Applied Computing with Python

SAT 4650

Lecture 5

Dr. Neerav Kaushal

Department of Applied Computing

Outline

- Reading and writing files
- System administration with python

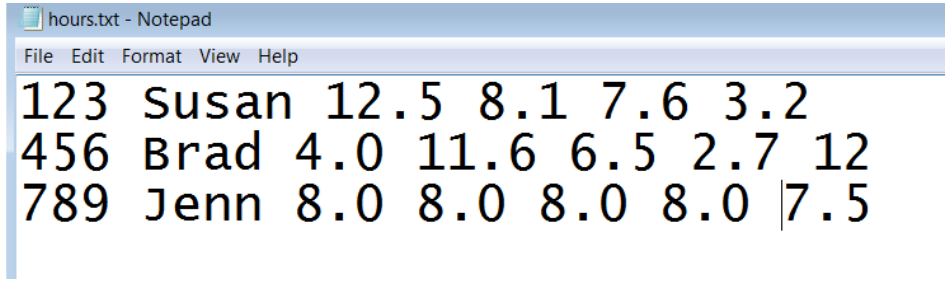
Reading and writing files

Michigan Tech

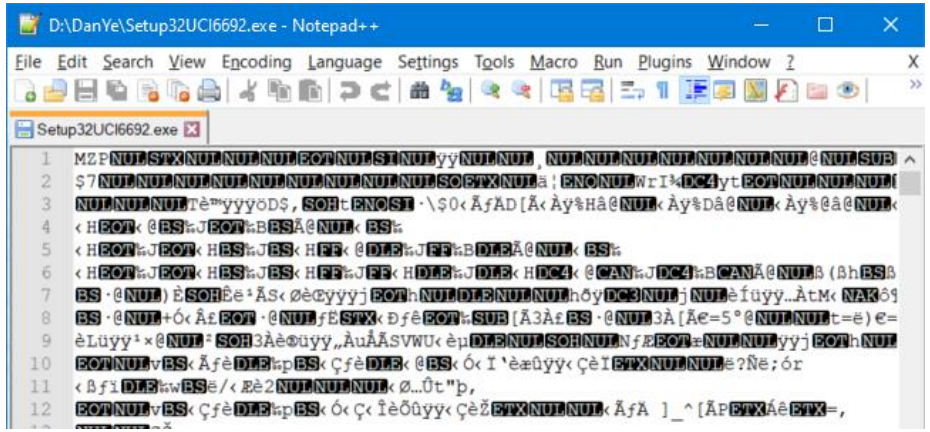
File types in python

- We often need to handle files in programming
- There are two types of files in python: Text files (default value) and binary files (such as images)
- Text files can be open with a text editor, and they are understandable by human beings directly while binary files are not understandable if you use text editor to open it.
- We will discuss how to handle text files

.txt file



.exe file



Michigan Tech

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. Format:

```
fhandle= open("filename", "Access mode")
```

"filename" Could be "directory+filename"

"Access 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

Mode:

- "r" - Read - Default value. Opens a file for reading, error if the file does not exist
- "a" - Append - Opens a file for appending, creates the file if it does not exist
- "w" - Write - Opens a file for writing, creates the file if it does not exist; overwrites the file if it exists
- "x" - Create - Creates the specified file, returns an error if the file exists

open() returns a "file handle object" to a variable - used to perform operations on the file

Examples:

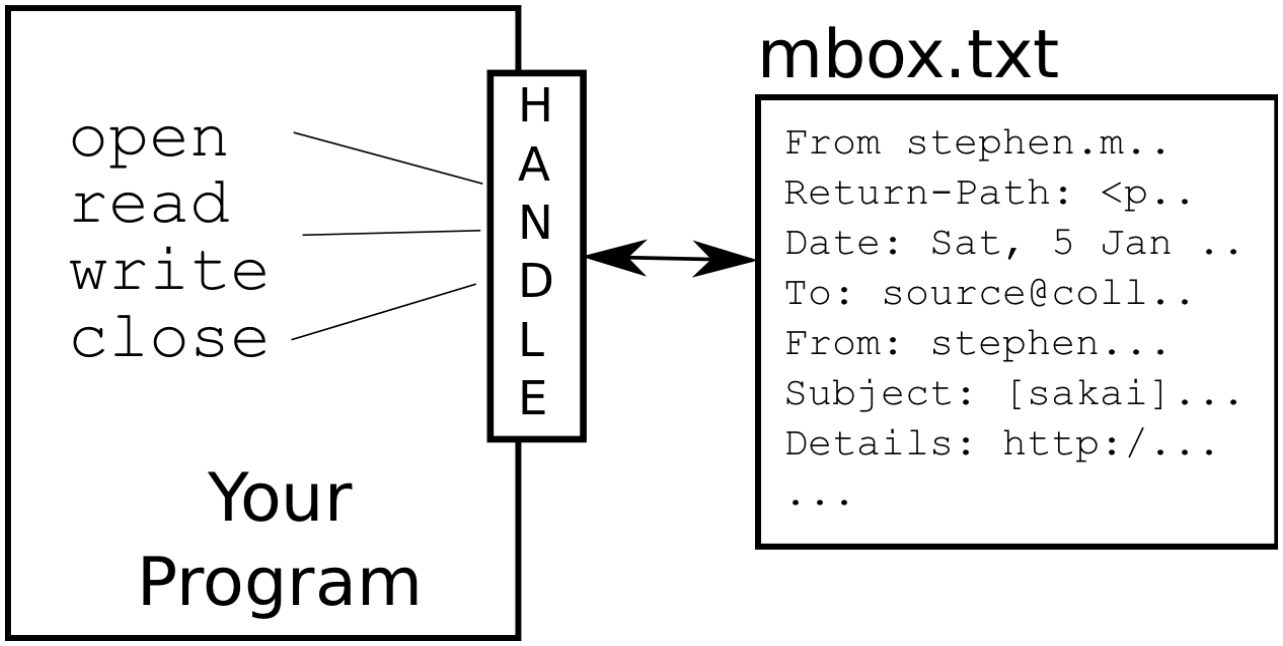
```
fhandle= open("C:/sat4310/temperature.txt", "r")
fhandle= open("C:/sat4310/output.txt", "w")
fhandle= open("C:/sat4310/output.txt", "a")
```

What is a File Handle?

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

"r" - Read - Default value

Don't forget to close the file after file operations: `fhandle.close()`



Ways to read a file

Show code

- fhandle= open("filename")
  - opens the given file for reading, and returns a file object
- fhandle.read()
  - reads entire contents as a string
- fhandle.readline()
  - reads next line from file as a string
- fhandle.readlines()
  - returns a list of all lines

Use a for loop
Each execution of the loop will read a line from the file into a string
Format:

```
for <variable to store a string> in <Fhandle>:
        <Do something with the string read from file>
```

Ways to write a file

```
fw = open("filename", "w")
fw = open("filename", "a")
```

- "w": opens file for write (deletes previous contents), or
- "a": opens file for append (new data goes after previous data)

```
fw.write(str)
```
- writes the given string to the file

```
fw.close()
```
- saves file once writing is done

Show real time code

```
>>> out = open("output.txt", "w")
>>> out.write("Hello, world!\n")
>>> out.write("How are you?")
>>> out.close()

>>> open("output.txt").read()
'Hello, world!\nHow are you?'
```

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
- Newlines are used for formatting outputs to the screen or to a file. Thus, they are often used in `print()` function and `write()` method

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

Example: Write to a file line by line

- `String1= 'He is a good person! He likes to eat fish! He is 21 years old!'`
- I want to write the above string to a file (out.txt) as follows:

  He is a good person!

  He like to eat fish!

  He is 21 years old!

Show code and output

```
# define the string
string1= 'He is a good person! He like to eat fish! He is 21 years old!'

# open string for writing
fhandle = open('out.txt','w')

# write to the file
fhandle.write(string1)
fhandle.close()
```

Michigan Tech

Modified Example for new line “\n”

```
# define the string
string1= 'He is a good person!\n
He like to  eat fish!\nHe is 21
years old!’

# open string for writing
fhandle = open('out.txt','w')
# write to the file
fhandle.write(String1)
fhandle.close()
```

Michigan Tech

Remove newline “\n” when reading file

- Use replace() method in string to replace “\n” with empty string “” to remove the new line.

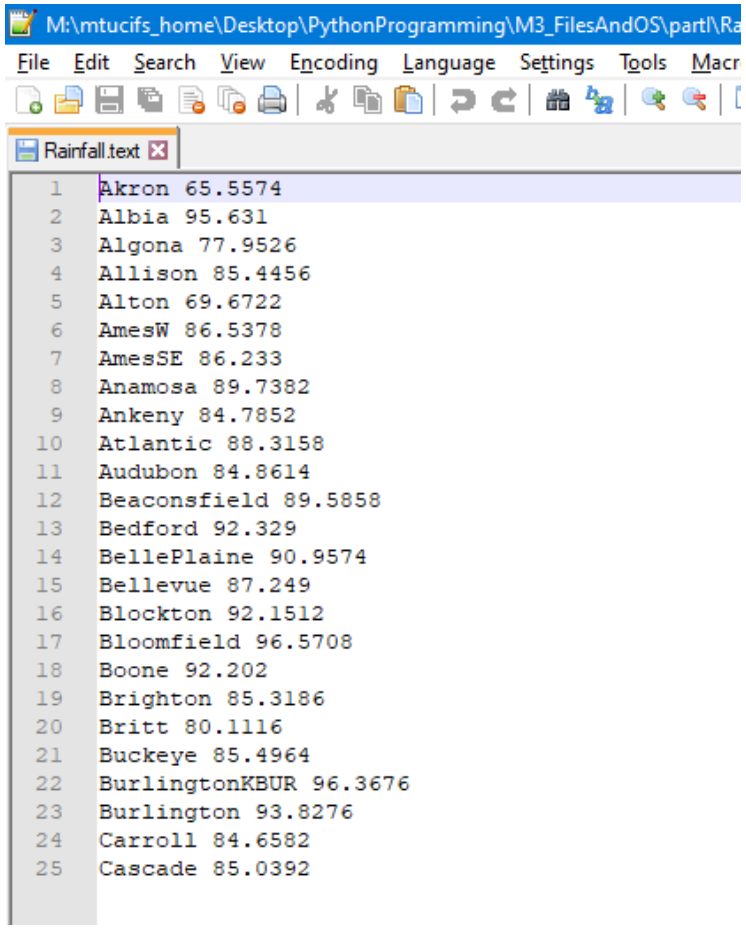
```
>>> out = open("output.txt", "w")
>>> out.write("Hello, world!\n")
>>> out.write("How are you?")
>>> out.close()
>>> open("output.txt").read()
    'Hello, world!\nHow are you?'

>>> open("output.txt").read().replace('\n','')
    'Hello, world! How are you?'
```

Another example

- Open a file called "**rainfall.text**" and display its information on the screen. Using the following format to display:

  The city of London had 55 .6 inches of rain




```
# open the file for reading
f_rain = open('rainfall.text', 'r')

#read the file
for aLine in f_rain:
        value = aLine.split() #new lines are treated as space
        print("The city of ", value[0], "had", value[1], "inches of rain")
```

Using in to select lines

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

```
fhand = open('c:/sat4310/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: nakamura.uits.iupui.edu: apache 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: nakamura.uits.iupui.edu: apache set sender to david.ho
rwitz@uct.ac.za using -f
```

Delete a file using python (Careful here!!!)

- To delete a file, you must import the OS module, and run its `os.remove()` function

- Example

```
import os
os.remove("demofile.txt")
```

- Check if file exist:

```
import os
if   os.path.exists("demofile.txt"):
     os.remove("demofile.txt")
else:
     print("The file does not exist")
```

Michigan Tech

When files are missing

```
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)
```

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

Enter the file name: TTTT
Traceback (most recent call last):
  File "L4_count1.py", line 2, in <module>
    fhand = open(fname)
FileNotFoundError: [Errno 2] No such file or directory: 'TTTT'
```

Errors handled by python itself

Python program recognizes and prints the error message as follows:

```
Traceback (most recent call last):
  File "L4_count1.py", line 2, in <module>
    fhand = open(fname)
FileNotFoundError: [Error 2] No such file or directory: 'TTTT'
```

Can we, as programmers, write code to handle errors and specify the output “message”?

and comes the infamous try and except

```
try:
        suite
except a_particular_error:
        suite 1
except a_particular_error:
        suite 2
…
```

- An `except` suite (perhaps multiple `except` suites) is associated with a `try` suite.
- Each exception names a type of exception it is monitoring for.
- If the error that occurs in the `try` suite matches the type of exception, then that `except` suite is activated.

- the `try` suite contains code that we want to monitor for errors during its execution.
- if an error occurs anywhere in that `try` suite, Python looks for a handler that can deal with the error.
- if no special handler exists, Python handles it, meaning the program halts and with an error message as we have seen so many times

So, let's handle the errors now

The `try` block lets you test a block of code for errors

The `except` block lets you handle the error if the errors found in `try` block

```
fname = input('Enter the file name:  ')
try:
    fhand = open(fname)
except FileNotFoundError:
    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)
```

- if no exception in the `try` suite, skip all the `try/except` to the next line of code
- if an error occurs in a `try` suite, look for the right exception
- if found, run that `except` suite and then skip past the `try/except` group to the next line of code
- if no exception handling found, give the error to Python

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

Enter the file name: TTTT
File cannot be opened: TTTT
```

Multiple errors

```
fname = input('Enter the file name:  ')
try:
    fhand = open(fname)
    print(fa)
except FileNotFoundError:
    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)
```

```
Enter the file name: mbox.txt
Traceback (most recent call last):
  File "L4_count2.py", line 5, in <module>
    print(fa)
NameError: name 'fa' is not defined
```

Michigan Tech

Handle multiple errors

```
fname = input('Enter the file name:  ')
try:
    fhand = open(fname)
    print(fa)
except FileNotFoundError:
    print ('File cannot be opened:', fname)
    exit()
except NameError:
    print("it is not file open error")
     exit()
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
it is not file open error
```

Modules for system administration with python

Michigan Tech

OS module

- Writing scripts for system administration automation is an important task for system administrator
- Many system administration tasks can be done by python scripting automatically.
- The OS module in Python provides a way to achieve that.
- The functions in OS module allows you to interface with the underlying operating system that Python is running on. (Windows, Mac or Linux.)
- Before we start, make sure that you have imported the OS module "`import os`"
- os module comes preinstalled with python, no need to install it.
- Show students how to install and import modules in a new environment (side learning)

Functions in OS module - 1

```
# Get current working directory with os.getcwd()
import os
print(os.getcwd())

# Get the status of a file with os.stat()
print("Getting the status of:" ,
os.stat('C:\\sat4310\\UIC\\CA.py'))

# Execute a shell command with os.system()
os.system('dir -l *')

# Get the PATH information
os.getenv('PATH')
```

Show code in real time

Functions in OS module - 2

Show code in real time

```
#Create a directory with os.makedir()
os.mkdir('new_directory')
#Recursive create directories with os.makedirs()
os.makedirs('dir_a\\dir_b\\dir_c')
#Remove a directory with os.rmdir()
os.rmdir('directory')
#Recursively remove empty directories with os.rmdirs()
os.removedirs('dir_a\\dir_b\\dir_c')
#Rename a file with os.rename(src, dst)
os.rename('c:\\sat4310\\tprint.txt', 'c:\\sat4310\\tprint_test.txt' )
#Move focus to a different directory with os.chdir()
os.chdir('c:\\sat4310\\temp')
print(os.getcwd())
os.chdir('c:\\sat4310')
```

Functions in OS module - 3

```
#Print out all directories, sub-directories and files with os.walk()
import os
for root, dirs, files in os.walk('c:\\sat4310'):
      print(root)
      print(dirs)
      print(files)

#Get the last time a directory was accessed with os.path.getatime()
os.path.getatime('c:\\sat4310')

#Get the last time a directory was modified with os.path.getmtime()
os.path.getmtime('c:\\sat4310 ')

#Returns a list of all files in a directory with os.listdir()
for filename in os.listdir('c:\\sat4310’):
    print("This is inside /temp", filename)
```

Show code in real time

Michigan Tech

Subprocess module

- The **subprocess** module allows you to spawn new processes, connect to their input/output/error pipes, and obtain their return codes.
- For a complete description of this module see the Python documentation.

  https://docs.python.org/3/library/subprocess.html#frequently-used-arguments
- run method in subprocess module:

  Subprocess.run() can be used to execute external command
- Example 1

```
import subprocess
#Launch windows application
subprocess.run("notepad.exe")
```

Example

Problem: Find a files named “hacker_attck.bin” and delete them in a specific directory and its subdirectory

```
import os
root ="C:\\sat4310"
for directory, subdir_list, file_list in os.walk(root):
      print('Directory:', directory)
      for name in subdir_list:
          print('Subdirectory:',name)
      for name in file_list:
             if name == 'hacker_attck.bin':
             print('I found a hacker file')
             os.remove(directory+"\\"+name)
          else:
             print('File:',name)
      print()
```

String comparisons, single char

- In system administration, we often need to handle strings and need to compare strings
- Python 3 uses the Unicode mapping for characters
- UTF-8, subset of Unicode, takes the English letters, numbers and punctuation marks and maps them to an integer
- Single character comparisons are based on that number
- You can use ord() function to get the number of the character

UTF-8 subset

UTF-8

	_0	_1	_2	_3	_4	_5	_6	_7	_8	_9	_A	_B	_C	_D	_E	_F
(1 byte) 0_	NUL 0000	SOH 0001	STX 0002	ETX 0003	EOT 0004	ENQ 0005	ACK 0006	BEL 0007	BS 0008	HT 0009	LF 000A	VT 000B	FF 000C	CR 000D	SO 000E	SI 000F
(1) 1_	DLE 0010	DC1 0011	DC2 0012	DC3 0013	DC4 0014	NAK 0015	SYN 0016	ETB 0017	CAN 0018	EM 0019	SUB 001A	ESC 001B	FS 001C	GS 001D	RS 001E	US 001F
(1) 2_	SP 0020	! 0021	" 0022	# 0023	$ 0024	% 0025	& 0026	' 0027	(0028	) 0029	* 002A	+ 002B	, 002C	- 002D	. 002E	/ 002F
(1) 3_	0 0030	1 0031	2 0032	3 0033	4 0034	5 0035	6 0036	7 0037	8 0038	9 0039	: 003A	; 003B	< 003C	= 003D	> 003E	? 003F
(1) 4_	@ 0040	A 0041	B 0042	C 0043	D 0044	E 0045	F 0046	G 0047	H 0048	I 0049	J 004A	K 004B	L 004C	M 004D	N 004E	O 004F
(1) 5_	P 0050	Q 0051	R 0052	S 0053	T 0054	U 0055	V 0056	W 0057	X 0058	Y 0059	Z 005A	[005B	\ 005C	] 005D	^ 005E	_ 005F
(1) 6_	` 0060	a 0061	b 0062	c 0063	d 0064	e 0065	f 0066	g 0067	h 0068	i 0069	j 006A	k 006B	l 006C	m 006D	n 006E	o 006F
(1) 7_	p 0070	q 0071	r 0072	s 0073	t 0074	u 0075	v 0076	w 0077	x 0078	y 0079	z 007A	{ 007B	| 007C	} 007D	~ 007E	DEL 007F

Michigan Tech

Comparisons within sequence and words

- Counterintuitively, it makes sense to compare within a sequence (lower case, upper case, digits)
  - `'a' < 'b'` → `True`
  - `'A' < 'B'` → `True`
  - `'1' < '9'` → `True`
  - `'a' < 'A'` → `False`
  - `'a' < '0'` → `False`
- Compare the first element of each string
  - if they are equal, move on to the next character in each
  - if they are not equal, the relationship between those to characters are the relationship between the string
  - if one ends up being shorter (but equal), the shorter is smaller

`'a' < 'b'` → `True`

`'aaab' < 'aaac'`

first difference is at the last char. 'b'<'c' so 'aaab' is less than 'aaac'. True

`'aa' < 'aaz'`

The first string is the same but shorter. Thus, it is smaller. True

Michigan Tech