What is Programming?

Decent definition from wikipedia:

Computer programming ... is a process that leads from an original formulation of a computing problem to executable programs. It involves activities such as analysis, understanding, and generically solving such problems resulting in an algorithm, verification of requirements of the algorithm, implementation of the algorithm in a target programming language, testing, debugging, and maintaining the source code. The purpose of programming is to find a sequence of instructions that will automate performing a specific task or solve a given problem.


What is a Program?

At an abstract level, a program is a sequence of commands, which produces an output for a given input.

Example 1:
1. Input: your income information
2. Program: stuff happens (Input ↦→ Output)
3. Output: how much tax you have to pay

Example 2:
1. Input: a text file containing all of Ulysses
2. Program: stuff happens (Input ↦→ Output)
3. Output: every bigram (two-word sequence) with its associated frequency

Algorithms

A program encodes an algorithm, i.e., a sequence of commands

Here’s what a sketch of an algorithm for printing out a text’s unigrams (i.e., wordlist) might look like:

1. Read in each word from the text
   1.1 Store each word
   1.2 Add to the count of each word, storing (word,count) pairs in some storage device
2. Read through the storage device
   2.1 Print each word with its count

But how do we “read in” something or “store” things?

Programming Languages

Programming languages share a lot in common:

- They often have similar data structures & features (lists, functions, modules, ...)
- They require you to use explicit syntax, e.g.:
  - Only well-defined functions can be used
    - exec is a legitimate command in Python
    - evac is not a legitimate command
  - The language forces you to follow particular formats
    - In Python, you have to indent within a for loop
    - In Perl, you have to enclose the contents of a loop within brackets.

Languages differ in the specifics of the syntax, but good programming practice in one carries over to another

Why Should Linguists Care?

- Linguists like to work with data
- Data is often electronically encoded, and there is often huge amounts of it

Computational linguists need to learn how to program, not just to analyze data, but also to develop technology

- NB: There are some programming books specifically geared at linguists; see, e.g., http://www.u.arizona.edu/~hammond/
What Will We Learn This Semester?

We'll examine one programming language in particular, Python, and you'll learn:
- basic command-line programming
- the basic & not-so-basic capabilities of Python
  - lists, tuples, strings, dictionaries, loops, functions, exceptions, objects, etc.
- how to convert an algorithm into program code
- fundamental concepts for writing good programs
- how to write programs for text processing

Why Python?

Why Python?
- It’s quick: It is very good for writing short scripts and for text processing.
- It’s powerful: At the same time, Python has much support for turning small programs into much larger projects (such as object-oriented programming).
- It’s easy: Function names are (arguably) rather transparent in Python.
- It’s free and available across systems (code is generally portable across platforms).
- It’s marketable: organizations like Google, Pixar, and the NSA use Python.

Python 3

It is important to note that we’re using Python 3, not Python 2
- Programs written in Python 3 are not backwards-compatible!
  - Python 2: print "Hello world!"
  - Python 3: print("Hello world!")

If you can learn one, you can learn the other
- But it is very important to note which version of Python is being referenced in documentation, third-party tools, etc.

Obtaining Python

- The latest python is available for different platforms at: http://www.python.org/download/
- Mac: Python 2 is probably pre-installed. Type python at a terminal to check.
  - You may have to install Python 3 and type python3 at a terminal.
- On Windows, if Python is only available in the directory where it was downloaded, you can:
  - work in the directory where Python was installed
  - include the full path of Python when you run your programs, e.g., C:\Python34\python program.py
  - change the environment variable PATH (check under “Control Panel”) to include C:\Python34, so the Command Prompt can find python from any directory.

Resources on Python

Online Books & Resources:
- Dive Into Python by Mark Pilgrim (for experienced programmers): http://www.diveintopython3.net
- Programming in Python 3 by Mark Summerfield (for those with some programming experience)
- Learning to Program by Alan Gauld: http://www.alan-g.me.uk/l2p/index.htm
- Guido van Rossum’s Python Tutorial: http://www.python.org/doc/current/tut/
  Or, of course, search for specific information . . .

NLTKit

Natural Language Toolkit (NLTK):

NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, and an active discussion forum.

http://www.nlpl.org/

We will use NLTK later in the semester.
Let’s step back from Python for just one second and talk about using a command line.

Run commands by typing, instead of clicking ...

- **Windows**: open a Command Prompt
  - Start → Programs → Accessories → Command Prompt
- **Mac**: open a Terminal
  - Applications → Utilities → Terminal

See the contents of a directory:

- **Windows**: `dir`
- **Mac (Unix)**: `ls`

This is where we’ll pick up with the next slides...