CS35L – Winter 2019

Slide set:	3.2
Slide topics:	Python
Assignment:	3

Python

- Not just a scripting language
- Object-Oriented language
  - Classes
  - Member functions
- Compiled and interpreted
  - Python code is compiled to bytecode
  - Bytecode interpreted by Python interpreter
- Not as fast as C but easy to learn, read and use
- Very popular at Google and other big companies

Why is it popular?

Uses English keywords frequently where other use different punctuation symbols

Fewer Syntactical Constructions

Automatic Garbage Collection

Easy integration with other programming languages

Different Modes

Interactive:

- Run commands on the python shell without actually writing a script/program.

Script Mode:

- Type a set of commands into a script
- Execute all the commands at once by running the script

Case sensitive

Start with _ (underscore) or letters followed by other letters, underscores or digits

Other special characters are not allowed as part of the variable name

Certain reserved words may not be used as variable names on their own unless concatenated with other words

Python Variables

Example: Python Variables

Python Script:

```
#!/usr/bin/python
counter = 100       # An integer assignment
miles = 1000.0      # A floating point
name = "John"       # A string
```

Python Lines and Indentation

No braces to indicate blocks of code for class and function definitions or flow control

Blocks of code are denoted by line indentation, which is why it is **strictly enforced**

Number of spaces for indentation may be variable but all the statements within the same block must be equally indented

Hence, *a single space has the ability to change the meaning of the code*

Python Decision Making

```
#!/usr/bin/python
var = 100
if ( var == 100 ) :
   print ”Correct”
print "Good bye!"
```

Python List

- Common data structure in Python
- A python list is like a C array but much more:
  - **Dynamic (mutable)**: expands as new items are added
  - **Heterogeneous:** can hold objects of different types
- How to access elements?
  - List_name[index]

Example

- `>>> t = [123, 3.0, ‘hello!’]`
- `>>> print t[0]`
  - 123
- `>>> print t[1]`
  - 3.0
- `>>> print t[2]`
  - hello!

Example – Merging Lists

- `>>> list1 = [1, 2, 3, 4]`
- `>>> list2 = [5, 6, 7, 8]`
- `>>> merged_list = list1 + list2`
- `>>> print merged_list`
  - **Output: [1, 2, 3, 4, 5, 6, 7, 8]**

Python Dictionary

- Essentially a hash table
  - Provides key-value (pair) storage capability
- Instantiation:
  - `dict = {}`
  - This creates an EMPTY dictionary
- Keys are unique, values are not!
  - Keys must be immutable (strings, numbers, tuples)

Example

```
dict = {}
dict[‘france’] = “paris”
dict[‘japan’] = “tokyo”
print dict[‘france’]

dict[‘germany’] = “berlin”
if (dict[‘france’] == ”paris”):
     print “Correct!”
else:
     print “Wrong!”

del dict[‘france’]
del dict
```

for loops

```
list1 = [‘Mary’, ‘had’, ‘a’, ‘little’, ‘lamb’]
```

```
for i in list1:
    print i
```

Result:

```
Mary
had
a
little
lamb
```

```
for i in range(len(list1)):
    print i
```

Result:

```
0
1
2
3
4
```

- Powerful library for parsing command-line options
  - **Argument:**
    - String entered on the command line and passed in to the script
    - Elements of sys.argv[1:] (sys.argv[0] is the name of the program being executed)
  - **Option:**
    - An argument that supplies extra information to customize the execution of a program
  - **Option Argument:**
    - An argument that follows an option and is closely associated with it. It is consumed from the argument list when the option is

Optparse Library

Homework 3

randline.py script

Input: a file and a number *n*

Output: *n* random lines from *file*

Get familiar with language + understand what code does

Answer some questions about script (Q3, Q4)

Implement shuf utility in python

Running randline.py

- Run it
  - ./randline.py –n 3 filename (need execute permission)
  - python randline.py –n 3 filename (no execute permission)

- randline.py has 3 command-line arguments:
  - filename: file to choose lines from
    - **argument** to script
  - n: specifies the number of lines to write
    - **option**
  - 3: number of lines
    - **option argument** to n

- Output: 3 random lines from the input file

shuf.py

- Support the options for shuf
  - --echo (-e)
  - --head-count (-n)
  - --repeat (-r)
  - --help
- Support all type of arguments
  - File names and – for stdin
  - Any number of non-option arguments
- Error handling

Homework 3

- shuf.py – this should end up working almost exactly like the utility ‘shuf’
  - Check $ man shuf for extensive documentation
- Use randline.py as a starting point!
  - Modify to accomplish logical task of shuf
- shuf C source code :
  - Present in coreutils
  - This will give you an idea of the logic behind the operation that shuf executes
- Python argparse module instead of optparse:
  - How to add your own options to the parser
  - -e -n --repeat --echo etc

Homework 3 Hints

- If you are unsure of how something should be output, run a test using existing shuf utility!
  - Create your own test inputs
- The shuf option --repeat is Boolean
  - Which action should you use?
- Q4: Python 3 vs. Python 2
  - Look up “automatic tuple unpacking”
- Python 3 is installed in /usr/local/cs/bin
  - export PATH=/usr/local/cs/bin:$PATH

Python Walk-Through

```
#!/usr/bin/python

import random, sys
from optparse import OptionParser

class randline:
    def __init__(self, filename):
        f = open (filename, 'r')
        self.lines = f.readlines()
        f.close ()

    def chooseline(self):
        return random.choice(self.lines)


def main():
     version_msg = "%prog 2.0"
     usage_msg = """%prog [OPTION]...
FILE Output randomly selected lines
from FILE."""
```

Tells the shell which interpreter to use

Import statements, similar to include statements
Import OptionParser class from optparse module

The beginning of the class statement: randline
The constructor
Creates a file handle
Reads the file into a list of strings called lines
Close the file

The beginning of a function belonging to randline
Randomly select a number between 0 and the size of lines and returns the line corresponding to the randomly selected number

The beginning of main function

version message

usage message

```
parser = OptionParser(version=version_msg,
                      usage=usage_msg)
parser.add_option("-n", "--numlines",
          action="store", dest="numlines",
          default=1, help="output NUMLINES     lines
(default 1)")

options, args = parser.parse_args(sys.argv[1:])

try:
    numlines = int(options.numlines)
except:
    parser.error("invalid NUMLINES: {0}".
                 format(options.numlines))
if numlines < 0:
    parser.error("negative count: {0}".
              format(numlines))
if len(args) != 1:
    parser.error("wrong number of operands")
input_file = args[0]
try:
    generator = randline(input_file)
    for index in range(numlines):
        sys.stdout.write(generator.chooseline())
except IOError as (errno, strerror):
    parser.error("I/O error({0}): {1}".
format(errno, strerror))

if __name__ == "__main__":
    main()
```

Creates OptionParser instance

Start defining options, action "store" tells optparse to take next argument and store to the right destination which is "numlines". Set the default value of "numlines" to 1 and help message.

options: an object containing all option args

args: list of positional args leftover after parsing options

Try block

get numline from options and convert to integer

Exception handling

error message if numlines is not integer type, replace {0} w/ input

If numlines is negative

error message

If length of args is not 1 (no file name or more than one file name)

error message

Assign the first and only argument to variable input_file

Try block

instantiate randline object with parameter input_file

for loop, iterate from 0 to numlines – 1

print the randomly chosen line

Exception handling

error message in the format of "I/O error (errno):strerror

In order to make the Python file a standalone program