TPOP PRACTICAL

FILES & EXCEPTIONS

PRACTICAL 05- Level 1

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.

The first thing we can do is write the content of a variable into a file. To do so we need to open the file for writing, this is done using the following statement:

```
output_file = open('output.txt', 'w')
```

The variable `output_file` is now a handle to an output stream. To write some text into that output stream we use the method `write`.

```
output_file.write("the first line of my file")
```

A good practice is to always close the I/O stream when it is not needed anymore, we do so by using the method close.

```
output_file.close()
```

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 `saveListToFile(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 `saveToLog(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.

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).

Now that we have managed to save data to a file, it would be quite useful to read the data back into variables. First of all, we need to open the file for read using the option `'r'` in the function open:

```
input_file= open('output.txt', 'r')
```

once we have open the file in read mode, we have several ways to read its content. The following script use a for loop to read and print the entire file line by line (a line is a text up to and including the newline character `'\n'`).

```
for line in input_file:
    print(line)
```

As said before a good practice is to always close the I/O stream when it is not needed anymore, we do so by using the method close.

```
output_file.close()
```

A summary of other methods available for reading the content of a file is given in Table 1Table 2.

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.

It should be noted that the open command raises an `IOError` upon failure, and therefore should be executed in a `try-except` statement. The following script is a much better way to deal with file, tidying the code by closing the I/O stream if the operation has been successful. It is also a nice example of how to use all the option of a `try-except` statement.

```
input_file = None
try:
    input_file = open('output.txt', 'r')
except IOError as err:
    print("There is an issue with the file!")
else:
    # Do something with the file
    for line in input_file:
        print(line)
finally:
    # cleaning, even if there has been an error
    if input_file is not None:
        input_file.close()
```

However, since the introduction of the with statement in Python 2.5, the following code is preferred:

```
with open('output.txt', 'r') as input_file:
    for line in input_file:
        print(line)
```

To know more about Python 3.x I/O, there is a more in-depth tutorial at:

http://www.diveintopython3.net/files.html

Exercise 4:

Write a script that read the content of a text file and print it in upper case.

Exercise 5:

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

Exercise 6:

We want to create a function `sumFromFile(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. It is sometime useful to decompose the problem into smaller problem. In this case it would be useful to have a function `sumNumbers(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:

```
>>> sumNumbers("1 30 4 5")
40
```

Note: It could be useful to remember /check some the methods already existing for `str` object.

2. Now that we have managed to write `sumNumbers`, it should be easier to write the function `sumFromFile`.