CLASSES AND METHODS

The simplest class possible is shown in the following example (save as `oop_simplestclass.py`).

```
class Person:
    pass # An empty block


p = Person()
print(p)
```

Output:

```
$ python oop_simplestclass.py
<__main__.Person instance at 0x10171f518>
```

How It Works

We create a new class using the `class` statement and the name of the class. This is followed by an indented block of statements which form the body of the class. In this case, we have an empty block which is indicated using the `pass` statement.

Next, we create an object/instance of this class using the name of the class followed by a pair of parentheses. (We will learn more about instantiation in the next section). For our verification, we confirm the type of the variable by simply printing it. It tells us that we have an instance of the Person class in the *main* module.

Notice that the address of the computer memory where your object is stored is also printed. The address will have a different value on your computer since Python can store the object wherever it finds space.

Methods

We have already discussed that classes/objects can have methods just like functions except that we have an extra self variable. We will now see an example (save as oop_method.py).

```
class Person:
    def say_hi(self):
        print('Hello, how are you?')


p = Person()
p.say_hi()
# The previous 2 lines can also be written as
```

```
# Person().say_hi()
```

Output:

```
$ python oop_method.py
Hello, how are you?
```

How It Works

Here we see the `self` in action. Notice that the `say_hi` method takes no parameters but still has the `self` in the function definition.

Source: http://www.swaroopch.com/notes/python/