Objects and equality

Python has 2 different operators to check equality: **is** and **==**

**is** checks “pointer equality”

**==** checks “value equality”
Equality and immutability

If 2 variables are set to be the same immutable object (e.g. a string), `is` and `==` are generally both True

```python
>>> a = 'abc'
>>> b = 'abc'
>>> a is b
True
>>> a == b
True
```
>>> a = ‘abc’
>>> b = ‘abc’

Python recognizes in some cases when it is possible to re-use the same immutable object.
Equality and mutability

If 2 variables are set to be mutable objects with the same contents, (such as lists), `is` and `==` are (possibly) different

```python
>>> a = [1, 2, 3]
>>> b = [1, 2, 3]
>>> a is b
False
>>> a == b
True
```
>>> a = [1, 2, 3]
>>> b = [1, 2, 3]
>>> a = [1, 2]
>>> b = [1, 2]
>>> c = b

```
>>> a
[1,2]
>>> b
[1,2]
>>> c
[1,2]
```
>>> a is b
False
>>> a == b
True

>>> b is c
True
>>> b == c
True
>>> a = [1, 2]
>>> b = [1, 2]
False
>>> c = b
>>> a == b
False
>>> c.append(3)
False
>>> a = [1, 2]

>>> b = [1, 2]

>>> c = b

>>> b is c
True

>>> c.append(3)

>>> b == c
True
More detailed diagram

```
list
[ [ 0, 0 ], [ 0, 0 ] ]

list
[ [ 0, 0, 0 ], [ 0, 0, 0 ] ]

int
1

int
2

int
3
```
Equality and user-defined classes

By default, `is` and `==` are the same for user-defined classes. Both check “pointer” equality

```python
>>> p1 = Point(1,2)
>>> p1
Point(1, 2)
>>> p1 == p2
False
>>> p2 = Point(1,2)
>>> p2
Point(1, 2)
>>> p2 == p1
False
>>> p3 = p2
>>> p3
Point(1, 2)
>>> p2 is p3
True
>>> p1 is p2
False
```
```python
>>> p1 = Point(1,2)
>>> p2 = Point(1,2)
>>> p1 == p2
False
```

We can change this

If we want
Lists containing mutable objects

A list can contain anything

```python
>>> p1, p2 = Point(1,2), Point(3,4)
>>> x, y = [p1, p2], [p1, p2]
>>> x is y
False
>>> x == y
True
```
>>> p1 = Point(1,2)  >>> x = [p1, p2]
>>> p2 = Point(3,4)  >>> y = [p1, p2]
Lists containing mutable objects

On the other hand...

```python
>>> x = [Point(1,2), Point(3,4)]
>>> y = [Point(1,2), Point(3,4)]
>>> x is y
>>> False
>>> x == y
>>> False
```