CMPSC 102
Discrete Structures
Fall 2018

Practical 3: Creating a secret message coder and decoder

Refer to your notes, slides and sample Python code from this week. Do not talk to people wearing trench coats, fedoras and sun glasses that you have only just noticed are in this class with you. They could be spies.

Summary

You, depicted in Figure 1, will use your knowledge creating the encoder to write a decoder tool to convert your secret spy-mail (from your covert-ops in the field) back into a readable language.

In this practical, you will to write a secret message encoder by following an online tutorial by *Code Club* at the below link.

`https://codeclubprojects.org/en-GB/python/secret-messages/`

The encoder and decoder Python source that you produce will be centered around the use of strings. In class and during your lab assignment, we have been studying and working with `sets`, `lists`, `tuples`, and `dictionaries`. Our efforts have centered around the notion that these data structures contain indexed elements which reside at specific locations. In this practical, we will work with strings to show that they are similar to `lists` and `tuples` for some applications. Specifically, we be indexing characters in strings to run the encoder and decoder algorithms.

Figure 1: You are a secret agent who writes and receives secret messages using an encoded language to prevent outside agents from learning the contents of the messages.

GitHub Starter Link

https://classroom.github.com/a/qXzLeUjm

To use this link, please follow the steps below.

- Click on the link and accept the assignment.

Figure 2: Secret messages are produced by encoders and have to be decoded before they can be read.

- Once the importing task has completed, click on the created assignment link which will take you to your newly created GitHub repository for this lab.
- Clone this repository (bearing your name) and work on the lab locally.
- As you are working on your lab, you are to commit and push regularly. You can use the following commands to add a single file, you must be in the directory where the file is located (or add the path to the file in the command):
  - `git commit <nameOfFile> -m ‘‘Your notes about commit here’’`
  - `git push`

  Alternatively, you can use the following commands to add multiple files from your repository:
  - `git add -A`
  - `git commit -m ‘‘Your notes about commit here’’`
  - `git push`

What to do for this practical: first build the encoder

You are to follow the online *Code Club* tutorial at `https://codeclubprojects.org/en-GB/python/secret-messages/` that will provide instructions on completing your project. Today you will be programming an encoder /decoder tool for secret message which you can use to encode top-secret-covert-ops-classified-secret-spy messages from your tree-house head-quarters to be sent securely to your under-cover, slealthy-contacts and operatives who are out in the field. You will use these tools and include an example of its output in the mini-reflection document, discussed below.

Your Secret *(Mum's-the-Word)* Decoder

Start by creating the encoder tool using the tutorial and then reuse some of its code to create the decoder tool. For this task, you are to invited to request the unlock-key from the user before embarking on decoding his or her top-secret, tell-no-one, message. Implement this key-obtaining code into your design. You might want to also modify your encoder tool to allow for user input of a key.

Remember to keep this information secret, especially, from the fedora guys who are likely reading a giant newspaper (purchased from a far-away place), with two small eye-holes punched in it to watch you, and are currently seated at your table. Once you have finished your encoder and decoder tools, please briefly address the questions for your mini-reflection document. *As always with secret communications, destroy this lab assignment after reading it. However just be sure to back-up all your files and materials on GitHub first!*

General questions

These questions are to be added to your `writing/miniReflection.md` file in your repository.

1. How are using strings to store characters similar to using `lists` or `tuples` to store characters?
2. How would you change your code to use only a `dictionary` to keep track of the locations and characters for replacement in your encoder and decoder tools. Change the following line reflect your solution using a `dictionary` (i.e., place the string or characters into a `dictionary` expression.)

   ```
   alphabet = 'abcde'
   ```

3. You and your covert-secret-op, code-named, "Jimmy," are exchanging secret messages. As you are picking up your secret messages from behind enemy lines – in the the secret-dead-drop-zone (where you have agreed to leave messages for each other), the neighbor's dog (called, "Daphiny) smells you and chases you from Jimmy's porch (where you were picking up your messages). After having been chased by Daphiny to your tree-house head-quarters, you realize that you have lost your jacket and that in its pocket was a piece of paper where you wrote the secret decoder key.

   Your first thought is to call up to ask Jimmy (or who-ever answers the phone at Jimmy's house) to give you the key but you later think better of this idea. Instead you begin to think about finding the key using mathematics and statistics. Eventually you find a way of looking over the secret messages to deduce the secret key. In Figure 2, you figure out how to do this. How was this done? (*Can you think of any way to break the code (and find the key) using only the messages, in the unlikely event that decoder key is lost?*)

Deliverables

1. Your completed python code (`src/encoder.py` and `src/decoder.py`)
2. Your mini-reflection Markdown document `writing/miniReflection.md` in which you show the encoder and decoder's output when working with secret messages. In this document, you are also to include your responses to the above three questions.