Method overriding in Python ¶

What is overriding?

Overriding is the ability of a class to change the implementation of a method provided by one of its ancestors.

Overriding is a very important part of OOP since it is the feature that makes inheritance exploit its full power.Through method overriding a class may "copy" another class, avoiding duplicated code, and at the same time enhance or customize part of it. Method overriding is thus a strict part of the inheritance mechanism.

Method Overriding in action

In Python method overriding occurs simply defining in the child class a method with the same name of a method in the parent class. When you define a method in the object you make this latter able to satisfy that method call, so the implementations of its ancestors do not come in play.

Example:

In [1]:

```
class Parent(object):
    def __init__(self):
        self.value = 5

    def get_value(self):
        return self.value

class Child(Parent):
    def get_value(self):
        return self.value + 1
```

Now **Child objects** behave differently

In [2]:

```
c = Child()
c.get_value()
```

Out[2]:

6

and taking a look inside the class we spot a difference

In [3]:

```
Parent.__dict__
```

Out[3]:

```
mappingproxy({'__dict__': <attribute '__dict__' of 'Parent' objects>,
              '__doc__': None,
              '__init__': <function __main__.Parent.__init__>,
              '__module__': '__main__',
              '__weakref__': <attribute '__weakref__' of 'Parent' obje
cts>,
              'get_value': <function __main__.Parent.get_value>})
```

In [4]:

```
Child.__dict__
```

Out[4]:

```
mappingproxy({'__doc__': None,
              '__module__': '__main__',
              'get_value': <function __main__.Child.get_value>})
```

Since now the Child class actually contains a get_value() method with a different implementation (the id of the two functions are different).

- This is of uttermost importance in Python. Inheritance delegation occours automatically, but if a method is overridden the implementation of the ancestors is not considered at all. So, if you want to run the implementation of one or more of the ancestors of your class, you have to call them explicitly.