6.2

Special requirements for implementing the project

Functions

The purpose of the project is to practice dividing the program into functions. To achieve this purpose, the program's source code must meet the following conditions:

* The program contains at least two functions which receive information as parameter(s) and also utilize these parameters for some sensible purpose.
* The program contains at least two functions that form at least one value needed by their caller and return this/these values to the caller using the return command.

The fact that minimum requirement is set as two functions of a certain kind does not mean that two is the optimal number of functions to use. It is just the absolute minimum. If you design your program carefully, you will realize that there are potential uses for more than two functions. Implmenting the bare minimum of two functions will not result very good grade from the TAs during the final grading.

Lists

Your implementation must use list data structure to store user inputs.

Task description

Finnish people drink too much coffee. The authorities are concerned and decided to investigate the situation by arranging a gallup to find out some accurate numbers. Gallup asks each passing person, how many cups of coffee, they drink on a daily basis and writes down the response. Implement a program that conducts some calculations for the responses.

First the program prints:

Enter one response per line. End by entering an empty row.

and reads the integers entered by the user until the user wants to finish entering the numbers and enters an empty row to mark this.

First the program removes the responses of non-coffee-drinkers. If there are values to be removed, their number is announced to the user by printing:

Removed X non-coffee-drinkers responses

where X is replaced with the number of the measurement results.

If there are responses left, the program prints interesting information related to coffee drinking habits. First the program prints the text:

Information related to coffee drinkers:

After this, the program prints a graphic presentation of the distribution of the responses. The graphic presentation is a bar diagram, that contains a bar for each number inside the range of the response (between the smallest and the largest response). The first thing at the left of each line of the graphic representation is the value of the response printed in a 2-character-wide column. After this, the program prints one space, and after this the character "#" as many times as there are responses with this value. The presentation of the graphic visualization can be seen more specifically in the example run in the end of the task description. After the graphic presentation print one empty line. Example, when the responses entered are: 2, 1, 2, 5, 1, 1

1 ###

2 ##

3

4

5 #

The recommendation is that you should not drink more than 4 cups of coffee per day. Next, the program prints the largest response, the most common response, and the percentage of the respondents that drink more than 4 cups of coffee per day:

The greatest response: X cups of coffee per day

The most common response: Y cups of coffee per day

Z% of the respondents drink more than 4 cups of coffee per day

where X, Y, and Z are replaced with correct values. If there are more than one most common responses, the smallest value of these responses is printed. The percentage value is printed to one decimal. Percentage value is printed only if it is greater than zero. If none of the respondents drink more than 4 cups of coffee per day, the printout is:

None of the respondents drink too much coffee

Example executions

To concretise the task description we present a couple of example executions. Note, that these test cases are not enough for testing the program properly. You are supposed to test your program much more extensively yourself.

Example 1. Gallup at the shopping mall Duo

Enter one response per line. End by entering an empty row.

3

9

4

0

6

5

2

3

4

0

7

5

4

12

5

3

0

6

8

4

5

9

10

5

4

3

7

5

6

0

2

1

4

5

Removed 4 non-coffee-drinkers responses

Information related to coffee drinkers:

1 #

2 ##

3 ####

4 ######

5 #######

6 ###

7 ##

8 #

9 ##

10 #

11

12 #

The greatest response: 12 cups of coffee per day

The most common response: 5 cups of coffee per day

56.7% of the respondents drink more than 4 cups of coffee per day

Example 2. Gallup in a coffeine rehab group

Enter one response per line. End by entering an empty row.

10

12

9

15

13

12

Information related to coffee drinkers:

9 #

10 #

11

12 ##

13 #

14

15 #

The greatest response: 15 cups of coffee per day

The most common response: 12 cups of coffee per day

100.0% of the respondents drink more than 4 cups of coffee per day

Example 3. Gallup in the kindergarten yard

Enter one response per line. End by entering an empty row.

0

0

0

0

0

0

Removed 6 non-coffee-drinkers responses

Example 4. Only one respondent

Enter one response per line. End by entering an empty row.

3

Information related to coffee drinkers:

3 #

The greatest response: 3 cups of coffee per day

The most common response: 3 cups of coffee per day

None of the respondents drink too much coffee

Programming tips

* Python includes the function sorted, which has a list as its' a parameter and which then returns an ordered copy of the list. This may be helpful.
* The list also includes a method, list.sort(), which sorts the list on its' place. As a result of this method call, the original list order is lost.
* You should consider the difference between these two.
* When testing your program you don't have to enter the measurements manually one by one. You can copy and paste them from this window's example executions to PyCharm's execution/run window.
* You can also store your own test inputs into a separate file and cut and paste them from there.

6.4 Vowels and consonants

Learning Goals:

Learning to go through the characters of a string using a repeatition structure.

Write a program that asks a user for a word and tells how many vowels and consonants the word contains.

Examples of how the program operates:

Enter a word: sassafrass

The word sassafrass contains 3 vowels and 7 consonants

Programming tips:

* Create a repetition structure that goes through all the characters of the string and checks each letter for whether it is a vowel.
* Use the first tasks of the list round to remind yourself how to iterate the list in two different ways using a for loop. The same two principles also apply to the for loop processing a string. Consider which way is better in this situation.
* Consider whether you need to calculate the consonants separately or if there is an easier way...

6.5 ROT-13 encryption of one line

Learning Goals:

Getting acquainted with string structure, ie. how to handle the characters in a string using a for command and the [] operator.

Implement a so-called ROT-13 encryption. In ROT-13 system, the characters are replaced with other characters in accordance with the following formula:

|  |  |
| --- | --- |
| unencrypted character | a b c d e f g h i j k l m n o p q r s t u v w x y z |
| encrypted character | n o p q r s t u v w x y z a b c d e f g h i j k l m |

Upper-case letters are changed to other upper-case letters using the same logic. The characters that were not listed in the previous table are maintained as they are. For instance the string

Happy, happy, joy, joy!

would be the following when encrypted in ROT-13:

Unccl, unccl, wbl, wbl!

Note that if the ROT-13 encryption is performed twice, first for the original text and then for the encrypted text, you will receive the original message in non-encrypted format. This can also be utilized in testing the program.

a)

First, we shall review the use of lists. Using the [attached template](https://plus.tuni.fi/graderB/static/tie02101-s2020/static/code/rot13.py), first implement the function encrypt, which performs the ROT13 transformation for one character. This means the function uses one character as its parameter and returns the encrypted character that matches the character in question. When testing a function in the interactive mode of the Python interpreter, it should operate as follows:

>>> encrypt("e")

'r'

>>> encrypt("E")

'R'

>>> encrypt("?")

'?'

Programming tips:

* Remind yourself of the prebuilt operations that exists for handling lists.
* To find the encrypted character that matches the character to be encrypted you have to:
  1. find the character you want to encrypt in the list REGULAR\_CHARS
  2. check what is the index in which it is located in the list
  3. you find the encrypted character in the same index in the list ENCRYPTED\_CHARS
* Note that the upper-case characters must be processed separately, but an upper-case character may be transformed to a matching lower-case character using a prebuilt function in Python. You should also get acquainted with the string functions lower and upper.
* Furthermore, all the characters that cannot be found from the REGULAR\_CHARS list (and where the matching lower-function character cannot be found from the REGULAR\_CHARS list) are processed separately.

b)

Implement a new function, row\_encryption, which perfoms a ROT13 transformation for an entire string. In other words, this function uses a string as a parameter and returns a string. When testing the function in the interactive mode of Python interpreter, it should operate as follows:

>>> row\_encryption("Happy, happy, joy, joy!")

'Unccl, unccl, wbl, wbl!'

Programming tips:

* The intent is to implement a new function that calls the encryption function implemented in section A.
* You can encrypt the entire string by going through the characters it contains using the for loop and encrypting each character separately (using the encrypt function you created in part "a").
* In this function, you can first create an empty string which is formatted as "", and after this add strings to this empty string using a + operator. A += operator exists for the string, just as for integers.
* You could implement this function in two alternative ways: EITHER the function takes a string to be encrypted for a parameter and forms a new, encrypted string, which it then returns as a return value, OR the function takes an encrypted string as a parameter and then prints the encrypted characters one by one, until it has gone through all the characters. Generally, the better and more commonly usable solution is creating a function that returns an encrypted string and prints the encrypted string in the main program. (Why?) This is also the way to operate in this task.

6.6 Saving a message

Learning Goals:

Learning to save strings in a list.

Implement the function read\_message to the [attached programcode template](https://plus.tuni.fi/graderB/static/tie02101-s2020/static/code/viesti.py). The function reads the input entered by the user, saves the rows in a list, and returns the list. The entry of the input is terminated by entering an empty row. This empty row is not saved in list.

Also implement a main program that calls the function to read a message and then prints the strings in the list using ALL CAPITALS. An example of how the program operates:

Enter the text rows of the message. End by entering an empty row.

Puff, the magic dragon lived by the sea,

And frolicked in the autumn mist, in a land called Honah Lee.

The same, shouting:

PUFF, THE MAGIC DRAGON LIVED BY THE SEA,

AND FROLICKED IN THE AUTUMN MIST, IN A LAND CALLED HONAH LEE.

Programming tips:

* We already practiced saving a user's entry in a list in the project concerning lists. This program operates in the same principle, but this time, the input consists of strings and not integers. (This means the input function returns correct information straight away and you don't need to convert the type.)
* You can use one of the string methods for capitalization.
* After calling the read\_message function, the main program receives a list containing strings as a return value. When you want to print this list's contents, you have to go through each string on the list and print it separately (in all capitals).

6.7 ROT-13 encryption for a whole message

Learning Goals:

Learn to implement a new program by combining previously implemented functions.

Implement the final version of the ROT-13 program so that it first reads the encrypted message from the user as a whole (the message is ended by entering an empty row). After this, the message is printed as encrypted. An example of how the program operates:

Enter text rows to the message. Quit by entering an empty row.

Puff, the magic dragon lived by the sea,

And frolicked in the autumn mist, in a land called Honah Lee.

ROT13:

Chss, gur zntvp qentba yvirq ol gur frn,

Naq sebyvpxrq va gur nhghza zvfg, va n ynaq pnyyrq Ubanu Yrr.

Programming tips:

* In this task, you only need to combine the functions you implemented in the previous ROT-13-task and the read\_message function you implemented in the previous task. The main-function needed for this is not large. Do you recognize now what are the benefits of implementing functions?
  1. Reverse the names to be correct

Learning Goals:

To get acquainted with the prebuilt string operations in Python.

In various lists of names, names are sometimes presented in reverse order, last name before the first name, so that there is a comma after the last name. Create a function reverse\_name, which changes a string containing such a name the "wrong way around" to be the "correct way around", meaning first name first and last name later with the comma off, and returns the reversed name. The function should thus also remove the unnecessary spaces from different parts of the string. Note the use of the spaces in the following example.

Examples of how the function operates when tested in the interactive mode of the Python interpreter:

>>> reverse\_name("Techie, Teddy")

'Teddy Techie'

>>> reverse\_name("Scumble, Arnold")

'Arnold Scumble'

>>> reverse\_name("Fortunato,Frank")

'Frank Fortunato

>>> reverse\_name("von Grünbaumberger, Herbert")

'Herbert von Grünbaumberger'

Programming tips:

* Use the Python [documentation](https://docs.python.org/3/library/) to check what operations can be utilized in this task.
* You can implement this task in a number of ways. One possible way is using the in-operator and the index-method for locating the comma, and then dividing the string in sections using the [:] operator. You can remove the unnecessary spaces using the strip-method. Furthermore, another way is using the split-method.
* Also note the exceptional situations! What if the name does not contain a comma? (In this case it is not in a wrong order and there is no need to do anything.) What other special cases should be noted in implementing such a function?
* If the name only contains a first name or a last name, it is returned without extra spaces.
* If the name does not contain a first name nor a last name (it is just a comma), return an empty string.

6.9 Forming an acronym

Learning Goals:

Familiarizing yourself with the prebuilt string operations in Python.

Implement the function create\_an\_acronym, which requests a name as a parameter and returns its acronym. An acronym is formed by taking the first letter of every word in a name and capitalizing it.

Examples of how the function operates when tested in the interactive mode of the Python interpreter:

>>> create\_an\_acronym("central intelligence agency")

'CIA'

Programming tips:

* Again, see the Python [documentation](https://docs.python.org/3/library/).
* Among other methods, the split method may prove useful for this task.

6.10 Capitalization

Learning Goals:

Getting acquainted with prebuilt string operations in Python. Learning to use Python's documentation.

Create the function capitalize\_initial\_letters, which uses a string as a parameter and returns it as written, with each word starting in upper case but the rest of the world in lower case.

Examples of how the function operates when tested in the interactive mode of the Python interpreter:

>>> capitalize\_initial\_letters("drIVING cAR")

'Driving Car'

The task can be performed several different ways, but in this task, it is **intended that you search the Python documentation for a method that fits this task, and then use it.**

In documentation, string features have been listed using a certain logic, so let's review how to read the documentation. Let's get acquainted with the reading of the method descriptions by researching a known method, "split".

First open the [split method description](https://docs.python.org/3/library/stdtypes.html#str.split) in Python documentation to its own tab in your browser.

The first row, which is coloured yellow, has described how to call the method. The first part of the row, the text before the dot, describes the data type that can be used to call the method. In this case, the type is str, meaning a string. After the dot come the method name and paraphrases, which contain the parameters used by the method. Some parameters may have a default value. In case of a split, this applies to both of its parameters. The default value parameters and the named parameters were presented in tasks [Triangle's angle](https://plus.tuni.fi/tie-02107/spring-2020/kierros04/reading__keyword_arguments/kulmat/) and [Improved box printing](https://plus.tuni.fi/tie-02107/spring-2020/kierros04/reading__keyword_arguments/nimetyt_parametrit_tehtava/).

The text after the first row states verbally what the method does and what its parameters are. The text may contain green boxes that are example codes, whose intent is clarifying the method's operations. The rows starting with ">>>" marks are code rows, and the ones without extra marks tell what the code row in question has returned.

Programming tips:

* The string documentation can be found in the [Library Reference](https://docs.python.org/3/library/) section of the [Python documentation](https://docs.python.org/3/), in the chapter [Text Sequence Type — str](https://docs.python.org/3/library/stdtypes.html#text-sequence-type-str)
* The string methods are listed in chapter [String Methods](https://docs.python.org/3/library/stdtypes.html#string-methods)
* The task includes a prebuilt method for solving the task. The code only needs to call the method and return the string returned by the method.

6.11 How many abbas?

Learning Goals:

Learning to design an algorithm related to processing the strings.

Implement the function count\_abbas, which returns the number of abbas (meaning the string "abba") that the parameter string contains.

Examples of how the function operates when tested in the interactive mode of the Python interpreter:

>>> count\_abbas("abbabbabba")

3

>>> count\_abbas("barbapapa")

0

Programming tips:

* A count method for Python strings exists, but it only counts non-overlapping substrings. It is thus not very helpful for solving this task.
* Create a repeating structure to review string indexes (just as the table's indexes were reviewed [in the latter part of the list indexing task](https://plus.tuni.fi/tie-02107/spring-2020/kierros05/reading_lists/listan_indeksointi_b/)).
* For each index, clarify whether it starts the string you wish to search for.
* You should get acquainted with the [:] operation for strings.

6.12 The longest substring in order

Learning Goals:

Learning to design a somewhat more challenging algorithm for processing strings.

Implement the function longest\_substring\_in\_order, which takes a string as its parameteri and searches for the longest substring with its characters in alphabetic order and then returns it. You can assume the string contains only lower-case letters, so you can compare the alphabetical order of the strings by using the comparison operator <, for instance.

If the string contains several equally long substrings in alphabetic order, the program returns the substring that is the closest to the beginning of the string.

Examples of how the function operates when tested in the interactive mode of the Python interpreter:

>>> longest\_substring\_in\_order("abcabcdefgabab")

'abcdefg'

>>> longest\_substring\_in\_order("acdkbarstyefgioprtyrtyx")

'efgioprty'

Programming tips:

* You should go through the string using a for loop.
* For every index, calculate the length of the alphabetically ordered substring found starting from there.
* Strings can be compared using the operators < and >. For example the comparison "aaa" < "aab" returns True, because the string "aaa" is before the sting "aab" if the strings are ordered alphabetically.
* If the found substring is longer than the longest one found earlier, take note of the information it contains. This way, after reaching the end of the string, you have stored information on the longest found substring.

6.13 Fully justified text

Learning Goals:

I learn to use a datastructure that contains multiple strings.

Implement a program that formats text in a fully justified typesetting. What this means is best demonstrated by an example:

Enter text rows. Quit by entering an empty row.

CHAPTER VIII - CONCERNING THOSE WHO HAVE OBTAINED A PRINCIPALITY BY

WICKEDNESS

Although a prince may rise from a private station in two ways, neither

of which can be entirely attributed to fortune or genius, yet it is

manifest to me that I must not be silent on them, although one could be

more copiously treated when I discuss republics. These methods are

when, either by some wicked or nefarious ways, one ascends to the

principality, or when by the favour of his fellow-citizens a private

person becomes the prince of his country. And speaking of the first

method, it will be illustrated by two examples--one ancient, the other

modern--and without entering further into the subject, I consider these

two examples will suffice those who may be compelled to follow them.

Enter the number of characters per line: 50

CHAPTER VIII - CONCERNING THOSE WHO HAVE OBTAINED

A PRINCIPALITY BY WICKEDNESS Although a prince may

rise from a private station in two ways, neither

of which can be entirely attributed to fortune or

genius, yet it is manifest to me that I must not

be silent on them, although one could be more

copiously treated when I discuss republics. These

methods are when, either by some wicked or

nefarious ways, one ascends to the principality,

or when by the favour of his fellow-citizens a

private person becomes the prince of his country.

And speaking of the first method, it will be

illustrated by two examples--one ancient, the

other modern--and without entering further into

the subject, I consider these two examples will

suffice those who may be compelled to follow them.

The user enters the number of characters that will be printed in one line. The text justification algoritm will divide the text into segments that are shorter than this and then fill in the line to the desired character length by adding space characters inbetween the words. The space characters are placed so that in the beginning of the line each word spacings contain one space character more than in the end of the line.

The last line of the text won't be filled with extra spaces. It can be shorter than the other lines.

Programming tips:

* Use the operators // and % to calculate the number of space characters.
* Remember to divide the program into functions!