

Excercise - Working With CSV

Using the CSV module

1. A CSV file is often used exchange format for spreadsheets and databases.
2. Each line is called a record and each field within a record is seperated by a delimiter such as comma, tab etc.
3. Keep in mind that Mac uses a different delimiter ('\r') than Microsoft ('\r\n'). Since the module we will use works well with Windows CSV files, we will save and use a Windows CSV file in our program.
4. We use the module "CSV" which is not included in the standard library of Python.

Let us write a program to read a CSV file (word_sentiment.csv). This file contains a list of 2000 + words and its sentiment ranging form -5 to +5. Write a function "word_sentiment" which checks if the entered word is found in the sentiment_csv file and returns the corresponding sentiment. If the word is not found it returns 0.

Step 1:Import the module CSV.

If the module is not included we will need to do "pip install csv" in the terminal (in case of mac) or in the command prompt (in case of windows).

In [2]:

```
import csv
```

Step 2: Assign the path of the file to a global variable "SENTIMENT_CSV"

In [3]:

```
SENTIMENT_CSV = "/Users/medapa/Dropbox/HEC/Teaching/Python Course for PhDs/Scripts/python4phd/word_sentiment.csv"
```

Step 3: Open the file using the "with open()" command and read the file

Before we read a file, we need to open it. The "with open()" command is very handy since it can open the file and give you a handler with which you can read the file. One of the benefits of the "with" command is that (unlike the simple open() command) it can automatically close the file, allowing write operations to be completed. The syntax is :

with open(**filename, mode, encoding**) as **fileobj** . Where **fileobj** is the file object returned by open(); **filename** is the string name of the file. **mode** indicates what you want to do with the file and **encoding** defines the type of encoding with which you want to open the file.

Mode could be:

- w -> write. if the file exists it is overwritten
- r -> read
- a -> append. Write at the end of the file
- x - > write. Only if the file does not exist. It does not allow a file to be re-written

For each, adding a subfix 't' refers to read/write as text and the subfix 'b' refers to read/write as bytes.

Encoding could be:

- 'ascii'
- 'utf-8'
- 'latin-1'
- 'cp-1252'
- 'unicode-escape'

After opening the file, we call the **csv.reader()** function to read the data. It assigns a data structure (similar to a multidimensional list) which we can use to read any cell in the csv file.

In [23]:

```
with open(SENTIMENT_CSV, 'rt', encoding = 'utf-8') as senti_data:
    sentiment = csv.reader(senti_data)
```

Let us package all of this into a nice function.

In [4]:

```
import csv
SENTIMENT_CSV = "/Users/medapa/Dropbox/HEC/Teaching/Python Course for PhDs/Scripts/python4phd/word_sentiment.csv"

def word_sentiment(word):
    '''This function uses the word_sentiment.csv file to find the sentiment of the word entered'''
    with open(SENTIMENT_CSV, 'rt', encoding = 'utf-8') as senti_data:
        sentiment = csv.reader(senti_data)

        for data_row in sentiment:
            if data_row[0] == word:
                sentiment_val = data_row[1]
                return sentiment_val

    return 0

def main():
    word_in = input("enter the word: ").lower()
    return_val = word_sentiment(word_in)
    print("the sentiment of the word '{:s}' is: {:s}".format(word_in, return_val))
main()
```

```
enter the word: insipid
the sentiment of the word 'insipid' is: -2
```

Now let us update this code so that we ask the user to enter a sentence. We then break the sentence into words and find the sentiment of each word. We then aggregate the sentiments across all the words to calculate the sentiment of the sentence and tell if the sentence entered is positive or negative. Hint: Use the `split()` command we saw in lesson 1.

In []:

```
# Enter code here
```

Can you improve this code to handle double negatives like "not" ? eg. not good . Do you think we can build a rudimentary learning algorithm to improve the corpus of sentiments ?

In []: