CS 8 Lab: pa03

October 29, 2014

1 Overview

This week we implement a cryptographic cipher called the Vigenère Cipher. You will write two functions in the same file. One will encrypt the text and one will decrypt the text. This pdf has suggested steps for completing the assignment.

2 Getting Set Up

Pull up the assignment description from `http://cs.ucsb.edu/~koc/cs8/hwexpa/pa03.html` and read through it. I also suggest looking through the linked Wikipedia page to learn more about the Vigenère Cipher and how it works.

Open up a new python file called `pa03.py` (use past handouts for help if needed). As the first three lines, type in comments containing your full name, lab section time, UCSB UMail email address, and perm number:

```
# Your name, Lab time
# Your UCSB email address
# Your perm number
```

In this programming assignment, you have to write two functions. Make sure you define them exactly as described on the assignment description. **Each function will return (not print) a string**. For now, type all of these function definitions in your file with "filler" code. In the code below, each function returns the string "answer will be here." This is called a **stub**. Stubs can be thought as "filler" code that you put in as placeholder before you type in the actual code. You'll also want to include the alphabet listed on the assignment description. Your file should look something like this:

```
alphabet = "abcdefghijklmnopqrstuvwxyz0123456789 "

#encryption function encVigenere
#accepts a plaintext message of any length and encrypts it using "key"
#which is another string (of at least 1 character),
#and returns the encrypted text (ciphertext)
#key and plaintext are string arguments
def encVigenere(key, plaintext):
    return "answer will be here"

#decryption function decVigenere
#accepts the ciphertext and decrypts using the key,
#obtains back the original message, and returns it
#key and ciphertext are string arguments.
```

```
#key is at least one character and ciphertext can be any length.
def decVigenere(key, ciphertext):
    return "answer will be here"
```

Save your file.

2.1 Running on the command line

If you're on a computer without IDLE or just want to run this on the command line, open up a Terminal window and navigate into the directory where you saved your pa03.py file. Type `python3` and press enter. You are now in the Python shell/command line environment. Remember you can exit at any time by typing `quit()`. After you type `python 3`, type `import pa03` to import your file. To test your encryption function, type:

```
pa03.encVigenere("a", "abc")
```

The parameters "a" and "abc" don't currently matter, as your code doesn't do anything with them yet. Keep in mind that when you test using actual input, the first parameter is the key and the second is the text to be encrypted.
If you used my stub above, it should show `'answer will be here'`. This will be replaced later when you do the code with the actual solution. If you want to test the decryption function, type:

```
pa03.decVigenere("a", "abc")
```

(Again, we're just using the random parameters "a" and "abc" right now.) If you used my stub above, it should show `'answer will be here'`. This will be replaced later when you do the code with the actual solution.

2.2 Running in IDLE

If you're on a computer with IDLE, go to `Run`→`Run Module` where you have your file. Then, over in the IDLE prompt (the window with the `>>>`), type `import pa03`. To test your encryption function, type:

```
encVigenere("a", "abc")
```

The parameters "a" and "abc" don't currently matter, as your code doesn't do anything with them yet. Keep in mind that when you test using actual input, the first parameter is the key and the second is the text to be encrypted.
If you used my stub above, it should show `'answer will be here'`. This will be replaced later when you do the code with the actual solution. If you want to test the decryption function, type:

```
decVigenere("a", "abc")
```

(Again, we're just using the random parameters "a" and "abc" right now.) If you used my stub above, it should show `'answer will be here'`. This will be replaced later when you do the code with the actual solution.

3 Write your code!

Now you want to go into your file and replace the stub `return "answer will be here"` with the correct code. Here are a few general tips:

- Remember to use mod 37 arithmetic (remember the % operator?), as explained on the assignment description.
- You might use `ord` and `chr` (but you don't have to in order to complete the assignment).

  - You can read about `ord` here: `https://docs.python.org/3.3/library/functions.html#ord`
  - You can read about `chr` here: `https://docs.python.org/3.3/library/functions.html#chr`
  - Keep in mind that `ord("a")` is 97, not 0.

- Refer to the Professor's in-class shift cipher example here: `http://cs.ucsb.edu/~koc/cs8/docx/Fall2014/shift.py`
  - You're allowed to have more functions in your code, as long as you have the two required ones.
  - You're allowed to use functions in this shift cipher example, just be sure to add a comment saying that you got the basis for the function from that URL.

- The following code loops through each character of a string called myString:

```
for c in myString:
     #do something with the current character c
```

- While going through each character in the plaintext/ciphertext, keep track of what index of the key you are currently using. For example, in the first character of the plaintext/ciphertext you'll use the first character of the key. Remember, though, that the key might be shorter than the plaintext/ciphertext. When you reach the last character of the key, you'll have to go to the first character again (hint: % is your friend!).

- To get the length of a string called myString, you type `len(myString)`.

- Say you have a string myString. To append or add another string called myOtherString to the end of myString, you type `myString +=  myOtherString` as seen in this example:

```
myString = "abc"
myOtherString = "def"
myString += myOtherString
#now myString = "abcdef"
```

- If your code isn't working, insert a bunch of helpful print statements so you can see what's going on, and go from there.

4 Testing and Turnin

Make sure you test your functions. Follow the procedure outlined in sections 2.1 or 2.2 above for testing. Check that it works with the example on the assignment description, both the encryption of it and the decryption of it. Feel free to make up your own examples as well.

Ready to submit? Make sure you move your file over to CSIL first (if it isn't there already). Then, in a Terminal, navigate the the directory containing your pa03.py file. To turn in, type the following command:

```
turnin pa03@cs8 pa03.py
```

and follow the on-screen directions. Remember, I will grade the last submission turned in before the deadline if you turn in multiple versions. **The deadline for this project is Thursday, October 30th, 2014 at 11pm. We will not be accepting late submissions like we did last week, so make sure you give yourself enough time to complete and turn in your project.**