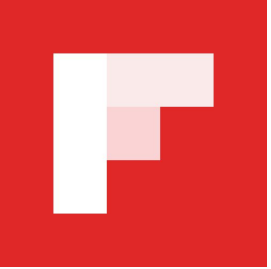

+

python™

Review of Basics & Control Flow

Variables, illustrated

Imagine we have a box. We can label that box and we can put something in a labeled box. This is how variables work.

```
x = 0

y = 0

x = 1

y = x

y = y + 1
```

Let's break into partners & work on "Practice Questions" (pg 28-29 or bottom of web page)

1. Operators: `*, -, /, +;` Values: `'hello', -88.8, 5`
2. `spam` is a variable name, `'spam'` is a string
3. string, integer, float
4. An expression is a programming instruction that consists of values and operators and evaluates into a single value.
5. An expression evaluates to a single value. A statement does not.
6. `bacon` still contains `20`, because the incremented value `20` was not reassigned to the variable named `bacon`
7. Both expressions evaluate to `'spamspamspam'`
8. Variable names cannot start with numbers, so `100` is invalid
9. `int(), float(), str()`
10. You cannot concatenate an int to a string without first casting the int to a string. This is a `ValueError`. So `'I have eaten ' + str(99) + ' burritos.'` works. Enclosing 99 in quotes would also work.

Introduction to Control Flow: Boolean Logic

A big part of programming is deciding whether or not to execute code based on a condition. For example, some logic we may have here at Flipboard:

If the user lives in Texas, show them a Whataburger ad. Otherwise if the user lives in California, show them an In-N-Out ad. If none of the previous conditions apply, show them a McDonald's ad.

Built into Python is another type called `bool`. A `bool` has two possible values: it can be `True` or it can be `False`.

Like the other types we've covered, `bool` can be used in expressions and stored in variables.

Acceptable uses:

```
True; False; rain = True; true = True;
```

Won't work:

```
True = True; false; true;
```

Comparison Operators

operator	meaning
==	Equal to
!=	Not equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to

As you can imagine, these operators resolve to boolean values depending on the two values being compared. So for an exercise, what would these values resolve to?

```
'Hello' == 'Hello'

False != False

27 < 21

23 >= 23

"Flipboard" == "flipboard"
```

Try this with variables too.

Boolean Operators

Boolean operators `and` & `or` take two boolean values and evaluate them together, providing an outcome based on the set of logical rules you see in these truth tables

a	b	a and b	a or b
True	True	True	True
True	False	False	True
False	True	False	True
False	False	False	False

Another boolean operator is called `not`. This modifies a boolean value to be the opposite. So for example if we have some variable:

```
enabled = True
```

```
not enabled # results in False
```

We can toggle between states with reassignment:

```
enabled = not enabled
```

Or save the result in another variable:

```
disabled = not enabled
```

Short exercise: Evaluate these expressions

```
(4 < 5) and (5 < 6)

(4 < 5) and (9 < 6)

(4 < 5) or (9 < 6)

(1 == 2) or (2 == 2)

2 + 2 == 4 and not 2 + 2 == 5

3 - 2 + 1 or 2 * 1 and 2
```

**Order of operations: comparison operators, then not, then and, then or