CSCI 1300

Intro to Computing

Gabe Johnson

Lecture 14 Feb 15, 2013

While Loops
Boolean Logic
Intro to Objects

Upcoming Homework Assignment

HW #3 **Due: Monday, Feb 18**

Dictionaries HW Pushed to Monday

The HW for the dictionaries assignment has been pushed to next Monday on account of (apparently) killer midterms in other classes.

The next homework will still be due on Friday, though.

Lecture Goals

1. While Loops
2. Boolean Arithmetic
3. Objects

Announcements

1. In Geany, there is an option in the menu to show whitespace and give you guides for where the code blocks should be.

This can help with indentation errors.

Announcements

2. HW 3 is pushed to Monday but HW 4 will still be due on Friday. Good luck on those midterms…

While Loops

while *condition*:
    <code>

This is sort of like a *for* loop, except we check for a condition at the start of each loop to see if we should continue. We'll run this <code> until *condition* is false.

Boolean Arithmetic: AND

AND	**TRUE**	**FALSE**
TRUE	TRUE	FALSE
FALSE	FALSE	FALSE

Boolean Arithmetic: OR

OR	**TRUE**	**FALSE**
TRUE	TRUE	TRUE
FALSE	TRUE	FALSE

Boolean Arithmetic

You've been doing this already for a while:

```
if something:
    do_something()
elif something_else:
    do_something_else()
```

Those 'something' parts are actually Truth statements. One way of getting them is via boolean arithmetic.

Boolean in Action

```
x = 4
y = 20

if x > 2 or y > 30:
    print “First condition”
else:
    print “Second condition”
```

Same thing using ‘and’

```
x = 4
y = 20

if x > 2 and y > 30:
    print “First condition”
else:
    print “Second condition”
```

Objects and Classes

We can create a *class* by using this syntax:

```
class Foo:
    <code>
```

Then we can create *instances* of this class like this:

```
my_foo = Foo()
```

‘Foo’ is a class, but ‘my_foo’ is an object.

Another class example

```
class Slug:
    hunger = 4

s1 = Slug() # make a slug
s2 = Slug() # make a different slug

# access the ‘member’ fields with a dot:
print “Slug number 1’s hunger level:”, s1.hunger
```

Objects are independent

Objects may be instances of the same class, but once we've created them their internal values are independent.

```
s1.hunger = 9
s2.hunger = 3
print "We just fed slug #1 but not the other."

print "Slug 1 has hunger:", s1.hunger
print "Slug 2 has hunger:", s2.hunger
```

The above tells us that slug 1's hunger s 9, but slug 2's hunger is 3. It is getting hungry :(