In [ ]:
```
 Create a sample class named Employee with two attributes id and name
employee :
    id
    name
object initializes id and name dynamically for every Employee object create

emp = Employee(1, "coder")
```

In [2]:
```
class Employee:
  def __init__(self, id, name):
    self.id = id
    self.name = name

emp = Employee(1, "coder")
print(emp.id)
print(emp.name)
```

```
1
coder
```

In [ ]:
```
Use del property to first delete id attribute and then the entire object
```

In [7]:
```
class Employee:
  def __init__(self, id, name):
    self.id = id
    self.name = name

emp = Employee(1, "coder")
print(emp.id)
```

```
1
```

In [8]:
```
class Employee:
  def __init__(self, id, name):
    self.id = id
    self.name = name

emp = Employee(1, "coder")
del emp.id

print(emp.id)

del emp
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-8-6c9319d59971> in <module>
      7 del emp.id
      8
----> 9 print(emp.id)
     10
     11 del emp

AttributeError: 'Employee' object has no attribute 'id'
```

In [ ]:
```
 create inheritance using animal Dog relation.
for example,
    Animal and Dog both has same habitat so create a method for habitat
```

In [7]:
```
class Animal:
    def __init__(self, habitat):
        self.habitat = habitat

    def get_habitat(self):
        return self.habitat

class Dog(Animal):
    def __init__(self, habitat, breed):
        super().__init__(habitat)
        self.breed = breed

    def get_breed(self):
        return self.breed

animal1 = Animal("jungle")
dog1 = Dog("city", "Golden Retriever")

print(animal1.get_habitat())
print(dog1.get_habitat())
print(dog1.get_breed())
```

```
jungle
city
Golden Retriever
```

In [ ]:
```
Create multiple inheritance on teacher,student and youtuber.
  if we have created teacher and now one student joins master degree with
```

In [8]:
```
class Teacher:
    def __init__(self, subject):
        self.subject = subject

    def teach(self):
        print(f"Teaching {self.subject}")

class Student:
    def __init__(self, major):
        self.major = major

    def study(self):
        print(f"Studying {self.major}")

class TeacherStudent(Teacher, Student):
    def __init__(self, subject, major):
        Teacher.__init__(self, subject)
        Student.__init__(self, major)

teacher_student = TeacherStudent("Computer Science", "Computer Science")
teacher_student.teach()
teacher_student.study()
```

```
Teaching Computer Science
Studying Computer Science
```

In [ ]:
```
Create a base class Shape with a method area that prints "Calculating area
  Create two subclasses Rectangle and Circle.
  Override the area method in each subclass to calculate the area of a rec
  Instantiate objects of both Rectangle and Circle and call their area met
```

In [11]:
```
import math

class Shape:
    def area(self):
        print("Calculating area...")

class Rectangle(Shape):
    def __init__(self, length, width):
        self.length = length
        self.width = width

    def area(self):
        return self.length * self.width

class Circle(Shape):
    def __init__(self, radius):
        self.radius = radius

    def area(self):
        return math.pi * self.radius ** 2
rectangle = Rectangle(4, 5)
circle = Circle(3)

print("Rectangle area:", rectangle.area())
print("Circle area:", circle.area())
```

```
Rectangle area: 20
Circle area: 28.274333882308138
```

In [ ]:
```
Create a class BankAccount with attributes account_holder_name, balance, an
Create a method deposit to add money to the account, and withdraw to withdr
Create a method check_balance to print the current balance
```

In [12]:
```
class BankAccount:
    def __init__(self, account_holder_name, balance, pin):
        self.account_holder_name = account_holder_name
        self.balance = balance
        self.pin = pin

    def deposit(self, amount):
        if amount > 0:
            self.balance += amount
            print("Deposit successful. New balance:", self.balance)
        else:
            print("Invalid deposit amount.")

    def withdraw(self, amount, entered_pin):
        if entered_pin == self.pin and amount <= self.balance:
            self.balance -= amount
            print("Withdrawal successful. New balance:", self.balance)
        elif entered_pin != self.pin:
            print("Incorrect PIN.")
        else:
            print("Insufficient funds.")

    def check_balance(self):
        print("Your current balance is:", self.balance)


account = BankAccount("John Doe", 1000, 1234)


account.deposit(500)
account.withdraw(200, 1234)
account.check_balance()
```

```
Deposit successful. New balance: 1500
Withdrawal successful. New balance: 1300
Your current balance is: 1300
```

In [ ]:
```
Now student is teacher as well as youtuber then what???
```

In [2]:
```
class Teacher:
    def __init__(self, name, subject):
        self.name = name
        self.subject = subject

    def teach(self):
        return f"{self.name} teaches {self.subject}."

class Student:
    def __init__(self, name, degree):
        self.name = name
        self.degree = degree

    def study(self):
        return f"{self.name} studies for {self.degree} degree."

class YouTuber:
    def __init__(self, channel_name):
        self.channel_name = channel_name

    def create_content(self):
        return f"Creating content for the {self.channel_name} YouTube chann

class StudentTeacherYouTuber(Teacher, Student, YouTuber):
    def __init__(self, name, subject, degree, channel_name):
        Teacher.__init__(self, name, subject)
        Student.__init__(self, name, degree)
        YouTuber.__init__(self, channel_name)

    def manage_roles(self):
        return f"{self.name} manages teaching {self.subject}, studying for


person = StudentTeacherYouTuber("Bob", "Physics", "Master", "LearnWithBob")
print(person.teach())
print(person.study())
print(person.create_content())
print(person.manage_roles())
```

```
Bob teaches Physics.
Bob studies for Master degree.
Creating content for the LearnWithBob YouTube channel.
Bob manages teaching Physics, studying for Master, and creating content fo
r LearnWithBob.
```

In [ ]: