Python Classes

Khatereh Mohajery
Feb 02,2017

summary

- Introduction to Objects
- Introduction to Procedural Programming
- Introduction to Object Oriented Programming (OOP)
- Python classes
- Examples

What is an Object?

In programming, elements of programs are called objects

- In Python : strings, dictionaries, integers, lists, functions, …

This means they have certain things in common

Procedural Programming

Dividing program into reusable chunks called procedures of functions

- Maintains separation between your code and your data
- Makes it easier to visualize what your code is doing and to maintain your code
- Avoid repetition

Object Oriented Programming

Lots of operations are common to objects of the same type

For example:

- Standard operations on strings: making a lowercase or uppercase version, splitting. Properties of objects in python: ***methods***
- In Python there is a blueprint string object called the string type. Its actual name is str.

Classes

When we create our own blueprints, these are called classes.

- We can define our own class of object - and from this create as many instances of this class as we want.
- They will all have the methods (and other properties) from the blueprint - the class.

Example of Class in Python:

```
class lunch:

    def print_sth(self):

        print 'I am hungry'

x = lunch()

x.print_sth()
```

Initialization

Or Constructor :

A function that is called when the class is created and do all the setup work. For example default values:

```
class Car(object):

        def __init__(self, wheels= 4):
                self.wheels= wheels

mustang = Car()
print mustang.wheels
4

motor_cycle = Car(2)
print motor_cycle.wheels
2
```

Calling functions in init

```
class Car(object):

        def __init__(self, wheels= 4):
                self.wheels= wheels
                self.print_sth()

        def print_sth(self):
                print "sth"


mustang = Car()
Sth

print mustang.wheels()
4
```

Inheritance: *child* class derives the data and behavior of *parent* class

```
class Car(object):

    def __init__(self, wheels =4):
        """Return a new Car object."""
        self.wheels = wheels


    def sale_price(self):
        """Return the sale price for this car as a float
amount."""
        return 5000.0 * self.wheels

ford = Car(4, None)
print ford.sale_price()
```

20000

```
class Truck(object):

    def __init__(self, wheels=8):
        """Return a new Car object."""
        self.wheels = wheels


    def sale_price(self):
        """Return the sale price for this car as a float
amount."""
        return 4000.0 * self.wheels

GMC = Truck(8)
print GMC.sale_price()
```

32000

Using Class object Vehicle

```
class Vehicle(object):

    base_sale_price = 0

    def __init__(self, wheels):

        self.wheels = wheels


    def sale_price(self):

        return self.base_sale_price * self.wheels
```

```
class Car(Vehicle):

    def __init__(self, wheels=4):
        """Return a new Car object."""

        self.wheels = wheels
        self.base_sale_price = 5000

class Truck(Vehicle):

    def __init__(self, wheels=8):
        """Return a new Truck object."""

        self.wheels = wheels
        self.base_sale_price = 4000
```

Food for thought !!!!