# File I/O

21 Oct 2009
CMPT140
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
    - Specify absolute pathname: 'z:\filename.txt'
- Read a line from the file:

  **myFile.readline()**

  - Returns a string, including the newline
  - Returns empty string when it hits the end-of-file
- Close the file when you're done:

  **myFile.close()**

**Also see**
**myFile.readlines()**

# Seeking in files

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

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

- Manually set the position of the file pointer:

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

- 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')
```

  - 'w' is the file mode (see next slide)
- Write text 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()**

# File modes

- Files may be opened in various modes:
  - 'r': read input from file (default)
  - 'w': write output to new file
    (if the file exists, it is cleared first)
  - 'a': append output to end of existing file
    (if file doesn't exist, it is created)
  - 'r+': both read and write to file
    (writing only overwrites existing bytes,
    will not insert new bytes in the middle of the file)
- On Windows, text I/O performs mangling of end-of-line characters; use 'b' (e.g., 'rb', 'rw') to prevent that for binary data

# Writing out variables in Python

- write() only accepts strings:

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

- str() formats a variable for human readability:

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

- Or we can use a format string:

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

# repr() and pickling

- How do we represent more complex types (e.g., lists) as strings?
- repr() gets a string representation suitable for re-reading by Python:

```
myFile.write( repr(numApples) )
```

  - Compare with str() (for human readability)
- A more general framework for file I/O of objects is Python's pickle library
  - Serialize an object for a stream
    - **pickle.dump(obj, file) and obj=pickle.load(file)**

# 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, sink is screen
- 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
```

# For more information

- Python Tutorial ch7 on I/O:
  - http://docs.python.org/tutorial/inputoutput.html
- Python I/O Library reference:
  - http://docs.python.org/lib/bltin-file-objects.html
- Python pickle library reference:
  - http://docs.python.org/library/pickle.html