LAB CYCLE 5

Experiment No :1

Date: 09/12/2024

Aim:

Write a program to determine whether a given year is leap year(use calendar module).

Pseudocode:

```
1. Import calendar module.
2. Read the year.
3. If calendar.isleap(year) then
        Print year is leap year
   Else
        Print year is not leap year
  End if
```

Method

Function	Description	Syntax
isleap()	Checks the given year is leap year or not	calendar.isleap(year)

Source code:

```
import calendar
year=int(input("Enter the year:"))
if calendar.isleap(year):
        print("Year ",year,"is a leap year")
else:
        print("Year ",year,"is not a leap year")
```

Output:

```
Enter the year : 2024
Year 2024 is a leap year

Enter the year : 1900
Year 1900 is not a leap year
```

Result:

The program is successfully executed and the output is verified.

Experiment No :2

Date: 09/12/2024

Aim:

Write a python script to display

a) current date and time

b) current year

c) month of the year

d) week no:of the year

e) weekday of the week

f) day of year

g) day of month

h) day of week

Pseudocode:

```
1. Import datetime from datetime module
2. Set current date and time as cur=datetime.now()
3. Print current date and time,cur
4. Print Month of the year =cur.strftime("%B")
5. Print Week no: of the year=cur.strftime("%U")
6. Print Weekday of the week=cur.strftime("%A")
7. Print Day of the year:",cur.strftime("%j")
8. Print Day of the month:",cur.day
9. Print Day of the week:",cur.strftime("%A")
```

Method

Function	Description	Syntax
strftime()	Used to convert date and time objects to its string representations	datetime_object. strftime (format)
now()	returns the current local date and time	datetime.now()

Source code:

```
from datetime import datetime
cur=datetime.now()
print("Current date and time:",cur)
print("Current year:",cur.year)
print("Month of the year:",cur.strftime("%B"))
print("Week no: of the year:",cur.strftime("%U"))
print("Weekday of the week:",cur.strftime("%A"))
print("Day of the year:",cur.strftime("%j"))
print("Day of the month:",cur.day)
print("Day of the week:",cur.strftime("%A"))
```

Output:

```
Current date and time: 2024-12-09 15:01:53.581771
Current year: 2024
Month of the year: December
Week no: of the year: 49
Weekday of the week: Monday
Day of the year: 344
Day of the month: 9
Day of the week: Monday
```

Result:

The program is successfully executed and the output is verified.

Experiment No :3

Date: 09/12/2024

Aim:

Write a python program to print yesterday,today,tomorrow.

Pseudocode:

```
1. Import datetime and timedelta from datetime module
2. Set today=datetime.now()
3. Set yesterday=today-timedelta(days=1)
4. Set tomorrow=today+timedelta(days=1)
5. Print yesterday
6. Print today
7. Print tomorrow
```

Source code:

```
from datetime import datetime,timedelta
today=datetime.now()
yesterday=today-timedelta(days=1)
tomorrow=today+timedelta(days=1)
print("Yesterday:",yesterday.strftime('%Y-%m-%d'))
print("Today:",today.strftime('%Y-%m-%d'))
print("Tomorrow:",tomorrow.strftime('%Y-%m-%d'))
```

Output:

```
Yesterday: 2024-12-08
Today: 2024-17-09
Tomorrow: 2024-17-10
```

Result:

The program is successfully executed and the output is verified.

Experiment No :4

Date: 09/12/2024

Aim:

Write a function in file palindrome.py to check whether a string is Palindrome or not. Import the module to find the longest palindromic substring in a given string by checking every possible substring and verifying if it is a palindrome.

Pseudocode:

```
1. Define function palindrome(s) in palindrome.py file
2. Import palindrome function from palindrome.py file
3. Define function long_palindrome(s)
4. Read the string
5. Call the function long_palindrome with str1 as argument
6. Store the result in result variable
7.  Print Longest palindrome substring, result
```

```
palindrome(s)
1.   Return s==s[::-1]
```

```
long_palindrome(s)
1.   Initialize longest as an empty string
2.   For each character index i from 0 to length of string s - 1
         For each character index "j" from i+1 to length of string s
             Extract the substring from index i to j (inclusive)
                 If the substring is a palindrome and its length
                    is greater than the length of longest
                        Set longest to this substring
                 End if
         End for
         Return the longest palindrome substring
```

```
    End for
```

Source code:

```
from fpalindrome import palindrome
def long_palindrome(s):
        longest=""
        for i in range(len(s)):
                for j in range(i+1,len(s)+1):
                        substring=s[i:j]
                        if palindrome(substring) and len(substring)>len(longest):
                                longest=substring
        return longest
str1=input("Enter string:")
result=long_palindrome(str1)
print("Longest palindrome substring:",result)
```

Output:

```
Enter string: amma is a malayalam word
Longest palindrome substring: malayalam

Enter string: 232 abc aa
Longest palindrome substring: 232
```

Result:

The program is successfully executed and the output is verified.

Experiment No :5

Date: 09/12/2024

Aim:

Create a package graphics with modules rectangle, circle and sub-package 3D-graphics with modules cuboid and sphere. Include methods to find area and perimeter of respective figures in each module. Write programs that find the area and perimeter of figures by different importing statements. (Include selective import of modules and import * statements).

Pseudocode:

```
1. Create package graphics
2. Create rectangle module
        Define function area(l,b)
        Define function perimeter(l,b)
3. Create circle module
        Import math module
        Define function area(r)
        Define function circumference(r)
4. Create sub-package 3D-graphics
5. Create cuboid module
        Define function area(l,w,h)
        Define function perimeter(l,w,h)
6. Create sphere module
        Import math module
        Define function area(r)
        Define function volume(r)
7. Create main program file
8. Import modules and sub-package modules from graphics package
9. Read length and breadth of rectangle
```

```
10. Print area and perimeter rectangle
11. Read radius of circle
12. Print area and circumference of circle
13. Read length,breadth and height of cuboid
14. Print area and perimeter of cuboid
15. Read radius of sphere
16.Print area and volume of sphere

area(l,b)
1.   Return l*b

perimeter(l,b)
1.   Return 2*(l+b)

area(r)
1.   Return math.pi*r*r

circumference(r)
1.   Return  2*math.pi*r

area(l,w,h)
1.   Return 2*(l*w+w*h+h*l)

perimeter(l,w,h)
1.   Return 4*(l+w+h)

area(r)
1.   Return  4*math.pi*r*r

volume(r)
1.   Return (4/3)*math.pi*(r**3)
```

Source code:

```
import graphics.rectangle as rect
import graphics.circle as cir
import graphics.threeD_graphics.cuboid as cub
import graphics.threeD_graphics.sphere as spr
length=int(input("Enter length for rectangle:"))
breadth=int(input("Enter breadth for rectangle:"))
print("Rectangle area:",rect.area(length,breadth))
print("Rectangle perimeter:",rect.perimeter(length,breadth))
r1=int(input("Enter radius for circle:"))
print("Circle area:",cir.area(r1))
print("Circle perimeter:",cir.perimeter(r1))
l=int(input("Enter length for cuboid:"))
w=int(input("Enter width for cuboid:"))
h=int(input("Enter height for cuboid:"))
print("Cuboid area:",cub.area(l,w,h))
print("Cuboid perimeter:",cub.perimeter(l,w,h))
r2=int(input("Enter radius for sphere:"))
print("Sphere area:",spr.area(r2))
print("Sphere volume:",spr.volume(r2))
```

```
//circle.py
From math import *
def area(r):
        return pi*r*r
def perimeter(r):
        return 2*math.pi*r
```

```
//rectangle.py
def area(length,breadth):
        return length*breadth
def perimeter(length,breadth):
        return 2*(length*breadth)
```

```
//cuboid.py
def area(l,w,h):
        return 2*(l*w+w*h+h*l)
def perimeter(l,w,h):
        return 4*(l+w+h)
```

```
//sphere.py
import math
def area(r):
        return 4*math.pi*r*r
def volume(r):
        return 4/3*math.pi*r*r*r
```

Output:

```
Enter the length of the rectangle:2
Enter the breadth of the rectangle:3
Area of rectangle=6
Perimeter of rectangle=10
Enter the radius of the circle:4
Area of circle=50.26548245743669
Circumference of the circle=25.132741228718345
Enter the length of the cuboid:2
Enter the breadth of the cuboid:3
Enter the height of the cuboid:4
Area of cuboid=52
Perimeter of cuboid=36
Enter the radius of the sphere:6
Area of sphere=452.3893421169302
Volume of sphere=904.7786842338603
```

Result:

The program is successfully executed and the output is verified.

LAB CYCLE 6

Experiment No :1

Date: 19/12/2024

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.

Pseudocode:

```
1.  Create class Bankaccount
2.  Define function __init__(name, account_number, account_type, balance=0)
3.  self.balance = balance
4.  Define function withdraw(self,account)
5.  Define function deposit(amount)
6.  Define function display()
7.  While True
        Print operations
        Read user choice
        If choice=1 then
            Get name,account number,account type,initial balance from user
            Create new bank account object with given details
            Print account created successfully
        Else If choice=2 then
            If account exists (account != None)
                Get deposit_amount from user
                Call account.deposit(deposit_amount)
            Else
                Print  Create an account first.
        Else If choice =3 then
            If account exists (account != None)
                Get withdraw_amount from user
                Call account.withdraw(withdraw_amount)
            Else
                Print Create an account first
        Else If choice = 4 then
            If account exists (account != None)
                Call account.display()
            Else
                Print Create an account first.
        Else If choice == 5:
            Print Exiting
            Break
        Else
            Print Invalid choice
        End if

__init__(name, account_number, account_type, balance=0)
1.  Set self.name = name
2.  Set self.account_number = account_number
3.  Set self.account_type = account_type
4.  Set self.balance = balance

withdraw(self,account)
1.  If amount>0 then
            If amount<=self.balance
                Set self.balance-=amount
```

```
                Print Amount withdrawn successfully. New balance:  +
                                              self.balance
            Else
                Print Insufficient balance.
            End if
        Else
            Print Withdraw amount must be positive.
        End if
```

```
deposit(amount)
1.  If amount > 0:
            self.balance += amount
            Print Amount deposited successfully. New balance: " + self.balance
    Else
        Print Deposited amount must be positive.
    End if
```

```
display()
1.  Print Account details
```

Source code:

```
class Bankaccount:
    def __init__(self,name,account_number,account_type,balance=0):
        self.name=name
        self.account_number=account_number
        self.account_type=account_type
        self.balance=balance
    def withdraw(self,account):
        if amount>0:
            if amount<=self.balance:
                self.balance-=amount
                print(f"{amount} withdraw successfully. new
                    balance:{self.balance}")
            else:
                print("Insufficient balance")
        else:
            print("Withdraw amount musy be positive")
    def deposit(self,account):
        if amount>0:
            self.balance+=amount
            print(f"{amount} deposited succesfully. new
                balance:{self.balance}")
        else:
            print("Deposited amount must be positive")
    def display(self):
        print("\nAccount details..")
        print(f"Name:{self.name}")
        print(f"Account_number:{self.account_number}")
        print(f"Account_type:{self.account_type}")
        print(f"Aalance:{self.balance}")
account=None
while True:
    print("\n1.Create a new account")
    print("2.Deposit money")
    print("3.Withdraw money")
    print("4.Display")
    print("5.Exit")
    choice=int(input("\nEnter your choice:"))
    if choice==1:
        name=input("\nEnter the account holder:")
        account_number=int(input("Enter the account number:"))
        account_type=input("Enter the account type(savings/current):")
        initial_balance=float(input("Enter the initial balance:"))
    account=Bankaccount(name,account_number,account_type,initial_balance)
        print("Account created successfully")
    elif choice==2:
        if account:
            amount=float(input("\nEnter the amount to depoist:"))
            account.deposit(amount)
        else:
            print("Create an account first")
    elif choice==3:
        if account:
            amount=float(input("\nEnter the amount to withdraw:"))
            account.withdraw(amount)
        else:
            print("Create an account first")
    elif choice==4:
        if account:
            account.display()
        else:
            print("Please create a account")
    elif choice==5:
        print("Exiting")
        break
    Else:
        print("Invalid choice")
```

Output:

```
1.Create a new account
2.Deposit money
3.Withdraw money
4.Display
```

```
5.Exit
Enter your choice:1
Enter the account holder:Nandana
Enter the account number:4567899976
Enter the account type(savings/current):savings
Enter the initial balance:3000
Account created successfully
1.Create a new account
2.Deposit money
3.Withdraw money
4.Display
5.Exit
Enter your choice:4
Account details..
Name:Nandana
Account_number:4567899976
Account_type:savings
Aalance:3000.0
1.Create a new account
2.Deposit money
3.Withdraw money
4.Display
5.Exit
Enter your choice:2
Enter the amount to depoist:5000
5000.0 deposited succesfully. New balance:8000.0
1.Create a new account
2.Deposit money
3.Withdraw money
4.Display
```

```
5.Exit
Enter your choice:3
Enter the amount to withdraw:2000
2000.0 withdraw successfully. New balance:6000.0
1.Create a new account
2.Deposit money
3.Withdraw money
4.Display
5.Exit
Enter your choice:5
Exiting
```

Result:

The program is successfully executed and the output is verified.

Experiment No :2

Date: 19/12/2024

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.

Pseudocode:

```
1. Create class Publisher
2. Define function  __init__(publisher_id, publisher_name)
3. Define function display()
4. Create class Book (inherits from Publisher)
        Define functions  __init__(publisher_id, publisher_name, title, author)
        Define function display()
6. Create class Python (inherits from Book)
        Define function __init__(publisher_id, publisher_name, title, author, price,
                        no_of_pages)
        Define function display()
9. Read details of the book from user
10. Create python_book as an instance of Python class with the provided inputs
11. Print "Python Book Information:"
12. Call python_book.display()
```

```
__init__(publisher_id, publisher_name)
1.  Set self.publisher_id = publisher_id
2.  Set self.publisher_name = publisher_name
```

```
display()
1.  Print “Publisher ID: " + self.publisher_id
2.  Print "Publisher Name: " + self.publisher_name

__init__(publisher_id, publisher_name, title, author)
1.  Call  Publisher.__init__(publisher_id, publisher_name)
2.  Set self.title = title
3.  Set self.author = author

display()
1.  Call Publisher.display()
2.  Print "Book Title: " + self.title
3.  Print "Author: " + self.author

__init__(publisher_id, publisher_name, title, author, price,  no_of_pages)
1.  Call Book.__init__(publisher_id, publisher_name, title, author)
2.  Set self.price = price
3.  Set self.no_of_pages = no_of_pages

display()
1.  Call Book.display()
2.  Print "Book Price: " + self.price
3.  Print "Number of Pages: " + self.no_of_pages
```

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(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(f"Book 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(self):
        super().display()
        print(f"Book Price: {self.price}")
        print(f"Number of Pages: {self.no_of_pages}")
print("Enter the details of book:")
publisher_id = int(input("Enter publisher ID: "))
publisher_name = input("Enter publisher name: ")
title = input("Enter the title of the book: ")
author = input("Enter the author name: ")
price = int(input("Enter the price: "))
no_of_pages = int(input("Enter the number of pages: "))
python_book = Python(
    publisher_id=publisher_id,
    publisher_name=publisher_name,
    title=title,
    author=author,
    price=price,
    no_of_pages=no_of_pages
)
print("\nPython Book Information:")
python_book.display()
```

Output:

```
Enter the details of book:
Enter publisher ID: 101
Enter publisher name: Bloomsbury
Enter the title of the book: Harry Potter
Enter the author name: J K Rowling
Enter the price: 700
Enter the number of pages: 2000
Python Book Information:
Publisher ID: 101
Publisher Name: Bloomsbury
Book Title: Harry Potter
Author: J K Rowling
Book Price: 700
Number of Pages: 2000
```

Result:

The program is successfully executed and the output is verified.

Experiment No :3

Date: 19/12/2024

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.

Pseudocode:

```
1. Create abstract class Polygon
2. Define function get_dimensions(self)
3. Define function calculate_area()
4. Create class Rectangle inherits from class Polygon
5. Create function   __init__()
6. Define function get_dimensions()
7. Define function calculate_area()
8. Create class  Triangle inherits from class Polygon
9. Define function __init__()
10. Define function get_dimensions()
11. Define function calculate_area()
12. Define main function
13. Call get_dimensions() on the chosen polygon
14. Call calculate_area() on the chosen polygon
15. Print area of the selected polygon
```

```
get_dimensions(self)
1.   pass
```

```
calculate_area()
1.   pass
```

```
__init__()
1.  Set length = 0
2.  Set breadth = 0

get_dimensions()
1.  Read breadth and length from user

calculate_area()
1.  Return length * breadth

__init__()
1.  Set base = 0
2.  Set height = 0

get_dimensions()
1.  Read height and base from user

calculate_area()
1.  Return 0.5 * base * height

main()
1.  Print the options to choose polygon
2.  Read the choice
3.  If choice = 1 then
        Create Rectangle object
    Else If choice = 2 then
        Create Triangle object
    Else
        Print invalid choice
        Exit the program
```

```
     End if
```

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.breadth = 0
        def get_dimensions(self):
                self.length = float(input("Enter the length of the rectangle: "))
                self.breadth = float(input("Enter the breadth of the rectangle: "))
        def calculate_area(self):
                return self.length * self.breadth
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):
                return 0.5 * self.base * self.height
def main():
        print("Choose a polygon:")
        print("1. Rectangle")
        print("2. Triangle")
        choice = int(input("Enter your choice (1 or 2): "))
        if choice == 1:
                polygon = Rectangle()
        elif choice == 2:
                polygon = Triangle()
        else:
                print("Invalid choice!")
                return
        polygon.get_dimensions()
        area = polygon.calculate_area()
        print(f"The area of the selected polygon is: {area}")
if __name__ == "__main__":
  main()
```

Output:

```
Choose a polygon:
1. Rectangle
2. Triangle
Enter your choice (1 or 2): 1
Enter the length of the rectangle: 9
Enter the breadth of the rectangle: 3
The area of the selected polygon is: 27.0
Choose a polygon:
1. Rectangle
2. Triangle
Enter your choice (1 or 2): 2
Enter the base of the triangle: 4
Enter the height of the triangle: 6
```

```
The area of the selected polygon is: 12.0
```

Result:

The program is successfully executed and the output is verified.

Experiment No :4

Date: 19/12/2024

Aim:

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

Pseudocode:

```
1. Create  class Rectangle
2. Define function __init__(self,length, breadth)
3. Define function perimeter(self)
4. Define function area(self)
5. Define function __lt__(self,other)
6. Define function __eq__(other)
7. Define function __gt__(other)
8. Define function main()

__init__(self,length, breadth)
1. Set self.length = length
2. Set self.breadth = breadth

perimeter(self)
1. Return self.length * self.breadth

area(self)
1. Return 2 * (self.length + self.breadth)

__lt__(self,other)
1. Return self.area() < other.area()
```

```
__eq__(other)
1.  Return self.area() == other.area()

__gt__(other)
1.  Return self.area() > other.area()

main()
1.  Read lenghth and breadth of first rectangle as length1, breadth1
2.  Create rect1 as an instance of Rectangle with length1, breadth1
3.  Read lenghth and breadth of second rectangle as length2, breadth2
4.  Create rect1 as an instance of Rectangle with length2, breadth2
5.  Print "Rectangle 1 - Area: ", rect1.area(), " Perimeter: ", rect1.perimeter()
6.  Print "Rectangle 2 - Area: ", rect2.area(), " Perimeter: ", rect2.perimeter()
7.  If rect1 > rect2
            Print  The first rectangle has a larger area
    Else If rect1 < rect2
            Print The second rectangle has a larger area
    Else
            Print Both rectangles have the same area
    End if
```

Source code:

```
class Rectangle:
        def __init__(self, length, breadth):
                self.length = length
                self.breadth = breadth
        def area(self):
                return self.length * self.breadth
        def perimeter(self):
                return 2 * (self.length + self.breadth)
        def __lt__(self, other):
                """Compare if this rectangle's area is less than another rectangle's
                        area."""
                return self.area() < other.area()
        def __eq__(self, other):
                """Check if this rectangle's area is equal to another rectangle's
                        area."""
                return self.area() == other.area()
        def __gt__(self, other):
                """Compare if this rectangle's area is greater than another
                        rectangle's area."""
                return self.area() > other.area()
def main():
        length1 = float(input("Enter the length of the first rectangle: "))
        breadth1 = float(input("Enter the breadth of the first rectangle: "))
        rect1 = Rectangle(length1, breadth1)
        length2 = float(input("Enter the length of the second rectangle: "))
        breadth2 = float(input("Enter the breadth of the second rectangle: "))
        rect2 = Rectangle(length2, breadth2)
        print(f"Rectangle 1 - Area: {rect1.area()}, Perimeter: {rect1.perimeter()}")
        print(f"Rectangle 2 - Area: {rect2.area()}, Perimeter: {rect2.perimeter()}")
        if rect1 > rect2:
                print("The first rectangle has a larger area.")
        elif rect1 < rect2:
                print("The second rectangle has a larger area.")
        else:
                print("Both rectangles have the same area.")
if __name__ == "__main__":
    main()
```

Output:

```
Enter the length of the first rectangle: 4
Enter the breadth of the first rectangle: 6
Enter the length of the second rectangle: 2
Enter the breadth of the second rectangle: 7
Rectangle 1 - Area: 24.0, Perimeter: 20.0
Rectangle 2 - Area: 14.0, Perimeter: 18.0
The first rectangle has a larger area.
```

Result:

The program is successfully executed and the output is verified.

Experiment No :5

Date: 19/12/2024

Aim:

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

Pseudocode:

```
1. Create class Time
2. Define function __init__(self, hour, minute, second)
3. Define function main()
4. Define function display(self)
5. Define function __add__(self,other)

__init__(self, hour, minute, second)
1. Set self.__hour = hour
2. Set self.__minute = minute
3. Set self.__second = second

main()
1. Set  total_seconds = self.__second + other.__second
2. Set  total_minutes = self.__minute + other.__minute + (total_seconds // 60)
3. Set  total_hours = self.__hour + other.__hour + (total_minutes // 60)
4. Set seconds = total_seconds % 60
5. Set  minutes = total_minutes % 60
6. Set hours = total_hours % 24
7. Return  a new Time object with hours, minutes, seconds

display(self)
1. Print self.__hour, self.__minute, self.__second in HH:MM:SS format
```

__add__(self,other)

1. Read hour, minute, second from user for first time
2. Create time1 as an instance of Time with hour1, minute1, second1
3. Read hour, minute, second from user for second time
4. Create time2 as an instance of Time with hour2, minute2, second2
5. Calculate total_time = time1 + time2
6. Print sum of time
7. Call total_time.display()

Source code:

```
class Time:
    def __init__(self, hour, minute, second):
        self.__hour = hour
        self.__minute = minute
        self.__second = second
    def __add__(self, other):
        total_seconds = self.__second + other.__second
        total_minutes = self.__minute + other.__minute +
            (total_seconds // 60)
        total_hours = self.__hour + other.__hour + (total_minutes // 60)
        seconds = total_seconds % 60
        minutes = total_minutes % 60
        hours = total_hours % 24
        return Time(hours, minutes, seconds)
    def display(self):
        print(f"{self.__hour:02}:{self.__minute:02}:{self.__second:02}")
def main():
    print("Enter the first time:")
    hour1 = int(input("Hour: "))
    minute1 = int(input("Minute: "))
```

```
        second1 = int(input("Second: "))
        time1 = Time(hour1, minute1, second1)
        print("Enter the second time:")
        hour2 = int(input("Hour: "))
        minute2 = int(input("Minute: "))
        second2 = int(input("Second: "))
        time2 = Time(hour2, minute2, second2)
        total_time = time1 + time2
        print("The sum of the two times is:")
        total_time.display()
if __name__ == "__main__":
    main()
```

Output:

```
Enter the first time:
Hour: 3
Minute: 30
Second: 25
Enter the second time:
Hour: 3
Minute: 10
Second: 5
The sum of the two times is:
06:40:30
```

Result:

The program is successfully executed and the output is verified.