

ENGG1811 Computing for Engineers

Week 7A: File handling

Motivations

- As an engineer, you may work with data
 - Sometimes these data come in data files and there can be many of them
- For example, in the following video, which was made especially for ENGG1811, Dr. Emily Moylan from Civil and Environmental Engineering at UNSW talked about her work on working with data coming from inductive loops, which are devices embedded below the road surface for measuring number of vehicles passing over the loop etc.
 - <https://www.youtube.com/watch?v=CR-bwYiT-IM#t=4m43s>
 - This link takes you to 4m43s into the video. From this point on till 5m46s, Dr. Moylan talked about the importance of using programming to deal with many data files
 - <https://www.youtube.com/watch?v=CR-bwYiT-IM>
 - Link to the full video

File types

- There are two types of files:
 - Text, e.g., Python source code
 - Binary, e.g., image files
- We won't go into the technical details and we will just say text files are human readable but binary files are not
- You can view the contents of text file using a text editor (e.g., Notepad on Windows, and TextEdit on Apple Mac)
 - For binary files, see next page
- We will only work with text files in ENGG1811

Binary files

- If you open a binary file with a text editor, you see gibberish

Image Viewer



Text Editor

```
âPNG
IHDR,»»ΩK pHYSöü
OICCPPhotoshop ICC
profilexûSgTSE~fiÛBKâÄiKoR RBãÄë&*! Jà!
*YQjEE»†àééÄâQ,â
ÿ%!çéÉ£ää ·{£k+°ÉÖ,μð>Á~Ûùzœ¿ñH3Q5Ä
©B†É«fΔ·%ø.@Ä
$pzdl$"#~-<<+~¿æx"¿Mö¿0á~ÍBô
VÄN¿të8KÄ@zëB¶@FÄüð&S†`Àcb,P-`É*Äü~ð{¶¶†ë
eâDh;~œVäEX0fKf9ÿ-0IW¶H=Σ¿œ≤0QaÖ)
{`»##xNðFÛW<Ö+ÆÄ*xð≤<π$9EA[-qWW.(œI
+6aaö@.~yð2Ä4†ÛÄ†ë†ÉÛ~xœÆœœœ6éð_-
lè~"bb,Äœ`p@t—,¿Ä;Äm,ç%Óh^
```

https://fileinfo.com/help/binary_vs_text_files

Revision on strings

- In order for you to work with text files, you will need to know how to manipulate strings.
- Skills needed are
 - Concatenating strings
 - Splitting strings
 - Conversion between int, float and str
- Python file: `working_with_strings.py`

File operations

- When you work with files, there are three standard steps
 1. Open the file
 2. Read data from the file or write data to the file
 3. Close the file

Opening and closing files

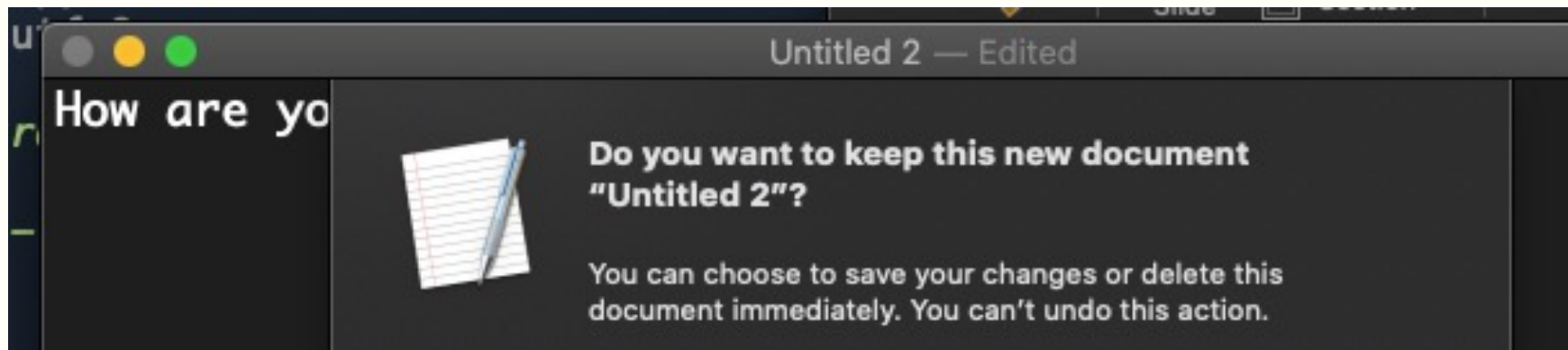
- Python provides two methods to open and close files
- Method 1: Use `open()` and `close()`
- Method 2: Using the keyword *with* and `open()`
 - The advantage of using the keyword *with* is that it will automatically close the file
- The function `open()` takes in two inputs:

```
open( file, mode)
```

- The input parameter `mode` is to specify what you want to do with the file
 - `'r'` is to read, `'w'` is to write, `'a'` to append
 - `'b'` for binary

Closing a file

- You probably have seen this type of message which asks whether you want to save a file



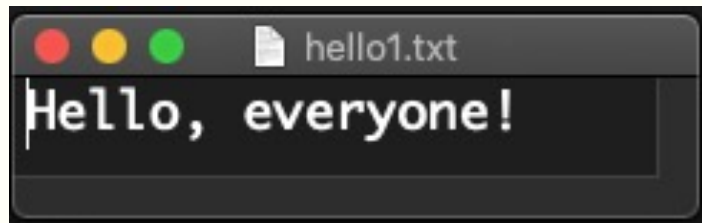
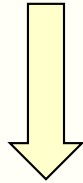
- Closing a file means telling the computer that:
 - I've finished with the file, please save it.

Writing file

- We will first learn how to write to files
- We will go through these Python programs one by one
 - `write_file_1A.py`, `write_file_1B.py`
 - `write_file_2.py`
 - `write_file_3.py`
- You will now work on an exercise in `write_file_ex_prelim.py`
 - In the directory `samples`, you will find three files `temp_Sydney.txt`, `temp_Hobart.txt`, `temp_Brisbane.txt` which are samples of the files that you want to produce in this exercise

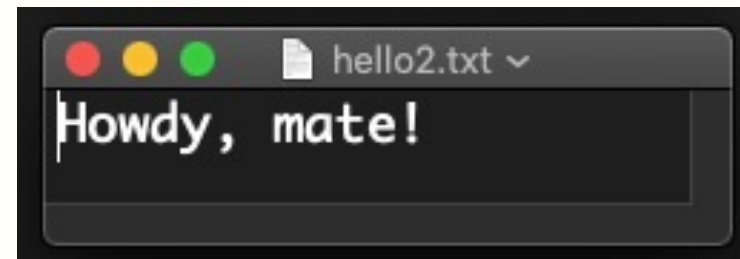
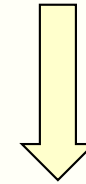
write_file_1A.py

```
# %% Method 1: open() and close()
outfile = open('hello1.txt', 'w')
outfile.write('Hello, everyone!')
outfile.close()
```



write_file_1B.py

```
# %% Method 2: Using with
with open('hello2.txt', 'w') as outfile:
    outfile.write('Howdy, ' + 'mate!')
```



- The two methods are very similar. The key difference is:
 - You need to close the file in write_file_1A.py
 - Once all the code under `with` is completed, the file is closed.

Outputs of write_file_2.py

1st cell

quote1.txt

```
What I cannot create, I cannot understand. (Richard Feynman)To live, to err, to  
fall, to trimuph, to recreate life out of life. (James Joyce)See things not as they  
are, but as they might be. (American Prometheus)
```

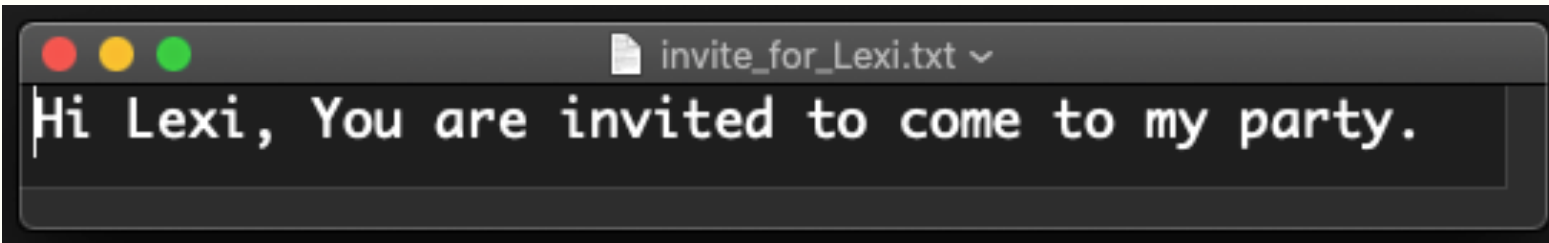
2nd cell

quote2.txt

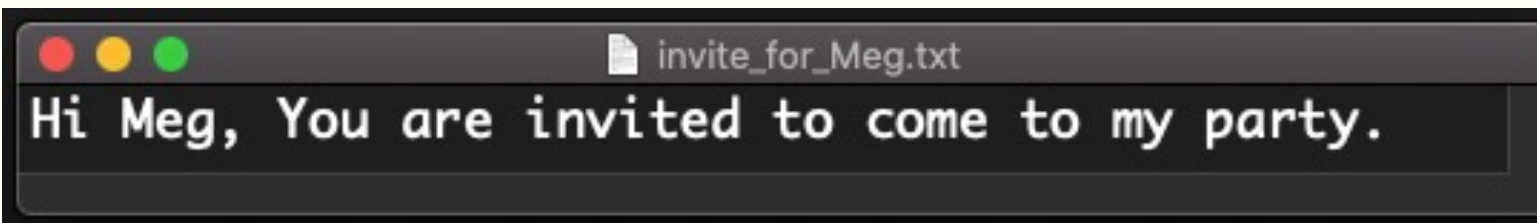
```
What I cannot create, I cannot understand. (Richard Feynman)  
To live, to err, to fall, to trimuph, to recreate life out of life. (James Joyce)  
See things not as they are, but as they might be. (American Prometheus)
```

Outputs of write_file_3.py

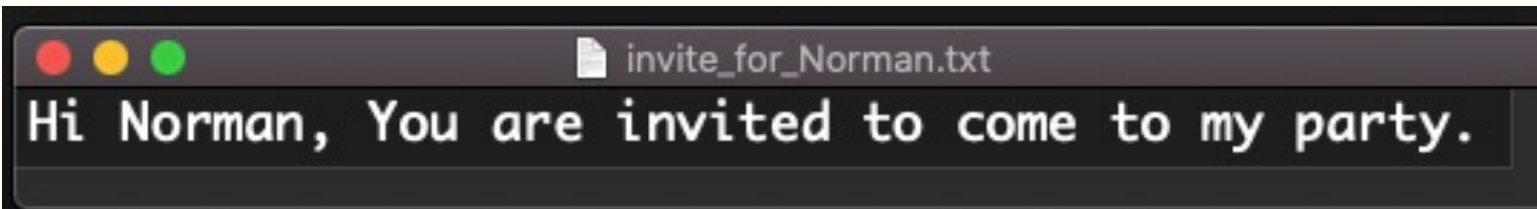
- It produces 3 files with similar names and similar contents
- Can use lists and loops to better organize the code



```
invite_for_Lexi.txt
Hi Lexi, You are invited to come to my party.
```



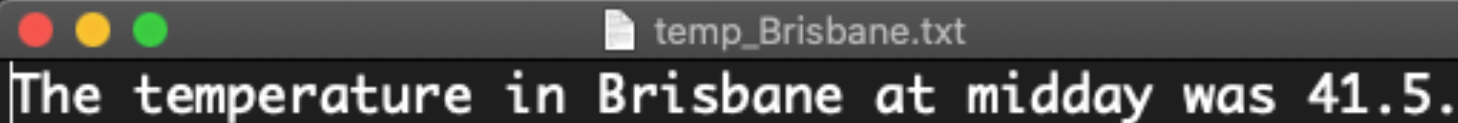
```
invite_for_Meg.txt
Hi Meg, You are invited to come to my party.
```



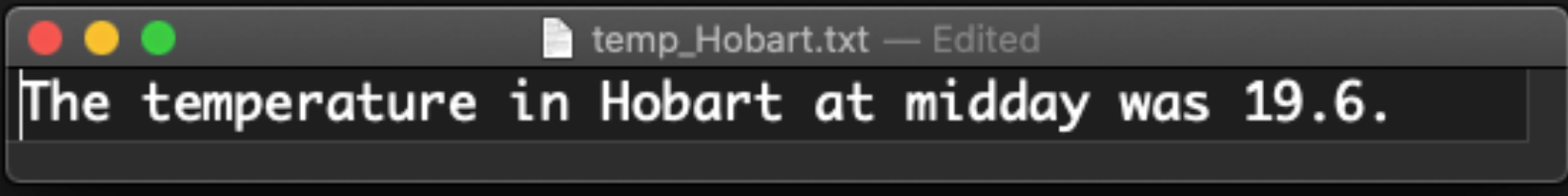
```
invite_for_Norman.txt
Hi Norman, You are invited to come to my party.
```

Exercise

- You will now work on an exercise in `write_file_ex_prelim.py`
 - The aim of the code is to create three files with names `temp_Sydney.txt`, `temp_Hobart.txt`, `temp_Brisbane.txt`
 - The contents of the file are shown below.
 - You need to complete
 - Line 36, the variable `filename` will be used in `open()` in Line 41
 - Line 37, the variable `line_to_write` will be used in `outfile.write()` in Line 42



The temperature in Brisbane at midday was 41.5.



The temperature in Hobart at midday was 19.6.



The temperature in Sydney at midday was 23.5.

Difference between write and append

- In many programming languages,
 - Write means
 - Creating the file if it doesn't exist
 - If the file exists, **erase** the existing contents
 - *Make sure you do want to erase*
 - Append means
 - Creating the file if it doesn't exist
 - If the file exists, add data to the end of the file

Reading files

- There are three functions to read the contents of a file:
 - `read()` returns the contents of the entire file in a string
 - `readlines()` returns the contents of the entire file in a list (note: it is a list of strings)
 - `readline()` reads a line at a time and returns the contents in a string
- See examples in `read_file.py`
- Since the output of the above functions is a string or a list of string, we explain how you can handle these strings in `read_file.py`

Python functions for reading or writing files

- Useful numpy functions:
 - loadtxt() can read in an array
 - See example in read_file.py
 - savetxt() writes an array to a text file
 - <https://docs.scipy.org/doc/numpy/reference/generated/numpy.savetxt.html>
 - save() and load() for saving and writing numpy arrays to binary format (file extension is npz)
 - <https://docs.scipy.org/doc/numpy/reference/generated/numpy.save.html>
 - <https://docs.scipy.org/doc/numpy/reference/generated/numpy.load.html>

Summary

- Writing and reading files
- Opening and closing files
- String manipulations
 - Convert to and from string
 - Split a string into substrings
 - Concatenate strings