CS214 - Advanced UNIX

Lecture 6 - More Python

Ymir Vigfusson

Adapted in part from material in

- *Python 201* (http://www.rexx.com/ dkuhlman/python_201/)
- and *Dive into Python* (http://www.diveintopython.org/)

Dictionaries

Recall that dictionaries are associative arrays, like hash-maps in Perl.

- `d = dict()` or alternatively
- `d = {}`
- `d["key"] = value`
- `del d["key"]` will remove `key` from the dictionary.

Dictionaries

Recall that dictionaries are associative arrays, like hash-maps in Perl.

- `d.clear()` clears all items from the dictionary
- `d.keys()` and `d.values()` gives you lists with keys and values.
- `d.items()` gives you a list of (key,value) pairs.

Lists

Creating and modifying lists.

- `l = list()` or alternatively
- `l = []`
- `l.append ('a')` adds 'a' to the end, giving `['a']`.
- `l.extend (list)` concatinates the list with l.

Lists

Creating and modifying lists.

- `l.push('a')` adds 'a' at the end (like a stack).
- `l += ['b']` also concatinates lists.
- `l = [1,2] * 2` gives `[1,2,1,2]`.

Lists

Searching and removing elements from a list.

- `l.index('a')` gives the first index for 'a', or the ValueError exception if not found.
- `'a' in l` gives True if 'a' is found in list, False otherwise.
- `l.remove('a')` removes the first occurrence of 'a' in l.
- `l.pop()` removes and returns the last element from the list (like a stack).

List

Python supposed advanced mapping to lists

```
>>> li = [1, 9, 8, 4]
>>> [elem*2 for elem in li]
[2, 18, 16, 8]
```

Sequences

- `tuple(seq)` converts a sequence object into a tuple.
- `list(seq)` converts a sequence object into a list.
- `min(seq)` returns the minimum element in the sequence.
- `max(seq)` returns the maximum element in the sequence.

String module

- `atof()` converts a string into a float
- `atoi()` converts a string into an integer
- `capitalize()` capitalizes the first character in the string
- `capwords()` capitalizes every word in a string.
- `lower()` and `upper()` change the case of the string.

String module

- `strip()` removes whitespace from beginning and end
- `replace()` replaces substrings with another.
- `split()` split string based on whitespace.

Operating system module

- `getcwd()` gets the current directory
- `listdir()` lists the files in the current directory
- `chown()` changes ownership of a file
- `chmod()` changes permissions of a file
- `rename()` renames a file
- `remove()` removes a file
- `mkdir()` creates a new directory
- `system()` executes a command in a subshell

Data conversion

We can perform conversions between basic objects with relative ease.

- `float(obj)` gives a floating point representation of obj.
- `int(obj)` gives the integer representation of obj. For instance `int("3")` gives 3.
- `str(obj)` gives the string representation of obj.
- `chr(num)` gives ASCII character #num.
- `ord(ch)` gives the ASCII code for character ch.

Regular expressions

- Python has routines for manipulating regular expressions in the `re` module.

```
import re
pat = re.compile ('(\d|Z)+([a-z]*)')
m = pat.search ("Z555characters")
    ...
```

Regular expressions

- Use `match` to match the beginning of a string, and `search` to match anywhere inside it.
- If `m` exists, `m.group(1,2)` extracts the substrings for the first and second parenthesized matches.

Regular expressions

```
>>> import re
>>> str = 'delicious apples'
>>> re.sub('[Aa]pple', 'pear', str)
'delicious pears'
```

Exceptions

Python handles exceptions with `try - except` blocks.

```
def ble:
      try:
           12345 / 0
      except ZeroDivisionError:
           print "Oh no!"
```

Exceptions

(From the official Python documentation)

```
try:
    i = int(open('myfile.txt').readline().strip(
except IOError, (errno, strerror):
    print "I/O error(%s): %s" % (errno, strerror
except ValueError:
    print "Could not convert data to an integer.
except:
    print "Unexpected error:", sys.exc_info()[0]
    raise
```

Clean-up

Some objects have standard clean-up routines.

Handles garbage collection, closing files, etc.

```
with open("myfile.txt") as f:
    for line in f:
        print line
```

This is new in Python 2.6, so use `'from __future__ use with_function`. (Thanks Ben).

Mathematics

- Mathematical operators are `+ - / *`, exponentiation x^y is done by $x**y$ (like in Gnuplot).
- The `math` module provides advanced mathematical functionality. Example:

```
import math

x = math.pi
f = math.sin(2.0*x)
```

Random numbers

- `random.random()` returns a random float in the range $[0, 1)$.
- `random.randrange(n)` returns a random integer in the range $[0, n)$.

```
> random.choice(['a', 'b', 'c'])
'b'
# sample two numbers without replacement
> random.sample(xrange(10), 2)
[2, 7]
```

Gnuplot

There is a Python module for interacting with Gnuplot

```
import Gnuplot
g = Gnuplot.Gnuplot()
g.title('Graph title')
g.xlabel('Range')
g.ylabel('Frequency')
g('set logscale')
g.plot ([ [1,1], [2,5], [3,15] ])
raw_input ('press any key')
```