Repeating with loops

Wrestling with Python



Overview

- The Story so far....
- The need for loops
- The while loop
 - Repeating code while a condition is true
- The for loop
 - Iterating through a collection

THE STORY SO FAR



What we can do so far

- Store data (using variables)
- Change data (using assignments)
- Make decisions (using conditions)
- There is not much more that we need to know how to do
 - But we do need to know how to create loops



Loops

- We create a loop so that we can repeat one or more statements
- A condition is used to determine whether or not the loop stops
- The condition is either true or false, just like that used in an if construction

THE WHILE LOOP



The While loop

- Sometimes you want a loop that will repeat while a condition is true
 - Read numbers from the user while they keep typing in ones that are not valid
- The Python language has a while construction that will do this for us
 - This repeats statements while a condition is true



A Stupid while Loop

• We can write never ending loops if we like:

```
while(True):
    print('hello')
```

• This loop will never finish (use CTRL+C to kill a program if you ever write this)..



A counting while loop

while continues while the condition is true

```
i=0
while(i<5):
    print('hello')
    i=i+1</pre>
```

- The end condition is tested each time round the loop and at the **start** of the loop
- We can use indenting to get more than one statement repeated



Reading in Numbers using While

```
width = -1
while(width < 0.5 or width > 10.0):
    widthString = input('Enter the width :')
    width = int(widthString)
print('Valid width entered: ',width)
```

- This will repeatedly read the width value until a valid one is entered
 - The condition is one we have seen before

THE FOR LOOP



For loops

- We have already seen how we can create code which will repeat something a particular number of times
- However, since this is something that we need to do a lot, Python provides a special constructions for this, the for loop



The For loop

The for loop has the following form:

```
for i in range (1,10):
    print(i)
```

- The variable i is given each value in the range
- The loop stops when i reaches the last value in the range (note that the value 10 is not in the range)



A working For Loop

```
for i in range (0,10):
print('Hello', i)
```



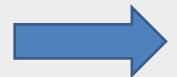
- This will print out Hello 10 times
- When the value in i reaches
 10 the loop stops

```
Hello 0
Hello 1
Hello 2
Hello 3
Hello 4
Hello 5
Hello 6
Hello 7
Hello 8
Hello 9
```



Iterating through other items

```
for i in 'Chicken':
   print('Hello', i)
```



Hello C
Hello i
Hello c
Hello k
Hello e
Hello n

- This will go round the loop once for each character in the string
- A string is a collection of items



For Loops in Python

- A For loop in Python is really something that works through a range of values
- That range can be created as a sequence of numbers (that is what we did first)
- Alternatively it can be a collection of items
 - A string can be regarded as a collection of characters



Breaking out of loop

```
for i in 'Chicken':
    print('Hello', i)
    if ( i=='k'):
        break
print('done')
Hello C
Hello h
Hello i
Hello c
Hello k
done
```

- You can use the break keyword to break out of a loop early
- This works in for loops or while loops



Iterating through other items

```
for i in 'Chicken':
    if ( i=='k'):
        continue
    print('Hello', i)
Hello C
Hello i
Hello c
Hello e
Hello n
```

- You can use the continue keyword to end an iteration early
 - In this case we don't print the 'k'
- This works in for loops or while loops



Summary

- We can now create constructions that can loop
- The while loop is controlled by a condition that can be true or false
- The for loop works on a collection of values
- We can break out of loops using break
- We can restart an iteration using continue