Python For Loops

For loops allow iteration over an object a set number of times

- One of the key goals in programming is automating a mundane, repeatable task
  - For example, doing a t-test on a big data table between every third column and every fourth column
- This can be done with loops
  - Python has **for loops** and **while loops**
  - **For loops** loop for a specific number of times, **while loops** loop until a condition is met (could be infinite)

For loops will loop for a specific number of times, set by you

- For loops (any loops) will only operate on iterable data types, like strings and lists
  - It will loop through every element of a list
  - It will loop through every character of a string
- It will loop through the data the number of times you tell it to do so
  - In general (and by default) you usually loop over the size of the thing you're iterating over

```
# This is pseudocode, NOT Python

for component in iterable_data:
    new_variable = component
    new_variable.method('option A')
    some_function(new_variable)
    save_transformations(new_variable)
```

For loops will loop for a specific number of times, set by you

- In Python, for loops have a very set structure:
  - Loops begin with a **for** and the name you choose to give the subobject of the thing you're iterating over (this name will take every element of the list in turn or every character of the string in turn) and an **in** followed by the name of your array/string. The first line always has a **colon**.
  - Each line in the loop after that begins with an **indent**
  - **There is no need for an end statement, just stop indenting**

```
# This is pseudocode, NOT Python

for component in iterable_data:
    new_variable = component
    new_variable.method('option A')
    some_function(new_variable)
    save_transformations(new_variable)
```

An example of string vs. list iteration

```
>>> dna = 'ATTGTCCCCAGATTT'
>>> for nt in dna:
...     print nt
...
A
T
T
G
T
C
C
C
C
A
G
A
T
T
T
```

```
>>> >>> mydata = ['chr1', 10878,
10945, '<DEL>']
>>> for thing in mydata:
...     print thing
...
chr1
10878
10945
<DEL>
>>> for c in mydata[0]:
...     print c
...
c
h
r
1
>>>
```

For loops will loop for a specific number of times, set by you

- Loops allow you to do a task to every item of the string or array (today we focus on arrays)
- Range(beg, end, step) allows you to build lists of increasing or decreasing numbers easily, which is good for today's exercises
- Save your transformations by appending to a blank list

```
>>> range(2,10,2)
[2, 4, 6, 8]
>>> numlist = range(2,10,2)
>>> for num in numlist:
...     num = num*8
...     print num
...
16
32
48
64
```

```
>>> newnum = [ ]
>>> for num in numlist:
...     num = num*8
...     newnum.append(num)
…
>>> newnum
[16, 32, 48, 64]
>>> # this appending to an empty list allows you to save your information in a new list
```

Looping with indexes allows you to loop through many arrays at once

- For loops are not just limited to one array at a time
  - If many arrays have the same length, you can use the indexing we've learned to loop through all of them

```
>>> alist = [6, 3, 7, 4, 5, 2, 8]
>>> blist = [4, 8, 2, 6, 12, 10, 14]
>>> clist = []
>>> lena = len(alist)
>>> for n in range(lena):
...     c = alist[n] + blist[n]
...     clist.append(c)
...
>>> clist
[10, 11, 9, 10, 17, 12, 22]
```

```
>>> clist = []
>>> import math
>>> for n in range(lena):
...     c = math.log(blist[n]*1.0/alist[n])
...     clist.append(c)
...
>>> clist
[-0.4054651081081644,
0.980829253011726,
-1.252762968495368,
0.405465108108164,
0.8754687373538999,
1.6094379124341003,
0.5596157879354227]
```

Looping Exercises

```
>>> alist = [6, 3, 7, 4, 5, 2, 8]
>>> blist = [4, 8, 2, 6, 12, 10, 14]
```

```
>>> for num in alist:
...     print num
```

```
>>> for num in alist:
...     print alist
```

```
>>> for num in alist:
...     print num*1.0/blist[0]
```

```
>>> for num in alist:
...     total = total + num
...
>>> total
```

Looping Exercises

```
>>> alist = [6, 3, 7, 4, 5, 2, 8]
>>> blist = [4, 8, 2, 6, 12, 10, 14]
```

```
>>> for num in alist:
...     print num
6
3
7
4
5
2
8
>>> for num in alist:
...     print alist
[6, 3, 7, 4, 5, 2, 8]
[6, 3, 7, 4, 5, 2, 8]
[6, 3, 7, 4, 5, 2, 8]
[6, 3, 7, 4, 5, 2, 8]
[6, 3, 7, 4, 5, 2, 8]
[6, 3, 7, 4, 5, 2, 8]
[6, 3, 7, 4, 5, 2, 8]
```

```
>>> for num in alist:
...     print num*1.0/blist[0]
...
1.5
0.75
1.75
1.0
1.25
0.5
2.0
>>> for num in alist:
...     total = total + num
...
>>> total
35
```

More Looping Exercises

```
>>> alist = ['bear', 'cat', 'koala',
'deer', 'mouse', 'marmoset', 'dog']
>>> blist = [4, 8, 2, 6, 12, 10, 14]
```

```
>>> for mammal in alist[::-1]:
...     print mammal
...
```

```
>>> for num in alist:
...     print blist
```

```
>>> mydata = []
>>> for m in range(len(blist)):
...     newrow = alist[m] + ' ' + str(blist[m])
...     mydata.append(newrow)
...
>>> mydata
```

More Looping Exercises

```
>>> alist = ['bear', 'cat', 'koala',
'deer', 'mouse', 'marmoset', 'dog']
>>> blist = [4, 8, 2, 6, 12, 10, 14]
```

```
>>> for mammal in alist[::-1]:
...     print mammal
...
dog
marmoset
mouse
deer
koala
cat
bear
>>> for num in alist:
...     print blist
[4, 8, 2, 6, 12, 10, 14]
[4, 8, 2, 6, 12, 10, 14]
[4, 8, 2, 6, 12, 10, 14]
[4, 8, 2, 6, 12, 10, 14]
[4, 8, 2, 6, 12, 10, 14]
[4, 8, 2, 6, 12, 10, 14]
[4, 8, 2, 6, 12, 10, 14]
```

```
>>> mydata = []
>>> for m in range(len(blist)):
...     newrow = alist[m] + ' ' + str(blist[m])
...     mydata.append(newrow)
...
>>> mydata
['bear 4', 'cat 8', 'koala 2', 'deer 6', 'mouse 12',
'marmoset 10', 'dog 14']
```

Today's Problem Set

- Problem 1: Create a text file and save it in your myPython folder as ttest.py
  - On the first line, write **#!/usr/bin/env python**
  - Copy and paste the next page's data (representing 4 patients of paired data).
  - Create an empty array called **ttest**
  - Import the scipy.stats module
- Problem 2:
  - Using for loops (hint: with indexing) and the **scipy.stats.ttest_ind(array1, array2)** function, append the results of the t-tests for the 4 patients into your **ttest** array
  - (Save your file)

logcancer = [[-0.099, -0.218, -0.103, -0.096, 0.265, 0.259, 0.316, -0.149, -0.038,
-0.279, -0.117, 0.026, 0.332, -0.205, -0.3, -0.545, -0.915, 0.029, -0.077, 0.171, 0.406,
0.151, -0.009, -0.062, -0.376],
[-0.564, 0.138, -0.261, -0.192, 0.186, -0.407, 0.134, -0.325, -0.019, -0.816, -0.249,
-0.406, -0.237, -0.47, -0.424, -0.291, -0.394, -0.427, 0.064, 0.314, 0.042, -0.677,
-1.417, -1.244, -0.64, -0.646, -0.141, -0.257, -0.246, 0.304, 0.204, 0.217, 0.086,
-0.497],
[-0.007, 0.204, -0.174, -0.01, 0.196, 0.095, -0.183, -0.461, -0.264, -0.094, -0.332,
-0.244, -0.677, -0.56, -0.011, -0.343, -0.213, -0.606, -0.436, -0.513, -0.168, -0.106,
-0.974, -0.398, -0.506, -0.641, -0.626, -0.167, -0.503, -0.774, -0.425, -0.557, -0.541],
[0.123, -0.284, -0.703, -0.491, -0.928, 0.166, -0.097, -0.68, -0.729, -0.549, -0.9,
-0.544, -0.765, -0.239, -0.793, -0.139, -0.218, -0.279, -0.028, 0.184, -0.997, -0.371,
-0.118, -0.924, -0.445, -0.345, -0.636, -1.571, -0.523, -0.639, -0.305, -0.277, -0.691]]

lognormal = [[-0.406, -0.07, -0.135, -0.059, 0.325, -0.472, -0.857, 0.197, 0.329, 0.29,
0.05, -1.508, -0.602, 0.039, 0.428, 0.259, 0.345, -0.077, -1.18, -1.557, -0.804, -0.179,
-0.289, -0.988, -1.336, -0.465, 0.215, 0.172, 0.816, 0.88, 0.942, 0.53, 0.498, 0.351],
[-0.556, -1.241, -1.003, -0.696, -1.421, -0.042, 0.163, -0.022, 0.287, 0.536, -1.287,
-0.485, -0.539, -0.593, -1.814, -0.807, -0.224, -0.267, -1.24, 0.848, 0.709, 0.144,
0.064, -1.381, -1.718, -0.245, -0.624, -0.21, -0.384, -0.418, -0.321, -0.288, -0.191],
[-0.433, -0.085, 0.546, 0.103, 0.128, 0.3, 0.437, 0.544, 0.046, 0.79, 0.436, 0.253,
0.145, -0.505, -0.155, 0.355, 0.8, 0.273, 0.151, 0.25, 0.746, -0.487, -0.154, 0.12,
-0.816, 0.044, 0.218, 0.144, 0.024, -0.375, -0.101, -0.036, -0.087, -0.444, -0.324],
[-0.988, -1.336, -0.465, 0.215, 0.172, 0.816, 0.88, 0.942, 0.53, 0.498, 0.351, -0.396,
0.262, 0.784, 0.939, 1.143, 0.2, 0.351, 0.151, -0.548, 0.658, 0.615, -0.556, -1.241,
-1.003, -0.696, -1.421, -0.042, 0.163, -0.022, 0.287, 0.536, -1.287, -0.485, -0.539]]