LECTURE 13

FILES

STRING FORMATTING

MCS 260 Fall 2020

Emily Dumas

REMINDERS

- Work on:
  - Worksheet 5
  - Quiz 5

FILES

A file is a named, ordered collection of data, usually in persistent storage (disks, flash, etc.).

Files are stored in a hierarchy of directories.

Basic operations: Read data from a file with a given name; Write data to file with a given name.

Typical request: Write the string `'print("hello world")\n'` to `c:\Users\ddumas\hello.py`.

DISKS

disks = shorthand for persistent storage devices.

OS typically sees each one as a list of 4KiB blocks.

Basic operations: Read a block by index, write to a block by index.

e.g. 466GiB disk is $\sim$122 million blocks. Typical request: Write these 4096 bytes to block 21430229.

FILESYSTEMS

A **filesystem** is the OS component that translates file-related requests into operations that disks support.

Filesystems decide where to put file data on disk, how to keep track of file and directory names, and tracking which disk parts are used/free.

FILES IN PYTHON

We will only cover **text files**, where you read and write strings. There are also **binary files**, where you read and write bytes.

The `open(filename,[mode])` function opens a file and returns an object with methods to read and write.

```
"""Write a string to a file"""
fout = open("out.txt","w")  # w means write allowed
fout.write("Hello world")
fout.close() # Done with this file (OS does cleanup)

fin = open("out.txt","r")   # r means read only (default)
s = fin.read() # Get entire file contents
```

MODES

- "r" - Read. The default. Allows reading.
- "w" - Write. **Dangerous: deletes the file if it exists**. Allows reading and writing.
- "a" - Append. Open for append. Allows reading and writing, but starts at the end of the file if it exists.

READING LINES

Often you want to process one line at a time. File objects are *iterable*, giving the lines. E.g. nl.py

```
"""Number the lines of a file specified on command line"""
import sys
fin = open(sys.argv[1],"r")
n = 0
for line in fin:
    n = n+1
    print(n,line,end="") # line usually has \n at the end
fin.close()
```

Sample output:

```
$ python nl.py nl.py
1 """Number the lines of a file specified on command line"""
2 import sys
...
```

Important: `file.write()` is not like `print()`. It doesn't add a newline, and it doesn't accept multiple arguments to print.

```
print(1,2,"three","four")       # ok
fout.write(1,2,"three","four") # FAILS
```

Must prepare a single string to write. The usual way is to use `str.format()`:

```
fout.write("{} {} three four\n".format(1,2)) # ok
# Writes "1 2 three four\n"
```

STRING FORMATTING

`str.format()` has many features to create a string based on a **template** and some values. In the string, placeholders (`{}` or `{...}`) are replaced by arguments of `str.format()`.

```
>>> "{1} taught {0}".format("MCS 260","Dumas")  # give indices
'Dumas taught MCS 260'
>>> for x in range(98,101):
...     print("{:4}".format(x))  # specify width
...
  98
  99
 100
>>> "{:04}".format(42)  # pad to width with zeros
'0042'
```

```
>>> "{:8.2f}".format(42)   # f = float, width 8, 2 digits after
'   42.00'
>>> "{:8x}".format(42)     # x = hex int, width 8
'      2a'
>>> "{:8d}".format(42)     # d = decimal int, width 8
'      42'
>>> "{:.2f}".format(13+2j) # f allows complex; no total width
'13.00+2.00j'
```

The general placeholder syntax is `{w:ot}` where w specifies **which** argument, o is a set of **options**, and t is the **type**.

`str.format()` has a lot of features we didn't discuss today.

Another file example: lettervalues.py, vals.txt

```
"""Compute the "value" of a word, if each letter is
worth a different number of points specified in a
file.
MCS 260 Fall 2020 Lecture 13 - Emily Dumas
"""

import sys

if len(sys.argv) != 3:
    print("Usage:",sys.argv[0],"valuefile word")
    print("Read values for each alphabet letter from `valuefile`")
    print("in the format letter,value, e.g.:")
    print("a,1\nb,3\nc,3\nd,2")
    print("Then, compute the value of `word` and print it.")
    sys.exit()  # exits the program immediately

fin = open(sys.argv[1],"r")
vals = dict()
for line in fin:
    ltr,valstr = line.split(",")
    vals[ltr] = int(valstr)
```

REFERENCES

- In *Downey*:
  - Chapter 14 discusses files, especially Sections 14.1, 14.2, and 14.4.
  - Section 14.3 discusses a different, older way of formatting strings.
- This Introduction to String Formatters in Python 3 by Lisa Tagliaferri at DigitalOcean is a good reference for the topics in string formatting we covered today.

REVISION HISTORY

- 2020-09-22 Initial publication