Introduction to Python

Objects 1: Classes and Instances

What is an Object?

An object is a single conceptual unit in a program that has

- data stored in **fields**
- functionality defined by functions called **methods**

Example

- We have already seen plenty of objects. Almost everything in Python is an object.

```
x = [] #x is a list object
y = 2 # y is an int object
x.append(y) # append is a method on list objects
```

Notice the '.' syntax. This is how you call methods and refer to fields.

Classes

- To define an object we have to decide what fields and methods it should have (although they can be changed later).

Classes

- To define an object we have to decide what fields and methods it should have (although they can be changed later).
- We do this with a 'blueprint' called a **class**.

Classes

- To define an object we have to decide what fields and methods it should have (although they can be changed later).
- We do this with a 'blueprint' called a **class**.

```
class myNewClass():
    def printMe(self): #Ignore self for now
        print "HI!"
```

Instances

- When we actually want to use the class we create an **instance** of it.

Instances

- When we actually want to use the class we create an **instance** of it.

```
class myNewClass():
    def printMe(self):
        print "HI!"

x = myNewClass()
x.printMe() # prints "HI"
```

Constructors

- More complicated classes need to be set up - for example with data only available at run time.

Constructors

- More complicated classes need to be set up - for example with data only available at run time.
- When you call `classname()` Python looks for a **constructor** with a special name `__init__`.

Constructors

- More complicated classes need to be set up - for example with data only available at run time.
- When you call `classname()` Python looks for a **constructor** with a special name `__init__`.

```
class myNewClass():
    def __init__(self, name):
        self.name = name
    def printMe(self):
        print "I'm" + name

x = myNewClass("Chris")
y = myNewClass("Bob")
x.printMe() # prints "I'm Chris"
y.printMe() #prints "I'm Bob"
```

self

- `self` is how you can refer to an instance from the class definition.

self

- `self` is how you can refer to an instance from the class definition.
- The method call `x.method()` passes the object `x` as the first parameter to method.

self

- `self` is how you can refer to an instance from the class definition.
- The method call `x.method()` passes the object `x` as the first parameter to method.
- All your methods should have `self` as their first parameter.

Demo

Demo Time!