Lesson 5: for and while Loops

Fundamentals of Text Processing for Linguists
Na-Rae Han
Objectives

- Loops
  - for
  - while
for loop

- for x in SEQ :
  - iterates through a sequence (list, str, tuple) for doing something to each element

```python
>>> simpsons = ['Homer', 'Marge', 'Bart', 'Lisa', 'Maggie']
>>> for s in simpsons :
    print s, 'is a Simpson.'
Homer is a Simpson.
Marge is a Simpson.
Bart is a Simpson.
Lisa is a Simpson.
Maggie is a Simpson.
```

← Iterates through every element s of the simpsons list and prints the value of s followed by 'is a Simpson.'.
for loop examples

```python
>>> for l in list('victory'):
    print 'Give me a', l
Give me a v
Give me a i
Give me a c
Give me a t
Give me a o
Give me a r
Give me a y
>>> for l in 'victory':
    produces the exact same result. (Why?)
```
for loop examples

```python
>>> chom = 'Colorless green ideas sleep furiously'
>>> for w in chom.split() :
    print(w, 'is', len(w), 'characters long.
"Colorless" is 9 characters long.
"green" is 5 characters long.
"ideas" is 5 characters long.
"sleep" is 5 characters long.
"furiously" is 9 characters long.
```
for loop examples

```python
>>> chom = 'Colorless green ideas sleep furiously'
>>> for w in chom.split():
    print '"'+w+'"', 'is', len(w), 'characters long.'

"Colorless" is 9 characters long.
"green" is 5 characters long.
"ideas" is 5 characters long.
"sleep" is 5 characters long.
"furiously" is 9 characters long.
```
range() function, tooltips

>>> range(10)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> range(3, 10)
[3, 4, 5, 6, 7, 8, 9]
>>> range(3, 10, 2)
[3, 5, 7, 9]
>>> range(10, 0, -1)
[10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
>>> range([start[, stop[, step]]) -> list of integers
range() function, tooltips

```python
>>> range(10)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> range(3, 10)
[3, 4, 5, 6, 7, 8, 9]
>>> range(3, 10, 2)
[3, 5, 7, 9]
>>> range(10, 0, -1)
[10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
>>> range([start,] stop[, step]) -> list of integers
```

** obligatory argument

** optional arguments in square brackets []
What does this script do?

total = 0
for i in range(10):
    print i,
    total += i
print total

**comma ',' at the end of print suppresses line break**
What does this script do?

There's a built-in function for summing up a list of integers: `sum()`

```python
total = 0
for i in range(10):
    print(i, end=',')
    total += i
print(total)
```

```python
>>> sum(range(10))
45
>>> sum([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
45
```
Write a program that takes a sentence from keyboard input, and then prints out:

- each word and its length, and then
- the average word length.

```python
>>> ============================ RESTART ============================
>>> Give me a sentence: Colorless green ideas sleep furiously Colorless 9
green 5
ideas 5
sleep 5
furiously 9
The average word length of this sentence is 6.6
```
# word_average_length.py
# Demonstrates for loop and summation
#

sent = raw_input('Give me a sentence: ')

words = sent.split()  # tokenize sentence into word list

total = 0  # total is 0 at first
for w in words:  # for every word
    print w, len(w)  # print out word and its length
    total += len(w)  # add its length to total

# average word length is the total number of characters
# divided by the number of words
# turn total (45) into float (45.0) so avrg will also be float
avrg = float(total)/len(words)
print 'The average word length of this sentence is', avrg
We need a looping mechanism that keeps going until a certain condition holds:

- "Keep counting until 10"
- "Keep adding 5 as long as the sum is less than 100"
- "Keep trying until you have the correct answer"
while loop: count from 1 to 10

number = 1

number <= 10

False ✗ True ✔

print number
number += 1

print 'Done!'
Terminating a process

- aborting a process: **Ctrl+c**

- try this in IDLE shell:

  ```python
  >>> num = 0
  >>> while num < 10:
      print 'Hello!
    num += 1
  ```

- bad things can happen when you forget to include a **condition-changing operation** in the while loop.

  **Ctrl+c**:
  When things go haywire.
while loop example

- What's the output?

```
num = 5
while num <= 100 :
    print num
    num += 10
print 'Done!'
```

```
>>> 5
15
25
35
45
55
65
75
85
95
Done!
```

- What's the output?

```
num = 5
while num <= 100 :
    print num
    num += 10
print 'Done!'
```

```
>>> 15
25
35
45
55
65
75
85
95
105
Done!
```
Try it out

- What's the output?

```python
num = 5
while num <= 100 :
    print num
    num += 10
print 'Done!'
```

```
>>> 5
15
25
35
45
55
65
75
85
95
Done!
```

- What's the output?

```python
num = 5
while num <= 100 :
    num += 10
    print num
print 'Done!'
```

```
>>> 15
25
35
45
55
65
75
85
95
105
Done!
```
What does this script do?

```python
secret = 'panda'
g = raw_input('What\'s the secret animal? ')

if g == secret:
    print 'CORRECT!', secret, 'is the secret animal.'
else:
    print 'Wrong guess. Try again.'
```

Let's make it persistent: keep prompting (loop back !) until correct.

We'll look at 2 methods.
Method 1: which part in loop?

```python
secret = 'panda'
g = raw_input('What\'s the secret animal? ')

if g == secret :
    print 'CORRECT!', secret, 'is the secret animal.'
else :
    print 'Wrong guess. Try again.'
```

1. Figure out which part should be repeated.
2. Put the part in the while loop block.
Method 1: which part in loop?

secret = 'panda'

g = raw_input('What\'s the secret animal? ')

if g == secret :
    print 'CORRECT!', secret, 'is the secret animal.'
else :
    print 'Wrong guess. Try again.'

1. Figure out which part should be repeated.
2. Put the part in the while loop block.
Method 1: while ... ?

```python
secret = 'panda'

while g != secret :
    g = raw_input('What\'s the secret animal? ')
    if g == secret :
        print 'CORRECT!', secret, 'is the secret animal.'
    else :
        print 'Wrong guess. Try again.'
```

1. Figure out which part should be repeated.
2. Put the part in the while loop block.
3. What's the condition for looping back?
Method 1: set initial value for $g$

```python
secret = 'panda'
g = ''

while g != secret :
    g = raw_input('What\'s the secret animal? ')
    if g == secret :
        print 'CORRECT!', secret, 'is the secret animal.'
    else :
        print 'Wrong guess. Try again.'
```

1. Figure out which part should be repeated.
2. Put the part in the while loop block.
3. What's the condition for looping back?
4. Before looping begins, the initial value of $g$ must be set
secret = 'panda'
g = ''

while g != secret :
    g = raw_input('What\'s the secret animal? ')

if g == secret :
    print 'CORRECT!', secret, 'is the secret animal.'
else :
    print 'Wrong guess. Try again.'
Success!

>>> ================================ RESTART
>>> 
What's the secret animal? fox
Wrong guess. Try again.
What's the secret animal? dog
Wrong guess. Try again.
What's the secret animal? cat
Wrong guess. Try again.
What's the secret animal? panda
CORRECT! panda is the secret animal.
>>>
Method 2: using Boolean variable

```python
secret = 'panda'
correct = False

while not correct:
    g = raw_input('What\'s the secret animal? ')

    if g == secret:
        print 'CORRECT!', secret, 'is the secret animal.'
    else:
        print 'Wrong guess. Try again.'
```

A new True/False variable which functions as the loop condition
Method 2: using Boolean variable

```python
secret = 'panda'
correct = False

while not correct :
    g = raw_input('What\'s the secret animal? ')

    if g == secret :
        print 'CORRECT!', secret, 'is the secret animal.'
    else :
        print 'Wrong guess. Try again.'
```

But there's something wrong here. What is it?
Method 2: using Boolean variable

secret = 'panda'
correct = False

while not correct :
    g = raw_input('What\'s the secret animal? ')

    if g == secret :
        print 'CORRECT!', secret, 'is the secret animal.'
        correct = True
    else :
        print 'Wrong guess. Try again.'

Variable value must be explicitly set to True in loop body. Otherwise the loop will never terminate!
Method 2: try it out

secret = 'panda'
correct = False

while not correct:
    g = raw_input('What\'s the secret animal? ')

    if g == secret:
        print 'CORRECT!', secret, 'is the secret animal.'
        correct = True
    else:
        print 'Wrong guess. Try again.'
secret = 'panda'
correct = False

while not correct :
    g = raw_input('What\'s the secret animal? ')

    if g == secret :
        print 'CORRECT!', secret, 'is the secret animal.'
        correct = True
    else :
        print 'Wrong guess. Try again.'
secret = 'panda'
g = ''

while g != secret:
g = raw_input('What''s the secret animal? ')

if g == secret:
    print 'CORRECT!', secret, 'is the secret animal.'
else:
    print 'Wrong guess. Try again.'
Indenting and de-indenting a block

- Select your lines, and then use the commands for changing indentation levels:
  - Indent Region: Ctrl + ]
  - Dedent Region: Ctrl + [
Practice

- Write a script that prompts for a word, and then prints out the word-initial consonant cluster.

```python
What is your word? cheese
ch
```
Using while

```python
word = raw_input('What is your word? ')

i = 0  # starting with the first char of word

while i < len(word) and word[i] not in 'aeiou':
    i += 1

print word[:i]  # print prefix up to index i
```
Using while

```python
word = raw_input('What is your word? ')

i = 0  # starting with the first char of word
vfound = False  # initially, verb has not been found

while not vfound and i < len(word):  # i shouldn't go over
    if word[i] in 'aeiou':
        vfound = True
        print word[:i]  # print prefix up to index i
    i += 1
```
Using for and break

```python
word = raw_input('What is your word? ')

for i in range(len(word)):
    if word[i] in 'aeiou':
        print word[:i]
        break  # exit for loop with first vowel seen
```

**break**

terminates the innermost for or while loop

Without break, the for loop would have continued until it reached the end of the string.

Try removing break and give 'structure' as the word!
Defining your own function with def:

```python
>>> def vowelCount(wd):
    wd = wd.lower()
    cnt = wd.count('a') + wd.count('e') + wd.count('i') \
         + wd.count('o') + wd.count('u')
    return cnt

>>> vowelCount('queue')
4
>>> vowelCount('sly')
0
>>> vowelCount('Hello, world!')
3
>>> vowelCount('Animal')
3
```
Wrap-up

Next class
- Loop examples, dictionary
- Defining your own function
- How to use help and learn on your own

Exercise #4
- [http://www.pitt.edu/~naraehan/ling1901/exercise.html#ex4](http://www.pitt.edu/~naraehan/ling1901/exercise.html#ex4)
- Due Tuesday midnight