Table of Contents

Table of Contents

Aim

Use Caesar Cipher technique to encrypt or decrypt an inputted message.

Introduction

A simple way to encrypt data is attributed to Julius Caesar, the Roman Emperor. This method takes each character in a message and replaces it with one which is a certain distance (offset) along the alphabet from it.

For example:

```
1  2  3  4  5  6  7  8  9  .  .  .  .  .
A  B  C  D  E  F  G  H  I  J  K  L  M  .  .  .
+3 offset →
         A  B  C  D  E  F  G  H  I  J  .  .  .
```

If the offset is 3 then A becomes D, B becomes E, C becomes F etc.

Encrypting DIG with the offset of +3 will result in GLJ. The decrypt GLJ, simply offset by the same amount in the opposite direction (i.e. with the negative offset -3).

Instead of restricting the cipher to the alphabetic characters only, we will use all the printable ASCII characters. That is, all the characters from ASCII 32 (Space) to ASCII 126 (~).

Mainly used functions

1. **ord(c)**

   If *c* is a string of length 1, *ord(c)* returns an integer representing the ASCII value of the string.

   For example: *ord('a')* returns the integer *97*.

2. **chr(i)**

   If *i* is an integer, *chr(i)* returns a string containing only one character with an ASCII code is equal to the integer *i*.

   For example: *chr(97)* returns the string '*a*'

Options

Includes many functions that collectively simulate a menu driven program that will allow the user to enter commands and process these commands until the quit command is entered.

The following commands should be allowed:

1. **Enter Message**:

   Prompt for and read (from the keyboard) a string to be encrypted.

2. **Encrypt Message**:

   Encrypts the previously entered message or displays an error message if no string was entered. The message will be encrypted using a randomly generated number between 32 and 126 as the offset/encryption key. This encryption key will be converted into a character using *chr(i)* and appended to the encrypted string.

3. **Decrypt Message**:

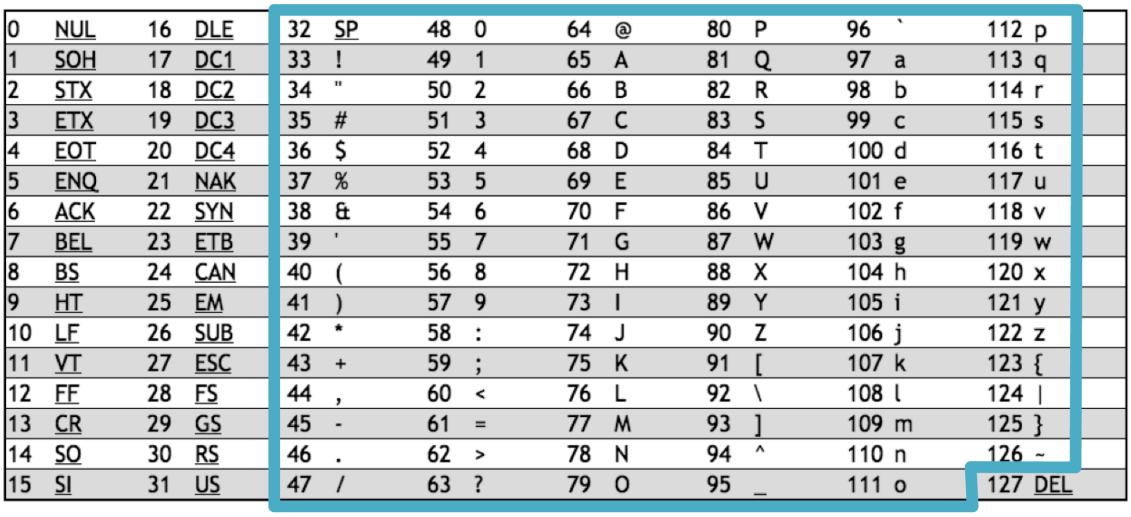
   Decrypts the previously entered message or displays an error message if no string was entered. The message will be decrypted by converting the last letter in the string (the encryption key) to its ASCII value, and using this to offset all other characters in the negative direction. The encryption key should then be removed from the message.

4. **Quit**:

   Displays a goodbye message to the screen and quits the program.

Encryption Process

To start with, choose 1 as the offset. In this case, if the message 'abG' is entered, after the encryption, the result should be 'bcH'. Now that it is working, use the *randint()* function from the *random* module to make the offset a random number between 32 and 126.



0	NUL	16	DLE	32	SP	48	0	64	@	80	P	96	`	112	p
1	SOH	17	DC1	33	!	49	1	65	A	81	Q	97	a	113	q
2	STX	18	DC2	34	"	50	2	66	B	82	R	98	b	114	r
3	ETX	19	DC3	35	#	51	3	67	C	83	S	99	c	115	s
4	EOT	20	DC4	36	$	52	4	68	D	84	T	100	d	116	t
5	ENQ	21	NAK	37	%	53	5	69	E	85	U	101	e	117	u
6	ACK	22	SYN	38	&	54	6	70	F	86	V	102	f	118	v
7	BEL	23	ETB	39	'	55	7	71	G	87	W	103	g	119	w
8	BS	24	CAN	40	(	56	8	72	H	88	X	104	h	120	x
9	HT	25	EM	41	)	57	9	73	I	89	Y	105	i	121	y
10	LF	26	SUB	42	*	58	:	74	J	90	Z	106	j	122	z
11	VT	27	ESC	43	+	59	;	75	K	91	[	107	k	123	{
12	FF	28	FS	44	,	60	<	76	L	92	\	108	l	124	|
13	CR	29	GS	45	-	61	=	77	M	93	]	109	m	125	}
14	SO	30	RS	46	.	62	>	78	N	94	^	110	n	126	~
15	SI	31	US	47	/	63	?	79	O	95	_	111	o	127	DEL

The program only work with the printable ASCII character set. That is, all the characters from ASCII 32 (Space) to ASCII 126 (~). When the ASCII value of the encrypted character points to a character beyond 126 it should *wrap* around to the beginning of the set.

For example, if the offset is 4 and the character is '}' (ASCII 125) then it will encrypt to ASCII 129. This is beyond 126 so you should subtract the total number of characters in the set (95) to wrap back to the beginning. The resulting encrypted character will be the double-quote character " (129-95, ASCII 34). You may have to subtract 95 multiple times until it is within the set, for example, '}' + '}' is 250, and minus 95 is 155. This is still out of bounds. Use a loop. Note that if the encrypted string contains characters that are not in the table above, then your ASCII values are not 'wrapping' correctly.

Decryption Process

The encryption key will be found as the last character in the string. Subtract this offset from the ASCII value of each other character in the message variable. These ASCII values will then need to be converted back to characters using the chr(i) function.

Again, the program only work with the printable ASCII character set. That is, all the characters from ASCII 32 (Space) to ASCII 126 (~). When the offset points to a character less than 32 it should *wrap* around to the end of the set.

For example, if the offset is 4 and the character is '!' (ASCII 33) then it will decrypt to ASCII 29. This is less than 32 so wrap back to the end by adding the total number of characters (95). This gives character '|' (29+95, ASCII 124). You may have to add 95 multiple times until it is within the set. Use a loop.

Sample Output

```
-------------------
     MAIN MENU
-------------------
1. Enter Message
```

```
2. Encrypt Message
3. Decrypt Message
4. Quit

Enter an option (1,2,3,4): 1
Please enter a new message: The rabbit has sprung.
Your message is: 'The rabbit has sprung.'.

-------------------
     MAIN MENU
-------------------
1. Enter Message
2. Encrypt Message
3. Decrypt Message
4. Quit

Enter an option (1,2,3,4): 2
Your message was successfully encrypted.
Your message is: '`tq,~mnnu!,tm , |~"zs:k'.

-------------------
     MAIN MENU
-------------------
1. Enter Message
2. Encrypt Message
3. Decrypt Message
4. Quit

Enter an option (1,2,3,4): 1
Please enter a new message: The conference will commence.
Your message is: 'The conference will commence.'.

-------------------
     MAIN MENU
-------------------
1. Enter Message
2. Encrypt Message
3. Decrypt Message
4. Quit

Enter an option (1,2,3,4): 3
Your message was successfully decrypted.
Your message is: '&:7Q5A@87D7@57QI;>>Q5A??7@57'.

-------------------
     MAIN MENU
-------------------
1. Enter Message
2. Encrypt Message
3. Decrypt Message
4. Quit
```

```
Enter an option (1,2,3,4): 2
Your message was successfully encrypted.
Your message is: 'BVSmQ]\TS`S\QSmeWZZmQ][[S\QS{'.

-------------------
     MAIN MENU
-------------------
1. Enter Message
2. Encrypt Message
3. Decrypt Message
4. Quit

Enter an option (1,2,3,4): 4
Your message is: 'BVSmQ]\TS`S\QSmeWZZmQ][[S\QS{'.

Goodbye
```