File I/O

26 Oct 2010
CMPT140
Dr. Sean Ho
Trinity Western University

More list operations

- Delete an element of the list:

```
del myApples[1]    # [ "Fuji", "Golden Delicious" ]
```

- List slice (start:end):

```
myApples[0:1]      # [ "Fuji" ]
```

- Lists are mutable, so assignment is aliasing:

```
yourApples = myApples      # points to same array
```

  - Changes to myApples are reflected in yourApples

- Use a whole-list slice to copy a list:

```
yourApples = myApples[:]
```

  - Shorthand for 0:len(myApples)

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
  - Or 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

Also see
myFile.readlines()

- Close the file when you're done:

  myFile.close()

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

Iterating over a file

- Just like iterating over a list or a string:

```
prov3File = open('prov3.txt')

for line in prov3File:
    line = line.strip()
    print( line.upper() )
```

  - Each line includes the newline; the .strip() method of strings removes trailing newlines

Handling file I/O errors

- File I/O errors raise exceptions (IOError):
  - file doesn't exist, no permissions, disk full, ...
  - More on exceptions next time
- The with clause ensures the file is closed tidily even if an I/O error happens along the way:

```
with open('prov3.txt') as provFile:
    for line in provFile:
        line = line.strip()
        # do stuff with line
```

- Don't need to .close(); with does it for you!

File output in Python

- Open a file for writing:

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

  - 'w' is the file mode (see next slide)
  - The with clause also works for writing
- Write text at the current position:

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

  - Newlines need to be explicit
- Writes are buffered in memory and are flushed (committed) to disk only in larger chunks
  - Force a flush: `myFile.flush()`
  - Writes are implicitly flushed on .close()

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 )
```

Reading data into variables

- We need to design our file format:
  - One number per line? Int? Float? #decimals? Order of variables? How to store a list?
- Variables in our programs can be in very complex data structures
  - e.g., list of Student record objects
- File I/O only operates on streams of bytes
- The process of converting a complex data structure to a stream of bytes is called serialization – see Python's pickle library

For more information

- Python Tutorial ch7 on I/O:
  - http://docs.python.org/py3k/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