

EE461L HW2 – Python Programming

Objectives

- Write Python functions for implementing and testing a particular feature, such as encrypting userid and passwords, in this 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 case.

Note: Please setup some way of editing Python programs on your laptop. For this class, we will be using Python 3.6 or higher. IDEs such as PyCharm or VSCode can be used.

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

Example:

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

Task 2: Test Customer Encryption Code function

In a Python file, write a function called `testCustomEncrypt` to test the function written in Task 1. The program shall:

- Prompt user for an input text string, which can contain any ASCII printable characters, with the exception 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 encryption 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 reverse 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 return 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 userids and passwords in the table below. Userid and password have been encrypted using the algorithm described above, with the values of N=3 and D=1. Think of the text file as a simple representation of the information stored in a database. Use the function that you have developed to answer the following questions.

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

In a Python file, write code to open the provided file and read the contents of the file. In the same code, write a test code that uses the functions that you have written above to answer the following questions. Write your answers as comments in your Python file:

1. Which of the userid and password combination(s) in the table above are present in the database?
2. Which userid(s) is/are present in the database, but the password does not match the password(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. Zip file <HW2-utid.zip> with code written for Task1, 2, and 3 and answers to Task 3 questions marked as Comments in the file.

Rubric

Task	Points
Task 1	2
Task 2	2
Task 3	2