**CYCLE 6**

**PROGRAM 1**

**Aim** : Define a class to represent a bank account. Include the following details like name of the depositor, account number, type of account, balance amount in the account. Write methods to assign initial values, to deposit an amount, withdraw an amount after checking the balance, to display details such as name, account number, account type and balance.

**Source code :**

class BankAccount:

def \_\_init\_\_(self, name, account\_number, account\_type, balance):

self.name = name

self.account\_number = account\_number

self.account\_type = account\_type

self.balance = balance

def deposit(self, amount):

"""Method to deposit an amount into the account."""

if amount > 0:

self.balance += amount

print(f"Deposited {amount}. New balance is {self.balance}.")

else:

print("Deposit amount must be positive.")

def withdraw(self, amount):

"""Method to withdraw an amount from the account."""

if amount <= 0:

print("Withdrawal amount must be positive.")

elif amount > self.balance:

print("Insufficient balance.")

else:

self.balance -= amount

print(f"Withdrew {amount}. New balance is {self.balance}.")

def display\_details(self):

"""Method to display account details."""

print(f"\nAccount Holder: {self.name}")

print(f"Account Number: {self.account\_number}")

print(f"Account Type: {self.account\_type}")

print(f"Balance: {self.balance}")

print("Enter account details to create a new account:")

name = input("Enter account holder name: ")

account\_number = input("Enter account number: ")

account\_type = input("Enter account type (e.g., Savings, Current): ")

balance = float(input("Enter initial balance: "))

account1 = BankAccount(name, account\_number, account\_type, balance)

account1.display\_details()

while True:

print("\nChoose an operation:")

print("1. Deposit")

print("2. Withdraw")

print("3. Display Account Details")

print("4. Exit")

choice = input("Enter your choice (1/2/3/4): ")

if choice == "1":

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

account1.deposit(deposit\_amount)

elif choice == "2":

withdraw\_amount = float(input("Enter withdrawal amount: "))

account1.withdraw(withdraw\_amount)

elif choice == "3":

account1.display\_details()

elif choice == "4":

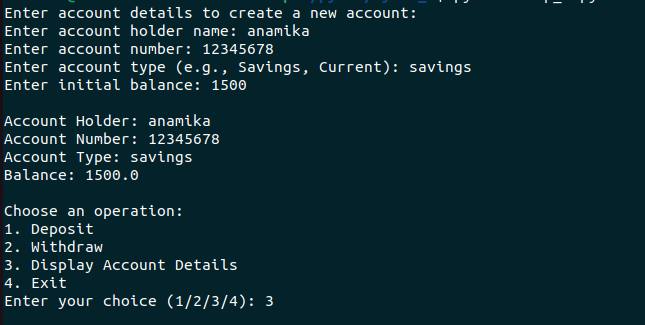
print("Exiting the program.")

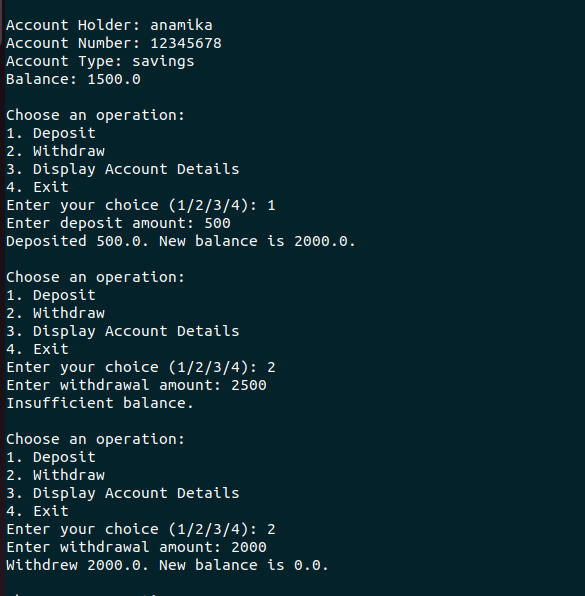
break

else:

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

**Output :**

****

****

**PROGRAM 2**

**Aim :** Create a class Publisher with attributes publisher id and publisher name. Derive class Book from Publisher with attributes title and author.

Derive class Python from Book with attributes price and no\_of\_pages. Write a program that displays information about a Python book. Use base class constructor invocation and method overriding.

**Source code :**

class Publisher:

def \_\_init\_\_(self, publisher\_id, publisher\_name):

self.publisher\_id = publisher\_id

self.publisher\_name = publisher\_name

def display\_publisher(self):

print(f"Publisher ID: {self.publisher\_id}")

print(f"Publisher Name: {self.publisher\_name}")

class Book(Publisher):

def \_\_init\_\_(self, publisher\_id, publisher\_name, title, author):

super().\_\_init\_\_(publisher\_id, publisher\_name)

self.title = title

self.author = author

def display\_book(self):

print(f"Title: {self.title}")

print(f"Author: {self.author}")

class Python(Book):

def \_\_init\_\_(self, publisher\_id, publisher\_name, title, author, price, no\_of\_pages):

super().\_\_init\_\_(publisher\_id, publisher\_name, title, author)

self.price = price

self.no\_of\_pages = no\_of\_pages

def display\_python\_book(self):

print(f"Price: {self.price}")

print(f"Number of Pages: {self.no\_of\_pages}")

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

publisher\_id = input("Enter Publisher ID: ")

publisher\_name = input("Enter Publisher Name: ")

title = input("Enter Book Title: ")

author = input("Enter Author Name: ")

price = float(input("Enter Book Price: "))

no\_of\_pages = int(input("Enter Number of Pages: "))

print()

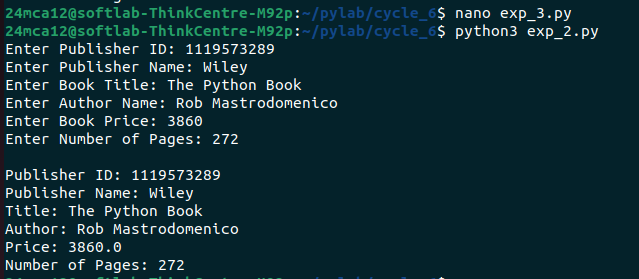
python\_book = Python(publisher\_id, publisher\_name, title, author, price, no\_of\_pages)

python\_book.display\_publisher()

python\_book.display\_book()

python\_book.display\_python\_book()

**Output**



**PROGRAM 3**

**Aim :** Write a program that has an abstract class Polygon. Derive two classes Rectangle and Triangle from Polygon and write methods to get the details of their dimensions and hence calculate the area.

**Source code :**

from abc import ABC, abstractmethod

class Polygon(ABC):

@abstractmethod

def get\_dimensions(self):

pass

@abstractmethod

def calculate\_area(self):

pass

class Rectangle(Polygon):

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

self.length = 0

self.width = 0

def get\_dimensions(self):

self.length = float(input("Enter the length of the rectangle: "))

self.width = float(input("Enter the width of the rectangle: "))

def calculate\_area(self):

area = self.length \* self.width

return area

class Triangle(Polygon):

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

self.base = 0

self.height = 0

def get\_dimensions(self):

self.base = float(input("Enter the base of the triangle: "))

self.height = float(input("Enter the height of the triangle: "))

def calculate\_area(self):

area = 0.5 \* self.base \* self.height

return area

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

print("Choose a polygon to calculate area:")

print("1. Rectangle")

print("2. Triangle")

choice = input("Enter your choice (1/2): ")

if choice == "1":

rectangle = Rectangle()

rectangle.get\_dimensions()

area = rectangle.calculate\_area()

print(f"The area of the rectangle is: {area}")

elif choice == "2":

triangle = Triangle()

triangle.get\_dimensions()

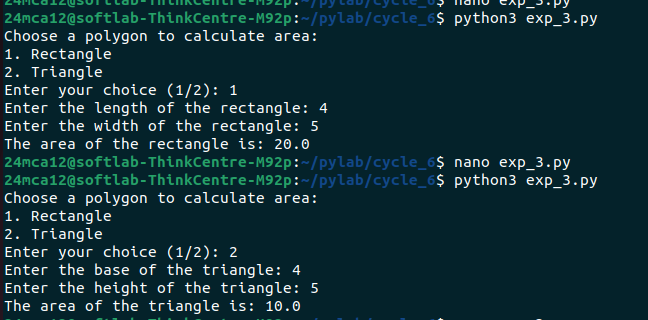
area = triangle.calculate\_area()

print(f"The area of the triangle is: {area}")

else:

print("Invalid choice. Please select either 1 or 2.")

**Output :**

****

**PROGRAM 4**

**Aim :** Create a Rectangle class with attributes length and breadth and methods to find area and perimeter. Compare two Rectangle objects by their area.

**Source code :**

class Rectangle:

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

self.length = length

self.breadth = breadth

def area(self):

"""Method to calculate the area of the rectangle."""

return self.length \* self.breadth

def perimeter(self):

"""Method to calculate the perimeter of the rectangle."""

return 2 \* (self.length + self.breadth)

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

print("Enter the dimensions for Rectangle 1:")

length1 = float(input("Enter the length of the rectangle: "))

breadth1 = float(input("Enter the breadth of the rectangle: "))

rect1 = Rectangle(length1, breadth1)

print("\nEnter the dimensions for Rectangle 2:")

length2 = float(input("Enter the length of the rectangle: "))

breadth2 = float(input("Enter the breadth of the rectangle: "))

rect2 = Rectangle(length2, breadth2)

print(f"\nRectangle 1 - Area: {rect1.area()} | Perimeter: {rect1.perimeter()}")

print(f"Rectangle 2 - Area: {rect2.area()} | Perimeter: {rect2.perimeter()}")

area1 = rect1.area()

area2 = rect2.area()

if area1 < area2:

print("\nRectangle 1 has a smaller area than Rectangle 2.")

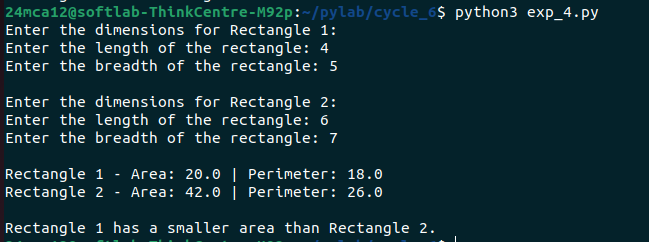
elif area1 == area2:

print("\nRectangle 1 and Rectangle 2 have the same area.")

else:

print("\nRectangle 1 has a larger area than Rectangle 2.")

**Output :**

****

**PROGRAM 5**

**Aim :** Create a class Time with private attributes hour, minute and second. Overload ‘+’ operator to find sum of 2 times.

**Source code :**

class Time:

def \_\_init\_\_(self, hour=0, minute=0, second=0):

self.\_\_hour = hour

self.\_\_minute = minute

self.\_\_second = second

def \_normalize(self):

if self.\_\_second >= 60:

self.\_\_minute += self.\_\_second // 60

self.\_\_second = self.\_\_second % 60

if self.\_\_minute >= 60:

self.\_\_hour += self.\_\_minute // 60

self.\_\_minute = self.\_\_minute % 60

if self.\_\_hour >= 24:

self.\_\_hour = self.\_\_hour % 24

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

if not isinstance(other, Time):

raise TypeError("Operand must be of type 'Time'")

total\_hour = self.\_\_hour + other.\_\_hour

total\_minute = self.\_\_minute + other.\_\_minute

total\_second = self.\_\_second + other.\_\_second

result = Time(total\_hour, total\_minute, total\_second)

result.\_normalize()

return result

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

return f"{self.\_\_hour:02d}:{self.\_\_minute:02d}:{self.\_\_second:02d}"

def get\_time\_from\_user():

while True:

try:

hour = int(input("Enter hours (0-23): "))

minute = int(input("Enter minutes (0-59): "))

second = int(input("Enter seconds (0-59): "))

if not (0 <= hour <= 23) or not (0 <= minute <= 59) or not (0 <= second <= 59):

print("Invalid time input. Please enter valid values for hour, minute, and second.")

continue

return Time(hour, minute, second)

except ValueError:

print("Invalid input. Please enter integers for hours, minutes, and seconds.")

print("Enter details for Time 1:")

time1 = get\_time\_from\_user()

print("\nEnter details for Time 2:")

time2 = get\_time\_from\_user()

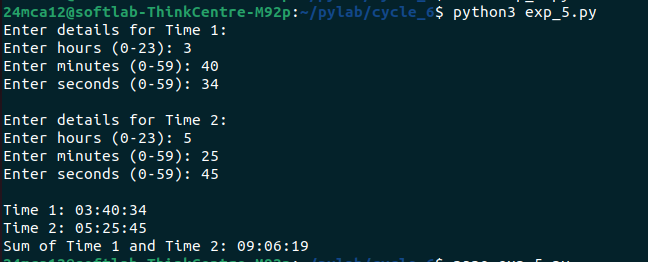
print("\nTime 1:", time1)

print("Time 2:", time2)

time\_sum = time1 + time2

print("Sum of Time 1 and Time 2:", time\_sum)

**Output :**

****