Programming in Python

GAVRILUT DRAGOS

COURSE 9

Classes

Classes exists in Python but have a different understanding about their functionality than the way classes are defined in C-like languages. Classes can be defined using a special keyword: **class**

Python 2.x/3.x

```
class <name>:
      <statement_1>
       …
      <statement_n>
```

Where **statement$_i$** is usually a declaration of a method or data member.

Documentation for Python classes can be found on:

- Python 2: https://docs.python.org/2/tutorial/classes.html
- Python 3: https://docs.python.org/3/tutorial/classes.html

Classes

Defining a function within a class without having the first parameter **self** means that that function is a static function for that class.

Python 2.x/3.x

```
class Point:
        def __init__(self):
                self.x = 0
                self.y = 0
        def GetY():
                return self.y

p = Point()
print (p.GetY())
```

Execution error (GetY is static)

Python 2.x/3.x

```
class Point:
        def __init__(self):
                self.x = 0
                self.y = 0
        def GetY():
                print("Test")

Point.GetY()
```

Output

```
Python 2: Error
Python 3: will print "Test" on the screen
```

Classes

It is not required for two instances of the same class to have the same members. A class instance is more like a dictionary where each key represent either a member function or a data member

Python 2.x/3.x

```
class Point:
        def __init__(self):
                self.x = 0
                self.y = 0
p1 = Point()
p2 = Point()
p1.z = 10
print (p1.x,p1.y,p2.z)
```

Error during runtine. “p2” does not have a data member “z” (only “p1” has a data member “z”)

Classes

You can also delete a member of a class instance by using the keyword **del**.

Python 2.x/3.x

```
class Point:
        def __init__(self):
                self.x = 0
                self.y = 0
p = Point()
print (p.x,p.y)
p.x = 10
print (p.x,p.y)
del p.x
print (p.x,p.y)
```

"x" is no longer a member of p. Code will produce a runtine error.

Classes

If a class member is like a dictionary – what does this means in terms of POO concepts:

A. method overloading is NOT possible (it would mean to have multiple functions with the same key in a dictionary). You can however create one method with a lot of parameters with default values that can be used in the same way.

B. There are no private/protected attributes for data members in Python. This is not directly related to the similarity to a dictionary, but it is easier this way as all keys from a dictionary are accessible.

C. CAST-ing does not work in the same way as expected. Up-cast / Down-cast are usually done with specialized functions that create a new object

D. Polymorphism is implicit (basically all you need to have is some classes with some functions with the same name). Even if this supersedes the concept of polymorphism, you don't actually need to have classes that are derived from the same class to simulate a polymorphism mechanism.

Classes

The same can be applied for class methods – however in this case there are some restrictions related to the **self** keyword.

Python 2.x/3.x

```
class MyClass:
        x = 10
        y = 20
        def Test(self,value):
                return ((self.x+self.y)/2 == value)
        def MyFunction(self,v1,v2):
                return str(v1+v2)+" - "+str(self.x)+","+str(self.y)
m = MyClass()
print (m.Test(15),m.Test(16))
m.Test = m.MyFunction
print (m.Test(1,2))
```

Output

```
True False
3 - 10,20
```

Classes

The same can be applied for class methods – however in this case there are some restrictions related to the **self** keyword.

Python 2.x/3.x

```
class MyClass:
        x = 10
        y = 20
        def Test(self,value):
                return ((self.x+self.y)/2 == value)
        def MyFunction(self,v1,v2):
                return str(v1+v2
                                                    lf.y)
m = MyClass()
print (m.Test(15),m.Test(
m.Test = MyClass.MyFunction
print (m.Test(1,2))
```

Runtime error because “MyFunction” is a method that needs to be bound to an object instance !