Lab2

Objectives

- Write Python functions for implementing and testing a particular feature, such as encrypting userid and passwords, in this particular case
- Write Python code for a custom encryption technique
- Learn how to use this feature in a (close-to) real-world application

Outcomes

- You will be incrementally better at using functions in Python
- You will develop a module that can be leveraged for your final project

In this lab, we will develop a customized encryption technique to encrypt any given text, userIDs and passwords, in this particular case.

Note: Please setup some way of running iPython notebooks (.ipynb) on your computer (this can be done using an ipynb IDE like Jupyter/Anaconda, Google Colab, or a Jupyter/ipynb plugin for VSCode, Intellij, PyCharm, etc.)

Task 1: Custom Encryption function

In cryptography, ciphertext or cyphertext is the result of encryption performed on plaintext using an algorithm, called a cipher. In an iPython notebook cell, write a function `customEncrypt` to convert plain text string into a ciphertext, encoded using your customized encryption technique, as described below (helper functions are allowed):

Inputs (please read Task 2 for information regarding valid and invalid inputs):

- string inputText: text to be encrypted
  - NOTE: inputText can contain any ASCII printable characters, with the exception of "space" (ASCII code 32) and "!" (ASCII code 33)
- Int N: number of positions to shift characters in the inputText by
  - NOTE: N has to be >= 1
- Int D = direction of shift
  - NOTE: +1 shift right, -1 shift left, any other values are invalid

Outputs:

- String encryptedText: encrypted version of inputText

The algorithm for this technique will follow these steps:

1. Reverse the input text
2. Shift all the ASCII characters in reversed input text by "N" positions.
   - In other words, each letter in the reversed text string is replaced by a letter N positions down the ASCII printable characters list cyclically.
   - The direction of shift is also taken into consideration here

- For example, if the inputs are N=3 and D=1(shift right), then then 'A' is replaced by 'D', 'B' by 'E', 'C' by 'F', ..., 'X' by '[', ..., 'Z' by ']'
- On the other hand, if the inputs are N=3 and D=-1(shift left), then then 'D' is replaced by 'A', 'E' by 'B', 'F' by 'C', ..., '[' by 'X', ..., ']' by 'Z'

Other rules:

1. "space" and "!" are not encrypted. This means that when encrypting, your modulo should start at 34, that way the cipher text does not contain "!" or "space". Please see the table of ASCII printable characters below:

ASCII printable characters

32	space	64	@	96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	'	71	G	103	g
40	(	72	H	104	h
41	)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	s
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	v
55	7	87	W	119	w
56	8	88	X	120	x
57	9	89	Y	121	y
58	:	90	Z	122	z
59	;	91	[	123	{
60	<	92	\	124	|
61	=	93	]	125	}
62	>	94	^	126	~
63	?	95	_		

Example:

If reversed text string is TEST, N=2, and D=1, the encrypted text is VGUV

<u>Task 2: Test Customer Encryption Code function</u>

In a new iPython notebook cell, write a function called `testCustomEncrypt` to test the func tion written in Task 1. The program shall:

- Prompt user for an input text string, which can contain any ASCII <u>printable</u> characters, with the <u>exception</u> of "space" (ASCII code 32) and "!" (ASCII code 33)
- **NOTE:** You will prompt the user twice, once for a userID, and once for a password
- Prompt user for a value for N (N must be >=1)
- Prompt user for a value of D (D must be +1 or –1)
- **NOTE:** You must gracefully handle checking for valid inputs for the input text, N, and D.
  - This can be done using try-catch statements, error codes, and re-prompting users if there is an invalid input
- Call the customEncrypt function to convert the text to ciphertext using custom encryptio n algorithm, and pass in the user inputs as parameters
- Print the returned ciphertext.
- In order to test if the customEncrypt function is written correctly, you now have to rever se the process. Pass the ciphertext into the customEncrypt function (this time with a sa

me value of N, but with a different value of D), and see if you can get the function to ret urn the original user input strings.

Example:

```
Testing Custom Encryption Algorithm

Enter UserID as text :  asamant

Enter password as text :  Temp123

Enter value of n:  3

Enter value of d:  1

encrypted userid: wqdpdvd

encrypted password: 654sphW

Original userid: asamant

Original passwd: Temp123
```

Task 3:

You have been provided a text file (database.txt) which contains encrypted versions of the useri ds and passwords in the table below. Userid and password have been encrypted using the algor ithm described above, with the values of N=3 and D=1. Think of the text file as a simple represe ntation of the information stored in a database. Use the function that you have developed to an swer the following questions.

Userid	Password
asamant	Temp123
aissa	TheKing%^&
bjha	$72messenger
skharel	Life15$
Ally!	God$12

Write your answers for the following into the Python notebook in a new Markdown (md) cell:

1. Which of the userid and password combination(s) in the table above are present in the d atabase?
2. Which userid(s) is/are present in the database, but the password does not match the pa ssword(s) in the table above?
3. Which userid(s) do/does not meet the requirements of a userid?

What to Submit

Upload these deliverables to Canvas.

Deliverables:

1. iPython Notebook file <Lab2-utid.ipynb> with two functions `customEncrypt` and `testCustomEncrypt` and answers to Task 3 questions in a Markdown (md) cell.

Rubric

Task	**Points**
Task 1	3
Task 2	2
Task 3	3