SOFTWARE 1 PRACTICAL

LOOPS & FUNCTIONS

Week 6 – Practical 5

Exercise 1: *reinventing the wheel! (From Seminar 1 exercise 3)*

For this question we are emulating the method `split()` from the type `str`. This exercise is more challenging than it may look like. It is crucial that you devise an algorithm before you start implementing. Implement your own pseudo-code or the one given in the model answers. There are methods in the `str` type that can help you identify between an alphabet character and a punctuation; this might be very useful.

Using the file `practical_5.py`, write a function `split_text(text, separators)` where `text` is a string to be split, `separators` is a string containing all the characters used to split the text (for example `',.!? '`). The function returns the list of tokens separated by one of the separators. For example:

```
>>> sample_text = "As Python's creator, I'd like to say a
few words about its origins."
>>> split_text(sample_text, ",'.") #no space delimiter
['As Python', 's creator', ' I',
 'd like to say a few words about its origins']
>>> split_text (sampleText, ", '.") [
'As', 'Python', 's', 'creator', 'I', 'd', 'like', 'to',
'say', 'a', 'few', 'words', 'about', 'its', 'origins']
```

You must NOT use the method `split()` from the `str` type, however other methods from the class are allowed. You must not use python libraries such as `string.py`.

Exercise 2:

In exercise 1, there could be many repetitions of the same word in the returned list. Rather than just returning the duplicated words, perhaps it would be more interesting to keep track of the number of occurrences of each word. For example, in the variable `sample_text` given in the file `practical_5.py`, 'a' occurs 9 times, while 'as' occurs twice (one 'As' and one 'as) and 'python' three times.

Using the file `practical_5.py` implement `get_words_frequencies(text)` which returns the result of our computation, that is for each word in `text`, provide its number of occurrences in `text`. The function should be case insensitive, meaning that 'As' and 'as' should be considered the same. The method returns a dictionary where the keys are the words in lowercase and the values are the frequency of the given word in the text passed in parameters.

Note the module `string` contains a variable `punctuation` which is a string containing all punctuation symbols but does not contain the white space.

Exercise 3:

Write a function `flatten(list_2D)` that transforms a 2D list passed as parameter into a 1D list. For simplicity, the input parameter is assumed to be a valid input. For example:

```
>>> flatten([[1,2],[3,4,5,6],[7],[8,9]])
[1,2,3,4,5,6,7,8,9]
>>> flatten([[1,2],[],[7],[]])
[1,2,7]
>>> flatten([[1,2,3,4,5]])
[1,2,3,4,5]
```

Exercise 4:

Write a function `rasterise(list_1D, width)` that transforms a 1D list passed as parameter into a 2D list, where each sub-list have `width` elements. If the length of the 1D list is not a multiple of `width`, the function returns `None`. If `width` is less than 1, the function returns `None`.

For example:

```
>>> rasterise([1,2,3,4,5,6,7,8],4)
[[1,2,3,4],[5,6,7,8]]
>>> rasterise([1,2,3,4,5,6,7,8],2)
[[1,2],[3,4],[5,6],[7,8]]
>>> rasterise([1,2,3,4,5,6,7,8],1)
[[1],[2],[3],[4],[5],[6],[7],[8]]
>>> rasterise([1,2,3,4,5,6,7,8],0)
None
>>> rasterise([1,2,3,4,5,6,7,8],3)
None
```