sOFTWARE 1 PRACTICAL

## Loops & Functions

Week 4 – Practical 6

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

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. Simulate your algorithm on paper before writing any code. There are methods in the str type that can help you identify between an alphabet character and a punctuation; this might be very useful. If you are stuck, ask me or one of the PTA to have look at your algorithm.

Using the file practical\_6.py, write and test a function split\_text(text) where text is a string and which returns the list of the words by splitting the string text. See example below:

>>> sample\_text = "As Python's creator, I'd like to say a few words about its origins.”

>>> split\_text(sample\_text)

['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 library such as string.py.

Exercise 2: *a more flexible split\_text.*

Using the file practical\_6.py, write and test a function split\_text\_by(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\_by(sample\_text, “,’.”)

['As Python', 's creator', ' I',

'd like to say a few words about its origins']

Exercise 3: *difficult (from seminar).*

As you can see in exercise 2, 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 sampleText, ‘a’ occurs 9 times, while ‘as’ occurs twice (one ‘As’ and one ‘as) and ‘python’ three times.

Using the file practical\_6.py implement getWordsFrequency(text)which returns the result of our computation, that is for each word in text, provide its number of occurrences in text. What kind of data structure would you use to represent the result of the computation?

Exercise 4:

Write a function flatten(list\_2D) that transforms a 2D list passed as parameter into a 1D list. 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 5:

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 should return 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],3)

None

Exercise 6:

Consider a 2D list table consisting of sub-list of floats. And example is given below.

table = [[1,2,3],

[4,5,6],

[7,8,9]]

Write a function sum\_column(a\_table) which computes the sum of each column and store the result into a list. For example, given the variable table defined above,

>>> sum\_column(table)

[12,15,18]

If any two rows of the table have a different size, you should return None.

other = [[1,2,3],

[4],

[7,8,9]]

>>> sum\_column(other)

None