant\_value = np.linalg.det(matrix\_3x3)

print("\nDeterminant of the 3x3 Matrix:")

print(determinant\_value)

# Solve a system of linear equations

# For example: 2x + y = 5, 3x - 2y = 8

coefficients\_matrix = np.array([[2, 1], [3, -2]])

constants\_vector = np.array([5, 8])

solution\_vector = np.linalg.solve(coefficients\_matrix, constants\_vector)

print("\nSolution to the System of Linear Equations:")

print(solution\_vector)

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

OUTPUT

Matrix Multiplication Result:

[[19 22]

[43 50]]

Determinant of the 3x3 Matrix:

0.0

Solution to the System of Linear Equations:

[ 2.57142857 -0.14285714]

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

**Lab 16. try except block**

try:

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

result = 10 / num # May raise ZeroDivisionError

print("Result:", result)

except ZeroDivisionError:

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

except ValueError:

print("Error: Invalid input! Please enter a number.")

except Exception as e:

print("An unexpected error occurred:", e)

finally:

print("Execution completed.")

**Lab 17. integration of Python basics, control flow, functions, OOP, file handling, and data processing into a single application that simulates a simple banking system**

"""

Comprehensive Python Program Demonstrating All Key Concepts

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

This program integrates Python basics, control flow, functions, OOP, file handling,

and data processing into a single application that simulates a simple banking system.

"""

**# Importing required modules**

import math

import pandas as pd

import numpy as np

from abc import ABC, abstractmethod

**# Step 1: Define a Bank Customer Class (OOP - Class, Encapsulation, Inheritance, Polymorphism)**

class BankAccount(ABC):  # Abstract Base Class

    def \_\_init\_\_(self, name, balance=0):

        self.name = name

        self.\_\_balance = balance  # Private attribute

    def deposit(self, amount):

        if amount > 0:

            self.\_\_balance += amount

            print(f"Deposited {amount} successfully!")

        else:

            print("Deposit amount must be positive!")

    def get\_balance(self):

        return self.\_\_balance

    @abstractmethod

    def account\_type(self):

        pass

**# Step 2: Create Derived Classes (Inheritance & Polymorphism)**

class SavingsAccount(BankAccount):

    def account\_type(self):

        return "Savings Account"

    def calculate\_interest(self):

        return self.get\_balance() \* 0.04  # 4% Interest

class CurrentAccount(BankAccount):

    def account\_type(self):

        return "Current Account"

**# Step 3: User Interaction (Input Handling, Conditional Statements, Loops)**

**main.py**

from SavingsAccount import SavingsAccount

from SavingsAccount import CurrentAccount

import pandas as pd

import numpy as np

def main():

    print("Welcome to the Python Bank System!")

    name = input("Enter your name: ")

    acc\_type = input("Choose account type (Savings/Current): ").strip().lower()

    if acc\_type == "savings":

        account = SavingsAccount(name, balance=1000)

    else:

        account = CurrentAccount(name, balance=500)

    print(f"Account Created: {account.account\_type()} for {name}")

    while True:

        print("\n1. Deposit\n2. Check Balance\n3. Interest (Only for Savings)\n4. Exit")

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

        if choice == 1:

            amount = float(input("Enter deposit amount: "))

            account.deposit(amount)

        elif choice == 2:

            print(f"Current Balance: {account.get\_balance()}")

        elif choice == 3 and isinstance(account, SavingsAccount):

            print(f"Interest Earned: {account.calculate\_interest()}")

        elif choice == 4:

            print("Thank you for banking with Python Bank!")

            break

        else:

            print("Invalid choice! Try again.")

**# Step 4: File Handling (Save Transactions)**

    with open("bank\_log.txt", "w") as file:

        file.write(f"Customer: {name}\nAccount Type: {account.account\_type()}\nFinal Balance: {account.get\_balance()}\n")

        print("Transaction details saved to bank\_log.txt")

**# Step 5: Data Processing (Pandas & NumPy)**

def process\_customer\_data():

    data = {'Customer': ['', 'Bob', 'Charlie'], 'Balance': [2500, 4000, 3200], 'Account': ['Savings', 'Current', 'Savings']}

    df = pd.DataFrame(data)

    print("\nCustomer Data:\n", df)

    # NumPy Operations

    balances = np.array(df['Balance'])

    print("\nBalance Statistics:")

    print("Mean Balance:", np.mean(balances))

    print("Maximum Balance:", np.max(balances))

    print("Minimum Balance:", np.min(balances))

try:

    main()

    process\_customer\_data()

except Exception as e:

    print("An error occurred:", e)

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

OUTPUT:

Welcome to the Python Bank System!

Enter your name: sachin

Choose account type (Savings/Current): savings

Account Created: Savings Account for sachin

1. Deposit

2. Check Balance

3. Interest (Only for Savings)

4. Exit

Enter choice: 1

Enter deposit amount: 2222

Deposited 2222.0 successfully!

1. Deposit

2. Check Balance

3. Interest (Only for Savings)

4. Exit

Enter choice: 2

Current Balance: 3222.0

1. Deposit

2. Check Balance

3. Interest (Only for Savings)

4. Exit

Enter choice: 3

Interest Earned: 128.88

1. Deposit

2. Check Balance

3. Interest (Only for Savings)

4. Exit

Enter choice: 4

Thank you for banking with Python Bank!

Transaction details saved to bank\_log.txt

Customer Data:

Customer Balance Account

0 Sachin 2500 Savings

1 Virat 4000 Current

2 Dhoni 3200 Savings

Balance Statistics:

Mean Balance: 3233.3333333333335

Maximum Balance: 4000

Minimum Balance: 2500

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