Most Basic Python

- Python is a simple scripting language
- Used for a lot of sysadmin tasks
- Has simple support for OOP, not not required

Invoking Python

- Since Python is an interpreted language, you need to use the “Bang Syntax” to write scripts

```
#!/bin/bash
#!/bin/perl
#!/bin/python
#!/usr/local/bin/python
#!/bin/bin/env python
```

Example 1
(example from Safari - Python for Unix and Linux System Administration, 1st Edition - by Noah Gift; Jeremy Jones)

```
#!/bin/bash
for a in 1 2; do
    for b in a b; do
        echo "$a $b"
    done
done

#!/usr/bin/perl
foreach $a ('1', '2') {
    foreach $b ('a', 'b') {
        print "$a $b\n";
    }
}

#!/usr/bin/env python
for a in [1, 2]:
    for b in ['a', 'b']:
        print a, b
```

Example 2

(example from Safari - Python for Unix and Linux System Administration, 1st Edition - by Noah Gift; Jeremy Jonesh)

```
#!/bin/bash
if [ -d "/tmp" ] ; then
    echo "/tmp is a directory"
else
    echo "/tmp is not a directory"
fi

#!/usr/bin/perl
if (-d "/tmp") {
    print "/tmp is a directory\n";
}
else {
    print "/tmp is not a directory\n";
}

#!/usr/bin/env python
import os

if os.path.isdir("/tmp"):
    print "/tmp is a directory"
else:
    print "/tmp is not a directory"
```

Example 3
(example from Safari - Python for Unix and Linux System Administration, 1st Edition - by Noah Gift; Jeremy Jones)

```
#!/usr/bin/env python

class Server(object):
    def __init__(self, ip, hostname):
        self.ip = ip
        self.hostname = hostname
    def set_ip(self, ip):
        self.ip = ip
    def set_hostname(self, hostname):
        self.hostname = hostname
    def ping(self, ip_addr):
        print "Pinging %s from %s (%s)" % (ip_addr, self.ip,
              self.hostname)

if __name__ == '__main__':
    server = Server('192.168.1.20', 'bumbly')
    server.ping('192.168.1.15')
```

Zen

```
Python 2.6 (r26:66714, Dec 11 2008, 09:53:18)
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
>>>
```

First Assignment

1) Add yourself to the class mailing list by going to:
   http://korgano.eecs.ohiou.edu/mailman/listinfo/python

2) re-code Ostermann's remv shell script in python

```
BSH:picard> remv
Usage: remv oldfile-regexp newfile-regexp files*
BSH:picard> remv FOO BAR FOO1 FOO2
BSH:picard> remv '---*' - *-*

BSH:picard> ls
first_last.txt shawn_ostermann.txt

BSH:picard> remv '\([a-z]*\)_\([a-z]*\).txt' '\2_\1' *
Old Name Regexp: "\([a-z]*\)_\([a-z]*\).txt"
New Name Regexp: "\2_\1"
mv -i "first_last.txt" "last_first"
mv -i "shawn_ostermann.txt" "ostermann_shawn"

BSH:picard> ls
last_first ostermann_shawn
```