Exploring Algorithms with Python

Dr. Chris Mayfield

Department of Computer Science
James Madison University

Oct 16, 2014

What is Python?

From Wikipedia:

- General-purpose, high-level programming language
  - One line of code translates to many CPU instructions
- Design philosophy emphasizes code readability
  - Syntax is minimal; indenting is part of the language
- Supports multiple programming paradigms
  - Imperative, object-oriented, procedural, functional, ...

Other features:

- Interpreted and interactive; very easy to learn
- Extensive standard library ("batteries included")
- Free and open source; very active community

Home page: http://www.python.org/

Who uses Python?

- Google (various projects)
- NASA (several projects)
- NYSE (one of only three languages "on the floor")
- Industrial Light & Magic (everything)
- Yahoo! (Yahoo mail & groups)
- RealNetworks (function and load testing)
- RedHat and Ubuntu (Linux installation tools)
- LLNL, Fermilab (steering scientific applications)

More success stories at http://www.pythonology.com/

Installing Python

Note: We will be using Python version 2.7
(the default that currently ships with Mac/Linux)

Linux / Mac: it's built in!

- Just run python (or idle) from the terminal

Windows:

- Go to http://www.python.org/download/
- Python 2.7.8 Windows X86-64 Installer
- Run python or idle from the start menu

What you need to know for CS 101

Algorithm representation

Section 5.2 presents four basic primitives:

1. Assignment
   - name $\leftarrow$ expression
2. Decisions
   - **if** condition **then** activity
   - **if** condition **then** activity **else** activity
3. Loops
   - **while** condition **do** activity
4. Functions
   - **def** name(parameters)

Example using all four

```
def greeting(hour):
    if hour < 12:
        print "Good morning!"
    else:
        print "Good afternoon!"
    # count up to 5:00 PM
    now = hour
    while now < 18:
        print "It's now", now, "o'clock."
        now = now + 1
    print "Time to eat!"

# test it out
greeting(9)
greeting(15)
```

NOTE: Files are executed from top to bottom

- The first statement *defines* a function
- The last two statements *call* the function

How does Python compare to other languages?

In case you can't wait until next week...

Python vs Java

Python

- Dynamically typed
  (bind names to values)
- Concise: "expressing much in a few words; implies clean-cut brevity, attained by excision of the superfluous"

```
# open the file d.txt
myFile = open("d.txt")
```

Java

- Statically typed
  (must declare variables)
- Verbose: "abounding in words; using or containing more words than are necessary"

```
import java.io.*;
...
// open the file d.txt
BufferedReader myFile;
myfile = new BufferedReader(
    new FileReader("d.txt"));
```

http://pythonconquerstheuniverse.wordpress.com/category/java-and-python/

Tips for CS 139/149 students

If you are currently learning Java...

```java
// Java example
int x;
x = 1;
if (x > 0) {
    System.out.println(
        "x is positive!");
}
```

```python
# Python example
x = 1
if x > 0:
    print "x is positive!"
```

- Forget about declarations
- Forget about semicolons
  - But think about colons
- Forget about parentheses
- Forget about braces
  - But remember to indent!

http://xkcd.com/353/

Okay, so now what?

CS is a creative activity

"Computational thinking is a way humans solve problems; it is not trying to get humans to think like computers. Computers are dull and boring; humans are clever and imaginative. We humans make computers exciting. Equipped with computing devices, we use our cleverness to tackle problems we would not dare take on before the age of computing and build systems with functionality limited only by our imaginations;"

Jeannette M. Wing, "Computational Thinking," CACM 2006
http://www.cs.cmu.edu/afs/cs/usr/wing/www/publications/Wing06.pdf

Algorithm discovery

Section 5.3 is a fantastic read:

- The art of problem solving
  - Don't follow steps; take the initiative
  - Trying examples helps you understand
  - Let your subconscious work for you
- Getting a foot in the door
  - Starting is often the hardest part
  - Extend knowledge, reverse engineer
  - Resist the temptation to use formulas!

The more problems you solve, the more creative you become!

Loop control

You don't need to read all of Section 5.4

- Focus on loop control, pages 224–228
- In particular, Figure 5.7 (components)

```
# while loop example
count = 3                   # initialize
while count > 0:            # test
    print 'Hello'
    count = count - 1       # modify
```

```
# for loop example
for i in range(3):          # all in one
    print 'Hello'
```

Note: Python does not have a post-test loop