ch6, Py ch14: Standard I/O and Libraries

10 Oct 2007
CMPT14x
Dr. Sean Ho
Trinity Western University

File input in Python

- Open a file for reading:

```
myFile = open('filename.txt')
```

  - myFile is a file object (file handle)
  - Filename is relative to current directory of IDLE
- Read a line from the file:

```
myFile.readline()
```

  - Returns a string, including the newline
  - Returns empty string when it hits the end-of-file

Also see
myFile.readlines()

- Close the file when you're done:

```
myFile.close()
```

Files and paths

- Specifying file pathnames: use forward slash
  - open('z:/directory/file.txt')
- Changing the current directory:
  - import os
  - os.chdir('z:/directory/')
  - open('file.txt')

Seeking in files

- Files are just streams of bytes
- Python maintains a file pointer:
  current position in the file
- Get the current position as an index:

```
myFile.tell()              # returns a number (long int)
```

- Manually set the position of the file pointer:

```
myFile.seek(0)             # go to start of file
myFile.seek(-128, 1)       # go 128 bytes back from current
```

- Read a certain number of bytes from the file:

```
myfile.read(256)           # read exactly 256 bytes
myfile.read()              # read whole file (yipes!)
```

  - Treats newlines like any other character

File output in Python

- Open a file for writing:

  ```
  myFile = open('file.txt', 'w')
  ```

  - Modes: 'r' (read), 'w' (write), 'r+' (both), 'a' (append)
  - Also 'b' (binary) for non-text files
- Write (insert) at the current position:

  ```
  myFile.write('Hello World!\n')
  ```

  - Newlines need to be explicit
- Writes are sometimes buffered before commit
- Force a flush:

  ```
  myFile.flush()
  ```

Writing out variables in Python

- write() only accepts strings:

```
numApples = 15
myFile.write( numApples )      # error
```

- str() gets the string representation of a variable:

```
myFile.write( str(numApples) )    # okay
```

- Or we can use a format string:

```
myFile.write( 'I have %d apples.\n' % numApples )
```

I/O channels

- Abstractly, a stream of input comes over a channel from a source
  - e.g., source can be keyboard, file, program, ...
- A stream is output over a channel to a sink
  - e.g., sink can be screen, file, program, etc.
- I/O channels (file descriptors, file handles) can be opened in one of three modes:
  - Read, write, and read/write
- Default source is keyboard, default sink is screen
- But we can redirect channels to other source/sink

Standard I/O channels

- The standard I/O channels are already open:
- Standard Input: sys.stdin
  - Usually the keyboard
- Standard Output: sys.stdout
  - Usually the screen
    - But often gets redirected to a file
- Standard Error: sys.stderr
  - Usually also the screen
- We've already used sys.stdout.write()
- Alternative to raw_input(): sys.stdin.readline()

Redirecting standard I/O

- You can redirect the standard I/O channels just by reassigning them:
- Make print go to a file:

```
old_stdout = sys.stdout              # save stdout
sys.stdout = open('log.txt', 'w')    # reassign
print 'Hello!'                       # goes to file
sys.stdout.close()                   # close file
sys.stdout = old_stdout              # restore stdout
```

Python standard math library

- Lots of fun stuff in here, just import math:
- pi, e
- sqrt, exp, pow(x,y)
- log(x, base) (default is natural log), log10
- sin, cos, tan, asin, acos, atan, sinh, cosh, tanh
- fabs (absolute value)
- ceil, floor
- Full list: http://docs.python.org/lib/module-math.html

TODO items

- Lab04 due tonight: ch5 # 26 / 32 / 38 / 39
- HW04 due Fri: Py §8.3 #1, Py §10.7 #1
- Lab05 due next Wed: ch6 # 33 / 35
- 140 Final / 141 midterm two weeks from today
  - Wed 24Oct 14:35-15:50 (part 1)
  - Thu 25Oct 13:10-14:15 (part 2)