Loops in Python (part 1)

Loops (while)
Basics
Warnings
Uses

Loops allow us to carry out one or more statements repeatedly.

Syntax:
while <test>:
do ...
do ...

(http://www.greenteapress.com/thinkpython/html/thinkpython008.html)

While

Example
counter = 1
while counter <= 100:
    print(counter)
    counter += 1

▶ The test is executed first, and if it is positive (True), the body of the loop is executed
▶ If the test is False from the beginning, the body of the loop is not executed at all

Counting down

From Think Python (section 7.3):
For any value of n, this will terminate:
Example
while n > 0:
    print(n)
    n = n - 1

Flow of execution

From Think Python (section 7.3):
1. Evaluate the condition, yielding True or False.
2. If the condition is false, exit the while statement and continue execution at the next statement.
3. If the condition is true, execute the body and then go back to step 1.

Proving that the loop will terminate at some point is important ...

Termination

From Think Python (section 7.3):
Does this code terminate?
Example
while n != 1:
    print(n, end='·')
    if n%2 == 0:  # n is even
        n = n/2
    else:  # n is odd
        n = n*3+1

Note also the way we keep printing on the same line
While

Beware of infinite loops!
It is easy to create infinite loops with `while`. Make sure that your `while` condition will return false at some point.

Common errors

Wrong incrementing
If you want to decrement, make sure that you do not automatically increment

Example
counter = 100
while counter > 0:
    print(counter)
    counter += 1

Off-by-1 errors
Getting the beginning and the end of a loop right can be tricky: Should it start with 0 or 1? And remember that the control variable is incremented before the loop is finished.

Example
```python
print( 'multiples of 33' )
counter = 0
while counter <= 100:
    counter += 33
    print(counter)
print( 'This is the end: ' + str(counter) )
```

Use #1: iteration

As we've just seen, `while` loops can be used to iterate over a sequence.

- Commonly done by iterating over integers: integers easily count how many times you do something.
- You can change the way you iterate: `i += 2`, `i -= 1`, ...

Use #2: until

Another, subtly different use is to perform the same actions until a certain condition is reached.

Example
```python
user_input = ''
while len(user_input) < 3:
    user_input = input( 'Please enter a long string: ' )
print("Thank you for entering a long enough string!")
```
The break command allows you to stop a loop prematurely.

Example

```python
while True:
    word = input('Please enter a word: ')
    if not word: break
    print('The word was ' + word)
```

Note also the one-line if statement, allowed when the block of code (e.g., break) is one line long.