SOFTWARE 1 PRACTICAL

I/O

Week 5 – Practical 4b

To date we have use variables to store set of data, however when the program exits, all data is lost. There is a need for persistent data. To alleviate the problem, we use files to store the data persistently. There are several types of files, binary files, random access files, but the simplest to use are text files. Our focus this year will be only on text files.

PART 1:

Table 1: Summary of method used to write to a text file.

Method name	use	Explanation
`open`	f = open(filename,'w')	Open a file called filename for write only. Return a <file> object. See help(file) in Python shell. When using the option 'w', the previous content of the file is erased.
`open`	f = open(filename,'a')	Open a file called filename for write only. Return a <file> object. There is an important difference compared to the 'w' option. The previous content of the file is not erased, and write statements append at the end of the file.
`close`	f.close()	Close the file.
`write`	f.write(aString)	Add the string aString at the end of the file. f must be opened (using open in row 2) and not closed (row 3 not used yet).

Exercise 1:

Write a script to save the content of a string variable `a_word` into a file called `exo1.txt`.

Exercise 2:

Write a function `save_list2file(sentences, filename)` that takes two parameters, where `sentences` is a list of string, and `filename` is a string representing the name of the file where the content of `sentences` must be saved. Each element of the list `sentences` should be written on its own line in the file `filename`.

Exercise 3:

Write a function `save_to_log(entry, logfile)` that takes two parameters, a string `entry` to be written at the end of the text file named `logfile` (also a string). The previous content of the `logfile` MUST NOT be erased.

PART 2

Now that we have managed to save data to a file, it would be quite useful to read the data back into variables. To know more about Python 3.x I/O, including binary files that are not in the Software 1 learning outcomes, there is a more in-depth tutorial on the website dive into python.

Table 2: Summary of method used to read a text file.

Method name	use	Explanation
`open`	f = open(filename,'r')	Open a file called filename for read only. Return a <file> object. See help(file) in Python shell.
`read`	f.read() f.read(n)	Reads and returns a string of `n` characters, or the entire file content as a single string if `n` is not provided
`readline`	f.readline()	Reads and returns the next line of the file with all text up to and including the newline character (`'\n'`).
`readlines`	f.readlines() f.readlines(n)	Reads and returns a list of `n` strings each representing a single line of the file. If `n` is not provided, then all lines of the file are returned.

Exercise 4:

Write a script that reads the content of a text file and prints it in upper case.

Exercise 5:

Write a function `to_upper_case(input_file, output_file)` that takes two string parameters containing the name of two text files. The function reads the content of the `input_file` and saves it in upper case into the `output_file`.

Exercise 6:

We want to create a function `sum_from_file(filename)` that calculate the sum of all `int` contained in the text file `filename`. The format of the text file is as follow, a series of `int` separated by a space spanning several lines as shown below. In this particular case, the returned value should be `100`.

```
1 30 4 5
8 12 19 1
5 5 10
```

1. Should the function raise an error? If yes, describe the case(s) where an error should be raised.
2. It is sometime useful to decompose the problem into smaller problem. In this case it would be useful to have a function `sum_numbers(a_string)` that calculates and returns the sum of all numbers contained in the string `a_string`. The format of the string is a series of `int` separated by a space. For example:

```
>>> sum_numbers('1 30 4 5')
40
```

   Note: It could be useful to remember /check some the methods already existing for `str` object.
3. Now that we have managed to write `sum_numbers`, it should be easier to write the function `sum_from_file`.
4. Write the docstring (python documentation) for this function, as explain in chapter 7 of the textbook.