

# Reading and Writing Text Files

Course: Python Programming

# File Processing

A text file can be thought of as a sequence of lines

```
From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008
Return-Path: <postmaster@collab.sakaiproject.org>
Date: Sat, 5 Jan 2008 09:12:18 -0500
To: source@collab.sakaiproject.org
From: stephen.marquard@uct.ac.za
Subject: [sakai] svn commit: r39772 - content/branches/
```

```
Details: http://source.sakaiproject.org/viewsvn/?view=rev&rev=39772
```

<http://www.py4e.com/code/mbox-short.txt>

# File Processing

A text file has **newlines** at the end of each line

```
From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008\nReturn-Path: <postmaster@collab.sakaiproject.org>\nDate: Sat, 5 Jan 2008 09:12:18 -0500\nTo: source@collab.sakaiproject.org\nFrom: stephen.marquard@uct.ac.za\nSubject: [sakai] svn commit: r39772 - content/branches/\n\nDetails: http://source.sakaiproject.org/viewsvn/?view=rev&rev=39772\n
```

# Opening a File

- 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
- Similar to “File -> Open” in a Word Processor

# Using open()

```
fhand = open('mbox.txt', 'r')
```

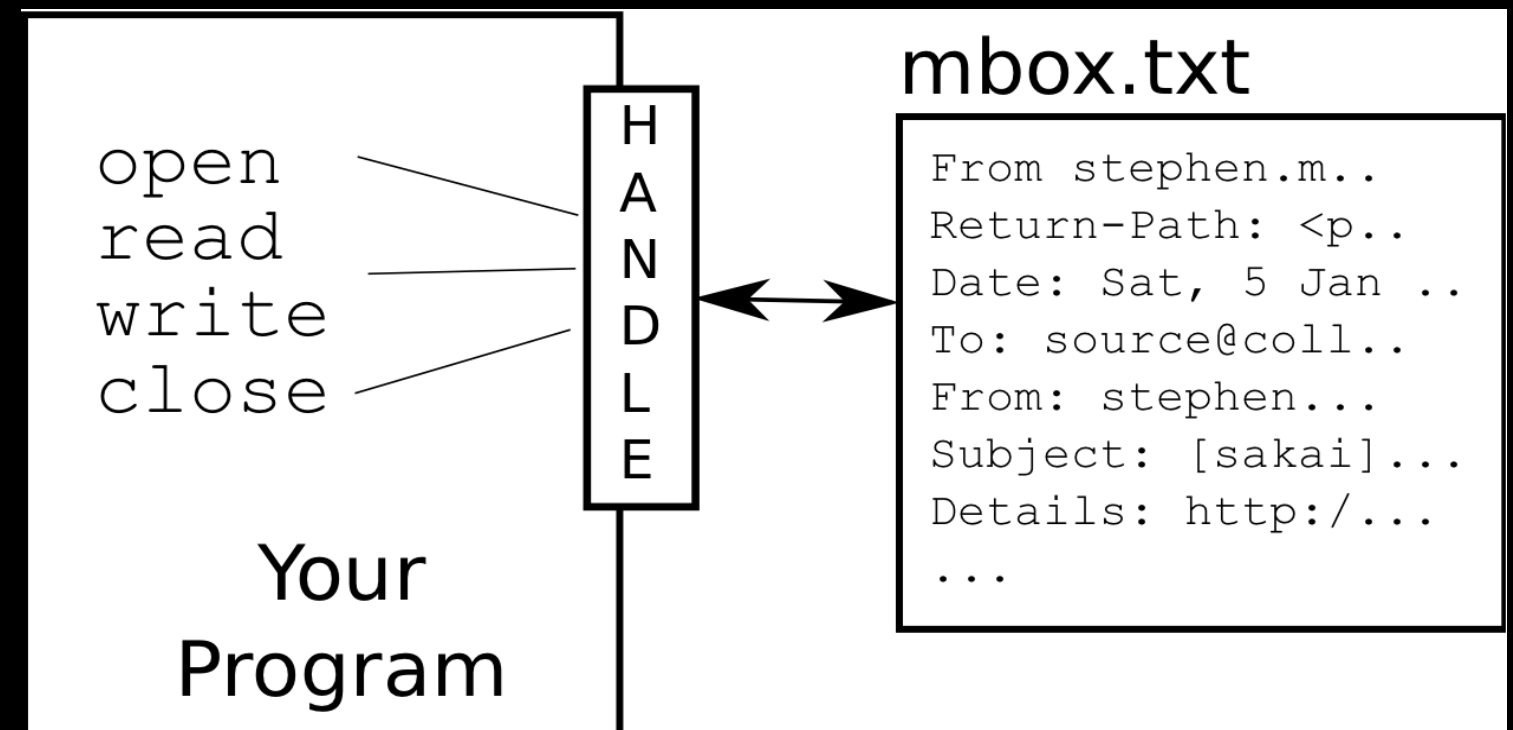
- `handle = open(filename, mode)`
- returns a handle use to manipulate the file
- filename is a string
- mode is optional and should be 'r' if we are planning to read the file and 'w' if we are going to write to the file

# File Access Modes in open() Function

- Access modes govern the type of operations possible in the opened file. It refers to how the file will be used once its opened.
- These modes also define the location of the File Handle in the file. File handle is like a cursor, which defines from where the data has to be read or written in the file.
- There are 6 access modes in python.

# What is a Handle?

```
>>> fhand = open('mbox.txt')
>>> print(fhand)
<_io.TextIOWrapper name='mbox.txt' mode='r' encoding='UTF-8'>
```



# File Access Modes (cont'd)

1. **Read Only ('r')** : Open text file for reading. The handle is positioned at the beginning of the file. If the file does not exist, raises I/O error. This is also the default mode in which file is opened.
2. **Read and Write ('r+')** : Open the file for reading and writing. The handle is positioned at the beginning of the file. Raises I/O error if the file does not exist.
3. **Write Only ('w')** : Open the file for writing. For existing file, the data is truncated and over-written. The handle is positioned at the beginning of the file. Creates the file if the file does not exist.
4. **Write and Read ('w+')** : Open the file for reading and writing. For existing file, data is truncated and over-written. The handle is positioned at the beginning of the file.
5. **Append Only ('a')** : Open the file for writing. The file is created if it does not exist. The handle is positioned at the end of the file. The data being written will be inserted at the end, after the existing data.
6. **Append and Read ('a+')** : Open the file for reading and writing. The file is created if it does not exist. The handle is positioned at the end of the file. The data being written will be inserted at the end, after the existing data.



# When Files are Missing

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

# Closing a File

`close()` function closes the file and frees the memory space acquired by that file. It is used at the time when the file is no longer needed or if it is to be opened in a different file mode.

`File_object.close()`

```
# Opening and Closing a file "MyFile.txt"
# for object name file1.
file1 = open("MyFile.txt","a")
file1.close()
```

# Reading Text Files in Python

# Reading Data from Text Files

There are three ways to read data from a text file.

1. **read()** : Returns the read bytes in form of a string. Reads n bytes, if no n specified, reads the entire file.

```
File_object.read([n])
```

2. **readline()** : Reads a line of the file and returns in form of a string. For specified n, reads at most n bytes. However, does not read more than one line, even if n exceeds the length of the line.

```
File_object.readline([n])
```

3. **readlines()** : Reads all the lines and return them as each line a string element in a list.

```
File_object.readlines()
```

**Note:** '\n' is treated as a special character of two bytes

# 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
- We can use the **for** statement to iterate through a **sequence**
- Remember - a **sequence** is an ordered set

```
xfile = open('mbox.txt')  
for cheese in xfile:  
    print(cheese)
```

# 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

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

```
$ python open.py
Line Count: 132045
```

# Reading the \*Whole\* File

We can **read** the whole file (newlines and all) into a **single string**

```
>>> fhand = open('mbox-short.txt')
>>> inp = fhand.read()
>>> print(len(inp))
94626
>>> print(inp[:20])
From stephen.marquar
```

# Searching Through a File

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

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



# OOPS!

What are all these blank  
lines doing here?

From: `stephen.marquard@uct.ac.za`

From: `louis@media.berkeley.edu`

From: `zqian@umich.edu`

From: `rjlowe@iupui.edu`

...

# OOPS!

What are all these blank lines doing here?

- Each line from the file has a **newline** at the end
- The **print** statement adds a **newline** to each line

```
From: stephen.marquard@uct.ac.za\n
\n
From: louis@media.berkeley.edu\n
\n
From: zqian@umich.edu\n
\n
From: rjlowe@iupui.edu\n
\n
...
```

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

```
From: stephen.marquard@uct.ac.za
From: louis@media.berkeley.edu
From: zqian@umich.edu
From: rjlowe@iupui.edu
....
```

# Skipping with `continue`

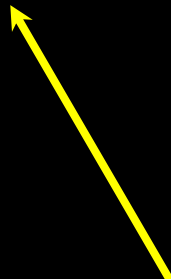
We can conveniently skip a line by using the `continue` statement

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

# Using `in` to Select Lines

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

```
fhand = open('mbox-short.txt')
for line in fhand:
    line = line.rstrip()
    if not '@uct.ac.za' in line :
        continue
    print(line)
```



```
From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008
X-Authentication-Warning: set sender to stephen.marquard@uct.ac.za using -f
From: stephen.marquard@uct.ac.za
Author: stephen.marquard@uct.ac.za
From david.horwitz@uct.ac.za Fri Jan  4 07:02:32 2008
X-Authentication-Warning: set sender to david.horwitz@uct.ac.za using -f...
```

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

# Prompt for File Name

Enter the file name: mbox.txt  
There were 1797 subject lines in mbox.txt

Enter the file name: mbox-short.txt  
There were 27 subject lines in mbox-short.txt

# Bad File Names

```
fname = input('Enter the file name: ')
try:
    fhand = open(fname)
except:
    print('File cannot be opened:', fname)
    quit()

count = 0
for line in fhand:
    if line.startswith('Subject:') :
        count = count + 1
print('There were', count, 'subject lines in', fname)
```

Enter the file name: mbox.txt

There were 1797 subject lines in mbox.txt

Enter the file name: na na boo boo

File cannot be opened: na na boo boo

# Writing to Text Files

There are two ways to write in a file.

1. **write()** : Inserts the string str1 in a single line in the text file.

```
File_object.write(str1)
```

2. **writelines()** : For a list of string elements, each string is inserted in the text file. Used to insert multiple strings at a single time.

```
File_object.writelines(L) for L = [str1, str2, str3]
```



# Writing vs. Appending to a Text File

- If the text file is opened in **write mode**, then using `write()` or `writelines()` methods overwrites the existing text in the file and the file is overwritten with the next text.

```
f_handle = open('my_file.txt', 'w')
```

- If the text file is opened in **append mode**, the file cursor moves to the end of the file, and the `write()` or `writelines()` methods append the new text at the end of the file.

```
f_handle = open('my_file.txt', 'a')
```

# Practice

- Write a program that maintains a to-do list.
- It repeatedly asks the user for to-do item, and inserts the to-do item in the file.

# Questions?