BOOLEANS

Integer Subclass

The `bool` class

PEP 285

Python has a concrete `bool` class that is used to represent Boolean values.

However, the `bool` class is a subclass of the `int` class

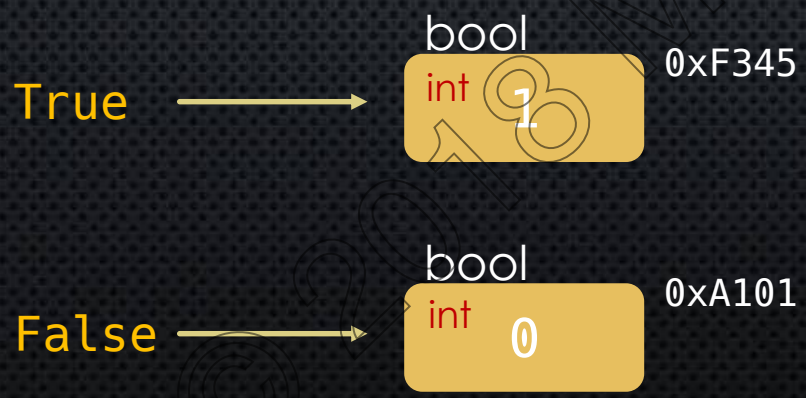
```
issubclass(bool, int)  → True
```

i.e. they posses all the properties and methods of integers, and add some specialized ones such as `and`, `or`, etc

Two constants are defined: `True` and `False`

```
isinstance(True, bool)  → True
isinstance(True, int)   → True
```

They are singleton objects of type `bool`



`is` vs `==`

Because `True` and `False` are singleton objects, they will always retain their same memory address throughout the lifetime of your application

So, comparisons of any Boolean expression to `True` and `False` can be performed using either the `is` (identity) operator, or the `==` (equality) operator

```
a == True     a is True
```

where `a` is a `bool` object

But since `bool` objects are also `int` objects, they can also be interpreted as the integers `1` and `0`

```
int(True) → 1     int(False) → 0
```

But: `True` and `1` are not the same objects — `id(True)` `id(1)`

`False` and `0` are not the same objects — `id(False)` `id(0)`

```
True is 1    → False
True == 1    → True
```

Booleans as Integers

This can lead to "strange" behavior you may not expect!

```
True > False  → True

(1 == 2) == False   → True        (1 == 2) == 0   → True
```

In fact, any integer operation will also work with booleans (//, %, etc)

```
True + True + True → 3      (True + True + True) % 2 → 1

-True → -1

100 * False → 0
```

The Boolean constructor

The Boolean constructor `bool(x)` returns `True` when `x` is `True`, and `False` when `x` is `False`

Wow, that sounds like a useless constructor! But not at all!

What really happens is that many classes contain a definition of how to cast instances of themselves to a Boolean – this is sometimes called the truth value (or truthyness) of an object *(upcoming video)*

Integers have a truth value defined according to this rule:

`bool(0)` → `False` (`0` is falsy)

`bool(x)` → `True for any int x  0` (x is truthy)

Examples

```
bool(0) → False

bool(1) → True
bool(100) → True
bool(-1) → True
```

Code