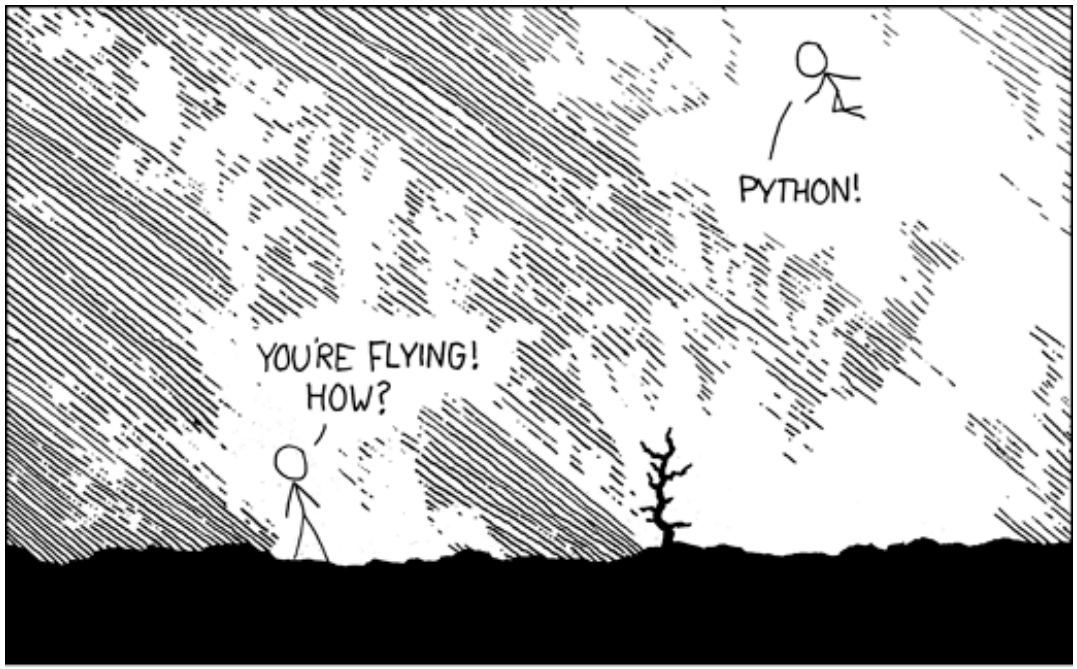
Lecture 38: Python

CS 51G
Spring 2018
Kim Bruce

Announcements

- Test program 2 now live
  - Design due Tuesday, April 24
    - It will not be returned before program is due!
    - Keep a copy for yourself!
  - Due last day of classes
- Quiz Friday (Strings) & Python lab
- Exercise


PYTHON!
YOU'RE FLYING! HOW?


I LEARNED IT LAST NIGHT! EVERYTHING IS SO SIMPLE!
HELLO WORLD IS JUST print "Hello, world!"
I DUNNO... DYNAMIC TYPING? WHITESPACE?
COME JOIN US! PROGRAMMING IS FUN AGAIN! IT'S A WHOLE NEW WORLD UP HERE!
BUT HOW ARE YOU FLYING?
I JUST TYPED import antigravity
THAT'S IT?
... I ALSO SAMPLED EVERYTHING IN THE MEDICINE CABINET FOR COMPARISON.
BUT I THINK THIS IS THE PYTHON.

Python

- Python is designed as a scripting language
  - Short programs to glue together calls to powerful libraries.
- Python is relatively slow compared to languages like Java, C, C++, etc.
  - but has highly optimized libraries written in other languages.
- Designed by BDFL Guido Van Rossum
  - Python 1 (1990), Python 2(2000), Python 3 (2008)

Python Resources

- Python for Java Programmers
  - http://python4java.necaiseweb.org
- Think Python 2e (free text) *for novices*
  - http://greenteapress.com/wp/think-python-2e/

Key Points of Python

- Indenting is significant (like Grace)
  - use spaces not tabs — *don't mix them!!*
  - Line breaks are important. Statements extending onto the next line are problematic. Surround by parens so Python knows it is a continuation!
    - Can also use backslash \ at end to signal next line is continuation
- No curly braces (blocks headed with ":" instead)
- No type declarations

Running Python

- Use PyCharm CE
  - Get from Applications folder and drag to dock
    - https://www.jetbrains.com/pycharm-edu/download/
    - See on-line documentation
  - Can use interactive mode in console or
  - Write programs as usual

Getting started

- print “hello world”
- count = 10
  - assignment
- count = “countString”
  - no type associated with names, can change on fly
- Comments start with #
  - x = 0 #assigns value 0 to x

Python programming

- Blocks use ":"
  - indentation counts!

```
i = 10
while i > 0:
    print(i)
    i = i - 1
print "That's it!", i

if i > 0:
    print "oops, terminated too soon!", i
elif i < 0:
    print "terminated too late", i
else:
    print 'terminated just right!', i
```

Defining functions

```
# Defines a "repeat" function that takes 2 arguments.
def repeat(s, exclaim):
    result = s + s + s
    if exclaim:
        result = result + '!!!'
    return result
```

def not *method*

Parameterless functions must have “( )”

Must use “return”

Primitive Types

- Numbers: Integers and floating point
  - have different results of division
  - can convert to other, i.e., float(3), int(3.7) #truncates
  - works for strings, too
- Boolean: False, True: not, and, or
- String: “hello” or ‘hello’,
  - str(3.7) converts number to string
- list: [0, 2, 4, ”hello”] — heterogeneous
- Tuple (immutable): (1,2,’a’)

Constants/Variables

- Python does not distinguish between constants and variables. However, by convention, writing an identifier in all caps says it should not be changed (though Python will not enforce it).
- Python has no multi-part function names
  - All parameters come after the name of the function

Lists

- vowels = [“a”, “e”, “i”, “o”, “u”]
- letterI = vowels[2]
- extendedVowels = vowels + [“y”]
  - exW = extendedVowels.append(‘y’)
- Other methods: del, pop, remove, reverse, sort

For Loops

```
for x in [2,3,5,7,11]:
    print "for",x

for x in range(2,8):
    print "range",x

if x in [2,3,5,7,11]:
     print x, "is in"
```

Tuples

- Like lists, but immutable:
  - triple = (5, True, “heel”)

Strings

- Treated like lists for indexing, slices
- if x = “abcde” then x[1] is “b”, x[-1] is e, x[1:3] is “bc”, x[3:] is “de”, x[:3] is “abc”, etc.
- Can also use slices with lists
- Methods: lower, upper, etc. (see documentation)

Imports

- Like Grace, can import functions from external files
  - from math import sqrt #specific function
    - can use it without writing math.sqrt, just sqrt(16).
  - from math import * # import everything

Example

```
from math import *

x = float( raw_input( "Enter a real value:" ) )
y = sqrt( x )
print "The square root of", x, "is", y
print (int(3.7))
```

- raw_input prompts for input, returns response as a string

Exceptions

- try-except rather than try-catch

```
try:
        cost = totalcost / days
except ZeroDivisionError:
        print "Division by zero error"
```

Object-Oriented Programming in Python

- Python has classes, but no object expressions
- Classes have
  - constructors
  - instance variables
  - methods

Class Definitions

```
class myClass(superClass):
  “"""myClass comment.”"""
  def __init__(self, otherParams):
    """Initialize object."""
    self.var = exp
    ...

  def someMeth(self, otherParams):
    “”"comment for method."""
    doStuff ....
```

Class Example

```
from math import sqrt

class Point():
    """Class representing point on screen"""
    def __init__(self, x, y):
        """weird syntax for constructor"""
        self.x = x
        self.y = y

    def translate(self,dx,dy):
        self.x = self.x + dx
        self.y = self.y + dy

    def distance(self,other):
        return sqrt((self.x - other.x)**2 + (self.y - other.y)**2)

    def asString(self):
        return "<" + str(self.x) + "," + str(self.y) + ">"
```

Example Using Class

```
p1 = Point(3,4)

print "p1 = ",p1.asString()

origin = Point(0,0)

print "distance =",p1.distance(origin)

origin.translate(6,8)

print "new origin = ", origin.asString()
print "p1 = ", p1.asString()

print "translated distance =",p1.distance(origin)
```

Questions?