expressions, variables, and for loops (oh my!)

spam, spam, spam, spam
http://www.youtube.com/watch?v=anwy2MPT5RE

expressions

- for the most part, similar to Java
  - plus ** for exponentiation
- () before ** before * / % before + -

variables

```java
int x = 2;
x++;
System.out.println(x);

x = x * 8;
System.out.println(x);

double d = 3.2;
d = d / 2;
System.out.println(d);
```

vs.

```python
x = 2
x += 1
print x

x *= 8
print x

d = 3.2
d /= 2
print d
```

- no type is written when declaring
- use `+=` and `-=` instead of `++` and `--`

types

- python and Java use different names for some types
- use the `type` function to determine something's type

```
>>> type(42)
<type 'int'>
>>> type(3.14)
<type 'float'>
>>> type("spam")
<type 'str'>
```

python doesn't care about types

- don't need to specify type when declaring a variable
- variables can be reassigned to have a different type

python cares about types

- types still govern which operations are allowed

```
>>> "23" - 5
TypeError: unsupported operand type(s) for -: 'str' and 'int'
```

- everything still has a type

```
>>> n = 23
>>> type(n)
<type 'int'>
```

concatenation

```
>>> x = 4
>>> print "Thou shalt not count to " + x + "."
TypeError: cannot concatenate 'str' and 'int' objects
```

- solution: explicitly cast to `str`

```
>>> print "Thou shalt not count to " + str(x) + "."
Thou shalt not count to 4.
```

- alternatively...

print, revisited

```
# prints two values, separated by a space
print value1, value2
```

- can be used to solve our concatenation problem

```
>>> print "Thou shalt not count to ", x
Thou shalt not count to 4
>>> print x + 1, "is out of the question."
5 is out of the question.
```

for loops

```
for name in range(max):
    statement
    statement
    ...
    statement
```

- repeats statements, from `0` (inclusive) to *max* (exclusive)

for loops, continued

```
for name in range(min, max):
    statements

for name in range(min, max, step):
    statements
```

- can specify a `min` other than `0`, and a `step` other than 1
- counts from `min` (inclusive) to `max` (exclusive) in increments of `step`

string multiplication!

```
"yo " * 10
```

string multiplication

- can often replace nested loops

```
1 for (int line = 1; line <= 5; line++) {
2     for (int j = 1; j <= (5 - line); j++) {
3         System.out.print(".");
4     }
5     System.out.println(line);
6 }
```

vs.

```
1 for line in range(1, 6):
2     print (5 - line) * "." + str(line)
```

constants

- don't exist in python!
- instead, use a variable and pretend it can't be changed

```
1 NUM_FISHES = 5
2
3 def how_many_fishes():
4     print "there are", NUM_FISHES, "fishes."
```

getting `help`

- use the `help` function to learn about a type

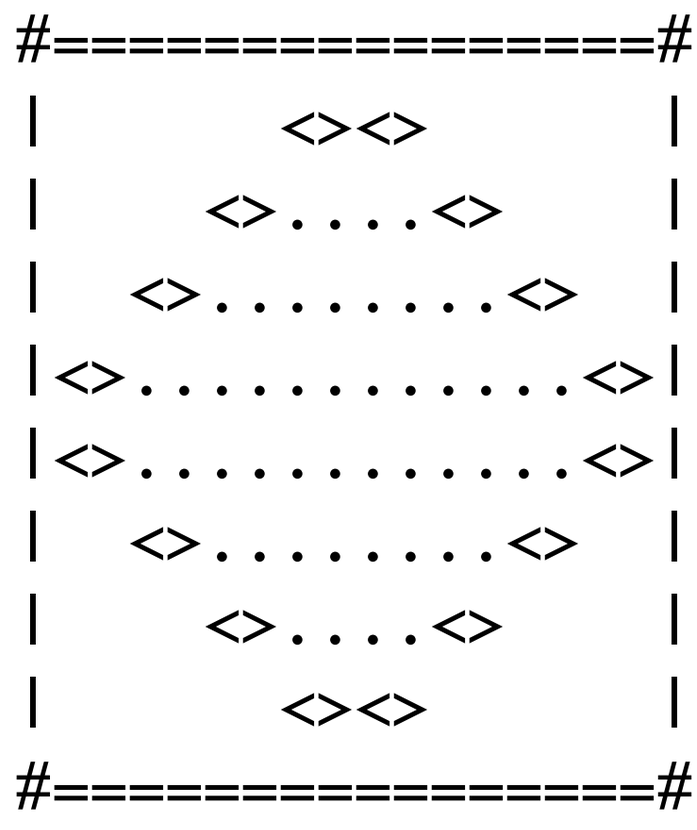
```
>>> help(str)
Help on class str in module __builtin__:

class str(basestring)
 |  str(object) -> string
 |
 |  Return a nice string representation of the object.
 |  If the argument is a string, the return value is the same object.

...
```

exercise

rewrite Mirror.java in python



(make sure your figure can be resized with a “constant”)

mirror.py

```
SIZE = 4

def bar():
    print "#" + 4 * SIZE * "=" + "#"

def top():
    for line in range(1, SIZE + 1):
        # split a long line by ending it with \
        print "|" + (-2 * line + 2 * SIZE) * " " + \
              "<>" + (4 * line - 4) * "." + "<>" + \
              (-2 * line + 2 * SIZE) * " " + "|"

def bottom():
    for line in range(SIZE, 0, -1):
        print "|" + (-2 * line + 2 * SIZE) * " " + \
              "<>" + (4 * line - 4) * "." + "<>" + \
              (-2 * line + 2 * SIZE) * " " + "|"

# main
bar()
top()
bottom()
bar()
```

range concatenation

- ranges can be concatenated with +
- can be used to loop over multiple ranges at once

```
>>> range(1, 5) + range(10, 15)
[1, 2, 3, 4, 10, 11, 12, 13, 14]

>>> for i in range(4) + range(10, 7, -1):
...     print i
0
1
2
3
10
9
8
```

mirror2.py

```
1 SIZE = 4
2
3 def bar():
4     print "#" + 4 * SIZE * "=" + "#"
5
6 def mirror():
7     for line in range(1, SIZE + 1) + range(SIZE, 0, -1):
8         print "|" + (-2 * line + 2 * SIZE) * " " + \
9               "<>" + (4 * line - 4) * "." + "<>" + \
10              (-2 * line + 2 * SIZE) * " " + "|"
11
12 # main
13 bar()
14 mirror()
15 bar()
```