

CYCLE 6

EXPEREMENT 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, balance=0):
        self.name = name
        self.account_number = account_number
        self.balance = balance

    def deposit(self, amount):
        if amount > 0:
            self.balance += amount
            print(f"Deposited ${amount}")
        else:
            print("Invalid deposit amount")

    def withdraw(self, amount):
        if amount > 0 and amount <= self.balance:
            self.balance -= amount
            print(f"Withdrew ${amount}")
        else:
            print("Invalid withdrawal amount")

    def display_details(self):
        print(f"\nName: {self.name}")
        print(f"Account Number: {self.account_number}")
        print(f"Balance: ${self.balance}")

    def get_valid_float(prompt):
        while True:
            value = input(prompt)
            if value.replace('.', '', 1).isdigit():
                return float(value)
            print("Please enter a valid number.")

name = input("Enter your name: ")
account_number = input("Enter account number: ")
name = input("Enter your name: ")
account_number = input("Enter account number: ")

while True:
    initial_balance = get_valid_float("Enter initial balance: $")
```

```

    if initial_balance >= 0:
        break
    print("Balance cannot be negative. Try again.")

account = BankAccount(name, account_number, initial_balance)

while True:

    print("\n--- Bank Account Menu ---")
    print("1. Deposit")
    print("2. Withdraw")
    print("3. Check Balance")
    print("4. Exit")

    choice = input("Enter your choice (1-4): ")

    if choice == '1':
        amount = get_valid_float("Enter deposit amount: $")
        account.deposit(amount)

    elif choice == '2':
        amount = get_valid_float("Enter withdrawal amount: $")
        account.withdraw(amount)

    elif choice == '3':
        account.display_details()

    elif choice == '4':
        print("Thank you for using our banking system")
        break

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

```

OUTPUT

```

24mca26@softlab-ThinkCentre-M92p: ~/pylab
24mca26@softlab-ThinkCentre-M92p:~/pylab$ python3 c6e1.py
Enter your name: devika
Enter account number: 1234
Enter initial balance: $250

--- Bank Account Menu ---
1. Deposit
2. Withdraw
3. Check Balance
4. Exit
Enter your choice (1-4): 1
Enter deposit amount: $200
Deposited $200.0

--- Bank Account Menu ---
1. Deposit
2. Withdraw
3. Check Balance
4. Exit
Enter your choice (1-4): 2
Enter withdrawal amount: $100
Withdraw $100.0

--- Bank Account Menu ---
1. Deposit
2. Withdraw
3. Check Balance
4. Exit
Enter your choice (1-4): 3
Name: devika
Account Number: 1234
Balance: $350.0

--- Bank Account Menu ---
1. Deposit
2. Withdraw
3. Check Balance
4. Exit

```

EXPERIMENT 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(self):
        print("Publisher Details:")
        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(self):
        super().display()
        print("\nBook Details:")
        print(f"Title: {self.title}")
        print(f"Author: {self.author}")

class PythonBook(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(self):
        super().display()
        print("\nPython Book Details:")
        print(f"Price: ${self.price}")
        print(f"Number of Pages: {self.no_of_pages}")

print("Enter Publisher Details:")
publisher_id = input("Publisher ID: ")
```

```
publisher_name = input("Publisher Name: ")

print("\nEnter Book Details:")
title = input("Book Title: ")
author = input("Book Author: ")

print("\nEnter Python Book Details:")
price = float(input("Price: $"))
no_of_pages = int(input("Number of Pages: "))

python_book = PythonBook(
    publisher_id,
    publisher_name,
    title,
    author,
    price,
    no_of_pages
)

print("\nComplete Book Information:")
python_book.display()
```

OUTPUT

```
Enter Publisher Details:
Publisher ID: 3
Publisher Name: devika

Enter Book Details:
Book Title: things happens for a reason
Book Author: preethi

Enter Python Book Details:
Price: $350
Number of Pages: 120

Complete Book Information:
Publisher Details:
Publisher ID: 3
Publisher Name: devika

Book Details:
Title: things happens for a reason
Author: preethi

Python Book Details:
Price: $350.0
Number of Pages: 120
```

EXPERIMENT 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):
        """Abstract method to get polygon dimensions"""
        pass

    @abstractmethod
    def calculate_area(self):
        """Abstract method to calculate polygon area"""
        pass

class Rectangle(Polygon):
    def get_dimensions(self):
        """Get rectangle dimensions from user input"""
        self.length = float(input("Enter rectangle length: "))
        self.width = float(input("Enter rectangle width: "))

    def calculate_area(self):
        """Calculate and return rectangle area"""
        return self.length * self.width

class Triangle(Polygon):
    def get_dimensions(self):
        """Get triangle dimensions from user input"""
        self.base = float(input("Enter triangle base: "))
        self.height = float(input("Enter triangle height: "))

    def calculate_area(self):
        """Calculate and return triangle area"""
        return 0.5 * self.base * self.height

while True:
    print("\n--- Polygon Area Calculator ---")
    print("1. Rectangle")
    print("2. Triangle")
    print("3. Exit")

    choice = input("Enter your choice (1-3): ")
```

```

if choice == '1':
    rect = Rectangle()
    rect.get_dimensions()
    print(f"Rectangle Area: {rect.calculate_area()}")

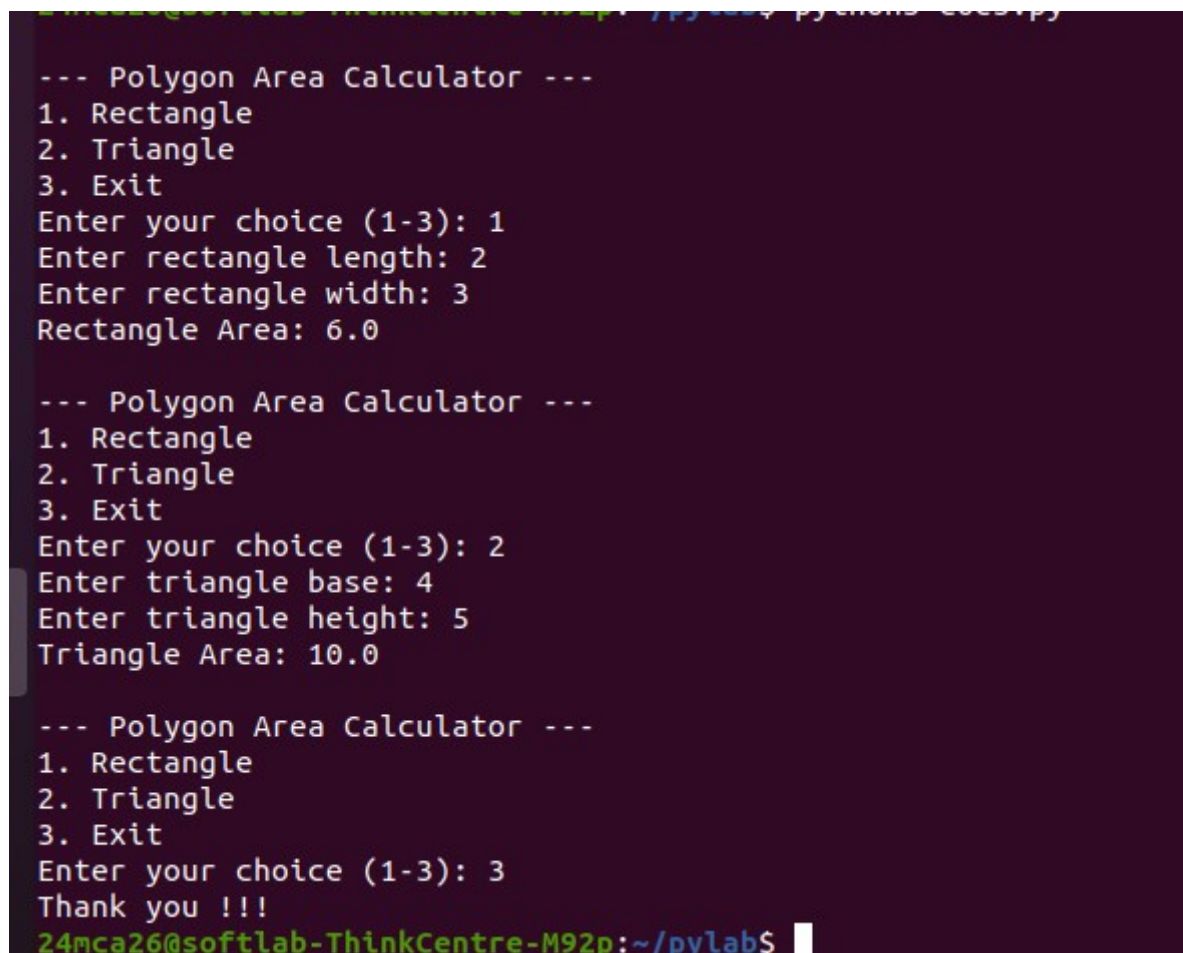
elif choice == '2':
    tri = Triangle()
    tri.get_dimensions()
    print(f"Triangle Area: {tri.calculate_area()}")

elif choice == '3':
    print("Thank you !!!")
    break

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

```

OUTPUT



```

--- Polygon Area Calculator ---
1. Rectangle
2. Triangle
3. Exit
Enter your choice (1-3): 1
Enter rectangle length: 2
Enter rectangle width: 3
Rectangle Area: 6.0

--- Polygon Area Calculator ---
1. Rectangle
2. Triangle
3. Exit
Enter your choice (1-3): 2
Enter triangle base: 4
Enter triangle height: 5
Triangle Area: 10.0

--- Polygon Area Calculator ---
1. Rectangle
2. Triangle
3. Exit
Enter your choice (1-3): 3
Thank you !!!
24mca26@softlab-ThinkCentre-M92p:~/pylab$

```

EXPERIMENT 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 calculate_area(self):
        """Calculate and return the area of the rectangle"""
        return self.length * self.breadth

    def calculate_perimeter(self):
        """Calculate and return the perimeter of the rectangle"""
        return 2 * (self.length + self.breadth)

    def compare_area(self, other):
        """Compare the area of this rectangle with another rectangle"""
        if not isinstance(other, Rectangle):
            raise TypeError("Can only compare with another Rectangle object")

        current_area = self.calculate_area()
        other_area = other.calculate_area()

        if current_area > other_area:
            return f"This rectangle (Area: {current_area}) is larger than the other rectangle (Area: {other_area})"
        elif current_area < other_area:
            return f"This rectangle (Area: {current_area}) is smaller than the other rectangle (Area: {other_area})"
        else:
            return f"Both rectangles have equal area: {current_area}"

print("Enter details for first rectangle:")
length1 = float(input("Enter length: "))
breadth1 = float(input("Enter breadth: "))
rect1 = Rectangle(length1, breadth1)

print("\nEnter details for second rectangle:")
length2 = float(input("Enter length: "))
breadth2 = float(input("Enter breadth: "))
rect2 = Rectangle(length2, breadth2)

print("\nFirst Rectangle:")
print(f"Area: {rect1.calculate_area()}")
print(f"Perimeter: {rect1.calculate_perimeter()}")

print("\nSecond Rectangle:")
print(f"Area: {rect2.calculate_area()}")
print(f"Perimeter: {rect2.calculate_perimeter()}")
```

```
print("\nComparison:")
print(rect1.compare_area(rect2))
```

OUTPUT

```
24mca26@softlab-ThinkCentre-M92p:~/pylab$ python3 c6e4.py
Enter details for first rectangle:
Enter length: 3
Enter breadth: 4

Enter details for second rectangle:
Enter length: 5
Enter breadth: 7

First Rectangle:
Area: 12.0
Perimeter: 14.0

Second Rectangle:
Area: 35.0
Perimeter: 24.0

Comparison:
This rectangle (Area: 12.0) is smaller than the other rectangle (Area: 35.0)
24mca26@softlab-ThinkCentre-M92p:~/pylab$
```

EXPERIMENT 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 = 0
        self.__minute = 0
        self.__second = 0
        self.set_time(hour, minute, second)

    def set_time(self, hour, minute, second):
        if (0 <= hour < 24 and
            0 <= minute < 60 and
            0 <= second < 60):
            self.__hour = hour
            self.__minute = minute
            self.__second = second
        else:
            raise ValueError("Invalid time values")

    def __add__(self, other):
        total_seconds1 = (self.__hour * 3600 +
                          self.__minute * 60 +
```



```

        self.__second)
total_seconds2 = (other.__hour * 3600 +
        other.__minute * 60 +
        other.__second)

total_seconds = total_seconds1 + total_seconds2

new_hour = total_seconds // 3600
remaining_seconds = total_seconds % 3600
new_minute = remaining_seconds // 60
new_second = remaining_seconds % 60

new_hour %= 24
return Time(new_hour, new_minute, new_second)

def display(self):
    print(f'{self.__hour:02d}:{self.__minute:02d}:{self.__second:02d}')

print("Enter first time:")
hour1 = int(input("Hour: "))
minute1 = int(input("Minute: "))
second1 = int(input("Second: "))

time1 = Time(hour1, minute1, second1)
print("First Time: ", end="")
time1.display()

print("\nEnter second time:")
hour2 = int(input("Hour: "))
minute2 = int(input("Minute: "))
second2 = int(input("Second: "))

time2 = Time(hour2, minute2, second2)
print("Second Time: ", end="")
time2.display()

result_time = time1 + time2
print("\nSum of Times: ", end="")
result_time.display()

```

OUTPUT

```

24mca26@softlab-ThinkCentre-M92p:~/pylab$ python3 co
Enter first time:
Hour: 3
Minute: 23
Second: 12
First Time: 03:23:12

Enter second time:
Hour: 2
Minute: 43
Second: 36
Second Time: 02:43:36

Sum of Times: 06:06:48
24mca26@softlab-ThinkCentre-M92p:~/pylab$ █

```