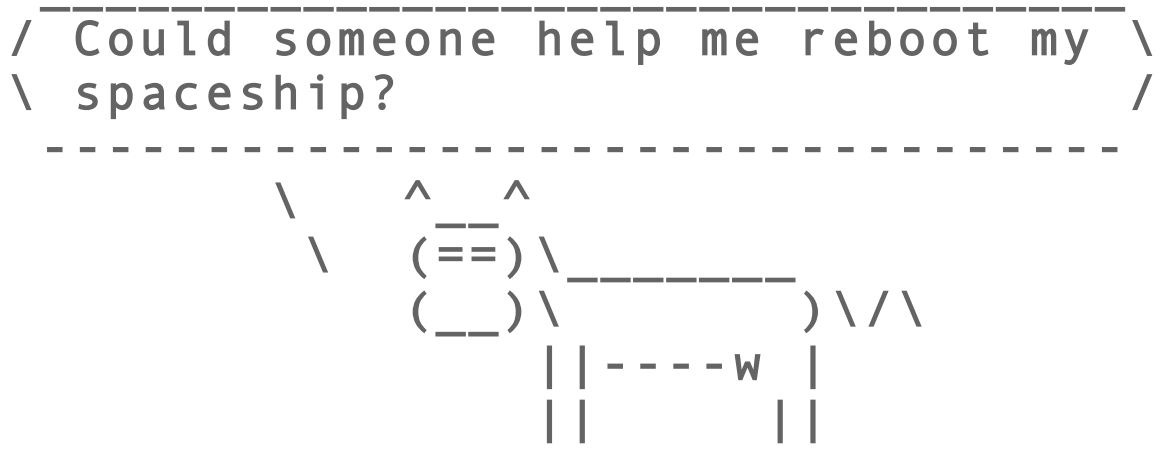
python™

10/9/07

```
 _______
< Moo? >
 -------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
```

>>> Reminders

* Code feedback for Pythony things only.
    * NOT logic errors that will also be in Java code.



>>> Last Time

* print
* escape sequences
* functions

```
>>> print "a string"
a string
>>> print 'a string'
a string
>>> print #a string#

>>> print $a string$
  File "<stdin>", line 1
    print $a string$
          ^
SyntaxError: invalid syntax
```

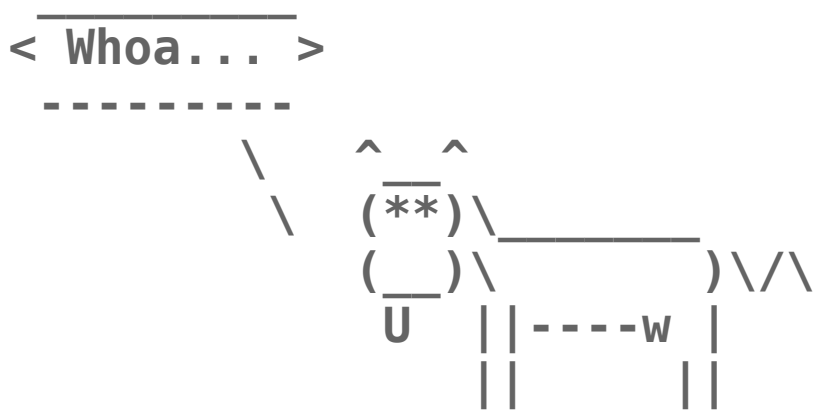
A whaa?

function – a subroutine independent of other code.

method – a subroutine associated with a class (think public class...) or object.

››› Overview

* types
* variables
* for loops
* 1337 ASCII art



>>> Types

Python cares very little about types. In Java, one must declare a variable with a particular type and maintain that type throughout the existance of that variable. In other words, ints can be only stored in places designated for ints, same for doubles etc. This is not the case in Python. Python does not care about types until the very last moment. This last moment is when values are used in certain ways, such as concatination.

	Java	python
178	int	int
175.0	double	float
“wow”	String	str
'w'	char	str
True	boolean	bool

```
>>> "Suppose " + 2 + " swallows carry it together."
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: cannot concatenate 'str' and 'int' objects
>>> "Suppose " + str(2) + " swallows carry it together."
'Suppose 2 swallows carry it together.'
```

>>> Expressions

Expressions.java

```
1  4 * 3/8 + 2.5 * 2
2  (2.5 + 3.5)/2
3  26 % 10 % 4 * 3
4  9/4 * 2.0 - 5/4
5  (5 * 7.0/2 - 2.5)/5 * 2
6  3 * 4 + 2 * 3
7  12/7 * 4.4 * 2/4
8  177 % 100 % 10/2
9  "hello 34 " + 2 * 4
10 9/2.0 + 7/3 - 3.0/2
11 "2 + 2 " + 3 + 4
12 813 % 100/3 + 2.4
13 3 + 4 + " 2 + 2"
14 27/2/2.0 * (4.3 + 1.7) - 8/3
15
16
```

expressions.py

```
1  4 * 3/8 + 2.5 * 2
2  (2.5 + 3.5)/2
3  26 % 10 % 4 * 3
4  9/4 * 2.0 - 5/4
5  (5 * 7.0/2 - 2.5)/5 * 2
6  3 * 4 + 2 * 3
7  12/7 * 4.4 * 2/4
8  177 % 100 % 10/2
9  "hello 34 " + str(2 * 4)
10 9/2.0 + 7/3 - 3.0/2
11 "2 + 2 " + str(3) + str(4)
12 813 % 100/3 + 2.4
13 str(3 + 4) + " 2 + 2"
14 27/2/2.0 * (4.3 + 1.7) - 8/3
15 “hello”*3+”wow”
16 “hello”*2+” world”
```

>>> Variables

As said earlier, Python cares less about types. This is the case when creating a variable. Unlike Java, Python does not care what type is in what variable. As a result, in Python to creating a variable is like Java without the type.

Variables.java

```
1  ...
2      int x = 2;
3      x++;
4      System.out.println(x);
5      x = x * 8;
6      System.out.println(x);
7
8      double d = 3.0;
9      d /= 2;
10     System.out.println(d);
11
```

expressions.py

```
1  x = 2
2  x += 1
3  print x
4  x = x * 8
5  print x
6
7  d = 3.0
8  d /= 2
9  print d
10
11 s = “wow”
12 print s
13
14
15
16
```

››› Constants

Continuing Python's free spirited ways, it has much less restrictions than Java. Because of this, constants are possible but not in a commonly used manner. Instead, we'll designate constants in Python solely by the variable capitalization.

constants.py

```
1  SIZE = 2
2  x = 10 * SIZE
3  print x
```

>>> Python's For

Unlike Java's for loop, Python's for loop loops over elements in a sequence. To loop over a certain sequence of integers use the range() function.

```
>>> range(4)
[0, 1, 2, 3]
>>> range(1,4)
[1, 2, 3]
>>> range(2,8,2)
[2, 4, 6]
>>> range(8,0,-2)
[8, 6, 4, 2]
```

for.py

```
1  for i in range(4): # (end)
2          print i
3
4  for i in range(1,4): # (start,end)
5          print i
6
7  for i in range(2,8,2): # (start,end,step_size)
8          print i
9
10 # for <var> in <sequence>:
11 #     <statements>
12
```

A whaa?

sequence – a series of elements.

Ex: [1,2,3,4,5] is a sequence produced by range(1,6)

Ex: [“milk”,”juice”,”bread”] is my grocery list

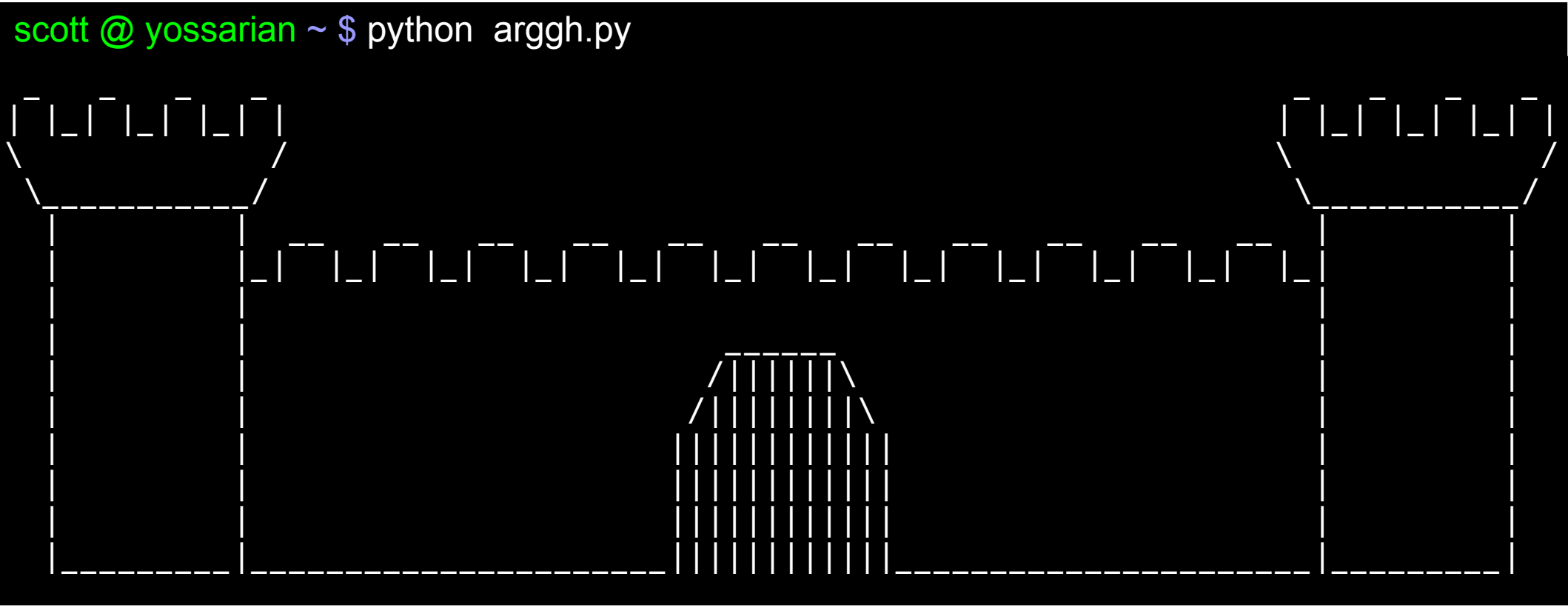
››› Mirror

```
scott @ yossarian ~ $ python  mirror.py
#================#
|       <><>     |
|     <>....<>   |
|   <>........<> |
|<>............<>|
|<>............<>|
|   <>........<> |
|     <>....<>   |
|       <><>     |
#================#
```


```
// Marty Stepp, CSE 142, Autumn 2007
// This program prints an ASCII text figure that
// looks like a mirror.
// This version uses a class constant to make the figure resizable.
public class Mirror2 {
    public static final int SIZE = 4;   // constant to change the figure size

    public static void main(String[] args) {
        line();
        topHalf();
        bottomHalf();
        line();
    }

    // Prints the top half of the rounded mirror.
    public static void topHalf() {
        for (int line = 1; line <= SIZE; line++) {
            // pipe
            System.out.print("|");

            // spaces
            for (int j = 1; j <= -2 * line + (2 * SIZE); j++) {
                System.out.print(" ");
            }

            // <>
            System.out.print("<>");

            // dots .
            for (int j = 1; j <= 4 * line - 4; j++) {
                System.out.print(".");
            }

            // <>
            System.out.print("<>");

            // spaces
            for (int j = 1; j <= -2 * line + (2 * SIZE); j++) {
                System.out.print(" ");
            }

            // pipe
            System.out.println("|");
        }
    }

    // Prints the bottom half of the rounded mirror.
    public static void bottomHalf() {
        for (int line = SIZE; line >= 1; line--) {
            // pipe
            System.out.print("|");

            // spaces
            for (int j = 1; j <= -2 * line + (2 * SIZE); j++) {
                System.out.print(" ");
            }

            // <>
            System.out.print("<>");

            // dots .
            for (int j = 1; j <= 4 * line - 4; j++) {
                System.out.print(".");
            }

            // <>
            System.out.print("<>");

            // spaces
            for (int j = 1; j <= -2 * line + (2 * SIZE); j++) {
                System.out.print(" ");
            }
```

››› Castle Arggh

```
scott @ yossarian ~ $ python  arggh.py
```



python™