CSI Lab 7

Tuesday, Feb 25th

Once again, this week I will ask you to work with a partner of your choice. You will turn in a SINGLE answer sheet to the instructor or TA before you leave and submit ONE version of your code under one person's eLearning account before the next class period. As a reminder, this means:

- Be working together
- Only be working at ONE computer
- Both people should work and you should share duties. Trade off typing from time to time so that no one person is in charge.

Failure to follow these guidelines may cause you to lose points for this activity.

Introductory Activity : Introduction to String Methods

Over the last few days we took a look at some string methods. Let's take a look at a string method called ***find***, which can be used to find the index of a particular character (or substring) inside of a string.

First, I would like you to get familiar with the built-in Python docs. Start IDLE, click on the Help menu, and select "Python Docs". You are now inside the Python language help documentation. Take a few minutes to click around. Once you are familiar with the layout of the Python docs, please find some information about the string method called ***find***.

[Q1] Please write down the steps you used to find the string method called ***find*** in the python docs.

(Note: If you can't find it after a few minutes, raise your hand to get a hint.)

Fun Fact: If you forget the name of a method while programming inside of IDLE, you can type the name of your string object followed by a "." and press the tab key.

IDLE also likes to give you hints on the types and numbers of arguments that you can give to any particular method. For example, if you type the name of your string, then a dot, then an open parenthesis, IDLE will show you a yellow-ish box with some helpful information.

At the command prompt type

```
>>> myStr = ""
>>> myStr.find(
```

[Q2] Please write down what you see in the yellow square. Based on what you read in the Python doc description of the find method, what does it mean?

Now let's actually play with the ***find*** string method. At the command prompt type

```
>>> myStr = "I hope it is a sunny day."
```

[Q3] What happens when you invoke each of these commands?

Command	Results
myStr[0]	
myStr.find("I")	
myStr[4]	
myStr.find("p")	
myStr.find("i")	
myStr.find("a")	
myStr.find("a",14)	
myStr.find("sunny")	
myStr.find("cloudy")	
myStr[myStr.find("I")]	
myStr[myStr.find("I")+2]	

[Q4] What is happening when the find method returns a -1?

[Q5] What does myStr.find("a",14) do?

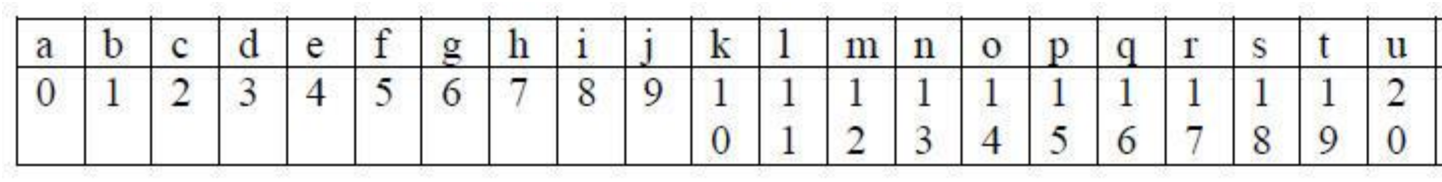
[Q6] Explain what is happening when you issue myStr[myStr.find("I")+2].

Introduction to Today's Lab Assignment -- A Cypher!

In this assignment, we are going to implement a simple encoding and decoding of text. As a result of working on this project, you will gain more experience with the use of:
1. strings and methods
2. if statement
3. for loop and while loop

Background

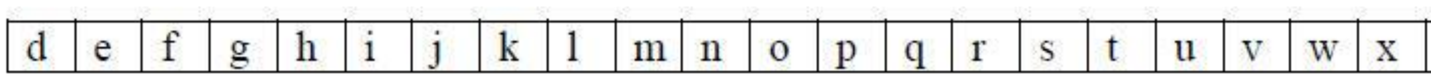
A rotation cipher is one of the simplest, plain-text ciphers, known since at least the time of Julius Caesar. It takes in a plain-text string, and translates it into a new string based on a rotation of the alphabet being used. The basis is a "rotation", a re-sequencing of an alphabet. Consider the following example. Consider the alphabet being a single string consisting of the lower case English letters as below (shown with each letter's associated index):



a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

then a rotation of 3 means that the first three letters of the alphabet are moved to the end of the sequence, while the other letters move up, as shown below. Notice that the

movement is done one
letter at a time.



d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x

A cipher is created by using the rotated alphabet to replace each letter in the original string with its rotated equivalent letter. Using the example above, the word "this" is translated as follows:

- the 't' is found at index 19 in the alphabet. The letter in the rotated alphabet is 'w'.
- the 'h' is found at index 7, the rotated letter is 'k'
- the 'i' is found at index 8, the rotated letter is 'l'
- the 's' is found at index 18, the rotated letter is 'v'

Thus the string 'this' becomes the string 'wklv' using a rotation of 3.

PART A

Write a program that:

1. Prompts the user for one of two commands:
    - 'e' to encode a string
    - 'q' to quit

   Any other command should raise an error and reprompt.

2. If the command is quit, then the program ends and prints a nice exit message.

3. If the command is encode, then the program prompts
    1. for a string to encode
    2. a rotation integer in the range of 1-25

The program then returns the encoded string. Important, the program should not encode any letter that is not in the lower case
alphabet. Those letters should simply be passed through to the encoded string

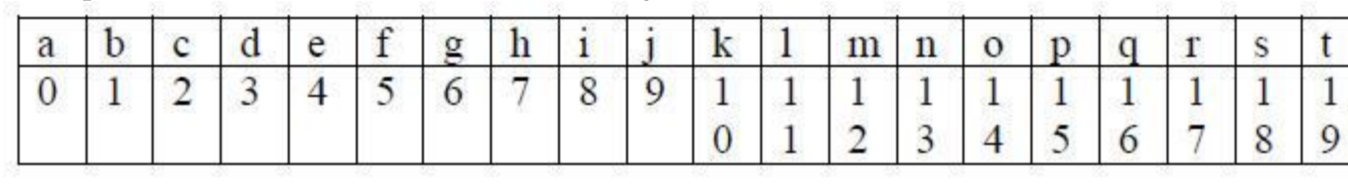
4. At this point you should print the menu again.

SUGGESTIONS/HINTS

There are two ways to make the encoding. The first way SEEMS easier but can be hard. The second way SEEMS hard but can be very easy once you see it. I would encourage you to try the second technique.

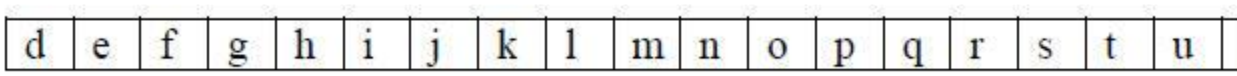
1. **Mathematically**. If the character is a lowercase letter you can do some mathematical manipulation to add the rotation value and then adjust down if you go over the top. (A "y" with a rotation of 3 will be above the value of a z. You need to figure out how to mathematically wrap back around to the "b" that would be it's value. This one seems easy at first, but has some things that make it tricky. *Hint*: If you really want to try this way, it might be helpful to use the functions ord and chr. Recall that the ord function will give you the numerical encoding of the character (example: ord('x') is 120) and chr will convert the numerical encoding back to a character (example: chr(120) is 'x').

2. **With two strings.** Generate a base string that is the alphabet. This is simply the alphabetical order and would look exactly like:



a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19

Now use slicing to generate the rotated string. This too is a 26 character string, but it starts with the character that "a" would "become" Think about how you can slice this into two pieces and then concatenate them back together in the new/different order. For example, if the rotation value is 3 then you would produce the string that is:



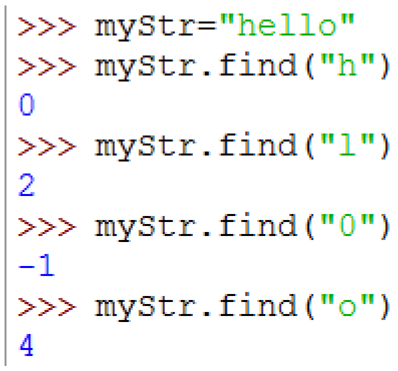
d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u

Given these two strings, one the lower-case alphabet (base) and one a rotated alphabet (rotated), how can you encode the secret message string? You need to go through each letter of the message string to encode it. To do this, find it in the base string, remember its location/index in that alphabet, then find the letter in the rotated alphabet at the same index.

For example, if I needed to encode the letter "h" I would find that it is in index 7 in the base string. Then I could find that letter "k" is at index 7 in the rotated string. Notice that "k" is three letters after "h" so this IS the correct encoding.

But how do you do that?

HINT: The python method **find** is helpful here. It indicates the location of a letter. Consider the string "hello":



```
>>> myStr="hello"
>>> myStr.find("h")
0
>>> myStr.find("l")
2
>>> myStr.find("O")
-1
>>> myStr.find("o")
4
```

I actually think that this later technique is harder to understand at first but EASIER to write in code.

Don't forget that you are only supposed to encode lower case letters. All other characters stay the same. When encoding a string, you should check to see if the letter from the original string is in the lower case alphabet. You may use the in operator. ('a' in 'abcde') returns True. ('A' in 'abcde') returns False.

Before you move on, test your code...

```
e for encode, q for quit: E
I don't recognize that option.  Try again.
e for encode, q for quit: e
String to encode This is a test!
Rotation value 3
Encoded string is:  Tklv lv d whvw!
e for encode, q for quit: e
String to encode abcdwxyz
Rotation value 2
Encoded string is:  cdefyzab
e for encode, q for quit: e
String to encode Testing bad characters = ABYZ @#$@#$
Rotation value 5
Encoded string is:  Tjxynsl gfi hmfwfhyjwx = ABYZ @#$@#$
e for encode, q for quit: q
Thanks for playing!
```

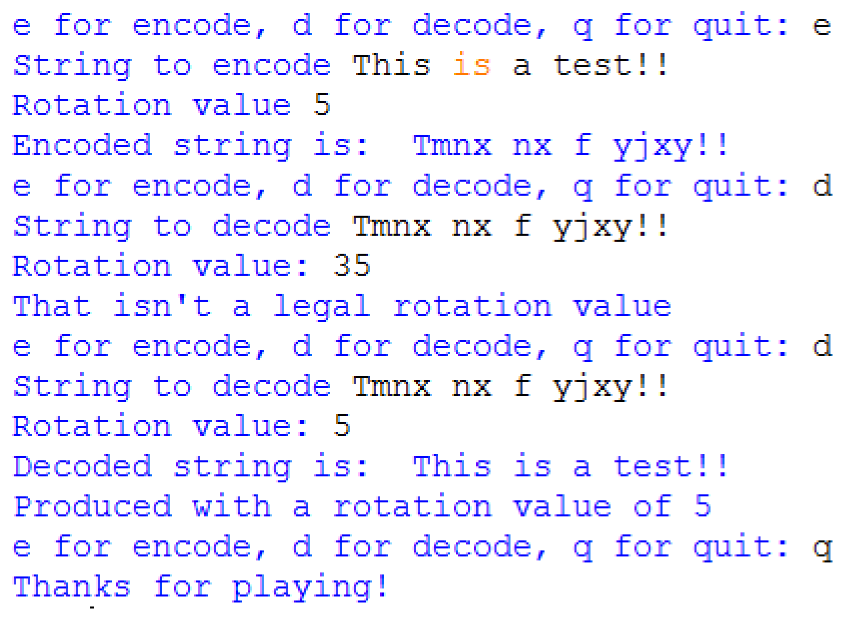
[SIG1] Please show this to an instructor BEFORE moving on

PART B

Add to your program so that it:

1. Prompts the user for one of three commands:
   - 'e' to encode a string
   - 'd' to decode the string using the rotation number
   - 'q' to quit
2. Everything from Part A should still work.
3. Now, if the command is 'd' to decode the string the program should prompt:
   1. for a string to decode
   2. a rotation integer in the range of 1-25
4. At this point the program should decode the program using the REVERSE of the technique from Part A. All of the original rules still apply.

For example:



```
e for encode, d for decode, q for quit: e
String to encode This is a test!!
Rotation value 5
Encoded string is:  Tmnx nx f yjxy!!
e for encode, d for decode, q for quit: d
String to decode Tmnx nx f yjxy!!
Rotation value: 35
That isn't a legal rotation value
e for encode, d for decode, q for quit: d
String to decode Tmnx nx f yjxy!!
Rotation value: 5
Decoded string is:  This is a test!!
Produced with a rotation value of 5
e for encode, d for decode, q for quit: q
Thanks for playing!
```

[SIG2] Please show this to an instructor BEFORE moving on

PART C

At this point I would save a copy of your code. If you do not get Part C completed before the end of the lab you should submit this copy. You will get credit for Part A and B only. However, I want you to TRY Part C.

Add to your program so that it:

1. Prompts the user for one of four commands:
   - 'e' to encode a string
   - 'd' to decode the string using the rotation number
   - 'c' to attempt to crack the code
   - 'q' to quit
2. Everything from Part B should still work.
3. Now, if the command is 'c' to crack the code the program should prompt:
   1. for a string to decode
   2. a single, plain text word that appears in the original text (decoded string).

The program then needs to calculate which rotation value will mean that the single word would be in the encoded message and use this value to decode the secret message.

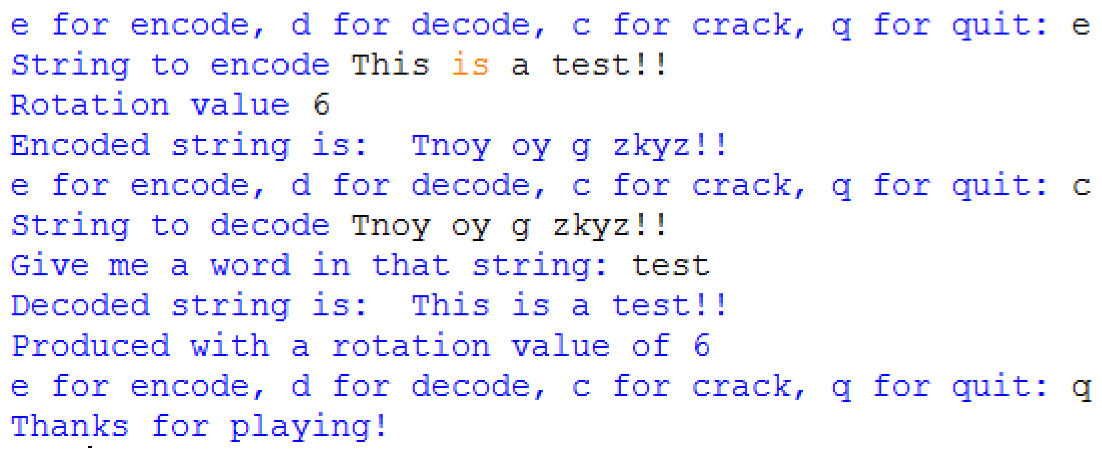
4. The output from the program is this rotation value and the decoded string (text).
5. If no rotation is found then the program should indicate this fact.

This probably sounds VERY hard. But in fact, it isn't that bad. You know that there are only 25 rotation values. Begin by encoding the plain text word with a rotation of 1. Then, check to see if that encoded word is in the larger encoded string.

- If it is, then the rotation value must be 1. Decode the large string with a rotation value of 1.
- If it isn't in the larger string, then repeat with a value of 2, then 3, etc.

As soon as you find the rotation value you can quit and print the results. If you check all 25 rotation values than there was an error and you should print an appropriate message.

A sample run might look like this:



```
e for encode, d for decode, c for crack, q for quit: e
String to encode This is a test!!
Rotation value 6
Encoded string is:  Tnoy oy g zkyz!!
e for encode, d for decode, c for crack, q for quit: c
String to decode Tnoy oy g zkyz!!
Give me a word in that string: test
Decoded string is:  This is a test!!
Produced with a rotation value of 6
e for encode, d for decode, c for crack, q for quit: q
Thanks for playing!
```

Final Submission

This week I will again ask you to submit your code for electronic grading, using the eLearning submission system.

Follow the directions on the system to select the appropriate course and assignment and submit

- caeser.py

Don't forget to hand in your answer sheet to the professor or TA before you leave!