Handin3 B2

The hand-ins in the course will consist of two parts: 1) a set of short exercises to practice a particular part of the curriculum (e.g. loops), and 2) a project part. In the project part, we will consider a specific Data Science case, working on the same (or similar) data each week and gradually building up a complete data analysis.

So that you know, the tools required to solve the hand-ins will be generally covered in the lectures in the week where the assignment is given. Sometimes, you might have to wait until the Wednesday lecture before you have all the tools to complete the assignment.

Part 1

In this week's exercise, we'll practice loops and functions more. We'll also try to write our own Python module.

Start by creating two files: handin3.py and handin3\_test.py. The idea is that handin3.py will be the module file containing the main code, and handin3\_test.py will contain code that uses (tests) our module. You will also need to download the following file to the **directory** into your project directory: [Download File (currently a txt file w/o a file ending)](https://absalon.ku.dk/courses/59965/files/6700794?wrap=1)[Download Download File (currently a txt file w/o a file ending)](https://absalon.ku.dk/courses/59965/files/6700794/download?download_frd=1)

1. Write a function called read\_word\_file that takes a single argument called filename.
   * The function should open the file, iterate over the lines, and remove the new lines at the end of each line.
   * For each line, it should construct a tuple consisting of two values: the line number (starting at zero), and the line string (without newlines).
   * For example, for a file containing "hello\n" on the first line and "world\n" on the second line, you should get the following list: [(0, "hello"), (1, "world")].
   * The function should return this list to the caller of the function. Inside the handin3\_test.py file, import handin3.py, and then call the read\_word\_file function using the british-english file we downloaded above. Save the result in a variable called word\_list.
2. Copy & Paste your read\_word\_file into a new function called read\_word\_file2. This time, the function takes two arguments: 1) filename and 2) word\_stem.
   * As before, the function should open the file corresponding to a filename, iterate over the lines, remove the new lines at the end of each line, and save tuples of (line-number, line-without-newline) to a list.
   * The difference is that this time, we have an extra argument called word\_stem, which is a string value. Your read\_word\_file2 use this string to search in each line whether it can find a match of the string in the line.
   * For instance, if I write read\_word\_file2("british-english","pyth"), it returns a list of tuples only for the lines that contain the letters pyth. The word\_stem argument should be optional - it should default to the empty string, which will return a list of tuples for all lines in the file.

Inside the handin3\_test.py file, call the read\_word\_file2 function on the british-english file. Specify a word\_stem so that only words containing "pyt" (in lowercase) are returned. Save the result in a variable called filtered\_word\_list.

Part 2: Project

Last week, we wrote a function called read\_data2, that read the temperature anomaly data into a list of strings. This week, we will process our data a bit more, such that our numerical data is more readily accessible.

1. Create a file called handin3\_project.py. Inside this file, create a function called read\_data3.
2. Like last week, the function should take two arguments: 1) filename and 2) year\_range, where year\_range has a default value, making this argument optional. This time, rather than returning a list of strings, the function should return a dictionary.
3. The keys in the dictionary should be the year values (as integers), and each dictionary value should be a list of the numbers (i.e. floating point) in that file line. So, for example, if the dictionary was called data, I should be able to able to write: data[1990][0] to get the value 0.366.
4. Hint: the file contains some NaN values (i.e. "not a number"), you can still convert these to a float - they will turn into nan float values.
5. Create a new file called handin3\_project\_test.py which calls the read\_data3 function with the arguments filename='Land\_and\_Ocean\_summary.txt' and year\_range=(2015,2018). Save the result in a variable called temp\_anomaly\_data.

When you are done, click the "Load Handin3 in a new window" button below, which will take you to the CodeGrade server. Here, please submit the handin3.py, handin3\_test.py, handin3\_project.py, and handin3\_project\_test.py files. CodeGrade will automatically check the code for you and upgrade your grade for the assignment within Absalon. You can submit as many times as you want.

**Note that the autotests will also test your code style from this week, giving errors for missing doc strings.**