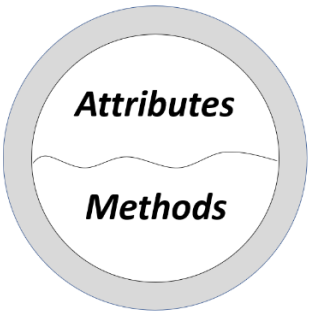
Objects in Python

We remember, that an object has Methods and Attributes:



- To add METHODS, you must add them to the CLASS definition.
- To add ATTRIBUTES, you typically add them to the CLASS definition, but it is possible to add them to the OBJECT definition also. You must add the values of the attributes to the OBJECT.

To declare a class in Python:

```
class <ClassName>:
    <Do stuff>
# END Class
```

For example:

```
class Point:
    pass
# END Class

p1 = Point()
p2 = Point()
```

To add attributes:

```
p1.x = 5
p1.y = 4
p2.x = 3
p2.y = 6
```

In Python the general form of declaring an attribute is as follows:

```
OBJECT.ATTRIBUTE = VALUE
```

We call this *dot notation*.

To add a method, we can simply put it in the class:

```
class Point:

    def reset(self):
            self.x = 0
            self.y = 0
    # END Reset

# END Class
```

In this case the use of the term `self` refers to the current instance of `Point`, so whichever object of `Point` you call the `reset` method with, is the `self`.

We can instantiate the class as follows:

```
p = Point()
p.x = 5
p.y = 4
```

We can call reset in two ways:

```
>> p.reset()
>> Point.reset(p)
```

Objects in Python

The Initialization Method

Sometimes when a developer creates a new class they forget to declare all of the attributes required, or forget to give those attributes a starting value ("initial value"). So Python has a special method called "`__init__()`" which forces the developer to make sure they declare and initialize all of the attributes that are required by each program. It can also be declared in such a way as to set default values for all attributes.

This is an Initialization method without any default values:	**This is an Initialization method out default values pre-set:**

`def __init__(self,x,y):`
`  self.move(x,y)`
`# END Init`
	`def __init__(self, x=0, y=0):`
	`  self.move(x,y)`
	`# END Init`

The Initialization method should be used in every class unless there is a very good reason.

Docstrings

One of the key motivations behind the object-oriented paradigm is the idea of "code reuse", so if you (or someone else) have already written a method or class, if it is at all possible you should not rewrite the code again. Good documentation and labelling are an important element of making the purpose and intent of programs clear. One feature that Python provides to make this easier is the *Doscstring* feature, where you can add descriptive sentence to each class and method, as the first line of that class or method.

```
class Point:
    "Represents a point in 2D space"

    def move(self,a,b):
    'Move the point to a new location'
        self.x = a
        self.y = b
    # END Move

# END Class
```

In the Python shell, all you need to do is type "help" with the name of the class in brackets, and you will see all the *Docstrings*.

```
>>> help (Point)

Help on class Point in module __main__:
class Point(builtins.object)
 |  Represents a point in 2D space
 |
 |  move(self, a, b)
 |      Move the point to a new location
 |  ----------------------------------------
```

So the purpose of all of the methods is made clear using the HELP command.