Print representation of an object

- Left to its own devices, Python uses a unique but uninformative print presentation for an object

```
>>> print c
<__main__.Coordinate object at 0x7fa918510488>
```

Print representation of an object

- Left to its own devices, Python uses a unique but uninformative print presentation for an object

```
>>> print c
<__main__.Coordinate object at 0x7fa918510488>
```

- One can define a `__str__` method for a class, which Python will call when it needs a string to print. This method will be called with the object as the first argument and should return a `str`.

```
class Coordinate(object):
    def __init__(self, x, y):
        self.x = x
        self.y = y
    def __str__(self):
        return "<"+self.x+","+self.y+">"
```

Print representation of an object

- Left to its own devices, Python uses a unique but uninformative print presentation for an object

```
>>> print c
<__main__.Coordinate object at 0x7fa918510488>
```

- One can define a `__str__` method for a class, which Python will call when it needs a string to print. This method will be called with the object as the first argument and should return a `str`.

```
class Coordinate(object):
    def __init__(self, x, y):
        self.x = x
        self.y = y
    def __str__(self):
        return "<"+self.x+","+self.y+">"
>>> print c
<3,4>
```

Type of an Object

- We can ask for the type of an object

```
>>> print type(c)
 <class __main__.Coordinate>
```

- This makes sense since

```
>>> print Coordinate, type(Coordinate)
<class __main__.Coordinate> <type ‘type’>
```

Type of an Object

- We can ask for the type of an object

```
>>> print type(c)
 <class __main__.Coordinate>
```

- This makes sense since

```
>>> print Coordinate, type(Coordinate)
<class __main__.Coordinate> <type ‘type’>
```

- Use `isinstance()` to check if an object is a Coordinate

```
>>> print isinstance(c, Coordinate)
True
```

Adding other methods

- Can add our own methods, not just change built-in ones

```
class Coordinate(object):
    def __init__(self, x, y):
        self.x = x
        self.y = y
    def __str__(self):
        return "<"+self.x+","+self.y+">"
    def distance(self,other):
        return math.sqrt(sq(self.x - other.x)
                                 + sq(self.y - 
other.y))
```