Lesson-12: Classes & Objects

Classes and Objects

Program 1:

```
class classy:

    def setValue(self):
        self.a = 6
        self.b = 7

    def getValue(self):
        return (self.a, self.b)

print("Hello")
c = classy( )
c.setValue( )
print(c.getValue( ))
```

Output:

```
Hello
(6, 7)
```

Program 2:

```
class classy:

    def setValue(self, a, b):
        self.a = a
        self.b = b

    def getValue(self):
        return (self.a, self.b)

print("Hello Main")

cobjone = classy( )
cobjtwo = classy( )
cobjone.setValue(5,6)
cobjtwo.setValue(7,9)
print(cobjone.getValue( ))
print(cobjtwo.getValue( ))
```

Output:

```
Hello Main
(5, 6)
(7, 9)
```

Program 3:

```
class classy:

    def __init__(self):
        self.a = 6
        self.b = 7

    def getValue(self):
        return (self.a, self.b)

print("Hello Main")

cobjone = classy( )
cobjtwo = classy( )
print(cobjone.getValue( ))
print(cobjtwo.getValue( ))
```

Output:

```
Hello Main
(6, 7)
(6,7)
```

Program 4:

```
class classy:

    def __init__(self, a, b):
        self.a = a
        self.b = b


    def getValue(self):
        return (self.a, self.b)

print("Hello Main")

cobjone = classy(5,6)
cobjtwo = classy(7,8)
print(cobjone.getValue())
print(cobjtwo.getValue())
```

Output:

```
Hello Main
(5, 6)
(7,8)
```

Program 5:

```
class classy:

    def __init__(self, a, b):
        self.a = a
        self.b = b

    def getValue(self):
        return (self.a, self.b)

print("Hello Main")

cobjone = classy(5,6)
cobjtwo = classy(7,8)

print( cobjone.a, cobjone.b )
print( cobjtwo.a, cobjtwo.b )
```

Output:

```
Hello Main
(5, 6)
(7,8)
```

Program 6:

```
class classy:

    def __init__(self, a, b):
        self.__a = a
        self.__b = b

print("Hello Main")

cobjone = classy(5,6)

print( cobjone.__a, cobjone.__b )
```

Output:

```
Hello Main
Traceback (most recent call last):
  File "classexamplefour.py", line
20, in <module>
    print( cobjone.__a,
cobjone.__b )
AttributeError: 'classy' object has
no attribute '__a'
```

Program 7:

```
class classy:

  def __init__(self, a, b):
    self.__a = a
    self.__b = b
    self.same( )

  def same(self):
    print("a = ", self.__a)
    print("b = ", self.__b)

print("Hello Main")

cobjone = classy(5,6)
```

Output:

```
Hello Main
a =  5
b =  6
```

Program 8:

```
class classy:

    def __init__(self, a, b):
        self.__a = a
        self.__b = b

    def getValue(self):
        return (self.a, self.b)

print("Hello Main")

cobjone = classy(5,6)

print( cobjone._classy__a, cobjone._classy__b )
```

Output:

```
Hello Main
5 6
```

Program 9:

```
class classy:

print("Hello main")
cobj = classy( )
print(cobj)
```

Output:

```
File "classexampleseven.py", line
4
  print("Hello main")
  ^
IndentationError: expected an
indented block
```

Program 10:

```
class classy:
    pass

print("Hello main")
cobj = classy( )
print(cobj)
```

Output:

```
Hello main
<__main__.classy object at
0x10d34ed60>
```

Program 11:

```
class classy:
    def __str__(self):
        return "I am a __str__ function"

    def __repr__(self):
        return "I am a __repr__ function"

print("Hello main")
cobj = classy()
print(cobj)
```

Output:

```
Hello main
I am a __str__ function
```

NOTE: The default version of **__str__** method calls the **__repr__** method.

Program 12:

```
class classy:

    def __repr__(self):
        return "I am a __repr__ function"

print("Hello main")
cobj = classy()
print(cobj)
```

Output:

```
Hello main
I am a __repr__ function
```

Just remember:

- The result of __str__ should be readable.
- The result of __repr__ should be unambiguous.
- Always add a __repr__ to your classes. The default implementation for __str__ just calls __repr__, so you get the best of both worlds.

The official Python documentation says __repr__ is used to find the "official" string representation of an object and __str__ is used to find the "informal" string representation of an object. The print statement and str() built-in function uses __str__ to display the string representation of the object while the repr() built-in function uses __repr__ to display the object. Let us take an example to understand what the two methods actually do.

```
>>> import datetime
>>> today = datetime.datetime.now( )
>>> str(today)                #internally calls __str__
'2018-01-12 09:21:58.130922'
>>> repr(today)
'datetime.datetime(2018, 1, 12, 9, 21, 58, 130922)'
```

Program 13:

```
class Person:

  def __init__(self, person_name, person_age):
    self.name = person_name
    self.age = person_age

  def __str__(self):
    return f'Person name is {self.name} and age is {self.age}'

  def __repr__(self):
    return f'Person(name={self.name}, age={self.age})'


p = Person('Subhash', 35)

print(p.__str__( ))
print(p.__repr__( ))
```

Output:

```
Person name is Subhash and age is
35
Person(name=Pankaj, age=35)
```

Program 14:

```
class classy:

    def __init__(self, a, b):
        self.__a = a
        self.__b = b

    def __str__(self):
        pass
        return "In __str__ " + str(self.__a) + " and " + str(self.__b)
    def __repr__(self):
        return "In __repr__ " + str(self.__a) + " and " +   str(self.__b)

print("Hello main")
cobj = classy(5,6)
print(cobj)
```

Output:

```
Hello main
In __str__ 5 and 6
```

Program 15:

```
class Angel:

    a = 10

    def imethod(self):
        self.a = 11

    def get(self):
        print(self.a)


a1 = Angel( )
a2 = Angel( )

a1.imethod( )
a1.get( )

a2.get( )
```

Output:

```
11
10
```

Program 16:

```
class Angel:

    a = 10

    @classmethod
    def cmethod(cls):
        cls.a = 11

    def get(self):
        print(self.a)


a1 = Angel( )
a2 = Angel( )

a1.cmethod( )
a1.get( )

a2.get( )
```

Output:

```
11
11
```

Output:

```
11
11
```

Program 17:

```
class Angel:

    a = 10

    @staticmethod
    def smethod(x):
        x = x * 4
        return x


a1 = Angel()
a2 = Angel()

res = a1.smethod(4)
print(res)

res = Angel.smethod(5)
print(res)
```

Output:

```
16
20
```

Program 18:

```
class Student:

    class dob:
        def __init__(self,date, month, year):
            self.date = date
            self.month = month
            self.year = year

        def __str__(self):
            return str(self.date) + "/" + str(self.month) + "/" + str(self.year)


    def __init__(self, name, age, d, m, y ):
        self.name = name
        self.age = age
        self.d_o_b = self.dob(d, m, y)

    def print_all(self):
        print(self.name)
        print(self.age)
        print(self.d_o_b)

s1 = Student("subhash", 34, 7, 6, 1985)
s1.print_all( )

s2 = Student("charan", 21, 4, 5, 1995)
s2.print_all( )
```

Output:

```
subhash
34
7/6/1985
charan
21
4/5/1995
```

Programming Assignments:

1. Implement Circle class and find out the area, perimeter and circumference of the circle.
2. Implement a TV class with volume up and down, channel up and down, power on and off, switch to a specific channel etc.