JBK3005-Python Assignment of Class

Example : 1

```
balance = 2000
def deposit(amount):
    global balance
    balance += amount
    return balance
def withdraw(amount):
    global balance
    balance -= amount
    return balance
print("After Deposited Amounta" ,deposit(100))
print("After Deposited Amount" ,deposit(100))
print("After Withdrowed Amount" ,withdraw(10))
print("After Withdrowed Amount", withdraw(10))
```

Example : 2

```
def scope_test():
    def do_local():
        spam = "local spam"
    def do_nonlocal():
        nonlocal spam
        spam = "nonlocal spam"

    def do_global():
        global spam
        spam = "global spam"

    spam = "test spam"
```

```
    do_local()
    print("After local assignment:", spam)
    do_nonlocal()
    print("After nonlocal assignment:", spam)
    do_global()
    print("After global assignment:", spam)

scope_test()
print("In global scope:", spam)
```

Example : 3

```
class A:
    def f(self):
        return self.g()

    def g(self):
        return 'A'

class B(A):
    def g(self):
        return 'B'

a = A()
b = B()
print(a.f(), b.f())
print(a.g(), b.g())
```

Example : 4

1]class Dog:

```
    tricks = []
    def __init__(self, name):
        self.name = name

    def add_trick(self, trick):
        self.tricks.append(trick)

d = Dog('Fido')
e = Dog('Buddy')
d.add_trick('roll over')
e.add_trick('play dead')
print(d.tricks)
```

2]

```
class Dog:
    def __init__(self, name):
        self.name = name
        self.tricks = []    # creates a new empty list for each dog

    def add_trick(self, trick):
        self.tricks.append(trick)

d = Dog('Fido')
e = Dog('Buddy')
d.add_trick('roll over')
e.add_trick('play dead')
print(d.tricks)
print(e.tricks)
```

Example : 5

```
class Employee:
```

```
  empCount = 0

  def __init__(self, name, salary):
    self.name = name
    self.salary = salary
    Employee.empCount += 1

  def displayCount(self):
   print("Total Employee %d" %
Employee.empCount)

  def displayEmployee(self):
    print("Name : ", self.name,  ", Salary: ", self.salary)

"This would create first object of Employee class"
emp1 = Employee("Zara", 2000)
"This would create second object of Employee class"
emp2 = Employee("Manni", 5000)
emp1.displayEmployee()
emp2.displayEmployee()
print("Total Employee %d" % Employee.empCount)
```

Example : 6 Built-In Class Attributes

```
class Employee:
  empCount = 0

  def __init__(self, name, salary):
    self.name = name
    self.salary = salary
    Employee.empCount += 1
```

```
  def displayCount(self):
    print("Total Employee %d" %
Employee.empCount)

  def displayEmployee(self):
     print("Name : ", self.name,  ", Salary: ", self.salary)

print("Employee.__doc__:", Employee.__doc__)
print("Employee.__name__:", Employee.__name__)
print("Employee.__module__:",
Employee.__module__)
print("Employee.__bases__:", Employee.__bases__)
print("Employee.__dict__:", Employee.__dict__)
```

Example : 7

```
class Point:
  def __init__( self, x=0, y=0):
     self.x = x
     self.y = y
  def __del__(self):
     class_name = self.__class__.__name__
     print(class_name, "destroyed")

pt1 = Point()
pt2 = pt1
pt3 = pt1
print(id(pt1), id(pt2), id(pt3))   # prints the ids of
the obejcts
del pt1
```

```
del pt2
del pt3
print(id(pt1), id(pt2), id(pt3))   #objects destroyed
so this line throw exception
```

Example : 8

1]

```
class MyClass:
    "This is my second class"
    a = 10
    def func(self):
        print('Hello')
print(MyClass.a)

print(MyClass.func)
print(MyClass.__doc__)
```

2]

```
class MyClass:
    "This is my second class"
    a = 10
    def func(self):
        print('Hello')


ob = MyClass()
print(MyClass.func)
print(ob.func)
ob.func()
```

Example : 9

```
class Parent:       # define parent class
```

```
  parentAttr = 100
  def __init__(self):
    print("Calling parent constructor")

  def parentMethod(self):
    print("Calling parent method")

  def setAttr(self, attr):
    Parent.parentAttr = attr

  def getAttr(self):
    print("Parent attribute :", Parent.parentAttr)

class Child(Parent): # define child class
  def __init__(self):
    print("Calling child constructor")

  def childMethod(self):
    print('Calling child method')

c = Child()         # instance of child
c.childMethod()     # child calls its method
c.parentMethod()    # calls parent's method
c.setAttr(200)      # again call parent's method
c.getAttr()         # again call parent's method
```

Example : 10

```
class BankAccount:
   def __init__(self):
      self.balance = 0
```

```
    def withdraw(self, amount):
        self.balance -= amount
        return self.balance

    def deposit(self, amount):
        self.balance += amount
        return self.balance
a = BankAccount()
b = BankAccount()
a.deposit(100)
b.deposit(50)
b.withdraw(10)
a.withdraw(10)

class MinimumBalanceAccount(BankAccount):
    def __init__(self, minimum_balance):
        BankAccount.__init__(self)
        self.minimum_balance = minimum_balance

    def withdraw(self, amount):
        if self.balance - amount < self.minimum_balance:
            print 'Sorry, minimum balance must be
maintained.'
        else:
            BankAccount.withdraw(self, amount)
```