Object Oriented Programming using Python

Object Oriented Programming(OOP) is an approach to problem solving where computations are carried out using objects. OOP allows us to represent data as real-world ob...More

Start

Learning Progress

Status: Completed

Completion Certificate

Click on Generate Certificate button to claim your completion certificate

Generate Certificate

Overview | Contents | Discussions

What you will learn

Provide insights into basics of object oriented programming Introduce class and objects Discuss Encapsulation and need for encapsulation Introduce static attributes and methods in Python Discuss on how to build a robust Python code Introduce various class Relationships Provide insights into Inheritance and their types Introduce abstract class and methods Discuss how to handle custom exceptions using exception handling

Authors/Creators

SL Seetha Lakshmi .

AA Avinash Ajay Malawade

MM Malathi M

Show Curators/Contacts

At a glance

- Course
- 46h 13m
- Beginner Level
- Free
- Infosys Wingspan
- EN
- Object Oriented Programming using Python, Object Oriented Programming, assessment for oops uisng python, python assessment, FP, constructor, static, inheritance, aggregation, weaker realtionship,

Thank you. Your test submitted.

You have cleared this assessment.

Obtained Percentage	Obtained Marks
100 %	15 / 15

Best Attempt Score:100 % on 23-03-2025

Review Your Attempt

What will be the output of the following Python code snippet?

```
class Sample:
    num=100
    def __init__(self,num):
        self.value=None

    def set_value(self,num):
        value=num

    def get_value(self):
        self.value=num
        return self.value

obj=Sample(200)
obj.set_value(10)
print(obj.get_value())
```

Warning

This operati

- ○ Code will not compile due to error in set_value() method
- ◉ Code will not compile due to error in get_value() method
- ○ 100
- ○ 0

Tech Mindz company has a set of employees. Employees can either be permanent employees or temporary employees. Salary is calculated for all employees but calculation is different for both type of employees. In future there might be a few more categories of employees that may be added.

Company wants an optimal OOP solution for this scenario.

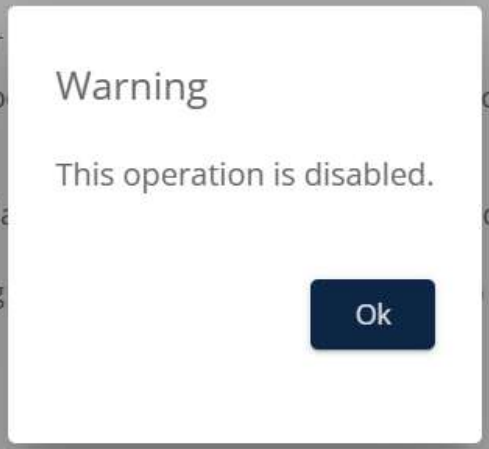
Suggestions from few employees are given below-

A) Jack: Create a class Employee with employeetyp calculate_salary() which would have logic to calculate salary for different types of employee

B) Tom: Create an Employee class and two child cl d TemporaryEmployee having method calculate_salary()

C) Olive: Create an abstract Employee class having

Choose the best statement from given options.



- ◯ Suggestions from Jack and Olive combined will result in an optimal solution
- ◯ Suggestion given by Jack is sufficient to get an optimal solution
- ◉ Suggestions from Tom and Olive combined will result in an optimal solution
- ◯ Suggestion given by Tom is sufficient to get an optimal solution

What will be the output of the below Python code?

```
class Demo:
    def __init__(self,num):
        self.__num=num
        self.value=0

    def display_total(self):
        total=self.__num+self.value
        return total

    def set_num(self,num):
        self.__num=num

obj=Demo(100)
obj.value=200
obj.set_num(obj.value)
print(obj.display_total())
```

- ○ 200
- ◉ 400
- ○ 300
- ○ 100

Consider the Python code given below.

```
class ClassAA:
    def method1(self):
        return 10
    def method2(self):
        return 20

class ClassBB:
    def method3(self):
        return 60
    def method4(self):
        return 70

class ClassCC(ClassAA):
    def method1(self):
        return 30
    def method4(self):
        return 40

class ClassDD(ClassBB):
    def method1(self):
        return 50
    def method2(self):
        return 60

obj1=ClassCC()
obj2=ClassDD()
```

Match the following print statements with corresponding output

Print Statements	Outputs
1. print(obj1.method2()+obj2.method4())	A. 100
2. print(obj1.method1()+obj2.method1())	B. 80
3. print(obj1.method4()+obj2.method3())	C. 90

- (●) 1-C, 2-B, 3-A
- ( ) 1-A, 2-B, 3-C
- ( ) 1-A, 2-C, 3-A
- ( ) 1-C, 2-A, 3-A

Consider the Python code given below.

```
class Base:
    def __init__(self):
        self.__value=200
    def get_value(self):
        return self.__value+1

class Child(Base):
    def get_num(self):
        num=100
        return num

class GrandChild(Child):
    def __init__(self):
        self.num=200

child=Child()
grandchild=GrandChild()
print(grandchild.get_value())
```


What changes should be done in the above code so as to get the output as 201?

- ◯ Make the instance variable of Base class public
- ◯ Add a constructor with statement super().__init__() in Child class
- ◉ Add statement super().__init__() in the constructor of GrandChild class
- ◯ Add a constructor in the Child class and initialize num to 201

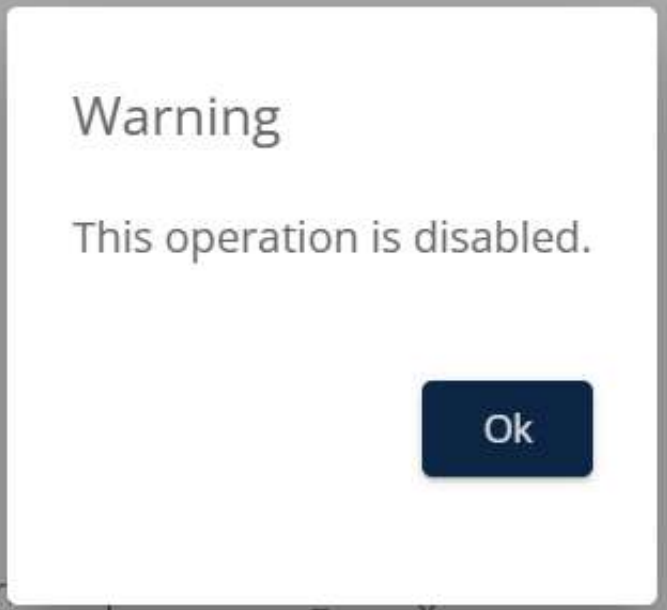
Consider the below Python code.
Assume that necessary imports have been done.

```
class Shape(metaclass=ABCMeta):
    def __init__(self):
        print("I am in init")

    @abstractmethod
    def draw_shape(self):
        pass

    @abstractmethod
    def set_color(self):
        pass

class Circle(Shape):
    def draw_shape(self):
        print("Draw Circle")
```



Which of the following statement(s) is/are TRUE?

i) Class Circle cannot be instantiated as it does r

ii) The above code will result in an error because class Shape has two abstract methods

- ◯ Both i) and ii) are True
- ◯ Neither i) nor ii) is True
- ◉ Only i) is True
- ◯ Only ii) is True

Consider the Python code given below.

```
class Alpha:
    def one(self):
        return 10
    def two(self):
        return self.one()

class Beta(Alpha):
    def one(self):
        return 10
    def three(self):
        return 10

class Gamma(Beta):
    def one(self):
        return 10
    def four(self):
        return self.one()
    def three(self):
        return super().two()
    def five(self):
        return 10

gobj=Gamma()
num1=gobj.two()+gobj.three()+gobj.four()
num2=gobj.three()+gobj.four()+gobj.one()
num3=gobj.one()+gobj.two()+gobj.three()
```

Which of the following statement(s) is/are TRUE?

i) Value of num1, num2, num3 will be the same
ii) Error in class Gamma as it cannot override method one() which has already been overriden by parent class Beta
iii) Error in class Gamma as method three() is giving call to method two() using super() which is not written in parent class Beta
iv) Value of num1 and num2 will be the same but num3 will be different

- ◯ Only iv)
- ◯ Both ii) and iii)
- ◉ Only i)
- ◯ Only ii)

What will be the output of the Python code given below?

```
class ExceptionOne(Exception):
    pass
class Test:
    counter=1
    @staticmethod
    def perform_test(temp_list,var1):
        try:
            if(temp_list[var1]>=5):
                Test.counter+=1
                temp_list[var1]-=2
                raise ExceptionOne()
                temp_list[var1]=5
            else:
                raise Exception()
        except Exception:
            Test.counter-=5
        except ExceptionOne:
            Test.counter+=5
        print("Data:",temp_list[var1])
try:
    test=Test()
    test.perform_test([2,4,7,5,1],3)
finally:
    print("Counter:",Test.counter)
```

- (●) Data: 3 Counter: -3
- ( ) Data: 3 Counter: 7
- ( ) Counter: -3
- ( ) Data: 3

What will be the output of the Python code given below?

```
class SomeException(Exception):
    pass
class Calculator:
    def div(self,var1,var2):
        var3=var1//var2
        raise SomeException
try:
    calculator=Calculator()
    calculator.div(10,0)
except SomeException:
    print("Some Exception occurred")
except:
    print("Error")
finally:
    print("Finally out of division meth
```

Warnir

This ope

- ○ Some Exception occurred
Error
Finally out of divison method
- ○ Finally out of divison method
- ◉ Error
Finally out of divison method
- ○ Some Exception occurred
Finally out of divison method

John has written the Python code given below:

```
class Account:
    def __init__(self,acc_number,balance_amt):
        self.__acc_number=acc_number
        self.__balance_amount=balance_amt
        self.account_type="Saving" #Line1
    def get_acc_number(self):
        return self.__acc_number
    def get_balance_amount(self):
        return self.__balance_amo
account=Account(123456,460000)

print(account.__acc_number,accoun                    _balance_amount) #Line2
```

Note: Line numbers are for reference on
John wants to display the account details error due to improper variable access.
Which of the following option should he d get his output?

- (●) Replace Line2 with: print(a.get_acc_number(),a.account_type,a.get_balance_amount())
- ( ) Replace Line2 with: print(a.get_acc_number(),a.account_type,a.__balance_amount)
- ( ) Remove methods get_acc_number() and get_balance_amount() from the class
- ( ) Replace Line1 to: self.__account_type="Saving"

Consider the Python code given below:

```
class Base:
    def base_method(self):
        print("Inside baseMethod")
    def common_method(self):
        print("Inside commonMethod: Base")

class Derived(Base):
    def common_method(self):
        print("Inside commonMethod: Derived")
    def derived_method(self):
        print("Inside derivedMethod")
```

Line#	Code at module level
1.	base_obj=Base()
2.	derived_obj=Derived()
3.	_____.common_method()
4.	_____.base_method()
5.	_____.common_method()

Which reference variable/s should be used at line numbers 3,4,5 so

Inside commonMethod: Derived
Inside baseMethod
Inside commonMethod: Base

Note: Line numbers are just for reference.
Choose TWO options that apply.

- [x] derived_obj,base_obj,base_obj
- [ ] derived_obj,derived_obj,derived_obj
- [ ] base_obj,base_obj,derived_obj
- [x] derived_obj,derived_obj,base_obj

Consider the Python code given below.
Note: Assume that necessary imports have been done.

```
class ClassA(metaclass=ABCMeta):

    def method1(self):
        return 45

    @abstractmethod
    def method2(self):
        pass

    @abstractmethod
    def method3(self):
        pass

class ClassB(ClassA):

    def method3(self):
        return 25
```


What should be done in so that object of ClassB gets created successfully?

- (●) Implementation of method2() must be provided in ClassB
- ( ) ClassB can be instantiated without any modification to the given code
- ( ) Implementation of method1() must be provided in ClassB
- ( ) Implementation of method3() must be removed from ClassB

Consider the Python code given below.

```
class Person:
    __belongs_to="Human Race"
    def __init__(self,name):
        self.__height=4.2
        self.__weight=54
        self.__name=name
    def can_speak(self,language):
        print(self.__name,"Can speak
```

Choose the correct option with respect to the type

Warning

This operation is disabled.

Ok

- (●) The class Person has 3 instance variables, 2
- ( ) The class Person has 3 instance variables, 1 local variable and 1 static variable
- ( ) The class Person has 2 instance variables, 3 local variables and 1 static variable
- ( ) The class Person has 4 instance variables, 2 local variables and no static variable

Identify the Python code to be written in lines Line 1, Line 2 and Line 3 so as to get the below output.
Emp id : 100
Emp id : 101
Emp Count 2

```
class Employee:
    __count=100
    def __init__(self,name):
        self.name=name
        #Line  1
        #Line  2

    @staticmethod
    def totalcount():
        #Line  3

emp1=Employee("Jack")
print("Emp id :",emp1.id)
emp2=Employee("Tom")
print("Emp id :",emp2.id)
Employee.totalcount()
```

Note: Line numbers are for reference only.

A) Line1: Employee.__count=100
Line2: self.id=Employee.__count+1
Line3: print("Emp Count",Employee.__count-100)

B) Line1: self.id=Employee.__count
Line2: Employee.__count+=1
Line3: print("Emp Count",Employee.__count-100)

C) Line1: self.id=100
Line2: Employee.__count+=1
Line3: print("Emp Count",Employee.__count)

D) Line1: self.id=Employee.__count+1
Line2: Employee.__count+=1
Line3: print("Emp Count",Employee.__count)

- ◯ A
- ◉ B
- ◯ C
- ◯ D

Consider the Python code given below.

```
class ClassA:
    def __init__(self):
        self.__var_one=100

    def method_one(self):
        return self.__var_one

class ClassB(ClassA):

    def __init__(self,var_two):
        #Line 1 _____
        self.__var_two=var_two

    def method_two(self):
        final_val=self.__var_two + self.me
        return final_val

bobj=ClassB(50)
print(bobj.method_two())
```

What changes should be done in the above code so as to g
Note: Line numbers are for reference only

Warning

This operation is

- (●) At Line 1 add: super().__init__()
- ( ) At Line 1 add: super().__init__(var_two)
- ( ) No need to make any change in the code, it will produce 150 as output
- ( ) No need to add anything at Line1.
  Change Line2 as: final_val=self.__var_two + super().method_one(