

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 :

```
Enter account details to create a new account:
Enter account holder name: anamika
Enter account number: 12345678
Enter account type (e.g., Savings, Current): savings
Enter initial balance: 1500

Account Holder: anamika
Account Number: 12345678
Account Type: savings
Balance: 1500.0

Choose an operation:
1. Deposit
2. Withdraw
3. Display Account Details
4. Exit
Enter your choice (1/2/3/4): 3
```

```
Account Holder: anamika
Account Number: 12345678
Account Type: savings
Balance: 1500.0

Choose an operation:
1. Deposit
2. Withdraw
3. Display Account Details
4. Exit
Enter your choice (1/2/3/4): 1
Enter deposit amount: 500
Deposited 500.0. New balance is 2000.0.

Choose an operation:
1. Deposit
2. Withdraw
3. Display Account Details
4. Exit
Enter your choice (1/2/3/4): 2
Enter withdrawal amount: 2500
Insufficient balance.

Choose an operation:
1. Deposit
2. Withdraw
3. Display Account Details
4. Exit
Enter your choice (1/2/3/4): 2
Enter withdrawal amount: 2000
Withdrew 2000.0. New balance is 0.0.
```

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

```
24mca12@softlab-ThinkCentre-M92p:~/pylab/cycle_6$ nano exp_3.py
24mca12@softlab-ThinkCentre-M92p:~/pylab/cycle_6$ python3 exp_2.py
Enter Publisher ID: 1119573289
Enter Publisher Name: Wiley
Enter Book Title: The Python Book
Enter Author Name: Rob Mastrodomenico
Enter Book Price: 3860
Enter Number of Pages: 272

Publisher ID: 1119573289
Publisher Name: Wiley
Title: The Python Book
Author: Rob Mastrodomenico
Price: 3860.0
Number of Pages: 272
```

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 :

```
24mca12@softlab-ThinkCentre-M92p:~/pylab/cycle_6$ nano exp_3.py
24mca12@softlab-ThinkCentre-M92p:~/pylab/cycle_6$ python3 exp_3.py
Choose a polygon to calculate area:
1. Rectangle
2. Triangle
Enter your choice (1/2): 1
Enter the length of the rectangle: 4
Enter the width of the rectangle: 5
The area of the rectangle is: 20.0
24mca12@softlab-ThinkCentre-M92p:~/pylab/cycle_6$ nano exp_3.py
24mca12@softlab-ThinkCentre-M92p:~/pylab/cycle_6$ python3 exp_3.py
Choose a polygon to calculate area:
1. Rectangle
2. Triangle
Enter your choice (1/2): 2
Enter the base of the triangle: 4
Enter the height of the triangle: 5
The area of the triangle is: 10.0
```

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: "))
```

```

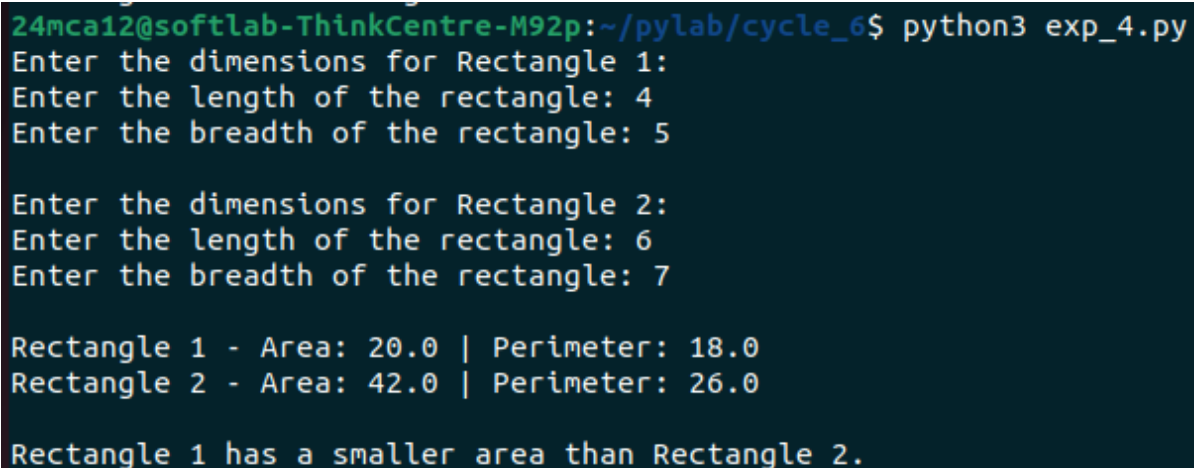
breadth1 = float(input("Enter the breadth of the rectangle: "))
rect1 = Rectangle(length1, breadth1)
print("\nEnter the dimensions for Rectangle 2:")

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 :



```

24mca12@softlab-ThinkCentre-M92p:~/pylab/cycle_6$ python3 exp_4.py
Enter the dimensions for Rectangle 1:
Enter the length of the rectangle: 4
Enter the breadth of the rectangle: 5

Enter the dimensions for Rectangle 2:
Enter the length of the rectangle: 6
Enter the breadth of the rectangle: 7

Rectangle 1 - Area: 20.0 | Perimeter: 18.0
Rectangle 2 - Area: 42.0 | Perimeter: 26.0

Rectangle 1 has a smaller area than Rectangle 2.

```

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 :

```
24mca12@softlab-ThinkCentre-M92p:~/pylab/cycle_6$ python3 exp_5.py
Enter details for Time 1:
Enter hours (0-23): 3
Enter minutes (0-59): 40
Enter seconds (0-59): 34

Enter details for Time 2:
Enter hours (0-23): 5
Enter minutes (0-59): 25
Enter seconds (0-59): 45

Time 1: 03:40:34
Time 2: 05:25:45
Sum of Time 1 and Time 2: 09:06:19
```