CS 35L Software Construction Laboratory (Lab3-B)
Wed, Jan 25, 2012, Ver 1.1

How to use a Makefile
  1. name it as default (Makefile) and run command "make" in the same directory.
  2. "make -f OtherMakefile" (not recommanded)

It is too complicated to write Makefile by ourselves
  Different system need different system
  autoconf -- generate the "configure" script
  automake -- generate Makefile
  -- a tutorial of using autoconf and automake with C++:
    http://www.openismus.com/documents/linux/automake/automake.shtml

Install program without sudo
  1. Generate a Makefile that will install the program into another directory
     ./configure --prefix=/tmp/cu
  2. Build the source code again (run "make")
  3. Run "make install" without sudo (you don't need it now)

$PATH environment variable
    Check you $PATH value:
      jwcai@eagles:/$ echo $PATH
      /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games
    When you type a command into the shell, the system will search it in
  directories listed in the $PATH.
    Modify the $PATH
    Option 1. export PATH=/tmp/cu/bin:$PATH
    Option 2. modify ~/.bashrc and append the export command.
  Check with "which" command

Python Reading:
    http://docs.python.org/release/2.4.1/tut/tut.html
    Ch1 ~ Ch7: must read, read the rest if you want more powerful tools
    http://learnpythonthehardway.org
    Lots of small exercises, very helpful

Python in Interactive mode
  Simply type "python" in the command line
  -- It is a good calculator
       e.g. >>> import math
            >>> math.sqrt ((3 + 4.5) ** 2)
            7.5
  -- It is a good helper for pyhton
       e.g. >>> help(open)

Revisit: Run a (Python) script
  Option 1: python myscript.py
  Option 2: #! line and chomd +x

Indent is important in Python
```
# something is wrong with the following code
def test():
    x = 0
```

```
  for i in range(0, 10):
    x += i
     print x
  test()

Build-in Data Structure in Python
  -- Basic type: int, float, string
  -- Tuple: immutable, (generally) sequences of different kinds of stuff
  -- List: mutable, (generally) sequences of the same kind of stuff
     Ref: http://news.e-scribe.com/397
  -- Access tuple and list using subscriptions
     a[1], a[1:], c[2:4], a[-1], a[:-1]
  -- Dict: key, value pairs
     e.g. d = {}
          d["tom"] = 20
          d["jerry"] = 19
     d.values(), d.keys(), d.items()
```