

Exp.No:21

Constructors - Parameterized Constructor

AIM

To write a Python code to create a class for a person with a parameterized constructor, which will take the **name** and **userid** of the person as parameters and print the **userid** of the person.

ALGORITHM

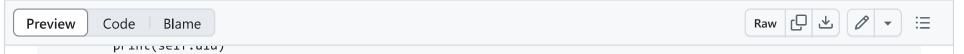
- 1. Begin the program.
- 2. Define a person class.
- 3. The person class should have a parameterized __init__ method that accepts two parameters: name and userid.
- 4. Inside the __init__ method, assign the name to self.name and the userid to self.userid.
- 5. Print the self.userid.
- 6. Prompt the user to enter their name (string) and userid.
- 7. Create an instance s1 of the person class by passing the entered name and userid to the constructor.
- 8. Terminate the program.

p=Person(input(),input())

PROGRAM

Reg no-212223020028 Name-Tharani devi.G write your code

Module-5 / Constructors - Parameterized Constructor.md



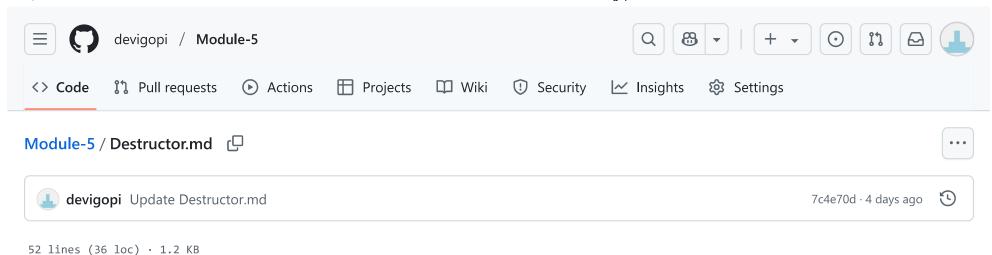
OUTPUT

↑ Top



RESULT

This program for code to create a class for a person with a parameterized constructor, which will take the name and userid of the person as parameters and print the userid of the person is successfully executed.



Exp.No:22

Destructor

AIM

To create a Python class Student with a destructor.

ALGORITHM

- 1. Begin the program.
- 2. Define the student class.

- 3. Inside the student class, define the __init__ method (constructor) and the __del__ method (destructor).
- 4. Create an object s2 of the student class. When the object s2 is created, the __init__ method is called, and its print statements are executed.
- 5. Use the del statement to delete the object s2. This triggers the __del__ method (destructor), and the respective print statements are executed.
- 6. Terminate the program.

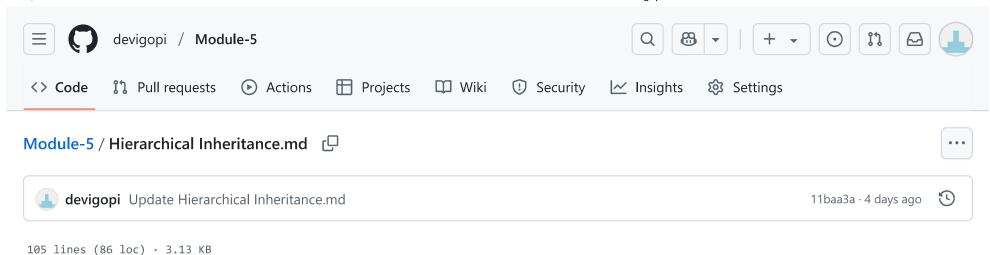
PROGRAM

```
Reg no-212223020028
    Name-Tharani devi.G
    write your code
    class Student:
        def init (self, name):
            self.name=name
            print('Inside Constructor')
            print('Object initialized')
            print("Hello, my name is",name)
        def del (self):
            print("Inside destructor")
            print("Object destroyed")
    name='Emma'
    obj=Student(name)
Module-5 / Destructor.md
                                                                                                                    ↑ Top
                                                                                                 Raw 📮 🕹
Preview
           Code
                  Blame
```



RESULT

This program for class Student with a destructor is successfully executed.



Exp.No:25

Hierarchical Inheritance

AIM

To write a Python program to get the employee and doctor details and display them using hierarchical inheritance. Create a parent (base) class named Details and two child (derived) classes named Employee and Doctor.

ALGORITHM

1. Begin the program.

- 2. Create a class Details with an __init__ method to initialize three attributes: id , name , and gender .
- 3. Define a method display_details() to print the values of id , name , and gender .
- 4. Create a class Employee that inherits from the Details class.
 - Add two additional attributes: company and department.
 - Override the display_details() method to print the employee-specific attributes (company and department) along with the inherited details.
- 5. Create a class Doctor that also inherits from the Details class.
 - Add two additional attributes: hospital and department.
 - Override the display_details() method to print the doctor-specific attributes (hospital and department) along with the inherited details.
- 6. Accept input for employee and doctor details.
- 7. Create objects of Employee and Doctor using the input.
- 8. Call the display_details() method for both objects to print the details.
- 9. Terminate the program.

PROGRAM

```
Reg no-212223020028

Name-Tharani devi.G

write your code

class Details:

    def __init__(self):
        self.__id="<No Id>"
        self.__name="<No Name>"
        self.__gender="<No Gender>"

    def setData(self,id,name,gender):
        self.__id=id
        self.__name=name
        self.__gender=gender
```

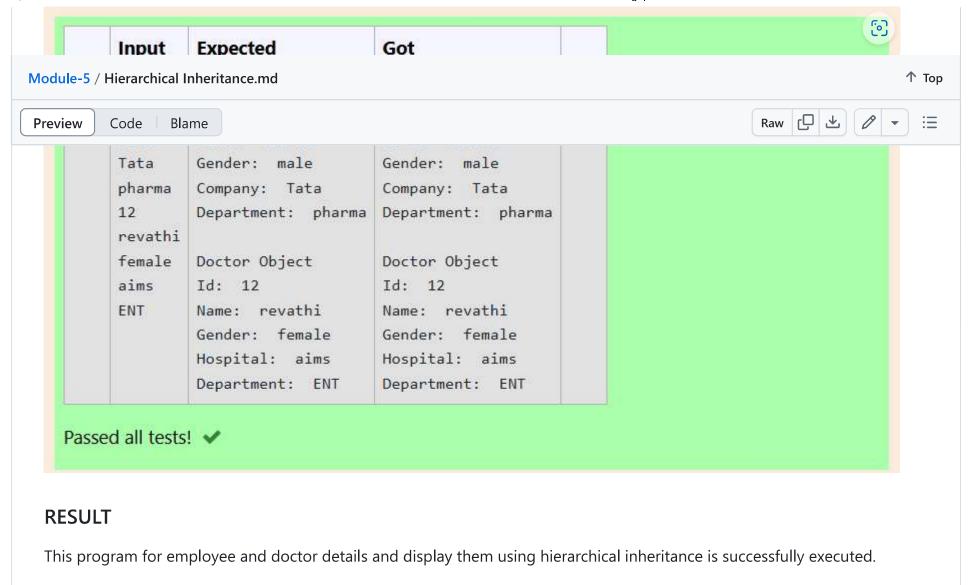
```
def showData(self):
        print("Id: ",self.__id)
        print("Name: ", self. name)
        print("Gender: ", self. gender)
class Employee(Details): #Inheritance
    def init (self):
        self.__company="<No Company>"
        self. dept="<No Dept>"
    def setEmployee(self,id,name,gender,comp,dept):
        self.setData(id,name,gender)
        self.__company=comp
        self.__dept=dept
    def showEmployee(self):
        self.showData()
        print("Company: ", self. company)
        print("Department: ", self. dept)
class Doctor(Details): #Inheritance
    def __init__(self):
        self. hospital="<No Hospital>"
        self. dept="<No Dept>"
    def setEmployee(self,id,name,gender,hos,dept):
        self.setData(id,name,gender)
        self. hospital=hos
        self.__dept=dept
    def showEmployee(self):
       self.showData()
        print("Hospital: ", self.__hospital)
        print("Department: ", self.__dept)
id=int(input())
name=input()
gender=input()
comp=input()
dept=input()
id1=int(input())
```

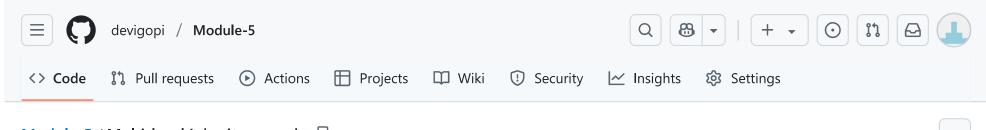
```
nam=input()
gen=input()
hosp=input()

dep=input()

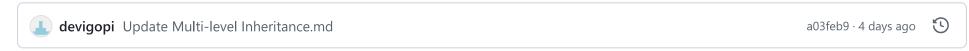
print("Employee Object")
e=Employee()
e.setEmployee(id,name,gender,comp,dept)
e.showEmployee()
print("\nDoctor Object")
d = Doctor()
d.setEmployee(id1, nam, gen, hosp, dep)
d.showEmployee()
```

OUTPUT





Module-5 / Multi-level Inheritance.md ☐



74 lines (56 loc) · 2.5 KB



Exp.No:24

Multi-level Inheritance

AIM

To write a Python program to get the name, age, and ID of a person and display them using multilevel inheritance.

ALGORITHM

- 1. Define the Person class:
 - o Inside the Person class, define the __init__ method (constructor) with two parameters: name and age.

- o Inside the __init__ method, assign the name to self.name and age to self.age.
- 2. Define the PersonDetails class that inherits from the Person class:
 - o Inside the PersonDetails class, define the __init__ method (constructor) with three parameters: name , age , and person_id .
 - o Inside the __init__ method, call the __init__ method of the Person class using super() to initialize name and age .
 - Assign person_id to self.person_id.
- 3. Define the DisplayDetails class that inherits from the PersonDetails class:
 - o Inside the DisplayDetails class, define the __init__ method (constructor) with three parameters: name , age , and person_id .
 - o Inside the __init__ method, call the __init__ method of the PersonDetails class using super() to initialize name, age, and person_id.
- 4. Inside the DisplayDetails class, define the show details method:
 - o Inside the show_details method, return a formatted string with self.name, self.age, and self.person_id.
- 5. Prompt the user to enter name (string), age (integer), and person id (integer).
- 6. Create an instance person of the DisplayDetails class, passing name, age, and person_id to the constructor.
- 7. Call the show_details method on the person object and print the result.
- 8. Terminate the program.

PROGRAM

```
Reg no-212223020028
Name-Tharani devi.G
write your code
class Parent:
  def init (self,name):
     self.name = name
  def getName(self):
     return self.name
class Child(Parent):
  def __init__(self,name,age):
     Parent. init (self, name)
     self.age = age
  def getAge(self):
     return self.age
class Grandchild(Child):
  def __init__(self,name,age,id):
    Child.__init__(self,name,age)
     self.id=id
  def getid(self):
     return self.id
name=input()
age=int(input())
id=int(input())
gc = Grandchild(name,age,id)
print(gc.getName(), gc.getAge(), gc.getid())
```

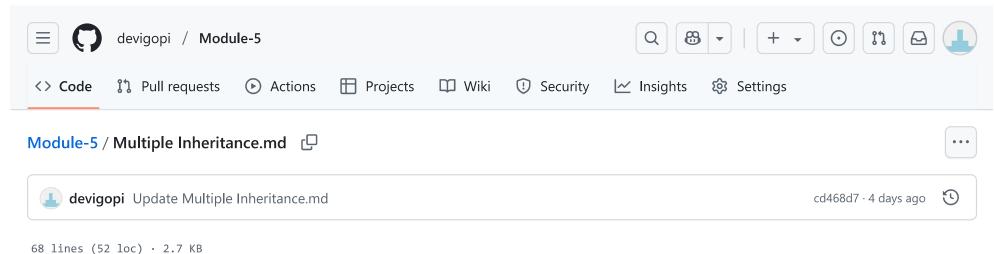
OUTPUT

	Input	Expected	Got	
~	srinivas 24 21223456	srinivas 24 21223456	srinivas 24 21223456	~

Passed all tests! 🗸

RESULT

This program for get the name, age, and ID of a person and display them using multilevel inheritance is successfully executed.



Exp.No:23

Multiple Inheritance

AIM

To write a Python program to get the name, attendance, and ID of a student and check if they are eligible for the next module using multiple inheritance. If attendance > 80, the student is eligible; otherwise, not eligible.

ALGORITHM

1. Define the Student class.

- 2. Inside the Student class, define the __init__ method (constructor). The __init__ method accepts two parameters: name and student_id.
 - o Inside the __init__ method: Assign the value of name to self.name and student_id to self.student_id.
- 3. Define the get_student_info method inside the Student class:
 - This method should return a string formatted with self.name and self.student_id.
- 4. Define the Attendance class, which inherits from the Student class.
- 5. Inside the Attendance class, define the __init__ method (constructor).
 - The __init__ method accepts three parameters: name , student_id , and attendance .
 - o Inside the __init__ method: Call the parent class constructor super().__init__(name, student_id) to initialize name and student_id. Assign the value of attendance to self.attendance.
- 6. Define the check_eligibility method inside the Attendance class:
 - o If self.attendance is greater than 80, return a formatted string indicating the student is eligible for the module exam.
 - Otherwise, return a formatted string indicating the student is not eligible for the module exam.
- 7. Prompt the user to enter the name (as a string), student_id (as an integer), and attendance (as an integer).
- 8. Create an instance student of the Attendance class, passing the entered name, student_id, and attendance to the constructor.
- 9. Call the check eligibility method on the student object and print the result.
- 10. Terminate the program.

PROGRAM

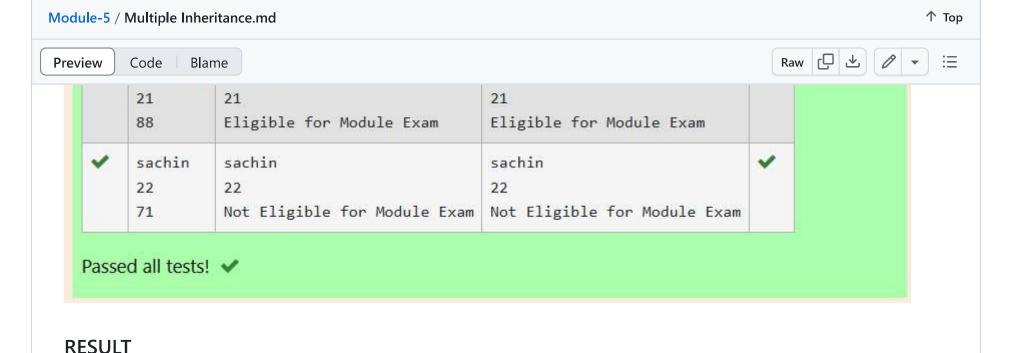
```
class one():
    def student1(self,name,att,id):
        print(name)
        print(att)
        print("Eligible for Module Exam")
class two():
```

```
def student2(self,name,att,id):
    print(name)
    print(att)
    print("Not Eligible for Module Exam")

class valid(one,two):
    def valid(self,name,att,id):
        if id>75:
            one().student1(name,att,id)
        else:
            two().student2(name,att,id)

name=input()
att=int(input())
id=int(input())
x=valid()
x.valid(name,att,id)
```

OUTPUT



This program for get the name, attendance, and ID of a student and check if they are eligible for the next module using multiple inheritance. If attendance > 80, the student is eligible; otherwise, not eligible is successfuly executed.

Started onTuesday, 23 April 2024, 9:12 AMStateFinishedCompleted onTuesday, 23 April 2024, 9:56 AM

Grade 43 mins 29 secs **100.00** out of 100.00

Question **1**

Correct

Mark 20.00 out of 20.00

Write a python program to define a function that accepts 3 values and return its multiplication.

Answer: (penalty regime: 0 %)

- 1 a=int(input())
- 2 b=int(input())
- 3 c=int(input())
- 4 d=a*b*c
- 5 print("Multiply is",d)

	Input	Expected	Got	
~	10 20 30	Multiply is 6000	Multiply is 6000	~
~	85 63 90	Multiply is 481950	Multiply is 481950	~

Passed all tests! ✓

```
Question 2
Correct
Mark 20.00 out of 20.00
```

Write a Python Program to Display the Employee Details

Empld, Emp Name., and Also Check Valid Employee or Not.

Note: If Employee id > 500000 Valid, Else Invalid

For example:

Input	Result			
563421 saveetha	(563421,	'saveetha')	Valid	Employee

Answer: (penalty regime: 0 %)

```
class detail1:
 2 🔻
        def detail1(self,empid,name):
 3
            print(f"({empid}, '{name}') Valid Employee")
 4 *
    class detail2:
 5 🔻
        def detail2(self,empid,name):
            print(f"({empid}, '{name}') Invalid Employee")
 6
 7 🔻
    class valid(detail1,detail2):
 8 ,
        def valid(self,empid,name):
 9
            if empid>500000:
10
                detail1().detail1(empid,name)
            else:
11 \
12
                detail2().detail2(empid,name)
    empid=int(input())
13
   name=input()
14
   obj=valid()
   obj.valid(empid,name)
```

	Input	Expected	Got	
~	563421 saveetha	(563421, 'saveetha') Valid Employee	(563421, 'saveetha') Valid Employee	~
~	237643 John	(237643, 'John') Invalid Employee	(237643, 'John') Invalid Employee	~

Passed all tests! 🗸

Question **3**Correct
Mark 20.00 out of 20.00

Write a Python program to Get the name, age and salary of a person and display using Multilevel inheritance.

For example:

Input	Result		
srinivas 24	srinivas	24	23456
23456			

Answer: (penalty regime: 0 %)

```
class Parent:
 2 🔻
        def __init__(self,name):
 3
            self.name=name
 4 ,
        def getName(self):
 5
            return self.name
    class Child(Parent):
 6 •
 7
        def __init__(self,name,age):
            Parent.__init__(self,name)
 8
            self.age=age
 9
        def getAge(self):
10
            return self.age
11
    class Grandchild(Child):
12 🔻
13 🔻
        def __init__(self,name,age,sal):
14
            Child.__init__(self,name,age)
15
            self.sal=sal
        def getSal(self):
16
            return self.sal
17
18
19
    name=input()
20
    age=int(input())
    sal=int(input())
21
   oc = Grandchild(name age sal)
```

	Input	Expected	Got	
~	srinivas 24 23456	srinivas 24 23456	srinivas 24 23456	~

Passed all tests! 🗸



Question 4
Correct
Mark 20.00 out of 20.00

Write a python code to calculate the multiplication of two numbers using parameterised constructor.

For example:

Input	Result		
5	ele 1 = 5		
6	ele 2 = 6		
	Total = 30		

Answer: (penalty regime: 0 %)

	Input	Expected	Got	
~	5	ele 1 = 5	ele 1 = 5	~
	6	ele 2 = 6	ele 2 = 6	
		Total = 30	Total = 30	

Passed all tests! 🗸

