Introduction

Ashwini Vaidya

February 25, 2017

0.1 Why learn Python?

- Because languages are interesting! Python is not a natural language, but one that does have its own style, grammar (syntax) and idioms
- It is a gateway to solving text processing problems independently, building your own systems or applications
- Gives access to a host of very useful libraries: for natural language processing, data visualization, machine learning
- No braces, ever!

```
>>> from __future__ import braces
```

0.2 What will we try to learn in this course?

- Cover the most important concepts for programming in python
- With a bias with respect to text processing & relevancy to linguists
- Thinking through a problem computationally
- Learn to write code, that's readable ("Readability Counts" is one of the maxims of Python)

0.3 Which version?

Python 2.7

0.4 References

Various books and online tutorials, that I will point to during the course of the class. Please look at the wealth of guides and examples online for Python, there are many of them!

0.5 Installing Python

- https://conda.io/miniconda.html
- Choose the installation that is required for your OS
- Quick install guide
- https://conda.io/docs/install/quick.html

0.6 Approach

- Analyze a program:- the way we analyze a text:
- Understand the individual components
- Discover the rules of their syntax
- Update and improve the working code with examples

Goals

- Ability to write short scripts to solve commonly encountered linguistic problems
- Readability of code important (efficiency will be improved later)
- Be able to find solutions on your own to solve more complex problems

1 Things to remember

Programming can be improved by trying, getting stuck, and trying again. Getting stuck is the most important part of learning how to program.

There are four important things to remember about every program

- Assignment (giving a name to a particular value)
- Loops (repeat the same act a certain number of times)
- Input and output (giving input to the program (data,files), getting the output (results, or other files)
- Control structures (statements that evaluate conditions)

Before beginning to write programs, it's very important to not think about *syntax*. It's like thinking about grammar when trying to speak fluently in a foreign language–disaster. First plan at an abstract level, the most important things that are required for the program to work. Then, understand the flow of the program from start to finish. If it makes sense, then only begin writing it out.

2 Setting things up

Go to the terminal and first locate where you are using `pwd`.
Make a directory where you will be working

```
pwd
mkdir pythonlinguists
cd pythonlinguists
python <name of program>
python simple_word.py
```

In the program below, pay attention to whitespace! A proper and consistent handling of whitespace is essential for writing good programs. The style guide advises the following:

- 4 spaces per indentation level
- No hard tabs (Don't use the tab key)
- Never mix tabs and spaces. (editors help here)

```
simple_word.py
#!/usr/bin/python

#This program is taken from Dirk Hovy's Python for Linguists tutorial
#https://github.com/dirkhovy/python_for_linguists

#strings, enclosed in double quotes (Single quotes are also allowed)
sentence1 = "Python is named after Monty Python"
sentence2 = "It's clean and readable"
sentence3 = "We are going to learn Python"

#create a list
mini_text = [sentence1, sentence2, sentence3]


##Loop begins
for current_sentence in mini_text:
    number_of_characters = 0
    # go through each character in the string
    for character in current_sentence: # increase the character count by 1
        number_of_characters = number_of_characters + 1
        number_of_words = 0

    #Another loop begins
    # split the sentence at spaces into a list and look at each element in turn
    for word in current_sentence.split():
        number_of_words += 1 # increase the word count by one

        # in order to make sure the result has a decimal point, we have to "cast"
        average_word_length = float(number_of_characters)/number_of_words


    # print everything out on screen
    print current_sentence
    print "number of words:", number_of_words
    print "number of characters:", number_of_characters
    print "this sentence has about %s characters per word" % (average_word_length)
    print
    # this just prints an empty line
```