Take-Home Exam (Resit)

Instructions

Answer all questions, paying attention to the instructions on file and function names. When you are done, zip the requested files into an archive and hand in that zip file, using your candidate number and the module code as the name (e.g., `KTJA9_SECU0012.zip`).

You are allowed to discuss the exam at a high level with anyone, but you may not share code or discuss specific answers. You may not use any Python packages in your answers. Your answers will be marked on the basis of not just the code itself but also the extent to which the code is well commented and tested. If your code for a question is identical (modulo variable renaming or other cosmetic tweaks) to code we find on the Internet or to the code of another student, we will consider that plagiarism and proceed accordingly.

1 Implementing a Caesar cypher [60 marks]

In this exercise we will go through a problem step by step until the coding tasks.

A Caesar Cypher is a cypher that shifts a letter into another one according to a key. The key indicates how many letters we need to pass before finding the cypher letter. If the key is the letter "d", it means that the letter in the uncyphered (plain text) message need to shift by 3 (from a to d) in the cyphered message. For example:

Plain text	**Key**	**Cypher text**
Ciao	d	Fldr
Hello, Everyone!	h	Olssv, Lclyfvul!
SECU0012 can be challenging	j	BNLD0012 ljw kn lqjuunwprwp

The aim of this exercise is to code a Caesar cypher that takes as input from the user whether the message is plain text that needs to be encoded into a cypher text or vice versa. Then, you ask for the message and the key. At the end you want to return the result of the encoding or decoding operation. Using print statements at the end of your code you can give three examples with which you have tested your code.

1.1 Part 1 - Breaking into subproblems [15 marks]

This first part of the assignment aims at destructuring the different parts of the exercise into smaller problems that can be solved more easily. Your aim is to submit a division of the problem into substeps. You can show even two layers of substeps (a more general one and a detailed one) to show your reasoning more in details. You can decide to write this on a word document as well as by hand and taking a picture of it. If you decide to submit the latter, the writing has to be perfectly understandable. Use the name "destructuring" for your file, whether word (.docx) or a picture (.jpg or .png).

1.2 Part 2 - Control flow [10 marks]

In this part you take the different steps identified and you transform them into a control flow. Control flows are the translation into logical steps of your ideas and the more detailed they are, the easier it is to transform them into code. As for the previous part, you can use software as well as pen and paper as long as it is completely understandable. Use the name "flow" followed by the correct extension for your file.

1.3 Part 3 - Coding the algorithm [20 marks]

This is when you start using spyder. Given the control flow, you can now implement your code to translate from plain text to cypher text and vice versa the messages. It is important that you comment your code, showing you understand it and how it relates to your control flow. Moreover, remember to add sanitization techniques that check the input and evaluates whether it is necessary to change the character (the letters) or not (punctuation, spaces, and numbers). Please, name the file "Caesar.py".

1.4 Part 4 - Testing [15 marks]

You need to test your code. As the code is based on the user input and we have not studied functions yet, it is enough for us to see at the end of the script used in part 3 the examples of your tests. You can use print functions where you say which are the input, the key, and the output. Show 5 test examples that attempt at different cases (3 encoding a plain text into a cypher text and two decoding a cypher text into a plain text), in particular, edge cases.

2 Phone snatching data collection [20 marks]

In this question, you'll be setting up a small interface that can be used by security around the campus to collect data on phone snatching from victims or witnesses. When you hand in your answers for this question, give us not only the `data_fill.py` and `data_eval.py` files but also `dataset.txt`.

2.1 Collecting data [10 marks]

Create a file `data_fill.py` and in it use `input` to collect from each interviewee their data. Using an `if` statement, make sure to collect data only from people over 18 and under 50. After asking the age, ask to the participant whether they were a victim or a witness. Allow the don't know answer for the following questions:

- Was the attacker using a bike, a scooter or were they running?
- Was it one or two attackers?
- Were they males or women?

In CSV format, add these results to a file `dataset.txt`. Fill the survey out at least twenty times (10 for victims and 10 for witnesses) by making up answers yourself. As always, make sure to test your code to ensure it works properly on all inputs, including edge cases. This will mean prompting users again if they enter an unexpected value. (Hint: this will require asking the user, in a loop, to enter an answer among the correct options. You also may require error handling in order to check if their answer is allowed.)

If your code is working, three sample interactions could be as follows:

```
> python data_fill.py
What is your age?       17
Thank you, but we want responses only from people over 18.

> python survey_fill.py
What is your age?       52
Thank you, but we want responses only from people under 50.

> python survey_fill.py
What is your age?       32
Were you the victim of phone snatching or a witness?  witness
Was the attacker using a bike, a scooter or were they running?      scooter
Was it one or two attackers? 0

Please answer only as one or two:      a
Please answer only as one or two:      1
Were they males or women?   Don't know

Thank you for participating in the survey.
```

2.2 Analyzing data [10 marks]

Now, create a file `data_eval.py`. In this file, write code to read in the dataset file. Now, print out the following basic statistics:

- The median and the average ages of the victims. [**2 marks**]
- Are there more don't know answers in victims or witnesses answers? [**4 marks**]
- The amount of times the crime is perpetrated by bike, scooter, or running. [**4 marks**]

Again, be sure to handle edge cases, such as when there are no respondents.

3 Dictionaries [20 marks]

3.1 Importing data [10 marks]

Please load the dataset.txt file from the phone snatching exercise in Question 4. Create a file dictionaries.py in which you store the information from this dataset within nested dictionaries. You should have:

- two dictionaries, one for witnesses and one for victims [**4 marks**]
- each dictionary should be keyed by case, and the value for each case should be a dictionary containing the responses to the other questions relating to that case (e.g. age, transport, etc) [**4 marks**]
- once the dictionaries have been created, print the lists of ages for each group (witnesses and victims) [**2 marks**]

3.2 Adding data [10 marks]

On the same file (use comments to separate the two parts), we will now add data in the dictionary. Please, add these three records and remember to check the inputs:

19, witness, bike, 1, male
24, victim, scooter, 2, Don't know
22, witness, scooter, 3, women

These three records should be added to the correct dictionary as a new case.