

Programming for Linguists

Lecture 1

1 Class Overview

1.1 Class Objectives

- Teach you to program in the Python language
- Expose you to some of the basic tasks in computational linguistics
- Familiarize you with computer programming ideas and techniques that will apply to any programming language/problem domain

1.2 Student Background

- This class is intended for someone who may have a little linguistic experience but no programming experience.
- I do assume a certain level of computer literacy, e.g. knowing how to install programs, download things from the web.
- If you already have some programming experience, the first part of the course might be easy for you. You should still be able to get something out of it once we move on to more linguistically specific applications.

1.3 Class Structure

- The class will consist of reading from an online textbook, lectures, and programming projects.
- The first few weeks will focus on the basics of Python without much linguistics. Once you've got a basic handle on the language, you'll practice by writing linguistically-interesting programs.
- The only grades come from weekly programming assignments. (Plus class participation.) There are no tests. (You should never have to memorize anything: that's what computers are for.)

- As much as is possible in a classroom setting, I'll try to make the assignments simulate programming you would do in the real world (e.g. in commercial software development or writing research code).

2 Python for Linguists

2.1 Why Linguists Should Learn to Program

- By writing your own program you can handle more complex data management tasks than you can with Excel.
- There are computer applications that work with natural language that are interesting in their own right.
- Computer programs are a way of precisely documenting chains of symbolic reasoning of the sort often employed in linguistics.
- Programming and linguistic reasoning are similar but different enough that a knowledge of one can fruitfully inform a knowledge of the other.

2.2 Why Python?

- Python is a popular, well-supported open source high-level programming language used in many commercial and scientific applications.
- Like similar high-level languages—e.g. Perl, Ruby, Java—Python is the right “weight” for linguistic tasks: abstract enough to be broadly applicable, but not so generic as to be hard to use.
- Python places a particular emphasis on writing code that is clear and self-documenting.

2.3 The Good News and Bad News About Learning a Programming Language

- *The bad news*: your knowledge will always be incomplete and outdated. Python is the best language to know right now; that almost certainly won't be the case ten years from now. Maybe not even five years from now.
- *The good news*: Computer languages are all alike. The same ideas and techniques get used over and over again. So if you master the way Python performs a few basic tasks, that knowledge will readily transfer to C++, Perl, Java, Ruby, etc. (It's like learning Italian if you already know Spanish, only easier.)
- In this class I will explicitly draw attention to those aspects of what we learn about Python that are more generally applicable.

3 A Quick Introduction to Python

There is a computer program called Python. It lives on your computer like any other program (e.g. Excel, Internet Explorer, Duke Nukem, etc.) You run it by typing `python` at a command prompt.

```
> python
Python 2.3 (#1, Aug 29 2003, 15:56:51)
[GCC 3.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Python prints some header information then gives you a `>>>` prompt. You type something after the prompt, hit enter, and the python program responds to what you typed. If you type something the program doesn't understand it'll give you an error message.

```
>>> gibberish
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
NameError: name 'gibberish' is not defined
>>>
```

There is a precisely defined set of commands that the program understands. This set is called the Python programming language, or just Python. The most simple command is probably the `print` statement.

```
>>> print "Hello, world"
Hello, world
```

The `print` command tells python to print something to the string. The double-quoted material that comes after `print` is what gets printed.

You can also do basic math

```
>>> 1 + 1
2
>>> 3*20
60
```

You can also assign values to variables and then work with the variables. Any resemblance to high school algebra is purely intentional.

```
>>> x = 5
>>> x * 2
10
```

The `>>>` prompt is called Python's *interactive mode*. We can also write Python code in text files for later use. Say we write the following in a file that we save as `hello.py`.

```
msg = "Hello, world"
print msg
```

If we go into the same directory and give the same of this file as an argument to the `python` program, `hello.py` will be run as a script.

```
> python hello.py
Hello, world
```

Alternately, we could start interactive mode and load `hello.py` as a library (or module) with the `import` statement. Variables defined in the library become available using the attribute operator `..`

```
>>> import hello
Hello, world
>>> hello.msg
'Hello, world'
```