Introduction to Computer Programming

Week 7.1: Reading & Writing Files

Open Spyder now

Reading files : Importing data (e.g. experiment results) into a program

Writing files : Exporting data - storing data outside of the program.
(e.g. output of a calculation)

Delimited file:
Uses a set character (tab, space, vertical bar etc) to separate values.

CSV (comma-seperated-value):
A *delimited* text file that uses a comma to separate values.

The CSV file is a widely used format for storing tabular data in plain text and is supported by software applications e.g. Microsoft Excel, Google Spreadsheet.

	A	B	C	D	E	F	G	H	I
1	sample	moisture	knotratio	treering	Edyn	density	beamheig	Estat	bstrength
2	units	%	-	mm	N/mm2	kg/m3	cm	N/mm2	N/mm2
3	DO1	13.3	0.04	3	14053	675	101	15452	58.4
4	DO2	12	0.16	2.5	20611	474	100	17272	74.35
5	DO3	12.8	0.14	3.88	18846	596	99	18456	49.82
6	DO4	11.7	0.13	2.02	18587	582	100	18940	78.52
7	DO5	12	0.16	2.13	19299	678	100	16864	79.31
8	DO6	12.4	0.04	2.98	21695	595	100	19440	64.34
9	DO7	12.5	0.32	3.67	16523	592	100	16152	58.19
10	DO8	11.5	0.07	3.67	18333	634	101	18480	88.39
11	DO9	13.1	0.19	2.44	18628	592	101	14604	33.02
12	DO10	11.7	0.25	3	15683	540	101	16628	60.28
13	DO11	13.2	0.2	3.75	18496	605	100	18476	91.86
14	DO12	11.9	0.11	1.96	19792	646	101	19212	81.88
15	DO13	12.3	0.03	2.4	20098	618	101	19608	91.02

Built-in Python functions for reading and writing text data files (.txt, .csv, .dat):

- `open()`
- `write()`
- `close()`

Before a file can be read or written to, it must be opened using the `open()` function.

```
open(file_path, mode_specifier)
```

Mode specifier:
An open file can be read, overwritten, or added to, depending on the mode specifier used to open it.

Mode specifier	Read (R)/Write (W)	File must already exist	If no file exists	`write()`	Stream position when opened
r	R	Yes	N/A	N/A	start
w	W	No	Creates new file	overwrites all previous contents	start
a	W	No	Creates new file	appends text to end of file	end
r+	R+W	Yes	N/A	overwrites previous contents	start
w+	R+W	No	Creates new file	overwrites all previous contents	start
a+	R+W	No	Creates new file	appends text to end of file	end

append: start writing at end of file
write: start writing at beginning of file

Importing a file from a different directory

When using `open` we must give the full file path.

Like when importing Python files/modules, often we want to read/write a file located in a different directory.

The directory must already exist.

The file does not need to already exit if writing (`a` , `a+` , `w` , `w+`)

Downstream file location

`/` is used to indicate a sub-directory downstream of the current location.

```
Documents/
│
├── Folder_1/
│   └── myScores.txt
│
├── Folder_2/
│   └── scores.txt
│
└── read_write.py
```

Example: Open a downstream file within `read_write.py` :

```
f = open('Folder_1/myScores.txt', 'w')
```

Upstream file location

`../` is used to indicate a location one directory upstream of the current location.

```
Documents/
│
├── Folder_1/
│   └── read_write.py
│
├── Folder_2/
│   └── scores.txt
│
└── myScores.txt
```

Example: Open an upstream file within `read_write.py` :

```
f = open('../myScores.txt', 'w')
```

Example: Open a file in a different directory at the same level as the directory containing `read_write.py:

```
f = open('../Folder_2/scores.txt', 'w')
```

Once the file is open, it creates a *file object*.

A class object (instance of a class) can have methods.

Methods are actions or functions that the object is able to perform (`write` , `close` ...)

Writing files `w`

`write` can be used to write string data to a text file.

```
f = open('my_file.txt', 'w') # mode specifier to write

f.write('hello world')

f.close()
```

A file type that is often used to store tabulated data is the .csv file.

.csv files can be opened in spreadsheet programs like excel

A .csv file is simply a text file, with row items separated (or *delimited*) by commas.

Example:
Write the high score table shown to a new file with the filename scores.csv

Elena	550
Sajid	480
Tom	380
Farhad	305
Manesha	150

In [362]:
```
names = ['Elena', 'Sajid', 'Tom', 'Farhad', 'Manesha']
scores = [550, 480, 380, 305, 150]
```

In [ ]:
```

```

Try it yourself

Example:
Write the high score table shown to a new file with the filename scores.txt

Elena	550
Sajid	480
Tom	380
Farhad	305
Manesha	150

In [ ]:
```
1
```

Closing Files

Why do we need to close a file?

1. Not automatically closed.
2. Saves changes to file.
3. Can prevent other prgrams from accessing file

`close` is just a method, belonging to the file object.

The simplest open-close process is shown.

This will erase the contents of / create a new file `file.txt` in the folder `sample_data`

In [363]:
```
1
2 open('sample_data/file.txt', 'w').close()
```

Appending files `a`

Example: Append (add a new entry to the end of) scores.txt so that the table reads

(Code structure identical to write, apart from mode specifier)

Elena	550
Sajid	480
Tom	380
Farhad	305
Manesha	150
Jen	100

In [ ]:
```
1
2
```

Reading Files `r`

(Default argument so mode specifier can be omitted.)

File object is:

- iterable (can use for loop etc)
- not subscriptable (cannot index individual elements)

In [ ]:
```
1
```

The **stream position**:

- can be thought of as a curser.
- goes to end of file when an operation run on file object
- can be returned to start (or any position) with `seek`

In [364]:
```
f = open('sample_data/scores.txt', 'r')


for line in f:  # iterable
    print(line, end='') # each line is a string

#f.seek(0)      # stream position goes to end of file when oper
                # can be returned to start with seek

for line in f:
    print(line, end='')

f.close()
```

```
Mia 700
Jo 20
Ola 500
Ria 460
```

If we convert the file object to a list:

- it is subsriptable
- the stream position of the list doesn't need to be reset after each operation
- the stream position of the file object is at the end of the file after the list conversion operation

Example:

Print the list of names and a list of scores from the file `'sample_data/scores.txt'`

Print the name and score of the winner from the file `scores.txt'`

In [ ]:
```

```

Try it yourself

Example:

Print the first three names and scores from the file you created earlier `'scores.txt'`

In [ ]:
```

```

Reading and Writing with `r+, w+, a+`

All modes can be used to read and write files.

Differences that determine which to use:

- Stream position when opened
- How the stream position when opened affects `write()`

Mode specifier	Read (R)/Write (W)	File must already exist	If no file exists	`write()`	Stream position when opened
r+	R+W	Yes	N/A	overwrites previous contents	start
w+	R+W	No	Creates new file	overwrites all previous contents	start
a+	R+W	No	Creates new file	appends text to end of file	end

a+

Example: When we want to edit (append only) and read.

The stream position is:

- at the *end* when opened (must be moved to the start to read).
- always moved to the *end* before writing when `write` is called (previous contents never overwritten).
- at the *end* after writing.

In [ ]:
```
1
```

r+

Example: When we want to read and edit.

The stream position is:

- at the *start* when opened
- at the *end* after reading

In [ ]:
```
1
```

w+

Example: When we want to overwrite file then read

The stream position is:

- at the *start* when opened (previous contents overwritten).
- at the *end* after writing (subsequent lines added using `write` will appended the file, not overwrite previous contents, until file is closed).

Writing *must* happen before reading.

In [ ]:
```
1
```

Editing file contents - a word of warning!

Unlike the `a+` mode specifier `r+` and `w+` allow writing from *anywhere* in the file.

When characters are overwritten, this can lead to unexpected behaviour.

Example: Replace the first two items in the table in scores.txt with two new entries.

In [365]:
```
1  f = open('sample_data/scores.txt', 'r+') # stream position at s
2
3
4  f.write('Sid 50\n')                      # overwrite
5  f.write('Jo 20\n')
6
7  f.seek(0)
8
9  for line in f:                           # read
10     print(line, end='')
11
12
13 f.close()
```

```
Sid 50
Jo 20

Ola 500
Ria 460
```

The original data is longer than the replacement data.

Some original charcters are overwritten with new letters and `'\n'` characters.

The extra characters are left as they were in the original file.

```
Tim 50\nMajid  500\n
Sid 50\nJo 20\n
```

It is advisable to:

1. convert the data you want to edit to an format to a easy-to-edit Python data structure
2. overwrite the original file

Example: Edit the file to remove the unwanted line that reads `500` (between Jo and Ola).

The file can be erased from a position onwards with `truncate()`, (default position is current position)

In [ ]:
```

```

In [366]:
```
f = open('sample_data/scores.txt', 'r+')

for line in f:                # read file contents
    print(line, end='')

f.close()
```

```
Sid 50
Jo 20

Ola 500
Ria 460
```

Automatically closing files

It can be easy to forget to close a file with `close()`

`with open()` can be used instead of `open()` to remove the need for `close()`:

In [367]:
```
with open('sample_data/scores.txt', 'a') as f:
    f.write('Ria 460 \n')

print('next bit of the program') # Code unindents. File automat
```

```
next bit of the program
```

In [368]:
```
with open('sample_data/scores.txt', 'r') as f:
    print(f.read())
```

```
Sid 50
Jo 20

Ola 500
Ria 460
Ria 460
```

Summary

- Python functions for reading and writing files: `open()`, `read()`, `write()`, `close()`
- The **mode specifier** defines operations that can be performed on the opened file
- Files must always be closed after opening
- Files can be automatically closed by opening with `with open`

Extra Example: Change the first row of scores.txt to Mia

In [369]:

```
with open('sample_data/scores.txt', 'r+') as f:

    file = list(f)                          # convert to list of

    L = [line.split() for line in file]     # list of lists
    print(L)
    names = [i[0] for i in L]               # names and scores
    scores = [i[1] for i in L]

    ###################################

    names[0] = 'Mia'                        # replace element 0
    scores[0] = '700'

    f.seek(0)                               # go to start

    for n, s in zip(names, scores):
        f.write(n + ' ' + s + '\n')         # over write original

    f.truncate()                            # trim any trailing c

```

```
[['Sid', '50'], ['Jo', '20'], [], ['Ola', '500'], ['Ria', '460'],
['Ria', '460']]

---------------------------------------------------------------------
---------
IndexError                                Traceback (most recent c
all last)
<ipython-input-369-6891327a88c9> in <module>
      5     L = [line.split() for line in file]   # list of lists
      6     print(L)
----> 7     names = [i[0] for i in L]              # names and scor
es
      8     scores = [i[1] for i in L]
      9

<ipython-input-369-6891327a88c9> in <listcomp>(.0)
      5     L = [line.split() for line in file]   # list of lists
      6     print(L)
----> 7     names = [i[0] for i in L]              # names and scor
es
      8     scores = [i[1] for i in L]
      9

IndexError: list index out of range
```

In [ ]:

```

```