OOP

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

```
class Students:
    def __init__(self, name, age, grade):
        self.name = name
        self.age = age
        self.grade = grade

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

student1 = Students("John Doe", 16, "10th Grade")

student1.display_info()
```

```
Name: John Doe, Age: 16, Grade: 10th Grade
```

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.

```
class Bank_Account:
    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):
        self.account_no = input("Enter Account Number: ")
        self.user_name = input("Enter User Name: ")
        self.email = input("Enter Email: ")
        self.account_type = input("Enter Account Type: ")
        self.account_balance = float(input("Enter Account Balance: "))

    def DisplayAccountDetails(self):
```

```
        print("\nAccount Details:")
        print(f"Account Number: {self.account_no}")
        print(f"User Name: {self.user_name}")
        print(f"Email: {self.email}")
        print(f"Account Type: {self.account_type}")
        print(f"Account Balance: {self.account_balance}")

def main():
    account = Bank_Account("", "", "", "", 0.0)
    account.GetAccountDetails()
    account.DisplayAccountDetails()

if __name__ == "__main__":
    main()
```

```
Enter Account Number:  161116111611
Enter User Name:  Parth
Enter Email:  parth@gmail.com
Enter Account Type:  current
Enter Account Balance:  25000


Account Details:
Account Number: 161116111611
User Name: Parth
Email: parth@gmail.com
Account Type: current
Account Balance: 25000.0
```

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

```
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(5)
print(f"Area of Circle: {circle1.area()}")
print(f"Perimeter of Circle: {circle1.perimeter()}")
```

```
Area of Circle: 78.53981633974483
Perimeter of Circle: 31.41592653589793
```

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

```
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(f"Employee Name: {self.name}")
        print(f"Age: {self.age}")
        print(f"Salary: {self.salary}")

employee1 = Employee("Alice", 30, 50000)
employee1.display_info()

employee1.update_info(age=31, salary=55000)
print("\nUpdated Employee Information:")
employee1.display_info()
```

```
Employee Name: Alice
Age: 30
Salary: 50000

Updated Employee Information:
Employee Name: Alice
Age: 31
Salary: 55000
```

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

```
class BankAccount:
    def __init__(self, account_number, account_holder, balance=0.0):
        self.account_number = account_number
        self.account_holder = account_holder
        self.balance = balance

    def deposit(self, amount):
```

```
        if amount > 0:
            self.balance += amount
            print(f"Deposited {amount}. New balance: {self.balance}")
        else:
            print("Deposit amount must be positive.")

    def withdraw(self, amount):
        if 0 < amount <= self.balance:
            self.balance -= amount
            print(f"Withdrawn {amount}. Remaining balance:
{self.balance}")
        else:
            print("Insufficient balance or invalid amount.")

    def check_balance(self):
        print(f"Account Balance: {self.balance}")

account_number = input("Enter Account Number: ")
account_holder = input("Enter Account Holder Name: ")
balance = float(input("Enter Initial Balance: "))

account = BankAccount(account_number, account_holder, balance)

while True:
    print("\n1. Check Balance\n2. Deposit\n3. Withdraw\n4. Exit")
    choice = input("Enter your choice: ")

    if choice == "1":
        account.check_balance()
    elif choice == "2":
        amount = float(input("Enter deposit amount: "))
        account.deposit(amount)
    elif choice == "3":
        amount = float(input("Enter withdrawal amount: "))
        account.withdraw(amount)
    elif choice == "4":
        print("Exiting... Thank you!")
        break
    else:
        print("Invalid choice! Please try again.")
```

```
Enter Account Number:  161116111611
Enter Account Holder Name:  Parth
Enter Initial Balance:  10000


1. Check Balance
2. Deposit
3. Withdraw
4. Exit
```

```
Enter your choice:  2
Enter deposit amount:  5000

Deposited 5000.0. New balance: 15000.0

1. Check Balance
2. Deposit
3. Withdraw
4. Exit

Enter your choice:  1

Account Balance: 15000.0

1. Check Balance
2. Deposit
3. Withdraw
4. Exit

Enter your choice:  4

Exiting... Thank you!
```

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

```python
class Inventory:
    def __init__(self):
        self.items = {}

    def add_item(self, name, price, quantity):
        if name in self.items:
            self.items[name]['quantity'] += quantity
        else:
            self.items[name] = {'price': price, 'quantity': quantity}
        print(f"Added {quantity} of {name} at ${price} each.")

    def remove_item(self, name, quantity):
        if name in self.items:
            if self.items[name]['quantity'] >= quantity:
                self.items[name]['quantity'] -= quantity
                print(f"Removed {quantity} of {name}.")
                if self.items[name]['quantity'] == 0:
                    del self.items[name]
            else:
                print("Insufficient quantity.")
        else:
            print("Item not found.")
```

```
    def display_inventory(self):
        if not self.items:
            print("\nInventory is empty.")
        else:
            print("\nInventory:")
            for name, details in sorted(self.items.items()):
                print(f"{name}: ${details['price']}, Qty:
{details['quantity']}")

inventory = Inventory()

while True:
    print("\n1. Add Item\n2. Remove Item\n3. Display Inventory\n4.
Exit")
    choice = input("Enter your choice: ")

    if choice == "1":
        name = input("Item name: ")
        price = float(input("Price: "))
        quantity = int(input("Quantity: "))
        inventory.add_item(name, price, quantity)
    elif choice == "2":
        name = input("Item name to remove: ")
        quantity = int(input("Quantity to remove: "))
        inventory.remove_item(name, quantity)
    elif choice == "3":
        inventory.display_inventory()
    elif choice == "4":
        print("Exiting... Thank you!")
        break
    else:
        print("Invalid choice! Try again.")


1. Add Item
2. Remove Item
3. Display Inventory
4. Exit

Enter your choice:  1
Item name:  Laptop
Price:  3
Quantity:  3

Added 3 of Laptop at $3.0 each.

1. Add Item
2. Remove Item
3. Display Inventory
4. Exit
```

```
Enter your choice:  4

Exiting... Thank you!
```

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

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

    def display_details(self):
        print(f"Name: {self.name}, Age: {self.age}, Grade:
{self.grade}")

student1 = Student("Alice", 20, "A")
student2 = Student("Bob", 22, "B")

student1.display_details()
student2.display_details()
```

```
Name: Alice, Age: 20, Grade: A
Name: Bob, Age: 22, Grade: B
```

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.

```
class StudentKit:
    principal_name = "Mr. ABC"

    def __init__(self, name):
        self.name = name
        self.attendance_days = 0

    def attendance(self, days):
        self.attendance_days = days

    def generate_certificate(self):
        print("\n----------------------------")
        print(f"Certificate of Attendance")
        print(f"Presented to: {self.name}")
```

```
        print(f"Principal: {StudentKit.principal_name}")
        print(f"Days Present: {self.attendance_days}")
        print("----------------------------\n")

name = input("Enter student name: ")
student = StudentKit(name)
days_present = int(input("Enter number of days present: "))
student.attendance(days_present)
student.generate_certificate()
```

```
Enter student name:  Parth
Enter number of days present:  90


----------------------------
Certificate of Attendance
Presented to: Parth
Principal: Mr. ABC
Days Present: 90
----------------------------
```

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

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

    def add_time(self, other):
        total_minutes = self.minute + other.minute
        extra_hours = total_minutes // 60
        final_minutes = total_minutes % 60
        final_hours = self.hour + other.hour + extra_hours
        return Time(final_hours, final_minutes)

    def display_time(self):
        print(f"Time: {self.hour} hours and {self.minute} minutes")

time1 = Time(2, 50)
time2 = Time(1, 30)
sum_time = time1.add_time(time2)
sum_time.display_time()
```

```
Time: 4 hours and 20 minutes
```