

Reading, writing files

Terence Parr
MSDS program
University of San Francisco

See notebook <https://github.com/parr/msds501/blob/master/notes/files.ipynb>

What are files?

- Both the disk and RAM are forms of memory
- RAM is much faster (but smaller) than the disk and RAM data disappears when the power goes out
- Disks, in contrast, are persistent / non-volatile
- A *file* is simply a chunk of data on the disk identified by a filename and living within a specific directory
- File data is less convenient to access because we have to explicitly load the file into working memory before operating on it
- If a file is too big to fit into memory all at once, we have to process the data in chunks, typically line by line if text format data

File state

- Files must be opened and then closed when we're done
- Files are opened for reading or for writing
- Files are opened with a mode: text or binary

```
f = open('foo.txt', mode='r')  # open for read text mode
    read from f
f.close()                      # ok, we're done
```

- **f** is a file descriptor

Avoiding confusion

- The filename is a string that identifies a file on the disk. It can include path information or can be just the name of the file itself
- The path can be fully qualified or relative to the current working directory
- The file descriptor object is not the filename and is also not the file contents itself on the disk. It's really just a descriptor that lets our program refer to and operate on the file
- The contents of the file is different than the filename and the file (descriptor) object that Python gives us

The WITH statement

- The **with** statement helps us to automatically close files

```
$ head -5 data/prices.txt
0.605
0.600
0.594
0.592
0.600
$
```

```
with open("data/prices.txt") as f:
    contents = f.read()
    lines = contents.split()
    print(lines[0:3])
    print(f.closed)
```

```
['0.605', '0.600', '0.594']
True
```

Most common programming pattern

- Load all file contents into a string

```
with open('data/IntroIstanbul.txt') as f:  
    contents = f.read() # read all content of the file  
print(contents[0:200]) # print just the first 200 characters
```

The City and ITS People

Istanbul is one of the worlds most venerable cities. Part of the citys allure is its setting, where Europe faces Asia across

2nd most common programming pattern

- Load all lines of a file or words of a file into a list

```
with open("data/prices.txt") as f:
    contents = f.read()
    lines = contents.split()
    print(lines[0:3])
    print(f.closed)
```

```
['0.605', '0.600', '0.594']
True
```

```
with open('data/prices.txt') as f:
    prices = f.readlines()
    prices[0:3]
```

```
['0.605\n', '0.600\n', '0.594\n']
```

```
with open('data/IntroIstanbul.txt') as f:
    contents = f.read() # read all content of the file
    words = contents.split()
    print(words[0:100]) # print first 100 words
```

```
['The', 'City', 'and', 'ITS', 'People', 'Istanbul', 'is',
```

Using Pandas to load CSV files

- If the text file is a comma separated value file (CSV), the easiest way to load the data is with Pandas

```
import pandas as pd
data = pd.read_csv('data/player-heights.csv')
data.head(5)
```

| | Football height | Basketball height |
|---|-----------------|-------------------|
| 0 | 6.33 | 6.08 |
| 1 | 6.50 | 6.58 |
| 2 | 6.50 | 6.25 |
| 3 | 6.25 | 6.58 |
| 4 | 6.50 | 6.25 |

```
$ head -5 data/player-heights.csv
Football height, Basketball height
6.329999924, 6.079999924
6.5, 6.579999924
6.5, 6.25
6.25, 6.579999924
$
```


Processing files line by line

- Loading everything into memory all at once doesn't work if a file is bigger than available RAM
- That limits the size of the data we can process with that method
- Instead, we can use a for-each loop iterating on the file descriptor:

```
with open('data/prices.txt') as f:  
    for line in f: # for each line in the file  
        print(float(line)) # process the line in some way
```

```
0.605  
0.6  
0.594  
0.592  
0.6
```

Pattern: Save strings into a text file

- (The way we encode strings for storage in a file is a complication we can ignore until MSDS692)
- Save a few lines representing a CSV file to /tmp dir

```
# Write an ASCII-encoded text file
with open("/tmp/names.csv", "w") as f:
    f.write("first,last\n")
    f.write("Terence,Parr\n")
    f.write("Xue,Li\n")
    f.write("Sriram,Srinivasan\n")
```

```
$ cat /tmp/names.csv
first,last
Terence,Parr
Xue,Li
Sriram,Srinivasan
```

The `\n` sequence represents a single newline character