Introduction to HW1, P1 and Python

EECS 388 – January 7, 2015

Contact Info/Office Hours

- Questions and concerns go to eecs388-staff@eecs.umich.edu
- OH: BBB Learning Center

Andrew:	M 230-330PM
Alishah:	Tue 1-2PM
Dylan:	Tue 2-3PM
Peter:	W 430-530PM
Luis:	Thu 4-5PM
Brad:	F 230-330PM
Zane:	W 230-330PM, F 330-430PM

- Piazza

Homework 1

- Due Thursday, January 21st at 6PM
- Thinking like an attacker and defender in a couple different scenarios

Project 1 - Crypto Project

- Due Thursday, January 28th at 6PM
- Read and understand the spec thoroughly
- Two parts:
    - Length Extension Attack
    - Hash Collision Attack

What is Python?

- Interpreted high-level programming language
- Used for a variety of contexts from web programming to RHEL system scripts
- Object-Oriented (less than Ruby or Smalltalk, more Java or C++)
- Dynamic Typing and Memory Management

Why Python?

- Designed to be readable (vs Perl)
- Mature and trusted (vs Ruby)
- Large standard library
- Ships with most Linux distros and BSDs
- Highly extensible (access to low level details)
- Very high level code
    - [x*x for x in range(10)]

Why not Python?

- Terrible unicode and regex support
  - If you're doing a ton of text parsing…

- Less Object-Oriented than Ruby
  - Decent but not amazing meta classes

- It is slow
  - Speed usually doesn't matter
  - You can make it fast

- Not the coolest language on the block

Notable Features

- Control-Flow based on white spaces not brackets
- (almost) everything is a first-class object
- Duck-Typing
- Static Scoping

Built-in Data Types

- Numbers:int, long, float, complex
- Strings (immutable)
- Lists (no arrays)
- Dictionaries (equivalent to hash tables)
- Types for binary data, regular expressions

Numbers

- The usual notations and operators
  - 12, 3.14, 0xFF, 0377, (-1+2)*3/4**5, abs(x), 0<x<=5
- C-style shifting & masking
  - 1<<16, x&0xff, x|1, ~x,x^y
- Integer division truncates
  - 1/2 -> 0 # float(1)/2 -> 0.5
- Long (arbitrary precision), complex
  - 2L**100 ->1267650600228229401496703205376L

Lists

- Flexible arrays
  - Lists resemble std::vector<Object *>

```
a = [99, "bottles of beer", ["on", "the", "wall"]]

a[0] => 99
a.append(10)
a.sort()
a.reverse()
```

Strings and Sequence Operations

```
"hello"+"world"   = "helloworld”          #concatenation
"hello"*3         = "hellohellohello”     #repetition
"hello"[0]        = "h"                   #indexing
"hello"[-1]       = "o"                   #(from end)
"hello"[1:4]      = "ell"                 #slicing
len("hello")      = 5                     #size
"hello" < "jello" = 1                     #comparison
"e" in "hello"    = 1                     #search
```

Dictionaries

- Associative Arrays / Hash Tables

- `x=MyObject()`

- `d= {"a":"foo", 7:x, x:{8:"mystring"}}`
  - `d[‘a’] => “foo”`
  - `d[x][8] => "mystring”`

- for k, v in d.iteritems():

```
for k, v in d.iteritems():
        print k, v
```

Dictionaries

- Indexing – d[key]
- Removal– del d[key]
- Membership testing – key in d, k not in d
- d.clear()
- Index w/default – d.get(key, default=None)
- Lists of elts–d.keys(), d.values(), d.items()
- Iteration: for key in d: ...
- Size:len(d)

Tuples

- Immutable grouping of elements
- x = ((1,2), (3,4), “mystring”)
- Unpacking
  - `one, two, three = (1, 2, 3)`

Variables

- No declaration -- only assignment:

```
x = 8; y = “mystring”; z = Object()
```

- Variables are not typed:

```
if myvar == 7:
   x = “mystring”
else:
   x = 8
```

Control Structures

- Whitespace!

```
if x == 7:
  print “success”
Elif x == 8:
  print “failure”
else:
  print “unknown”
for x in [1,4,5]:
  print x
for x in range(10):
  print x
x= 0
while x < 10:
  print x
  x = x + 1
```

Reference Semantics

```
>>>a = [1, 2, 3]
>>> b = a
>>>a.append(4)
>>> print b

[1, 2, 3, 4]
```

Need a copy?

```
import copy
copy.copy(a)        # Shallow Copy
copy.deepcopy(a) # Deep Copy
```

Functions

```
defgcd(a, b):
  "greatest common divisor"
  if b < a:
    a, b = b, a
  while a != 0:
    a, b = b%a, a
  return b

>>>gcd.__doc__
'greatest common divisor’

>>>gcd(12, 20)
4
```

Classes

```
class Animal(object):
“””This class represents an animal”””
  def__init__(self, name, secret=None):
    self.name == name
    self.__secret= secret

  deftalk(self):
    raise Exception(“I don’t know how to talk”)

  def__str__(self):
    return “<%s (%s)>” % (self.__class__.__name__,self.name)

Class Dog(Animal):
  deftalk(self):
    print “Bark!”
```

Exception Handling

```
import sys

try:
with open(“myfile”, r”) asfd:
for line infd:
print line
except Exception, e:
print “unable to open file”
sys.exit(1)
```

Help?

- The help function displays doc strings
- The dir function lists attributes & methods
- http://docs.python.org/lib is best
- Google is your friend

Help?

- Python Tutorial
  - http://docs.python.org/tutorial
- PEP8: Style Guide for Python Code
  - http://www.python.org/dev/peps/pep-0008/
- PEP20: The Zen of Python (short!)
  - http://www.python.org/dev/peps/pep-0020/
- DiveInto Python
  - http://diveintopython.org

Conclusion

- Questions?