python™

Week 2

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

>>> Last Time

* print
* escape sequences
* functions

```
>>> print "a string"
a string
>>> print 'a string'
a string
>>> print "A word", "and another "
A word and another
>>> print "Hello," + "world!"
Hello, world!
```

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


```
 _____________
< Whoa... >
 -------------
        \   ^__^
         \  (**)\_______
            (__)\       )\/\
             U  ||----w |
                ||     ||
```

>>> Types

Python cares very little about types. In Java, one must declare a variable with a particular type and maintain that type throughout the existence 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 concatenation of strings.

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

>>> String concatenation

Like Java, we can concatenate strings using a “+”. Unlike Java, when a number has to be concatenated with a string in python, you need to explicitly perform the conversion to a string using the str() function because it hasn’t made sure that the types match before run time.

```
>>> "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.'
```

>>> Python expressions

Python is very similar to Java in the way that it handles expressions such as:

- + - * / %
- Integer division – rounds down to nearest int
- Precedence – same rules
- Mixing types – numbers change to keep precision
- Real numbers are kept as “floats” or floating point numbers

>>> Differences in expressions

There are a few things that differ between Python and Java, such as:

You can multiply strings in python!

There are no increment operators in python (++, --) so we have to use -= and +=

```
>>> "Hello!"*3
'Hello!Hello!Hello!'
>>> x = 1
>>> x += 1
>>> print x
2
```

>>> Variables

As we said earlier, Python cares less about types. When we create a variable in Python, the type of the variable doesn't matter. As a result, in Python, creating a variable has the same syntax as setting a value to a variable.

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. We do need to write the constants at the top of our program so that every function can see them!

constants.py

```
1  SIZE = 2
2  x = 10 * SIZE
3  print x
4
5  # main
6  …
```

>>> 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. Later we will learn objects that we can use a for loop to go through all of the elements.

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 <name> in range([<min>,] <max> [,<step>]):
11  #     <statements>
12
```

>>> Complex Printing

Sometimes more complex output is needed.
To produce output but not go to the next line, just write a comma after the last quotes.
This adds whitespace, so sometimes you need “sys.stdout.write()” which just writes what is in the quotes. You also have to import “sys”!

Hello.java

```
1 System.out.print("Hello world! ")
2 System.out.print("This will all be")
3 System.out.println(" on the same line.")
4
```

hello2.py

```
1 import sys
2
3 sys.stdout.write("Hello world! ")
4 print "This will all be",
5 print " on the same line."
```

>>> Nested loops

In Python, a lot of the time we can do nested loops in a much more straightforward way using string multiplication.

```
    5
   44
  333
 2222
11111
```

Nested.java

```
1  for (int i = 1; i <= 5; i++) {
2  for (int j = 1; j <= (5 - i); j++) {
3          System.out.print(" ");
4  }
5  for (int k = 1; k <= i; k++) {
6          System.out.print(i);
7  }
8  System.out.println();
9  }
```

nested1.py

```
1  for i in range(5,0,-1):
2          print " " * (i-1) + str(i)*(6-i)
3
```

nested2.py

```
1  import sys
2          for i in range(5,0,-1):
3                  sys.stdout.write(" " * (i-1))
4                  sys.stdout.write(str(i)*(6-i))
5                  print
```

>>> 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(" ");
         }
```