Python for Programmers

a little bit of Python for experienced programmers

Dr. Lisa Ball
Last Update: Nov 2012

Why Python (or not)?

- Python is a high level language suited for
  - Rapid app development
    - Can take 1/5th the time to code equivalent in Java or C
- Large-scale programming

Java & others	Python

int temp = var1;
var1 = var2;
var2 = temp;
	var2, var1 = var1, var2

Why Python (or not)?

- Not so much for big number crunching
  - Science and engineering (FORTRAN or C)
  - but it easily integrates with C or C++

- "Batteries included". Many libraries, including
  - image processing
  - numeric processing
    - e.g., computational biologists seem to like it
  - user interfaces & web programming
  - can integrate ones in other languages
    - (C/C++ especially)

easy & expressive

- A single line of code can do more than that in most other languages.
  - faster to write
  - fewer lines of code ⊏⊐ easier to maintain, debug, modify
- Simple syntax rules
  - Watch out: does require proper indentation!
- Variables
  - variable assigned to any type
  - lists can include different data type

Built-in libraries, Cross-Platform

- Many (others available to download)
  - handling email, web pages, databases, OS calls, GUIs, and more
  - Example: write web server to share files in a directory

```
import http.server
http.server.test(Handlerclass=http.server. SimpleHTTPRequest Handler)
```

- Cross Platform
  - Windows, Mac, Linux, …
  - Interpreted, so same code runs on all
  - Can run on Java (Jython) and .Net (IronPython)

Python is readable

```
#Perl version
sub pairwise_sum {
  my($arg1, $arg2) = @_;
  my(@result)=();
  @list1 = @$arg1;
  @list2 = @$arg2;
  for ($i=0;$i < length(@list1); $i++ {
    push(@result, $list1[$i] + $list2[$i];
  }
  return(\@result);
}
```

*From Ceder book, page 19 (see refs) 4/25/16

Python is readable

```
#Python version for the same task
def pairwise_sum(list1, list2):
    result=[]
    for i in range(len(list1)):
        result.append(list1[i] + list2[i])
    return result
```

*From Ceder book, page 19 (see refs) 4/25/16

Things to consider

- Can be slower to execute than C, …
  - byte-code ⎵ Python interpreter
- But not always, e.g., regular expressions
  - as fast or faster than C programs
- This may all be ok, since
  - with fast computers, development costs often outweigh
  - Easy to integrate with C/C++ modules for CPU intensive components

*From Ceder book, page 19 (see refs)

Things to consider

- Excellent library support
  - others (C, Perl Java) have more
    - but this is changing
- Speed
  - byte-code ⊏ Python interpreter
  - But not always, e.g., regular expressions
    - as fast or faster than C programs
  - This may all be ok, since
    - with fast computers, development costs often outweigh
    - Easy to integrate with C/C++ modules for CPU intensive components

Python 3.x and 2.7

- Not backward compatible
  - 3.x suggested for new development
- Python 2.7
  - existing code to use, maintain
  - need certain libraries
  - tools available to help transition to 3.x

Python 2.x	Python
Print “hello”	Print(“hello”)

*From Ceder book, page 19 (see refs)

Python 3.x and 2.7

- Make sure existing code works 1st!
- Some changes, e.g.,
  - strings are Unicode by default
  - see table below
- Tools are available to help, e .g
  - 2to3

Python 2	Python 3
½ is 0	½ is 0.5 1//2 is 0

Overview for programmers

Python Synopsis*

Python has a number of built-in data types such as integers, floats, complex numbers, strings, lists, tuples, dictionaries, and file objects. These can be manipulated using language operators, built-in functions, library functions, or a data type's own methods.

[… object stuff omitted, but it's in the language]

Python provides conditional and iterative control flow through an `if-elif-else` construct along with `while` and `for` loops. It allows function definition with flexible argument-passing options. Exceptions (errors) can be raised using the `raise` statement and caught and handled using the `try-except-else` construct.

Variables don't have to be declared and can have any built-in data type, user defined object, function, or module assigned to them.

Built-in datatypes

- Numbers
  - Integers
  - Floats
  - Complex Numbers (3 + 2j)
  - Booleans (True, False) behave like 1 and 0
- Numeric Operators
  - +, -, *, /, %, ** (exponentiation)
  - built in operators and libraries
    - import math
  - Complex numbers
    - x = (3 + 2j) * (4 + 9j) is (-6 + 35j)
    - x.real is -6
    - x.imag is 35j

Other datatypes

- more datatypes
  - strings
  - lists
  - dictionaries
  - tuples
  - sets
- same indexing & slicing functions

Strings

- operators and methods return new strings
  - operators: in, +, *
  - re library (regular expressions)
- printing is similar to C’s printf
  - print (“the value of %s is %5.2f” % (“cat”, c)
- delimiting
  - “Can contain ‘single quotes’ here”
  - ‘Can contain “single quotes” here’
  - ''' \t starts with tab, end with newline char\n '''
  - """triple quoted can contain real newlines, like in a file??? """

Lists

- can contain mixture of other types, including
  - numbers, strings, lists, dictionaries, functions, file objects, and tuples.
- can add, remove, replace, extract elements
- built-ins include
  - len, min, max,
- library includes
  - append, count, pop, remove, sort, reverse

x = [1, (2,3,)'piggie']
x[2] is 'piggie'
x[1] is (2,3)
x[-1]is 'piggie'
x[0:3] is [1, (2, 3), 'piggie']
x[-2:] is [(2, 3), 'piggie'] notice the colon after -2

*From Ceder book, page 19 (see refs) 4/25/16

List comprehensions

```
>>> [n*n for n in range(1,11)]
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
>>> [c.upper() for c in 'piggie']
['P', 'I', 'G', 'G', 'I', 'E']
>>> nums=[2,-4,13,-17, 2]
>>> result=[n for n in nums if n>0]
>>> result
[2, 13, 2]
```

*From Ceder book, page 19 (see refs)

Example comprehensions and function def (eatvowels.py)

```
# eat_vowels -- example from a Python text
# to use
# >>> eat_vowels('Texas weather is weird')

def eat_vowels(s):
    "Remove vowels from s."
    print s
    x = ' '.join(c for c in s if c.lower() not in 'aeiou')
    print x
    return

eat_vowels('Texas weather is weird')
```

Control flow statments

- if …elif
  - for i in range (10,0,-1)…print(i)
  - see airfare.py
- while loop

```
x=2
y=4
while y > x:
    y = y-1
print y
```

- for loop
  - includes continue and break statements (as in C)

More stuff.

- functions
  - flexible. simple example:
    - funct1(u,z=v,y=2) # returns 23
    - see demofunctions.py
- exceptions
  - try-except-finally-else (for 3, maybe for 2)
- creating your own modules
  - can import and use just like Python's built in libraries
- File handling
  - basic open, close, reading & writing
  - sys library for access to stdin, stdout, stderr
  - Pickle to easily read/write Python data types
  - struct library to read/write files for use with C programs
- OO (not covered :-)

Resources

- The Quick Python Book 2nd ed.
  - Vernon L. Cedar, Manning Publishing, 2010
  - Source of most of this document
  - Covers Python 3 and differences with ver. 2
- Python.org
- Good series (both 2.7 and 3 versions)
  - for example, from the series by Derek Banas
    - Tkinter – GUI's
- Google Python Class Day 1 Part 2:
  - YouTube - Google Python Class Day 1 Part 2
  - This is one part of a 2 day seminar