Day 4: I/O and Exceptions

Suggested reading: *Learning Python* (3rd Ed.)

Chapter 9: Tuples, Files, and Everything Else

Chapter 27: Exception Basics
Turn In Homework
Homework Review
Write code. At least a little. Every day. Play around!
Files
What Is a File?

- Sequence of bytes on disk
- Up to Python and your script to interpret the bytes

```
% cat hw4.txt
This
is
the
Speech
Thought
and
Writing
Corpus
compiled
by
...
```

```
54 68 69 73 0a 69 73 0a |This.is.
74 68 65 0a 53 70 65 65 |the.Spee
63 68 0a 54 68 6f 75 67 |ch.Thoug
68 74 0a 61 6e 64 0a 57 |ht.and.W
72 69 74 69 6e 67 0a 50 |riting.P
72 65 73 6e 74 61 74 69 0a 43 6f 70 65 64 0a 6c 65 64 0a 62 79 0a 45 |ion.Corpus
75 73 0a 6d 69 63 0a 43 6f 70 65 64 0a 62 79 0a 45 |led.by.E
...```
Preparing to Read a File

```python
my_file = open(filename, 'r')
```

- `filename` can be absolute or relative
- `'r'` means “read”, is the default, can be omitted

```python
data = open('seq_03_T.txt', 'r')
parameters = open(parameter_filename)
```

- when done with file, close it

```python
my_file.close()
```
Reading One Line at a Time

```python
for line in file_object:
    # Note: most lines have newline at end
    print line.rstrip('\n')
```

```python
total = 0
count = 0
input = open('my-data.txt')
for line in input:
    total += int(line)
count += 1
input.close()
mean = float(total) / float(count)
print 'Mean value = %.1f' % (mean)
```
Reading Whole Files

```python
line_list = file_object.readlines()
```

- One list element per line
- Trailing newlines on each element

```python
input = open('name-list.txt')
lines = input.readlines()
input.close()

names = set()
for line in lines:
    set.add(line.rstrip('
'))
print '%d names, %d unique' % 
    (len(lines), len(names))
```
Digression #1: Failure
Run Time Failures

- Sometimes, things go badly at run-time
- Have seen error messages like this already:

```python
>>> a[2]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: list index out of range
```

- Working with files is particularly prone to failures

```python
f = open('this_filename_does_not_exist')
- Missing file
- Not allowed to read from or write to file
- Disk full when writing file
```
Exceptions

- **Raised** when a run-time failure occurs
- Allows Python to get out of arbitrarily deep code
- **YOU** decide when and where to *handle* exceptions
- Can raise your own
- *Exceptions are objects!!!*

BaseException
  - Exception
    - StandardError
      - ArithmeticError
      - FloatingPointError
      - OverflowError
    - ZeroDivisionError
  - AssertionError
  - AttributeError
  - BufferError
  - EOFError
  - EnvironmentError
    - IOError
    - OSError
  - ...
Handling Exceptions

```python
try:
    # Living dangerously
    exception TypeErrorA, e:
        print 'Caught exception A:', e
    except ExceptionTypeErrorB, e:
        print 'Caught exception B:', e
    print 'Continue here'

• If code in try block raises exception:
  – Check except clauses in order
  – Exception variable (e.g., e) may contain extra info

• Execution continues after last except block
```
array = ['Tim', 'Scot', 'Alain', 'Nick']
index = raw_input('Enter a number: ')

try:
    name = array[int(index)]
except ValueError, e:
    print '"%s" is not an int" % (index)
    print 'Assuming first element'
    name = array[0]
except IndexError, e:
    print '%s is out of bounds' % (index)
sys.exit(1)

print 'Hello, %s!' % (name)
Not Handling Exceptions

```python
def foo:
    raise ValueError()

def bar:
    foo()

bar()
```
Back to Files
try:
    input = open(filename)
    data = input.readlines()  # or other...
    input.close()
except IOError, e:
    print 'Cannot read', filename
    print e.strerror  # see help(IOError)

# What happens here if the read failed?
Writing a File

- Prepare to create (or overwrite):

```python
output = open(filename, 'w')
```

- Prepare to append (or create):

```python
output = open(filename, 'a')
```

- Options for writing data to a file:

```python
output.write('Must add newline!\n')

output.writelines(list_of_strings)

print >> output, 'No newline here'
```
Writing a File Carefully

```python
try:
    output = open(filename, 'w')
    output.writelines(data_list)  # e.g.
    output.close()
except IOError, e:
    print 'Cannot write', filename
    print 'Error:', e  # different output
```

# What happens here if the write failed?
Digression #2: Modules
Brief Introduction to Modules

```python
import sys
# help(sys)
print 'Python', sys.version
```

- Extra functionality is bundled into **modules**
- To use functionality, must **import** the module
- Prefix functions or data with **module name + dot**
Standard File Objects

- `sys.stdin` = standard (interactive) input
- `sys.stdout` = standard output (default for `print`)
- `sys.stderr` = standard error (alternative output)

```python
import sys

sys.stdout.write('Enter name: ') # no 
input = sys.stdin.readline().strip()
if input != 'Tim Cartwright':
    print >> sys.stderr, 'Wrong person!
sys.exit(1)
```
Directories
File Test Operators

- path exists: `os.path.exists(path)`
- path is absolute (starts with `/`): `os.path.isabs(path)`
- path is regular file: `os.path.isfile(path)`
- path is directory: `os.path.isdir(path)`
- path is symbolic link: `os.path.islink(path)`
- join path parts: `os.path.join(p1, p2, ...)`
- directory part of path: `os.path.dirname(path)`
- filename part of path: `os.path.basename(path)`

```python
import os.path
if os.path.exists(filename):
    if os.path.isdir(filename):
        print 'Skipping dir', filename
    else:
```

Directory Contents I

```
import os
files = os.listdir(directory)
```

- All entries in directory, except . or ..
- Arbitrary order

```
import os
import os.path
for e in sorted(os.listdir('.')):  
    if os.path.isdir(e): print 'dir:', e  
    elif os.path.isfile(e): print 'file:', e  
    else: print 'other:', e
```
Shell-Like Operations

- create a directory: `os.mkdir(path, mode)`
- create a directory recursively: `os.makedirs(path, mode)`
- remove file: `os.remove(path)`
- rename/move a path: `os.rename(old, new)`
- remove (empty) directory: `os.rmdir(path)`
- change permissions: `os.chmod(path, mode)`
- change ownership: `os.chown(path, uid, gid)`
- create a symlink: `os.symlink(path, link)`
- read the path from a symlink: `os.readlink(path)`
Last 2 Slides!
Other Scripting Languages

- Most have similar I/O operations

- Check for different or additional:
  - Operation names (-d vs. isdir() vs. directory?())
  - Syntax
  - Operations

- Not all languages have (real) exceptions (e.g., Perl)
Homework

- Read a large file of words (or subset, for practice)
- Count words, except for UPPERCASE
  - What collection type works well here?
- Find and print, e.g., \texttt{freq(“Yes”) > freq(“yes”)}

- \textbf{BE SURE TO LABEL YOUR PRINTOUT!!!}

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

"""Homework for CS 368-2 (2012 Spring)
Assigned on Day 04, 2012-03-22
Written by <Your Name>"""
```