08-Loops

Unknown Author

August 20, 2013

1 Python: Flow Control

Materials by: John Blischak and other Software Carpentry instructors (Joshua R. Smith, Milad Fatenejad, Katy Huff, Tommy Guy and many more)In this lesson we will cover howto automate repetitive tasks using loops.

2 Loops

Loops come in two flavors: `while` and `for`.

In [1]:
```
fruits = ['apples', 'oranges', 'pears', 'bananas']
i = 0
while i < len(fruits):
    print fruits[i]
    i = i + 1
```

```
apples
oranges
pears
bananas
```

In [2]:
```
for fruit in fruits:
    print fruit
```

```
apples
oranges
pears
bananas
```

In [3]:
```
# Use range for a range on integers
for i in range(len(fruits)):
    print i, fruits[i]
```

```
0 apples
1 oranges
2 pears
3 bananas
```

In [4]:
```
# Use zip to iterate over two lists at once
fruits = ['apples', 'oranges', 'pears', 'bananas']
prices = [0.49, 0.99, 1.49, 0.32]
for fruit, price in zip(fruits, prices):
    print fruit, "cost", price, "each"
```

```
apples cost 0.49 each
oranges cost 0.99 each
pears cost 1.49 each
bananas cost 0.32 each
```

In [5]:
```
# Use "items" to iterate over a dictionary
# Note the order is non-deterministic
prices = {'apples': 0.49, 'oranges': 0.99, 'pears': 1.49, 'bananas': 0.32}
for fruit, price in prices.items():
    print fruit, "cost", price, "each"
```

```
pears cost 1.49 each
apples cost 0.49 each
oranges cost 0.99 each
bananas cost 0.32 each
```

In [6]:
```
# Calculating a sum
values = [1254, 95818, 61813541, 1813, 4]
sum = 0
for x in values:
    sum = sum + x
sum
```

Out [6]:
```
61912430
```

2.1 Short Exercise

Using a loop, calculate the factorial of 42 (the product of all integers up to and including 42).

In [6]:
```

```

2.2 break, continue, and else

A break statement cuts off a loop from within an inner loop. It helps avoid infinite loops by cutting off loops when they're clearly going nowhere.

In [7]:
```
reasonable = 10
for n in range(1,2000):
    if n == reasonable :
        break
    print n
```

```
1
2
3
```

```
4
5
6
7
8
9
```

Something you might want to do instead of breaking is to continue to the next iteration of a loop, giving up on the current one.

In [8]:
```
reasonable = 10
for n in range(1,20):
    if n == reasonable :
      continue
    print n
```

```
1
2
3
4
5
6
7
8
9
11
12
13
14
15
16
17
18
19
```

What is the difference between the output of these two?

A slightly confusing bit of unusual syntax

Feel free to ignore this part!

Python allows you to use an `else` statement in a `for` loop. It will execute if you finish the loop without seeing a `break` statement. I have never once used this, but if you are curious, here is an example:

In [9]:
```
knights={"Sir Belvedere":"the Wise", "Sir Lancelot":"the Brave", \
        "Sir Galahad":"the Pure", "Sir Robin":"the Brave", "The Black Knight":"John Cl

favorites=knights.keys()
favorites.remove("Sir Robin")
for name, title in knights.iteritems() :
    string = name + ", "
    for fav in favorites :
        if fav == name :
            string += title
            break
    else:
        string += title + ", but not quite so brave as Sir Lancelot."
    print string
```

```
Sir Galahad, the Pure
Sir Belvedere, the Wise
Sir Lancelot, the Brave
The Black Knight, John Cleese
Sir Robin, the Brave, but not quite so brave as Sir Lancelot.
```

Working with files isn't covered until Lesson 12 in Codecademy. However, in the interest of getting to interesting ways to deal with data, we introduce the basics here!

3 Reading from a file

```
In [10]: less example.txt
```

```
In [11]: my_file = open("example.txt")
         for line in my_file:
             print line.strip()
         my_file.close()
```

```
This is line 1.
This is line 2.
This is line 3.
This is line 4.
This is line 5.
```

4 Writing to a file

```
In [12]: new_file = open("example2.txt", "w")
         dwight = ['bears', 'beets', 'Battlestar Galactica']
         for i in dwight:
             new_file.write(i + '\n')
         new_file.close()
```

```
In [13]: less example2.txt
```

5 Longer Exercise: Convert genotypes

If most of this material is brand new to you, your goal is to complete Part 1. If you are more experienced, please move on to Part 2 and Part 3. And don't forget to talk to your neighbor!

Motivation:

A biologist is interested in the genetic basis of height. She measures the heights of many subjects and sends off their DNA samples to a core for genotyping arrays. These arrays determine the DNA bases at the variable sites of the genome (known as single nucleotide polymorphisms, or SNPs). Since humans are diploid, i.e. have two of each chromosome, each data point will be two DNA bases corresponding to the two chromosomes in each individual. At each SNP, there will be only three possible genotypes, e.g. AA, AG, GG for an A/G SNP. In order to test the correlation between a SNP genotype and height, she wants to perform a regression with an additive genetic model. However, she cannot do this with the data in the current form. She needs to convert the genotypes, e.g. AA, AG, and GG, to the numbers 0, 1, and 2, respectively (in the example the number corresponds the number of G bases the person has at that SNP). Since she has too much data to do this manually, e.g. in Excel, she comes to you for ideas of how to efficiently transform the data.

5.1 Part 1:

Create a new list which has the converted genotype for each subject ('AA' -> 0, 'AG' -> 1, 'GG' -> 2).

In [14]:
```
genos = ['AA', 'GG', 'AG', 'AG', 'GG']
genos_new = []
# Use your knowledge of if/else statements and loop structures below.
```

Check your work:

In [15]:
```
genos_new == [0, 2, 1, 1, 2]
```

Out [15]:
```
False
```

5.2 Part 2:

Sometimes there are errors and the genotype cannot be determined. Adapt your code from above to deal with this problem (in this example missing data is assigned NA for "Not Available").

In [16]:
```
genos_w_missing = ['AA', 'NA', 'GG', 'AG', 'AG', 'GG', 'NA']
genos_w_missing_new = []
# The missing data should not be converted to a number, but remain 'NA' in the new lis
```

Check your work:

In [17]:
```
genos_w_missing_new == [0, 'NA', 2, 1, 1, 2, 'NA']
```

Out [17]:
```
False
```