Introduction to Python

Genome 559: Introduction to Statistical and Computational Genomics

Prof. James H. Thomas

If you have your own PC, download and install a syntax-highlighting text editor and Python 2.6.4:

http://www.flos-freeware.ch/notepad2.html

http://www.python.org/download/releases/2.6.4/

If you have your own Mac, download Python (same site) and TextWrangler:

http://www.barebones.com/products/TextWrangler/download.html

Why Python?

- Python is
  - easy to learn
  - relatively fast
  - object-oriented
  - strongly typed
  - widely used
  - portable

- C is much faster but much harder to use.
- Java is somewhat faster and harder to use.
- Perl is slower, is as easy to use, but is not strongly typed.

Getting started on the Mac

- Start a terminal session
- Type “python”
- This should start the Python interpreter (often called “IDLE”)

```
> python
Python 2.6.4 (something something)
details something something
Type "help", "copyright", "credits" or "license"
for more information.
>>> print “Hello, world!”
Hello, world!
```

The interpreter

- Try printing various things
  - Leave off the quotation marks.
  - Print numbers, letters and combinations.
  - Print two things, with a comma between.
  - Enter a mathematical formula.
  - Leave off the word "print".
- The interpreter allows you to try things out interactively and quickly.
- Use the interpreter to test syntax, or to try commands that you're not sure will work when you run your program.

Your first program

- In your terminal, Ctrl-D out of python.
- Type "pwd" to find your present working directory.
- Open TextWrangler.
- Create a file containing one line:

```
print "hello, world!"
```

- Be sure that you end the line with a carriage return.
- Save the file as "hello.py" in your present working directory.
- In your terminal, type "python hello.py"

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

Notice that, once you save the file with ".py" as the extension, WordWrangler automatically colors the text according to the syntax.

Objects and types

- We use the term object to refer to any entity in a python program.
- Every object has an associated type, which determines the properties of the object.
- Python defines six types of built-in objects:

Number	10 or 2.71828
String	“hello”
List	[1, 17, 44] or [“pickle”, “apple”, “scallop”]
Tuple	(4, 5) or (“homework”, “exam”)
Dictionary	{“food” : “something you eat”, “lobster” : “an edible arthropod”}
File	more later…

- Each type of object has its own properties, which we will learn about in the next several weeks.
- It is also possible to define your own types, comprised of combinations of the six base types.

Literals and variables

- A variable is simply a name for an object.
- For example, we can assign the name “pi” to the Number object 3.14159, as follows:

```
>>> pi = 3.14159
>>> print pi
3.14159
```

- When we write out the object directly, it is a literal, as opposed to when we refer to it by its variable name.

Assignment operator

```
>>> pi = 3.14159
```

This means assign the value 3.14159 to the variable pi.
(it does NOT assert that pi equals 3.14159)

```
>>> pi = 3.14159
>>> pi = -7.2
>>> print pi
-7.2
```

The “import” command

- Many python functions are available only via “packages” that must be imported.

foo and bar mean something-or-other-goes-here

```
>>> print log(10)
Traceback (most recent call last):
  File foo, line 1, in bar
NameError: name 'log' is not defined
>>> import math
>>> print math.log(10)
2.30258509299
>>> print log(10)
Traceback (most recent call last):
  File foo, line 1, in bar
    print log(10)
NameError: name 'log' is not defined
```

The command line

- To get information into a program, we can use the command line.
- The command line is the text you enter after the word "python" when you run a program.

```
python my-program.py 17
```

- The zeroth argument is the name of the program file.
- Arguments larger than zero are subsequent elements of the command line.

zeroth argument

first argument

Reading command line arguments

Access in your program like this:

```
import sys
print sys.argv[0]
print sys.argv[1]
```

zeroth argument

first argument

```
> python my-program.py 17
my-program.py
17
```

There can be any number of arguments, accessed by sequential numbers (sys.argv[2] etc).

Sample problem #1

- Write a program called "print-two-args.py" that reads the first two command line arguments after the program name, stores their values as variables, and then prints them on the same line with a colon between.
- Remember to use the python interpreter for quick syntax tests.

```
> python print-two-args.py hello world
hello : world
```

Hint – to print multiple things on one line, separate by commas:

```
>>> print 7, "pickles"
7 pickles
```

Solution #1

```
import sys
arg1 = sys.argv[1]
arg2 = sys.argv[2]
print arg1, ":", arg2
```

Sample problem #2

- Write a program called "add-two-args.py" that reads the first two command line arguments after the program name, stores their values as variables, and then prints their sum.

```
> python add-two-args.py 1 2
3.0
```

Hint - to read an argument as a number, use the syntax

```
num1 = float(sys.argv[1])
```

Solution #2

```
import sys
arg1 = float(sys.argv[1])
arg2 = float(sys.argv[2])
print arg1 + arg2
```

Challenge problems

Write a program called "circle-area.py" that reads the first command line argument as the radius of a circle and prints the area of the circle.

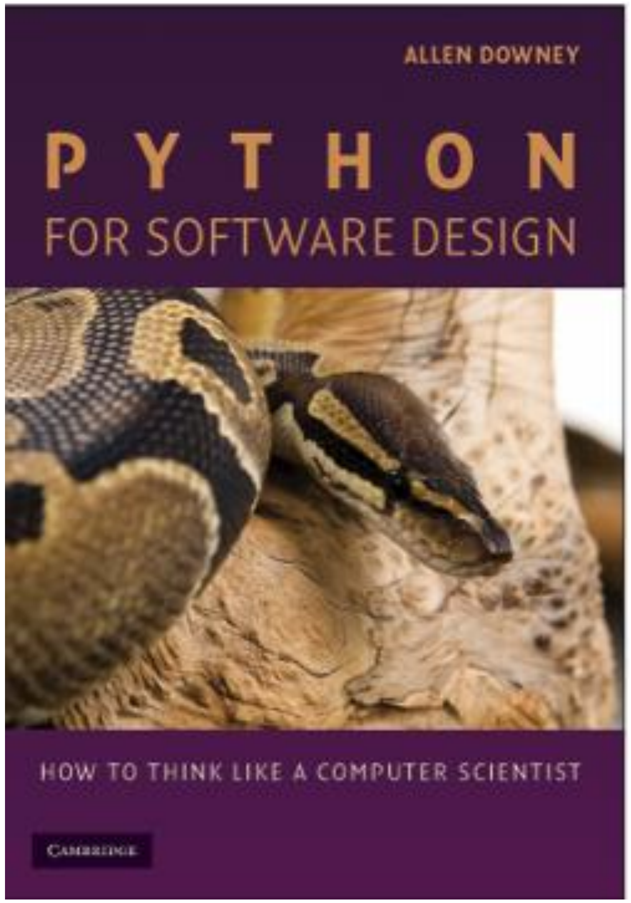
```
> python circle-area.py 15.7
774.371173183
```

Do the same thing but read a second argument as the unit type and include the units in your output.

```
> python circle-area2.py 3.721 cm
43.4979923683 square cm
```

Reading

- Chapters 1-2 of *Python for Software Design* by Downey.



One-minute feedback

Take an index card, write one thing that you liked or want more explanation for, hand in.