lab_13

March 11, 2025

Python Programming - 2301CS404

Lab - 13

OM BHUT | 23010101033 | 122

1 OOP

1.0.1 01) Write a Program to create a class by name Students, and initialize attributes like name, age, and grade while creating an object.

[11]:
```
class Student:
    def __init__(self, name, age, grade):
        self.name = name
        self.age = age
        self.grade = grade

    def display_info(self):
        print("Name:",self.name)
        print("Age:",self.age)
        print("Grade:",self.grade)

student1 = Student("Yash", 20, "O")
student1.display_info()
```

```
Name: Yash
Age: 20
Grade: O
```

1.0.2 02) Create a class named Bank_Account with Account_No, User_Name, Email,Account_Type and Account_Balance data members. Also create a method GetAccountDetails() and DisplayAccountDetails(). Create main method to demonstrate the Bank_Account class.

[12]:
```
class BankAccount:
    def __init__(self, Account_No, User_Name, Email, Account_Type,␣
 ↪Account_Balance):
        self.Account_No = Account_No
        self.User_Name = User_Name
        self.Email = Email
        self.Account_Type = Account_Type
        self.Account_Balance = Account_Balance

    def GetAccountDetails(self):
        return self.Account_No, self.User_Name, self.Email, self.Account_Type,␣
 ↪self.Account_Balance

    def DisplayAccountDetails(self):
        print("Account Number:", self.Account_No)
        print("User Name:", self.User_Name)
        print("Email:", self.Email)
        print("Account Type:", self.Account_Type)
        print("Account Balance:", self.Account_Balance)

account1 = BankAccount(1234567890, "Yash", "yash77@gmail.com", "Savings",␣
 ↪70000000)

account_details = account1.GetAccountDetails()

account1.DisplayAccountDetails()
```

```
Account Number: 1234567890
User Name: Yash
Email: yash77@gmail.com
Account Type: Savings
Account Balance: 70000000
```

1.0.3 03) WAP to create Circle class with area and perimeter function to find area and perimeter of circle.

[13]:
```
import math
class Circle:
    def __init__(self, radius):
        self.radius = radius

    def area(self):
        return math.pi* (self.radius ** 2)

    def perimeter(self):
        return 2 * math.pi * self.radius

circle1 = Circle(10)

print("Area:", circle1.area())
print("Perimeter:", circle1.perimeter())
```

```
Area: 314.1592653589793
```

```
Perimeter: 62.83185307179586
```

1.0.4 04) Create a class for employees that includes attributes such as name, age, salary, and methods to update and display employee information.

[14]:
```python
class Employee:
    def __init__(self, name, age, salary):
        self.name = name
        self.age = age
        self.salary = salary

    def update_info(self, name=None, age=None, salary=None):
        if name:
            self.name = name
        if age:
            self.age = age
        if salary:
            self.salary = salary

    def display_info(self):
        print("Name:", self.name)
        print("Age:", self.age)
        print("Salary:", self.salary)

employee1 = Employee("Yash Kakadiya", 20, 70000)

employee1.display_info()

employee1.update_info(salary=100000)

employee1.display_info()

employee1.update_info(age=21)

employee1.display_info()
```

```
Name: Yash Kakadiya
Age: 20
Salary: 70000
Name: Yash Kakadiya
Age: 20
Salary: 100000
Name: Yash Kakadiya
Age: 21
Salary: 100000
```

1.0.5 05) Create a bank account class with methods to deposit, withdraw, and check balance.

[15]:
```python
class BankAccount:
    def __init__(self, account_number, initial_balance):
        self.account_number = account_number
        self.balance = initial_balance

    def deposit(self, amount):
        self.balance += amount
        print("Deposited:", amount)
        self.check_balance()

    def withdraw(self, amount):
        if amount <= self.balance:
            self.balance -= amount
            print("Withdrawn:", amount)
            self.check_balance()
        else:
            print("Insufficient balance.")

    def check_balance(self):
        print("Current Balance:", self.balance)

account1 = BankAccount(1234567890, 10000)

account1.deposit(5000)

account1.withdraw(2000)

account1.withdraw(3000)

account1.check_balance()
```

```
Deposited: 5000
Current Balance: 15000
Withdrawn: 2000
Current Balance: 13000
Withdrawn: 3000
Current Balance: 10000
Current Balance: 10000
```

1.0.6 06) Create a class for managing inventory that includes attributes such as item name, price, quantity, and methods to add, remove, and update items.

[16]:
```python
class Product:
    def __init__(self, item_name, price, quantity):
        self.item_name = item_name
        self.price = price
        self.quantity = quantity

    def add_item(self, quantity):
        self.quantity += quantity
        print("Item added:", self.item_name)
        self.display_info()

    def remove_item(self, quantity):
        if quantity <= self.quantity:
            self.quantity -= quantity
            print("Item removed:", self.item_name)
            self.display_info()
        else:
            print("Not enough items in stock.")

    def update_price(self, new_price):
        self.price = new_price
        print("Price updated:", self.item_name)
        self.display_info()

    def display_info(self):
        print("Item Name:", self.item_name)
        print("Price:", self.price)
        print("Quantity:", self.quantity)

product1 = Product("Laptop", 10000, 5)

product1.add_item(3)

product1.remove_item(2)

product1.update_price(12000)

product1.display_info()
```

```
Item added: Laptop
Item Name: Laptop
Price: 10000
Quantity: 8
Item removed: Laptop
Item Name: Laptop
```

```
Price: 10000
Quantity: 6
Price updated: Laptop
Item Name: Laptop
Price: 12000
Quantity: 6
Item Name: Laptop
Price: 12000
Quantity: 6
```

1.0.7 07) Create a Class with instance attributes of your choice.

[22]:
```
class MyClass:
    def __init__(self, attribute1, attribute2):
        self.attribute1 = attribute1
        self.attribute2 = attribute2

    def display_attributes(self):
        print(f'Attribute 1: {self.attribute1}')
        print(f'Attribute 2: {self.attribute2}')

my_object = MyClass('Hello', 'World')

my_object.display_attributes()
```

```
Attribute 1: Hello
Attribute 2: World
```

1.0.8 08) Create one class student__kit

Within the student__kit class create one class attribute principal name (Mr ABC)

Create one attendance method and take input as number of days.

While creating student take input their name .

Create one certificate for each student by taking input of number of days present in class.

[21]:
```
class StudentKit:
    PrincipalName='Mr.ABC'
    def __init__(self,name):
        self.name=name
        self.attendance=0
        self.certificate=0

    def attendance_method(self,days):
        self.attendance=days
        print(f'{self.name} has attended {self.attendance} days in class.')
    def certificate_method(self,days):
        self.certificate=days
        print(f'{self.name} has obtained {self.certificate} days of certificate.
 ↪')

    def display_info(self):
        print(f'Student Name: {self.name}\nPrincipal Name: {self.
 ↪PrincipalName}\nAttendance: {self.attendance} days\nCertificate: {self.
 ↪certificate} days')

student1=StudentKit('Yash Kakaiya')

student1.attendance_method(25)

student1.certificate_method(30)

student1.display_info()
```

```
Yash Kakaiya has attended 25 days in class.
Yash Kakaiya has obtained 30 days of certificate.
Student Name: Yash Kakaiya
Principal Name: Mr.ABC
Attendance: 25 days
Certificate: 30 days
```

1.0.9 09) Define Time class with hour and minute as data member. Also define addition method to add two time objects.

[20]:
```
class Time:
    def __init__(self, hour, minute):
        self.hour = hour
        self.minute = minute

    def add_time(self, other_time):
        new_minute = self.minute + other_time.minute
        new_hour = self.hour + other_time.hour + new_minute // 60
        new_minute = new_minute % 60
        return Time(new_hour, new_minute)

    def display_time(self):
        print(f"{self.hour:02d}:{self.minute:02d}")

time1 = Time(10, 30)

time2 = Time(5, 45)
```

```python
sum_time = time1.add_time(time2)

sum_time.display_time()
```

```
16:15
```