Reading & writing files

Lecture 4

CSCI 3351 & CSCI 6651
Dr. Frank Breitinger

Overview

- There are three common ways to "communicate" with a program
  - Last lecture: user input, command line arguments
  - Today: reading files

Opening a file

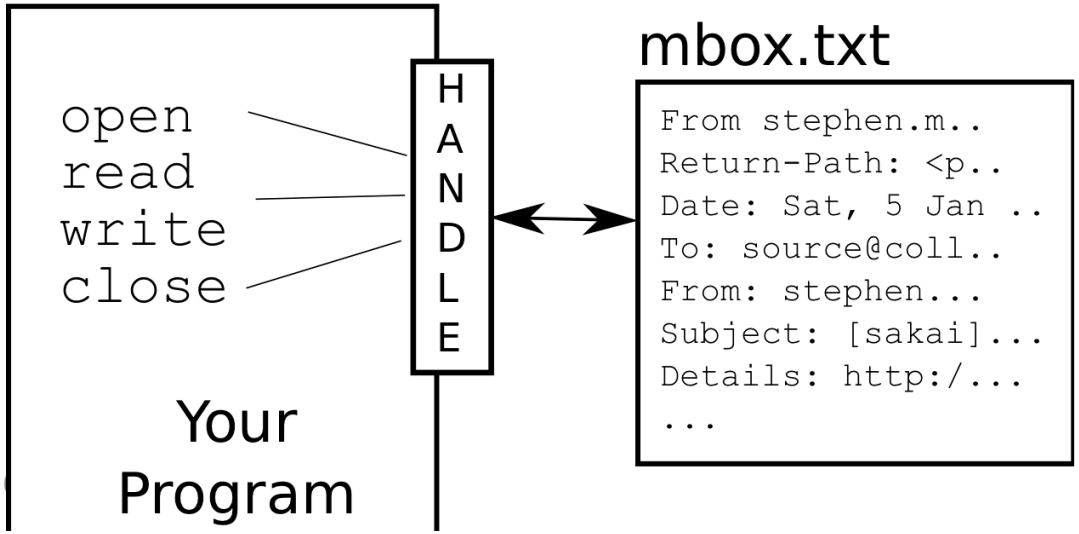
- A text file can be thought of as a sequence of lines
- Before we can read the contents of the file we must tell Python which file we are going to work with and what we will be doing with the file
- This is done with the open() function
- open() returns a “file handle” - a variable used to perform operations on the file
  - Kind of like “File -> Open” in a Word Processor

Using open()

- handle = open(filename, mode)
  - `handle = open('myFile.txt', 'r')`
  - default mode is 'r'
- returns a handle use to manipulate the file
- filename is a string
- mode is optional and should be 'r' if we are planning reading the file and 'w' if we are going to write to the file.

What is a handle?

```
>>> fhand = open('mbox.txt')
>>> print fhand
<open file 'mbox.txt', mode 'r' at 0x1005088b0
```



When Files are Missing

```
>>> fhand = open('stuff.txt')
Traceback (most recent call last):  File
"<stdin>", line 1, in <module>IOError: [Errno 2]
No such file or directory: 'stuff.txt'
```

The newline Character

- We use a special character to indicate when a line ends called the "newline"
- We represent it as \n in strings
- Newline is still one character - not two

```
>>> stuff =
'Hello\nWorld!’
>>> stuff'Hello\nWorld!’
>>> print stuff
HelloWorld!
>>> stuff = 'X\nY’
>>> print stuff
X
Y
>>> len(stuff)3
```

File Handle as a Sequence

- A file handle open for read can be treated as a sequence of strings where each line in the file is a string in the sequence
  - Remember - a sequence is an ordered set
- We can use the for statement to iterate through a sequence

```
handle = open('mbox.txt')
for line in handle:
    print(line)
```

Counting Lines in a File

- Open a file read-only
- Use a for loop to read each line
- Count the lines and print out the number of lines

```
handle = open('mbox.txt')
count = 0
for line in handle:
     count = count + 1
print('Line Count:',
count)
```

Reading the *Whole* File

- We can read the whole file (newlines and all) into a single string.
  - Be careful, might crash a program.

```
handle = open('topics.txt')
input = handle.read()
print(input)
print(len(input))
```

Reading files

- Functions to read a file:
  - `handle.read()` - file's entire contents as a string
  - `handle.readline()` - next line from file as a string
  - `handle.readlines()` - file's contents as a list of lines

- Copy the following in a text file:

From: FBreitinger@newhaven.edu
this is the text of email1
From: Kaplan@newhaven.edu
this is the text of email2
From: zqian@newhaven.edu
this is the text of email3
From: mycharger@gmail.com
this is the text of email4

Exercise - Searching through a File

- Only print the lines that contain ‘From’.
  - Tipp: Since each line is a string, we can use string functions.

Solution - Searching through a File

- We can put an if statement in our for loop to only print lines that meet some criteria

```
fhand = open(’text.txt')
for line in fhand:
      if line.startswith('From:’):
            print(line)
```

???

- Why do we have newlines?

From: FBreitinger@newhaven.edu

From: Kaplan@newhaven.edu

From: zqian@newhaven.edu

From: mycharger@gmail.com

Because ...

- What are all these blank lines doing here?
  - Each line from the file has a new line at the end
  - The print statement adds a newline to each line.

From: FBreitinger@newhaven.edu\n

\n

From: Kaplan@newhaven.edu\n

\n

From: zqian@newhaven.edu\n

\n

From: mycharger@gmail.com\n

\n

How can we fix that?

Searching Through a File (fixed)

- We can strip the whitespace from the right hand side of the string using rstrip() from the string library
- The newline is considered "white space" and is stripped

```
fhand = open('mbox.txt')
for line in fhand:
  #line = line.rstrip()
  if line.startswith('From:'):
    print(line.rstrip())
```

Skipping with continue

- We can conveniently skip a line by using the continue statement
  - Continue works for every loop!

```
fhand = open('mbox.txt')
for line in fhand:
  #line = line.rstrip()
  if not line.startswith('Fr:'):
    continue
  print(line.rstrip())
```

Using in to select lines

- We can look for a string anywhere in a line as our selection criteria

```
fhand = open('mbox.txt')
for line in fhand:
    line = line.rstrip()
    if not '.edu' in line:
        continue
    print(line)
```

Print word by word

- `split` breaks a string into tokens that you can loop over.

  name.`split()# break by whitespace`

  name.`split(`delimiter`)  # break by delimiter`

- Splitting into variables is possible as well
  - `>>> s = "Jessica 31 647.28"`
  - `>>> name, age, money = s.split()`

Prompt for file name

```
fname = raw_input('Enter the file name:  ')
fhand = open(fname)
count = 0
for line in fhand:
    if line.startswith(’From:') :
        count = count + 1
print('There were', count, ’From: lines in', fname)
```

closing files

- After using a file, handles should be closed.
  - Will eventually be closed when the file object is garbage collected
  - you will be wasting system resources by holding to file handles you no longer need.

```
f = open("mbox.txt")
for line in f:
        print line
f.close()
```

Writing and appending

name = `open(`**"filename",** `"w")`
name = `open(`**"filename",** `"a")`

- opens file for <u>write</u> (deletes previous contents), or
- opens file for <u>append</u> (new data goes after previous data)

name.`write(`**str**`)` – writes the given string to the file

Exercise

- Write a short program that reads an input and saves it into a file.
  - You can also ask for the filename!

Modify

- Unfortunately there is no way to insert into the middle of a file without re-writing it.
- You can append to a file or you'll have to rewrite it (if you want to make changes at the beginning or middle).
- Exercise: change your program so that it appends.

seek

- The method seek() sets the file's current position at the offset

  `handle.seek(offset[, whence])`
  - offset -- This is the position of the read/write pointer within the file.
  - whence -- This is optional and defaults to 0 which means absolute file positioning, other values are 1 which means seek relative to the current position and 2 means seek relative to the file's end.
- E.g., helpful to to back to the start of the file
  - `handle.seek(0,0)`

Example seek

- Assume a text:

```
From: FBreitinger@newhaven.edu
foobar
```

- To print the email address:

```
handle = open('testfile.txt',
'r')
handle.seek(6)
print(handle.readline().strip())
handle.close()
```

- Very helpful if the file read has a unique structure (e.g., all lines have the same length)

tell()

- You can get the current position of the handle using tell:
  - `>>> handle.tell()`

Random access to text lines

- The linecache module allows one to get any line from a Python source file.
- Will not be discussed (yet).

Try and Except will be discussed next lecture!

Bad file names

```
fname = raw_input('Enter the file name:  ')
try:
    fhand = open(fname)
except:
    print 'File cannot be opened:', fname
    exit()
count = 0
for line in fhand:
    if line.startswith('Subject:') :
        count = count + 1
print 'There were', count, 'subject lines in', fname
```