python-programming-lab-13-1

March 13, 2025

Python Programming - 2301CS404

Gohel Neel

Enrollnment No. : 23010101089

Roll No. 340

Date: 03-03-2025

Lab - 13_1

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.

[5]:
```
class Student:
    school = "DIET"
    def __init__(self , name , age , grade):
        self.name = name
        self.age = age
        self.grade = grade
    def printStudent(self):
        print(self.name)
        print(self.age)
        print(self.grade)
stu1 = Student("Neel" , "20" , "A")
stu1.printStudent()
print(Student.school)
```

```
Neel
20
A
DIET
```

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.

[18]:
```
class Bank_Account:

    def getAccountDetail(self):
        self.account_No = input("Enter account_No ")
        self.user_name = input("Enter user_name ")
        self.email = input("Enter email ")
        self.account_type = input("Enter account_type ")
        self.account_balance = input("Enter account_balance ")

    def displayAccountDetail(self):
        print("Your account_No is" , self.account_No)
        print("Your user_name ",self.user_name)
        print("Your email ",self.email)
        print("Your account_type ",self.account_type)
        print("Your account_balance ",self.account_balance)
B1 = Bank_Account()
B1.getAccountDetail()
B1.displayAccountDetail()
```

```
Enter account_No abc
Enter user_name abc
Enter email abc
Enter account_type abc
Enter account_balance abc
Your account_No is abc
Your user_name  abc
Your email  abc
Your account_type  abc
Your account_balance  abc
```

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

[26]:
```
import math

class Circle:

    def __init__(self):
        self.r = int(input("Enter Radius "))
    def area(self):
        area = math.pi * self.r * self.r
        print("Area is " ,area)
    def perimeter(self):
```

```
        peri = 2 * math.pi * self.r
        print("Perimeter is " ,peri)
c1 = Circle()
c1.area()
c1.perimeter()
```

```
Enter Radius 5
Area is  78.53981633974483
Perimeter is  31.41592653589793
```

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.

[27]:
```
class Employee:

    def __init__(self):
        self.name = input("Enter name ")
        self.age = int(input("Enter Age "))
        self.salary = int(input("Enter Salary "))
    def updateEmployee(self):
        n1 = input("Enter Updated Name ")
        a1 = int(input("Enter updated age "))
        s1 = int(input("Enter updated Salary "))
        self.name = n1
        self.age = a1
        self.salary = s1
    def display(self):
        print("Employee name is ",self.name)
        print("Employee age is " ,self.age)
        print("Employee salary is ",self.salary)
e1 = Employee()
e1.display()
e1.updateEmployee()
e1.display()
```

```
Enter name neel
Enter Age 20
Enter Salary 12345
Employee name is  neel
Employee age is  20
Employee salary is  12345
Enter Updated Name neeeeel
Enter updated age 21
Enter updated Salary 9876
Employee name is  neeeeel
Employee age is  21
Employee salary is  9876
```

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

[29]:
```
class bank_account():

    def __init__(self):
        self.user_name = input("Enter user_name ")
        self.account_balance = int(input("Enter account_balance "))

    def deposit(self):
        money = int(input("Enter money to deposit "))
        self.account_balance+=money
        print("Your account has ",self.account_balance)

    def withdraw(self):
        money = int(input("Enter money to withdraw "))
        if(money >self.account_balance ):
            print("Invalid Money")
        else:
            self.account_balance-=money
        print("Your account has ",self.account_balance)


    def checkBalance(self):
        print("Your Money in bank is " ,self.account_balance)

user1 = bank_account()
user1.deposit()
user1.withdraw()
user1.checkBalance()
```

```
Enter user_name neel
Enter account_balance 500
Enter money to deposit 100
Your account has  600
Enter money to withdraw 200
Your account has  400
Your Money in bank is  400
```

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.

[ ]:
```
class managing_inventory():

    def __init__(self):
        self.name = input("Enter name ")
        self.price = int(input("Enter price "))
        self.quantity = int(input("Enter quantity "))
```

```
    def add(self):
        add = int(input("Enter no of add items "))
        self.quantity += add
        print("Updated " ,self.quantity )
```

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

[3]:
```
class Car:
    def __init__(self, make, model, year, color):
        self.make = make
        self.model = model
        self.year = year
        self.color = color

    def display_car_info(self):
        print(f"Car Information:")
        print(f"Make: {self.make}")
        print(f"Model: {self.model}")
        print(f"Year: {self.year}")
        print(f"Color: {self.color}")

car1 = Car("Toyota", "Century", 2021, "Blue")
car1.display_car_info()
car2 = Car("Honda", "Civic", 2020, "Red")
car2.display_car_info()
```

```
Car Information:
Make: Toyota
Model: Century
Year: 2021
Color: Blue
Car Information:
Make: Honda
Model: Civic
Year: 2020
Color: Red
```

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.

[1]:
```
class StudentKit:
    principal = "Mr ABC"

    def __init__(self, student_name):
        self.student_name = student_name
        self.attendance = 0

    def record_attendance(self, days_present):
        self.attendance = days_present
        print(f"Attendance recorded: {self.attendance} days")

    def generate_certificate(self):
        if self.attendance >= 75:
            print(f"Certificate of Attendance\n")
            print(f"Principal: {StudentKit.principal}")
            print(f"Student: {self.student_name}")
            print(f"Days Present: {self.attendance}")
            print(f"Status: Passed (Attendance is sufficient)")
        else:
            print(f"Certificate of Attendance\n")
            print(f"Principal: {StudentKit.principal}")
            print(f"Student: {self.student_name}")
            print(f"Days Present: {self.attendance}")
            print(f"Status: Failed (Attendance is insufficient)")

student_name = input("Enter the student's name: ")
student = StudentKit(student_name)
days_present = int(input("Enter the number of days present in class: "))
student.record_attendance(days_present)
student.generate_certificate()
```

```
Enter the student's name:  neel
Enter the number of days present in class:  28

Attendance recorded: 28 days
Certificate of Attendance

Principal: Mr ABC
Student: neel
Days Present: 28
Status: Failed (Attendance is insufficient)
```

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

[2]:
```
class Time:
    def __init__(self, hour=0, minute=0):
        self.hour = hour
        self.minute = minute

    def add_time(self, other):
        total_minutes = (self.hour * 60 + self.minute) + (other.hour * 60 +␣
 ↪other.minute)
        total_hour = total_minutes // 60
        total_minute = total_minutes % 60
        return Time(total_hour, total_minute)

    def display_time(self):
        print(f"{self.hour} hour(s) and {self.minute} minute(s)")

time1 = Time(2, 45)
time2 = Time(3, 30)
total_time = time1.add_time(time2)
print("Time 1:")
time1.display_time()
print("Time 2:")
time2.display_time()
print("Total Time after adding:")
total_time.display_time()
```

```
Time 1:
2 hour(s) and 45 minute(s)
Time 2:
3 hour(s) and 30 minute(s)
Total Time after adding:
6 hour(s) and 15 minute(s)
```

[ ]:
```

```