More scripting

Week 3 - Wednesday

Interpreted Language & Compiled Language

- Interpreted language
  - A programming language that avoids explicit compilation
  - Execute the program source code directly
  - Rely on the just-in-time compilation of interpreters
  - E.g., shell script, Python
- Be contrasted with Compiled Language
  - For compiled language, user must explicitly translate source code into a lower-level machine language executable (binary)
  - E.g., C, C++

Interpreted Language: Pros & Cons

- Pros
  - Platform independence
  - Reflection
    - E.g., “eval” translate a string into an expression
  - Dynamic typing
  - Smaller executable size
- Cons
  - Way Too Slow!

Java: A third choice

- A compromise between interpreted and compiled language
  - Combine compiling and interpreting
  - Source code need to be explicitly compile into Byte Code
    - Byte code is platform independent
  - Virtual machine interprets the byte code into machine language/binary code
    - The implementation of the virtual machines depends on the platform
  - Tradeoff for flexibility & efficiency
  - No directly access to memory management
    - No delete/free, No pointers

Python

- A widely used general−purpose, high-level programming language
- Emphasize code readability
- Support multiple programming paradigms
  - Object-oriented
  - Functional programming
- Often used as a scripting language

Python 2 & Python 3

- Python 2.0
  - Released in 2000
  - Many major new features
    - Garbage collector
    - Support for Unicode
  - Still being developed, current version 2.7
- Python 3.0
  - Released in 2008
  - A major, backwards-incompatible release
  - Many features been backported to Python 2
- NOTE: which version you are using!
  - “python --version”
  - “python3”
  - Use symbolic link to switch your default Python

Prerequisite 1: Basics of other languages

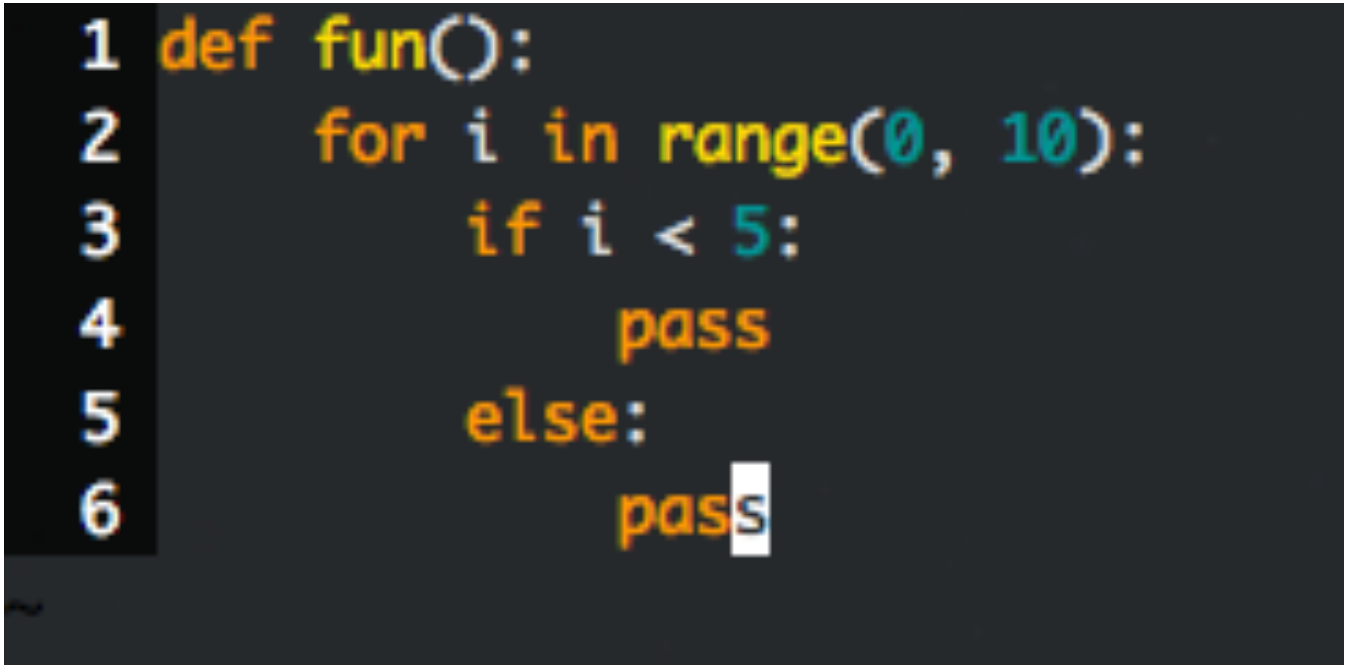
- C/C++
  - Variable, type
  - Control structure
  - Function
- C++
  - class
- Shell scripting

Prerequisite 2: Indent not “{}”

- “{}” in C/C++: use to encapsulate code block

Prerequisite 2: Indent not “{}”

- In Python, use indent instead of “{}”
  - Indent must be consistent
  - Space is different from tab



Variable

- Similar to shell scripting
  - No need to declare before using
- Dynamic typing
  - Not bound to a type
  - E.g., first assign a int, later assign a string

String

- String can be expressed in
  - Single quotes
  - Double quotes
  - Triple quotes

```
'''This is a multi-line string. This is the first line.
This is the second line.
"What's your name?," I asked.
He said "Bond, James Bond."
'''
```

Control Structures

- If-elif-else
- for-in
- while

Function & Class

- Function
  - Define a function: def fun()
- Class
  - __init__(): constructor
  - All the member variables declared in the constructor
  - self: the “this” pointer

Runtime Error Handling

- The try-except structure
  - Error handling mechanism to avoid corruption
  - try: encapsulate the code that may have runtime error
  - except: code to handle errors

Basic data types

- Int
- Long
- Float
- Boolean: True, False
- None: = NULL in C
- http://docs.python.org/2/library/stdtypes.html

Convert to a string

```
a = 1
b = str(a)
print type(a)
print type(b)
```

Data Type: List

- A more flexible array
  - No fixed-length
  - No limitation on the type of items
- Visit the element with index
  - Get the length of a list with len(l)
- Slice operation: extract a sub-list
- Check whether an item is in the list
- Add/remove an item
- Add to another list
- http://docs.python.org/2/tutorial/introduction.html#lists

A list example

```
l = [1, 'a', [1, 2]]
print l[0]
print l[-1]
print l[1:2]
```

Data Type: Sequence Types

- Both string and list are Sequence Types
  - in
  - +, *
  - Index
  - Slicing
  - len(), min(), max()
  - .index(), .count()
- http://docs.python.org/2/library/stdtypes.html#sequence-types-str-unicode-list-tuple-bytearray-buffer-xrange

Data Type: Dictionary

- An mapping object: maps hashable values to arbitrary objects
- Key & Value
- Why hashable?
  - Hash table
  - Two strings with same value taken as the same key or two different keys?
- Example
  - Use a dictionary to record whether a number has been seen
- http://docs.python.org/2/library/stdtypes.html#mapping-types-dict

A dictionary example

```
d = {'one' : 1, 2 : 'two'}
d['one'] = 1111
print d['one']
print d.has_key(1)
print d.keys()
print d.values()
```

Data Type: Set

- An unordered collection of distinct hashable objects
- Convert a set from/to a list
- http://docs.python.org/2/library/stdtypes.html#set-types-set-frozenset

Modules

- How to use a module?
  - http://docs.python.org/2/library/
- Option parser
  - Helps to parse the command line options & arguments
    - ./randline.py –n 10 input.txt
  - http://docs.python.org/2/library/optparse.html

diff & patch

- diff
  - Compare files line by line
- patch
  - Apply a diff file to an original
- Combination of diff & patch
  - Developer diff old version and new version; save the output into a patch file
  - User apply the patch file to the old version of source code and get the new version

Hints

- Decompress the source code
  - “tar zxvf xxx.tar.gz”
- Prepare the compilation environment
  - Entry the directory of the source code
  - Read INSTALL to see how to set your temporary install location
  - “./configure”
- Compile
  - “make”
- Install
  - “make install”
- Run the “ls” you build instead of that provided by the system
  - Enter you directory, run “./ls”
- Use Python3 on SeasNet server
  - export PATH=$PATH:/usr/local/cs/bin