Python

1. Find Output of following:

```
class Student:
        pass

s = Student()
s.name="Guido"
s.age=62
print(s.name)
print(s.age)
```

```
class Student:
        pass

s1 = Student()
s1.name="Guido"
s1.age=62
s2 = Student()
s2.name="Bjarne"
s2.age=67
print(s1.name, s1.age)
print(s2.name, s2.age)
```

2. For the **Student** class in above example, add **constructor** with 2 arguments for name and age, to set the **name** and **age** attributes. Create a student object, initialize it with some values and print its attributes.

3. Find Output Again:

```
class Test:
        def __init__(self):
                print("Constructor")
        def __del__(self):
                print("Destructor")
s1 = Test()
s2 = Test ()
```

```
class Test:
        def __init__(self):
                print("Constructor")
        def __del__(self):
                print("Destructor")
s1 = Test()
Test()
s2 = Test()
s3 = s1
del(s1)
```

4. Add a method **set_marks(marks_ list)**, that takes a list of marks in 5 subjects and stores in a new attribute **marks**. Also add a method **print_details()**, to student class to print **average** of marks and all details of student. (Hint : **average** will be calculated as **(total marks)/5**)
Test your class against the following code:

```
if __name__ == '__main__':
  s = Student('abc', 20)
  s.set_marks([80,60,90,70,99])
  s.print_details()
```

5. Find Output Once Again:

```
class Test:
        def __str__(self):
                print("I am a Good Student")
        def __repr__(self):
                print("I am Still Good Enough")
t = Test()
print(t)
print(str(t))
print(repr(t))
```

```
class Test:
        def __mul__(lhs, rhs):
                t=Test()
                t.val = lhs.val*rhs.val
t1 = Test()
t2 = Test()
t1.val = 10
t2.val = 30
t3 = t1*t2
print(t3.val, t2.val, t1.val)
```

6. Add **str** method to Student class in place of the **print_details** method, so that the student object can be converted directly to string and can also be printed on the screen.

7. Create a class Circle, that stores the radius and contains 2 methods:
get_area, get_perimeter, which give the area and perimeter respectively of the circle.

8. Create a class SelfManaged such that it keeps track of the number of objects currently alive.
Create a class method get_current_count(), that gives the number of objects currently alive in memory.
[Hint: use a class attribute to keep count of number of objects and use __init__ and __del__ methods to update the value of count count]

9. Add the multiplication operator overload to the Complex class.
Logic for Complex number multiplication is:
c1 = x + yi
c2 = p + qi
c1 * c2 = (x.p – y.q) + (x.q + p.y)i
Ex : (2+1i) * (3 + 5i) = 1 + 13i

10. Create a class BankAccount, which contains attributes balance and name, and methods deposit() and withdraw(), to add and deposit some money in account.
the balance should be set to 0 in the constructor, and withdrawal should be allowed only if sufficient balance is there. Also overload the str method to allow printing the details directly.