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.

PROGRAM

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



```
        print(self.uid)
p=Person(input(),input())
```

OUTPUT

	Input	Expected	Got	
✔	John John@abc.com	John@abc.com	John@abc.com	✔

Passed all tests! ✔

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)
```

	Expected	Got	
✔	Inside Constructor Object initialized Hello, my name is Emma Inside destructor Object destroyed	Inside Constructor Object initialized Hello, my name is Emma Inside destructor Object destroyed	✔

Passed all tests! ✔

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

	Input	Expected	Got	
	Tata pharma 12 revathi female aims ENT	Gender: male Company: Tata Department: pharma Doctor Object Id: 12 Name: revathi Gender: female Hospital: aims Department: ENT	Gender: male Company: Tata Department: pharma Doctor Object Id: 12 Name: revathi Gender: female Hospital: aims Department: ENT	

Passed all tests! ✔

RESULT

This program for employee and doctor details and display them using hierarchical inheritance is successfully executed.

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

- 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:
   - Inside the `PersonDetails` class, define the `__init__` method (constructor) with three parameters: `name`, `age`, and `person_id`.
   - 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:
   - Inside the `DisplayDetails` class, define the `__init__` method (constructor) with three parameters: `name`, `age`, and `person_id`.
   - 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:
   - 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`.
   - 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`.
   - 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:
   - 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

Module-5 / Multiple Inheritance.md ↑ Top

Preview Code Blame Raw

	21 88	21 Eligible for Module Exam	21 Eligible for Module Exam	
✔	sachin 22 71	sachin 22 Not Eligible for Module Exam	sachin 22 Not Eligible for Module Exam	✔

Passed all tests! ✔

RESULT

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 on	Tuesday, 23 April 2024, 9:12 AM
State	Finished
Completed on	Tuesday, 23 April 2024, 9:56 AM
Time taken	43 mins 29 secs
Grade	**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 %)

```
a=int(input())
b=int(input())
c=int(input())
d=a*b*c
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! ✔

Correct

Marks for this submission: 20.00/20.00.

Question **2**

Correct

Mark 20.00 out of 20.00

Write a Python Program to Display the Employee Details

EmpId , 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:
    def detail1(self,empid,name):
        print(f"({empid}, '{name}')  Valid Employee")
class detail2:
    def detail2(self,empid,name):
        print(f"({empid}, '{name}')  Invalid Employee")
class valid(detail1,detail2):
    def valid(self,empid,name):
        if empid>500000:
            detail1().detail1(empid,name)
        else:
            detail2().detail2(empid,name)
empid=int(input())
name=input()
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! ✔

Correct

Marks for this submission: 20.00/20.00.

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 23456	srinivas 24 23456

Answer: (penalty regime: 0 %)

```
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,sal):
        Child.__init__(self,name,age)
        self.sal=sal
    def getSal(self):
        return self.sal

name=input()
age=int(input())
sal=int(input())
gc = Grandchild(name,age,sal)
```

	Input	Expected	Got	
✔	srinivas 24 23456	srinivas 24 23456	srinivas 24 23456	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

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

Answer: (penalty regime: 0 %)

```
class mul:
    def __init__(self,a,b):
        self.a=a
        self.b=b
        print("ele 1 =",self.a)
        print("ele 2 =",self.b)
        print("Total  =",self.a*self.b)
a=mul(int(input()),int(input()))
```

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

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

Question **5**

Correct

Mark 20.00 out of 20.00

Add destructor in the following python code.

For example:

Result
Hello World! Hello from the __del__ method.

Answer: (penalty regime: 0 %)

Reset answer

```
class Awesome:

    # some method
    def greetings(self):
        print("Hello World!")
    def __del__(self):
        print("Hello from the __del__ method.")

# object of the class
obj = Awesome()

# calling class method
obj.greetings()
```

	Expected	Got	
✔	Hello World! Hello from the __del__ method.	Hello World! Hello from the __del__ method.	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.