Python OO Practicals 1

Please download PyCharm IDE

1) Create an empty class named Actor.

```
class Actor:
    pass
```

a) Create an object of Actor (a1 = Actor())
b) Type a1.name = ‘Ranveer’ and print it to console. Do you see the name printed? How did this Actor get the name when we have an empty class definition?
c) Create another object, a2 = Actor() and directly print a2.name. What happens? So the learning is in Python, instance variables can be inj…?
d) Now add a behaviour to Actor called act() like this:

```
def act(self):
        print(‘Actor acting’)
```

Now invoke a1.act() and verify if this behaviour is getting invoked. Try calling Actor.act() and see if it works. Why not? Do you understand what is self? Change it to “me” and see if it works. Using self in every instance method is a Be...P…?

e) Now in act(), write this code -> self.name = ‘sonakshi’. After creating the object, invoke a1.act() & then print a1.name. Do you see the name being printed? Why? Is act() behavior the correct place to indicate that Actor has a name? Where should this be done then?
f) Create the init dunder function thus:

```
def __init__(self):
    self.name = ‘deepika’
    print(‘in init() of Actor’)
```

Now test when you create an object whether init() is called. A function is called a dunder / magic method cuz …?

g) Print self.name in act() as well. Create another object and invoke act(). What name is being printed? What should you do to ensure each Actor will have a different name that is given by the user?

```
def __init__(self, n):
        self.name = n
        print(‘in init() of Actor’)
```

Test:

```
a1 = Actor('Deepika')
a2 = Actor('Sonakshi')
a1.act()
a2.act()
```

This should have made you understand how to create instance variables, instance methods in a class definition.

h) Now print a1 as print(a1) and check the output.

2) Code the following in First.py:

```
y = 10
print('y outside = '+str(y))
def t():
   y = 5 # what variable is this?
   print('y inside t() = '+str(y))

def p():
   global y
   y = 15 # which value is getting changed?
   print('y inside p() after changing = ' + str(y))
t()
print('y outside = '+str(y))
p()
print('y outside = '+str(y))
```

Do you understand the scoping of variables? Try permutation, combination to see what you can access and what you cannot.

3) Create a Second.py file
Try to access y of First.py & invoke p() & t() functions as well.

Sample code:

```
import First

print(First.y)
```

Check when you import whether the code in First is getting executed. Why so? What is another way of importing where you do not need to use <module name>.<member name> to access. What happens if you have a y in Second module as well? Try it out.

3) A Pen has inkQty (int), colour (string) and can be used to write and refill. A text must be given for it to write. A quantity must be given to refill. If there is ink then the pen will write the text given to it (SOP). Refill works by taking in the int qty to add to the existing inkQty. First as a class designer, on paper apply OOAD and arrive at the class design. Then create the class implementation and create a tester class to create 2 pen objects, give it inkQty and ask it to write.

4) Create a Person class. Implement the functionality to count the number of objects created. How would you go about this?

5) There are TVs. A TV has a name and channel that is being displayed. You can increment/decrement channel. You can change the channel to a given number as well. You can ask the TV to display. When a TV is asked to display, it will print the channel num, the volume.TV has volume (int). You have to switch on the TV first before you can operate the channels or increase or decrease the volume. Design and test TV working.

6) There are Employees. An Employee has a name and salary (monthly). As Employee objects are created, you need to keep a track of the total salary that the company has to pay on a monthly basis and it should be returned when you invoke Employee.getTotalSalary() function. Employee can work. An emp must be given a Job to work. A Job has a description and effort as attributes. Accept Job as a parameter in work() and ensure that no other data typed args are accepted. How can you validate this?

In Employee..
```
def work(self, job):
        job.description = ‘change’
        job.effort = ‘20 days’
```

Now when you invoke e.work(job), print the description and effort before and after invoking work(). Do you understand why the state has changed? What is being passed as parameter to work() in the first place.

In Employee..
```
def work(self, job):
        job = Job()
        job.description = ‘this is something new’
        Job.effort = ‘1day’
```

Now test the same. What is the difference? By this, you can understand that job was passed by …? Now try doing the same for int parameters.

Example code (First.py):

```
class Person:
    '''this is the design of Person who has a name and age'''
    count = 0 # this is a class scoped variable

    def __init__(self,n,a): # this is the initializer / constructor
        print("in Persons init()")
        self.name = n # injecting the name attribute
        self.age = a # injecting the age attribute
        Person.count += 1

    def printName(self):
        print("I am a person with name "+self.name)

    def dance(self):
        if self.age <= 40: # state affects behaviour
            print(self.name+' doing break dance')
        else:
            print(self.name + ' doing naagin dance')

    def getOlder(self):
        print(self.name+' getting older')
        self.age += 1

    def test(): # class scoped method
        print('in Persons test() ')

print('Person.count = '+ str(Person.count))
p1 = Person('Ramu',20)
print('Person.count = '+ str(Person.count))
print('p1.name = '+p1.name+ ' p1.age = '+ str(p1.age))
p1.printName()
p2 = Person('Bheemu',60)
print('Person.count = '+ str(Person.count))
print('p2.name = '+p2.name+ ' p2.age = '+ str(p2.age))
#Person.printName(p2)
p2.printName()

p1.dance()
p2.dance()
```

```
p1.getOlder()
p2.getOlder()
print('p1.name = '+p1.name+ ' p1.age = '+ str(p1.age))
print('p2.name = '+p2.name+ ' p2.age = '+ str(p2.age))
```