7i19wqiyj

March 12, 2025

Python Programming - 2301CS404

23010101202 || KHUSHI PATEL || 07-03-2025

Lab - 13

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.

[3]:
```
class Students:
    def __init__(self,name,age,grade):
        self.name=name;
        self.age=age;
        self.grade=grade;
st=Students('Khushi',18,'A')
print(st.name)
```

```
Khushi
```

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.

[42]:
```
class Bank_Account:

    def GetAccountDetails(self):
        self.Account_No=int(input("enter accno"))
        self.User_Name=(input("enter username"))
        self.Email=(input("enter Email"))
        self.Account_Type=(input("enter Account_Type"))
        self.Account_Balance=(input("enter Account_Balance"))

    def DisplayAccountDetails(self):
        print(self.User_Name)
        print(self.Email)
        print(self.Account_No)
```

```
        print(self.Account_Type)
        print(self.Account_Balance)



C=Bank_Account()
C.GetAccountDetails()
C.DisplayAccountDetails()
```

```
enter accno 101
enter username khushi
enter Email jpda
enter Account_Type savings
enter Account_Balance 100000
khushi
jpda
101
savings
100000
```

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

[48]:
```
class Circle:
    def __init__(self,r):
        self.r=r
    def area(self):
        return 3.14*self.r*self.r
    def per(self):
        return 2 * 3.14 * self.r
C=Circle(5)
print(C.area())
print(C.per())
```

```
78.5
31.400000000000002
```

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.

[53]:
```
class Employee:
    def GetAccountDetails(self):
        self.name=(input("enter name"))
        self.age=(input("enter age"))
        self.salary=(input("enter salary"))
    def DisplayAccountDetails(self):
        print(self.name)
```

```
        print(self.age)
        print(self.salary)

    def update(self,name,age,salary):
        self.name=name
        self.age=age
        self.salary=salary
Emp=Employee()
Emp.GetAccountDetails()
Emp.DisplayAccountDetails()
Emp.update('vanita',18,20000)
Emp.DisplayAccountDetails()
```

```
enter name khushi
enter age 18
enter salary 1000
khushi
18
1000
vanita
18
20000
```

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

[57]:
```
class bank:
    def __init__(self,balance):
        self.balance=balance

    def deopsit(self,amount):
        self.balance=self.balance+amount

    def withdraw(self,amount):
        self.balance=self.balance-amount
    def checkbalance(self):
        print(self.balance)
B=bank(10000)
B.checkbalance()
B.withdraw(5000)
B.checkbalance()
B.deopsit(5000)
B.checkbalance()
```

```
10000
5000
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.

[4]:
```
class Inventory:
    def __init__(self):
        self.items = {}

    def add_item(self, item_name, price, quantity):
        if item_name in self.items:
            print(f"{item_name} exists. Use update_item.")
        else:
            self.items[item_name] = {'price': price, 'quantity': quantity}

    def remove_item(self, item_name):
        if item_name in self.items:
            del self.items[item_name]
        else:
            print(f"{item_name} not found.")

    def update_item(self, item_name, price=None, quantity=None):
        if item_name in self.items:
            if price is not None:
                self.items[item_name]['price'] = price
            if quantity is not None:
                self.items[item_name]['quantity'] = quantity
        else:
            print(f"{item_name} not found. Use add_item.")

    def display_inventory(self):
        if not self.items:
            print("Inventory is empty.")
        else:
            print("Inventory:")
            for item_name, details in self.items.items():
                print(f"{item_name}: Price: {details['price']}, Quantity:␣
↪{details['quantity']}")
```

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

[2]:
```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

person1 = Person("Alice", 30)
person2 = Person("Bob", 25)
```

```
print(person1.name)
print(person1.age)
print(person2.name)
print(person2.age)
```

```
Alice
30
Bob
25
```

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.

[80]:
```
class student_kit:
    principal='KUSHI'
    def __init__(self,name):
        self.name=name;
    def attendace(self):
        self.day=int(input("enter days"))
    def certificate(self):
        print(f'----------- | {self.name} |   {self.day }  days present   |  ␣
 ↪{self.principal} |--------------');
s=student_kit('VENITA')
s.attendace()
s.certificate()
```

```
enter days 10
----------- | VENITA |   10  days present   |   KUSHI |--------------
```

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

[1]:
```
class Time:
    def __init__(self, hour, minute):
        self.hour = hour
        self.minute = minute

    def add_time(self, other_time):
        total_minutes = self.minute + other_time.minute
        total_hours = self.hour + other_time.hour + total_minutes // 60
```

```
        remaining_minutes = total_minutes % 60
        return Time(total_hours, remaining_minutes)

# Example usage
time1 = Time(5, 30)
time2 = Time(2, 45)

sum_time = time1.add_time(time2)

print(sum_time.hour)
print(sum_time.minute)
```

```
8
15
```

[ ]: