CS 101 – Text Steganography Lab

Today you will write a couple of Python programs that will practice steganography. This means we are going to hide a secret message inside an otherwise innocent looking text file. You may want to refer to the Caesar cipher Python implementation handout as a guide for how to type certain statements in Python. Here are the steps to follow.

1. First, create a folder called Lab3 on your memory stick or account. We will do all work in this folder. From the class Web site, please download the folder lab03, which already contains a large input file called input.txt. This is a famous speech given by President Reagan in 1987 in Berlin. We want to write two programs:
   1. Let’s write steg.py: This program will hide a short secret message, e.g. “may the force be with you” among the characters of the speech (input.txt), and it will write the resulting output to a new file, output.txt.

This program will have the following structure.

* Create a variable called secretMessage, and set it equal to some sentence of your choice.
* Open the input and output files, and associate each one with a variable.
* Read the entire input file into a string variable. For example, the name of this string variable could be text.
* Write a loop that examines each character in the string. The loop needs to be written in such a way that the variable i refers to which letter in the string we are looking at. For example, i = 0 is the first character, i = 1 is the next character, and so on. In other words, all letters in the input file are numbered, all the way up to the total number of bytes in the file, which in this case is about 15,000.
* The purpose of the loop is to copy the input file into the output file. But we have to hide our secret message. How? Let’s say that every 100th character of the input file is going to be replaced with the next character of the secret message. Let me show you how this is done:

if i % 100 == 0 and I /100 < len(secretMessage):

outFile.write(secretMessage[i/100])

else:

outFile.write(text[i])

* At the end of the program, tell Python to close the files.
  1. Run the program steg.py. Look at the output.txt file. Can you tell if anything has changed relative to the input.txt file? If so, what is different? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
  2. Let’s write steg2.py: This program will attempt to discover the hidden message that is contained inside output.txt. The program will print the secret message to the screen. In other words, steg2.py only needs to open 1 file.
* At the beginning of the program, create an empty string called secretMessage. This is where we are going to store the message that was left for us in output.txt.
* This program will have the same general structure as steg.py. But the idea here is that on every 100th character of the text, the program needs to grab this character and place it in our secretMessage. For instance, try this if-statement inside your loop that looks at every character in the text:

i % 100 == 0:

secretMessage = secretMessage + (text[i])

* Finally, at the end of the program, print the secretMessage to the screen.

Run steg2.py. Is the secret message being printed out correctly? Why is the program printing more text than the length of this message? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Changing every 100th character in the input file is not stealthy. It’s especially obvious to a casual reader if there is an alteration very early in the text file, such as in the first few words. It’s also better to hide our message inside a larger body of text. How about a whole book? Copy steg.py and steg2.py into new files steg3.py and steg4.py, respectively. Change steg3.py so that the input file is alice.txt instead of input.txt.

In addition, let’s modify steg3.py and steg4.py so that they do not change letters so often. Instead, let’s change every 1,000th character. Also, if we write a condition that says that we are going to modify a character if i % 1000 == 0, this statement is true when i is zero. This means we will be changing the first letter in the text file. Does this sound like a good idea?

Instead, let’s arrange for the program to change the following characters. Pick some number from 1-1000, say 687 (not zero or a very small number). In this case, we’ll change the 687th character; and then every 1000th character after that. In other words, we want to ask if the remainder of dividing i by 1000 is 687, rather than zero.

Modify steg3.py and steg4.py accordingly, and verify that the secret message is preserved. You should look at output.txt to see that it’s a little more difficult to find letters of your secret message. Where are its first three letters? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Experiment with a third input file. Especially a file that is large and already has a lot of typos or other weird looking text in it. Also use a different secret message and a different formula for placing characters in the output file. Create complementary programs steg5.py and steg6.py (analogous to our original steg.py and steg2.py) to do the encoding and decoding, respectively. Be stealthy!