**Problem Definition**:

In this task I have been assigned to program and develop Caesar Cipher which involves Encryption and Decryption plus it’s all about Cryptography. I have to make a program which will ask the user to encrypt or decrypt and the user will input the choice of encrypting or decrypting after that they will enter the message when they have done so it will ask how many numbers do they want to shift the letters by, after it has calculated depending on the users choice then it will output the final message with the shifted letters. On the other hand if they have the choice of decrypting then they will input the encrypted message and the amount it has been shifted by, and then it will output the original message without any shifts.

The Caesar Cipher works by moving the original alphabet by a number amount of spaces for example “A” moved five places would be “F”/ “Apple” would be “Fuuqj” and you can call this encryption. This can be used to communicate to people without them knowing the original message. Decryption is when an Encrypted message is converted back to its original message for example “Fuuqj” would be “Apple”. This is the most effective way to send message which would be difficult for other people to decrypt.

The Caesar cipher is named after an ancient roman emperor Julius Caesar. He used it with a shift of three to protect messages for military reasons. At the time it is unknown how effective the Caesar cipher was but was most secure, because most of Caesar’s enemies were illiterate and would have assumed that it was written in an unknown foreign language. The Caesar cipher was created 2000 years ago but it is not used by people today because there are 26 shift key and you can use a computer to crack it quicker because it can go through 26 times for it to show the message.

**Success Criteria:**

In order for my program to be successful I will list the following points

- To ask the user whether to encrypt or decrypt.

- The program will ignore numbers and symbols and blanks in selection.

- The message can contain symbols number and letters and must contain at least one character.

- My program will only accept positive integers for the key and will reject symbols, letters, non-positive integers and blanks.

- My program will not be case sensitive.

- My program should return back to A to Z if choice is decrypt.

- My program should return back to Z to A if choice is encrypt.

**Test Plan:**

The following test plans will determine if my success criteria has been met.

|  |  |  |  |
| --- | --- | --- | --- |
| No | Test Description | Test Data | Expected Outcome |
| 1 | Check if the choice will accept encrypt or decrypt and will not accept symbols numbers or blank | Choice: encrypt  Message:  Key: | Accept and ask for message |
| 2 | Check if the choice will accept encrypt or decrypt and will not accept symbols numbers or blank | Choice: decrypt  Message:  Key: | Accept and ask for message |
| 3 | Check if the choice will accept encrypt or decrypt and will not accept symbols numbers or blank | Choice: e  Message:  Key: | Reject the user and ask to enter encrypt or decrypt |
| 4 | Check if the choice will accept encrypt or decrypt and will not accept symbols numbers or blank | Choice: d  Message:  Key: | Reject the user and ask to enter encrypt or decrypt |
| 5 | Check if the choice will accept encrypt or decrypt and will not accept symbols numbers or blank | Choice: code  Message:  Key: | Reject the user and ask to enter encrypt or decrypt |
| 6 | Check if the choice will accept encrypt or decrypt and will not accept symbols numbers or blank | Choice: [Blank ]  Message:  Key: | Reject the user and ask to enter encrypt or decrypt |
| 7 | Check if the choice will accept encrypt or decrypt and will not accept symbols numbers or blank | Choice: encrypt2  Message:  Key: | Reject the user and ask to enter encrypt or decrypt without numbers |
| 8 | Check if the choice will accept encrypt or decrypt and will not accept symbols numbers or blank | Choice: $&%encrypt  Message:  Key: | Reject the user and ask to enter encrypt or decrypt  without symbols |
| 9 | Check if the message will ignore symbols and numbers and accept letters or capital letter. | Choice: encrypt  Message: javed is in prison  Key: 4 | “Nezih mw mr tvmwsr” |
| 10 | Check if the message will ignore symbols and numbers and accept letters or capital letter. | Choice: encrypt  Message: error 45  Key: 6 | “kxxux 45” |
| 11 | Check if the message will ignore symbols and numbers and accept letters or capital letter. | Choice: encrypt  Message: HELLO mr  Key: 2 | “JQNNQ ot” |
| 12 | Check if the message will ignore symbols and numbers and accept letters or capital letter. | Choice: encrypt  Message: how are you!?  Key: 5 | “Mtb fwj dtz!?” |
| 13 | Check if the message will ignore symbols and numbers and accept letters or capital letter. | Choice: encrypt  Message: [Blank]  Key: | Reject the user and ask to enter at least one letter. |
| 14 | Check if the key for encrypt will accept whole numbers between 1 and 26 and reject letters or symbols and Z should go back to A vice versa. | Choice: encrypt  Message: Z  Key: 1 | “A” |
| 15 | Check if the message for decrypt will ignore symbols and numbers and accept letters or capital letter. | Choice: decrypt  Message: nezih mw mr tvmwsr  Key: 4 | “javed is in prison” |
| 16 | Check if the message for decrypt will ignore symbols and numbers and accept letters or capital letter. | Choice: decrypt  Message: kxxux 45  Key: 6 | “error 645” |
| 17 | Check if the message for decrypt will ignore symbols and numbers and accept letters or capital letter. | Choice: decrypt  Message: JGNNQ ot  Key: 2 | “HELLO mr” |
| 18 | Check if the message for decrypt will ignore symbols and numbers and accept letters or capital letter. | Choice: decrypt  Message: Mtb fwj dtz!?  Key: 5 | “How are you!?” |
| 19 | Check if the message for decrypt will accept whole numbers between 1 and 26 and reject letters or symbols and A should go back to Z vice versa. | Choice: decrypt  Message: A  Key: 1 | “Z” |
| 20 | Check if the message for decrypt will ignore symbols and numbers and accept letters or capital letter and will reject blanks. | Choice: decrypt  Message: [Blank]  Key: | Reject the user and ask to enter at least one letter. |
| 21 | Check if the key will accept whole numbers between 1 and 26 and reject letters or symbols and reject blank | Choice: encrypt  Message: Run Away!  Key: ab | Reject the user and ask to enter a whole positive key number between 1-26 |
| 22 | Check if the key will accept whole numbers between 1 and 26 and reject letters or symbols and reject blank | Choice: encrypt  Message: Run Away!  Key: -15 | Reject the user and ask to enter a whole positive key number between 1-26 |
| 23 | Check if the key will accept whole numbers between 1 and 26 and reject letters or symbols and reject blank | Choice: encrypt  Message: Run Away!  Key: 0 | Reject the user and ask to enter a whole positive key number between 1-26 |
| 24 | Check if the key will accept whole numbers between 1 and 26 and reject letters or symbols and reject blank | Choice: encrypt  Message: Run Away!  Key: 34 | Reject the user and ask to enter a whole positive key number between 1-26 |
| 25 | Check if the key will accept whole numbers between 1 and 26 and reject letters or symbols and reject blank | Choice: encrypt  Message: Run Away!  Key: £ | Reject the user and ask to enter a whole positive key number between 1-26 |
| 26 | Check if the key will accept whole numbers between 1 and 26 and reject letters or symbols and reject blank | Choice: encrypt  Message: Run Away!  Key: 22ty | Reject the user and ask to enter a whole positive key number between 1-26 |
| 27 | Check if the key will accept whole numbers between 1 and 26 and reject letters or symbols and reject blank | Choice: encrypt  Message: Run Away!  Key: [Blank] | Reject the user and ask to enter a whole positive key number between 1-26 |

**Proposed Solution:**

The program language that I am going to use in my controlled assessment is python. I will be using this to create my encryption and decryption program. My program will start off with an input to ask the user which choice to input, the choice of encrypt or decrypt, also a while loop until the user has made a correct selection, which will be a string.

After the selection the program will ask the user to input the key between 1-26, a while loop will be used until the user enters a valid key, which should be a positive integer. Finally the user will enter a message for the program to calculate, the message should accept lowercase and uppercase but will ignore symbols and numbers. These three information will be stored in three variables (Choice, Key, Message).

The program will perform a selection depending on the users choice of encrypt or decrypt. And after the decision it will calculate.

In my program, to encrypt and decrypt messages I will be using the python built in functions such as Ord and Chr. Ord is a function that will convert letters into their unique ASCII number and using this it will convert the message into its unique ASCII number then the key will be added to the ASCII message if the user has selected encrypt otherwise if the user has selected decrypt the ASCII message will be subtracted with key, when this is done the Chr function will convert the ASCII message back into its alphabetic letters.

To make sure that my program works correctly, if the shifted letter has gone beyond the end letter in the alphabet which is “z” then it will be returned back to “a” if the user has selected encrypt. Otherwise if the selection is decrypt and it has gone beyond the beginning letter of the alphabet “a” then it will be returned back to “z”.

After my program has done calculating, the calculated message will be printed out.

**Pseudo code:**

I have created a pseudo code to show the code of my program that will help me throughout my actual program

**FUNCTION** getSelection()

**WHILE TRUE**:

**INPUT** Selection

**IF** Selection == “Encrypt” **OR** Selection == “Decrypt” **THEN**

**RETURN** Selection

**ELSE**

**OUTPUT** “Enter Encrypt or Decrypt only”

**LOOP**

**END FUNCTION**

**FUNCTION** getMessage()

**WHILE TRUE**:

**INPUT** Message

**IF** Letter in Message **THEN**

**RETURN** Message

**ELSE**  **OUTPUT** “The Message contains no letters”

**LOOP**

**END FUNCTION**

**FUNCTION** getKey()

**WHILE TRUE**:

**INPUT** Key

**IF** Key >= 1 **AND** Key <= 26

**RETURN** Key

**ELSE** :

**OUTPUT** “Enter a key between 1-26”

**EXCEPT** ValueError**:**

**OUTPUT** “Integer Only”

**LOOP**

**END FUNCTION**

**IF** Selection == Decrypt **THEN**

currentNumber = CapitalA

**WHILE** currentLoop < currentKey:

**WHILE** currentNumber = < CapitalZ:

currentLetter = CHR(currentNumber)

**OUTPUT** currentNumber+ “:” + currentLetter

currentNumber = currentNumber + 1

**LOOP**

currentNumber = currentNumber - 26

currentLoop = currenLoop + 1

**LOOP**

**ELSE**

currentNumber = CapitalZ

**WHILE** currentLoop < currentKey:

**WHILE** currentNumber = < CapitalA:

currentLetter = CHR(currentNumber)

**OUTPUT** currentNumber+ “:” + currentLetter

currentNumber = currentNumber + 1

**LOOP**

currentNumber = currentNumber + 26

currentLoop = currenLoop + 1

**LOOP**

**END IF**

Key = getKey()

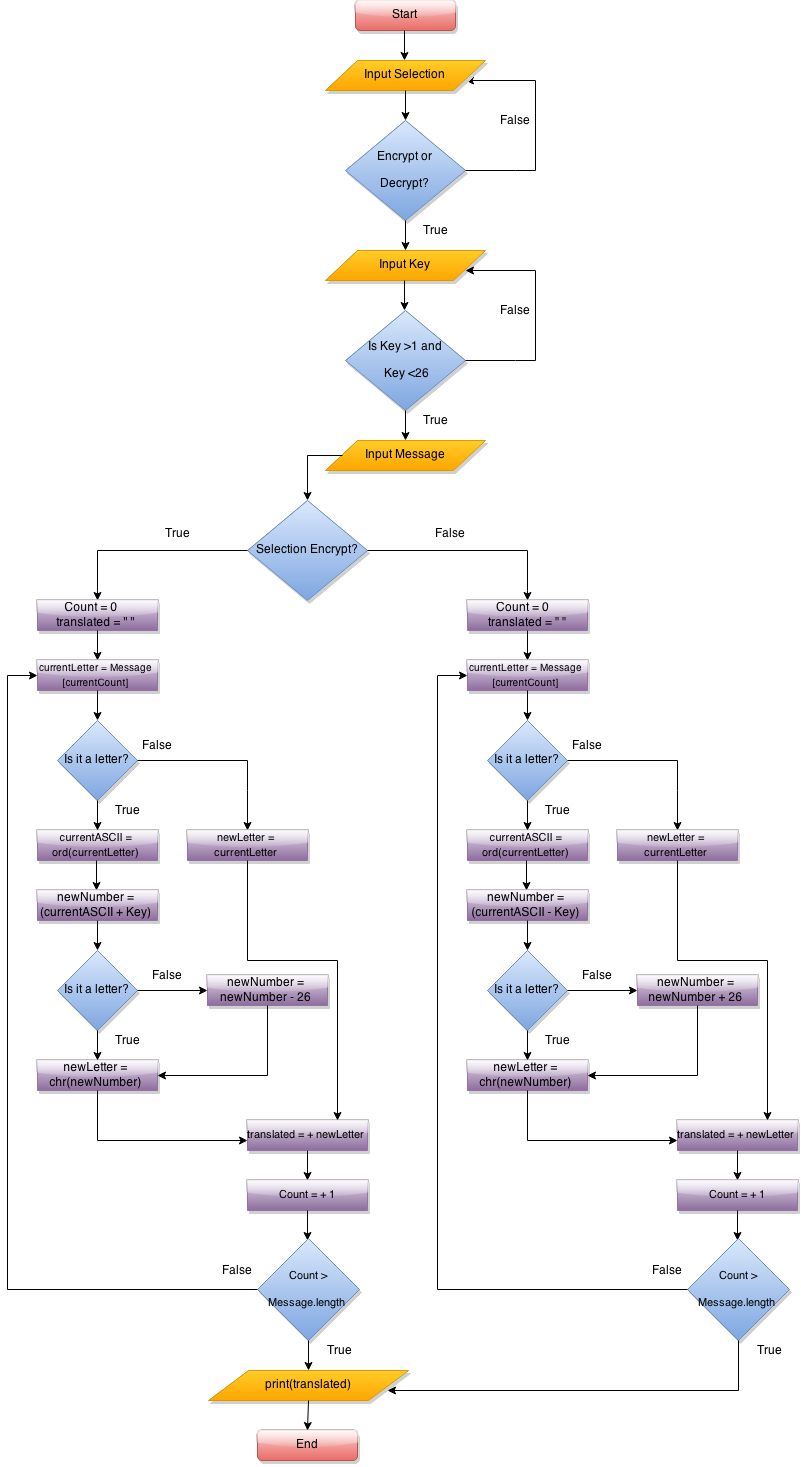
Choice = getSelection()

Msg = getMessage()

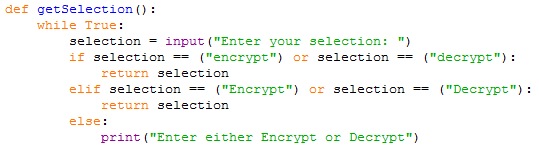
translatedMSG = translatedMessage(Choice,Msg,Key)

**Flow Chart:**

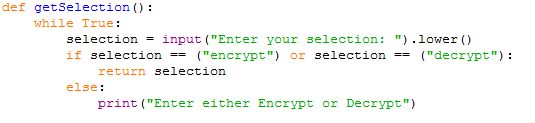
I have created a flow chart to show the steps involved in my encryption and decryption program.



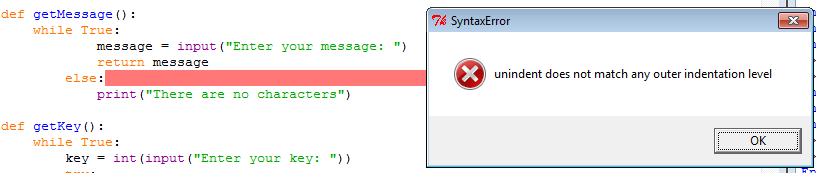
**Development:**

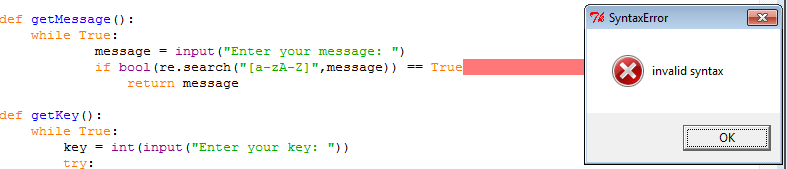


I have started my first function with an inefficient code.

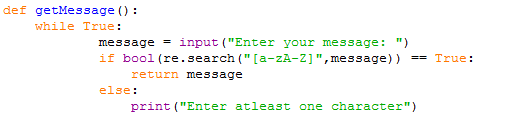


Then I have made this function more efficient, by adding the .lower() function which will make the first letter of the selection entered by the user lowercase even if it

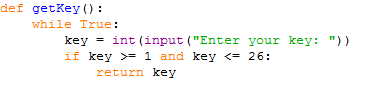
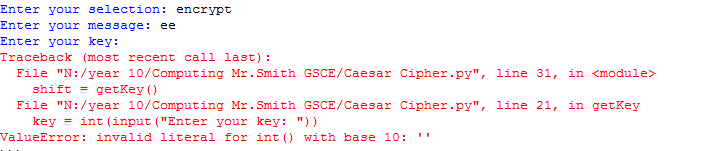


I had run across an indent error 

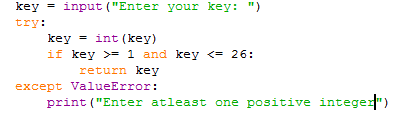
But which I had removed by adding a line of code which will search the message if it includes at least one letter, but I made a mistake of never adding colon.

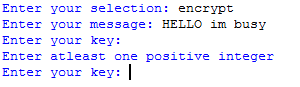


I have fixed the error by adding a colon at the end of the word ‘True’

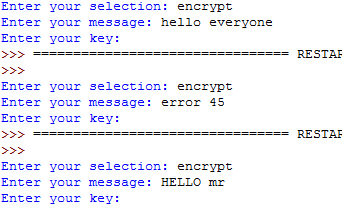
****

As I was testing my program and I had come across this error of which it would give an error if I left the key variable blank and pressed enter, and the above code is an old error code.

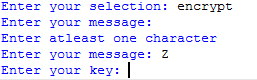




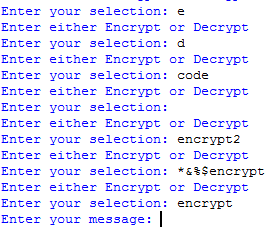
But know I have fixed my error by leaving the Int code next to my ‘input(“Enter your key: “)’ and adding except value error to end my code without any errors making it more reliable.



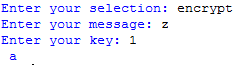


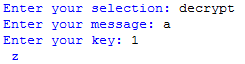


I have carried on testing my message variable which has proven to work as you can see from above.

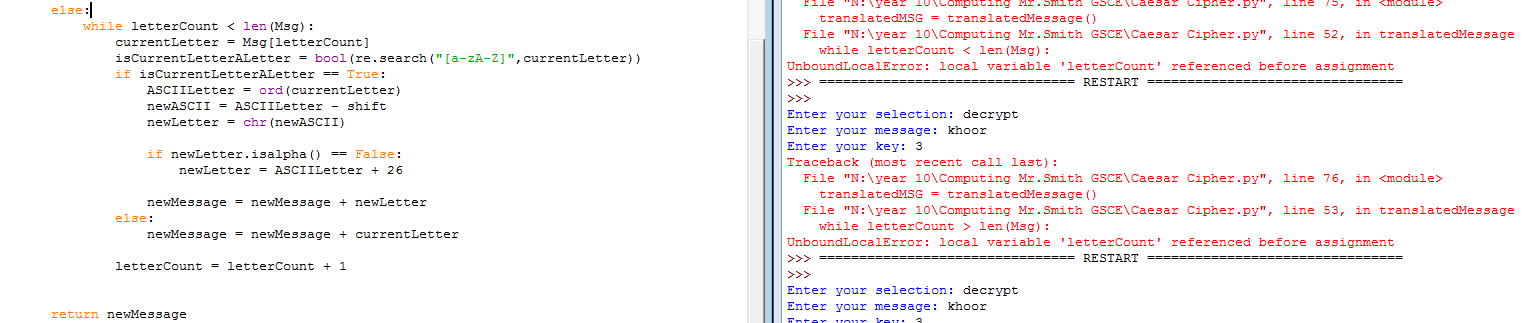




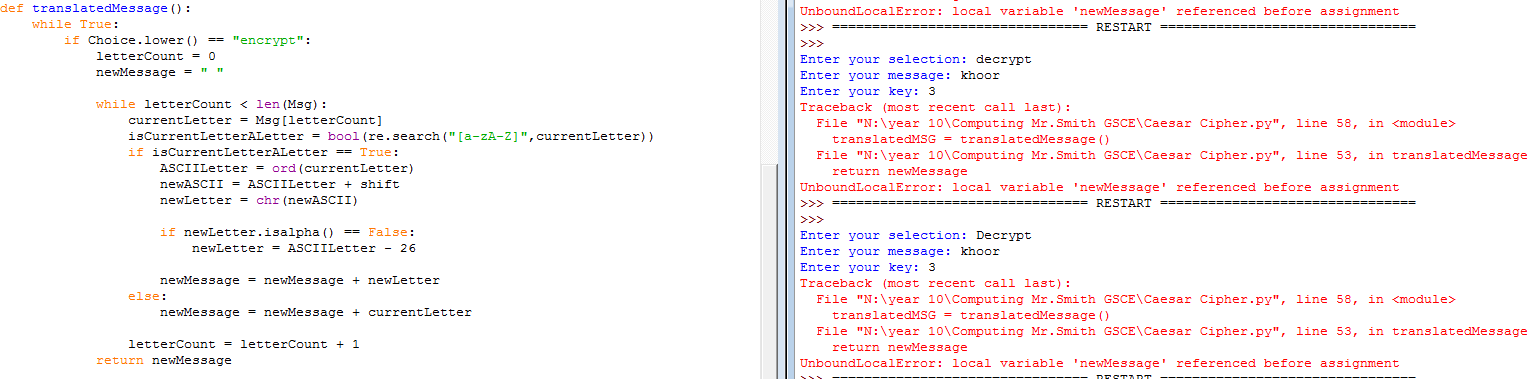




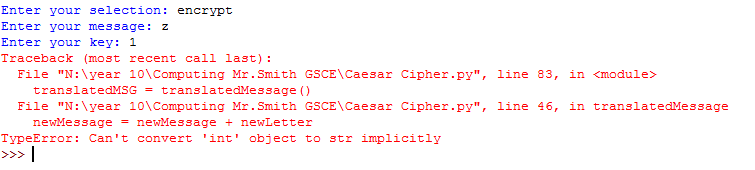
Next I have tested the encrypt and decrypt selection which works fine and inspect from the above output.



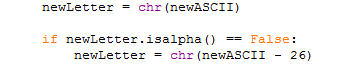
This error has been caused by not putting any variables at the top of my second part of my code.

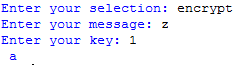


By putting variables at the top of my code solved my error, these variables will store my letters which will be added one by one and printed out at the end.

****

This part of my code wouldn’t return back from Z to A

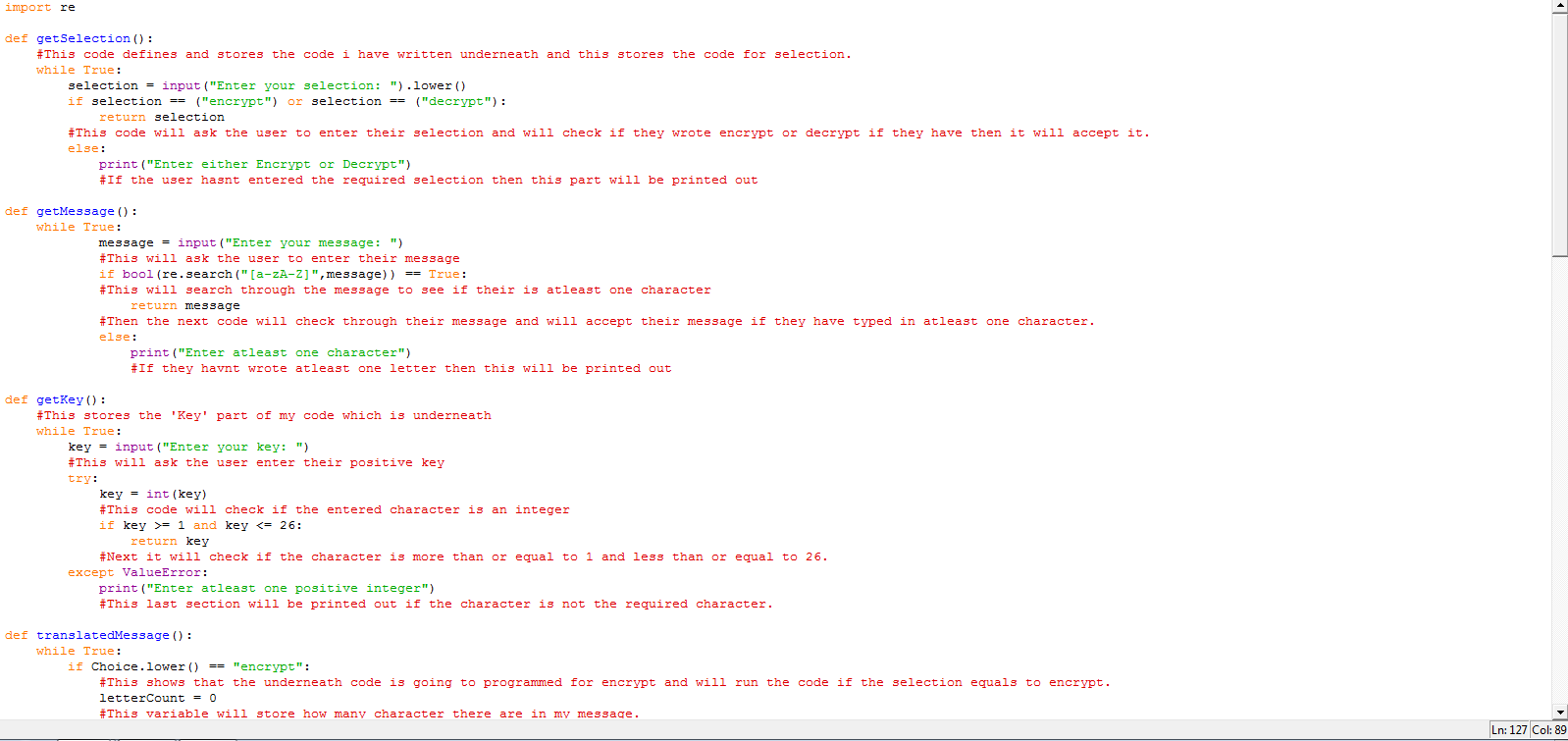


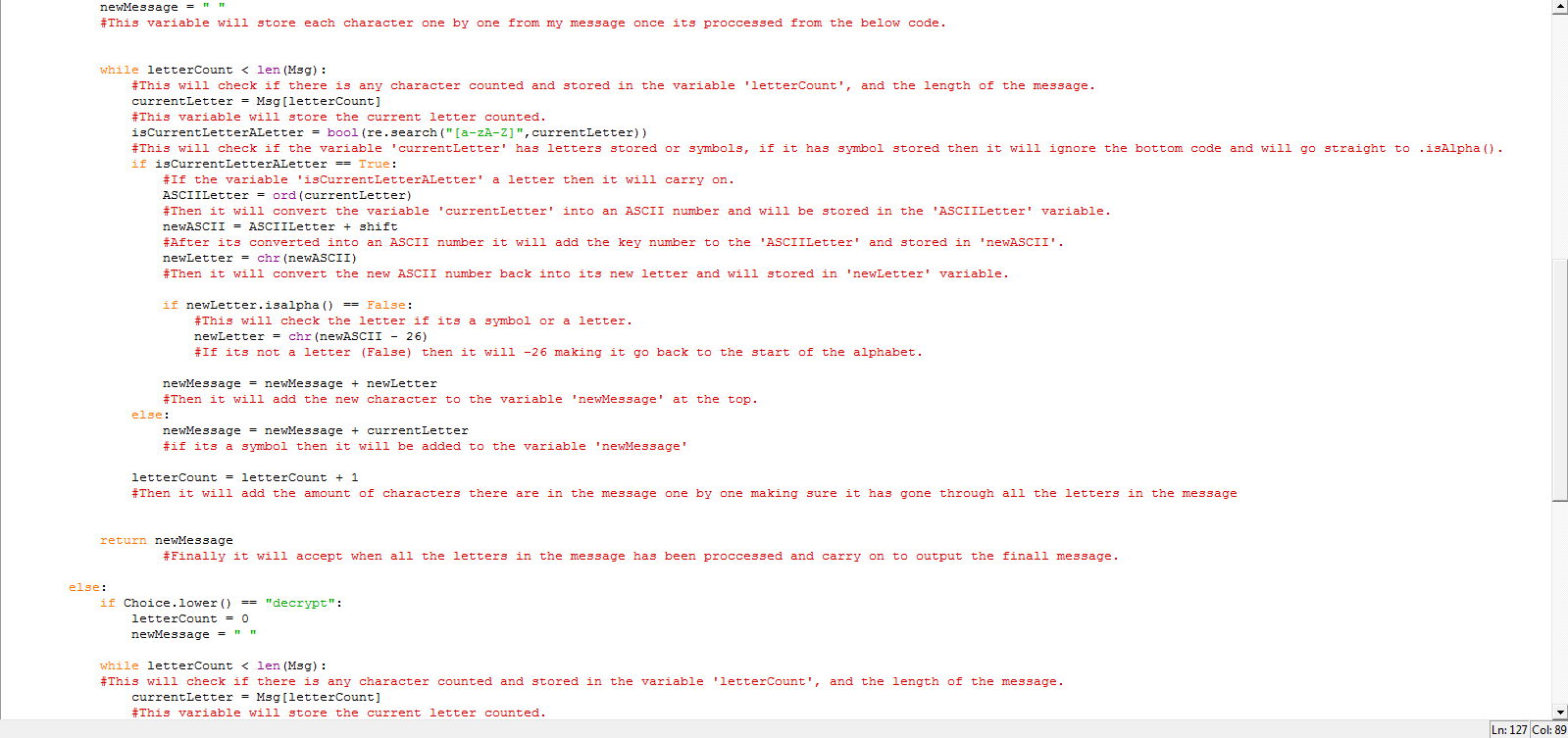


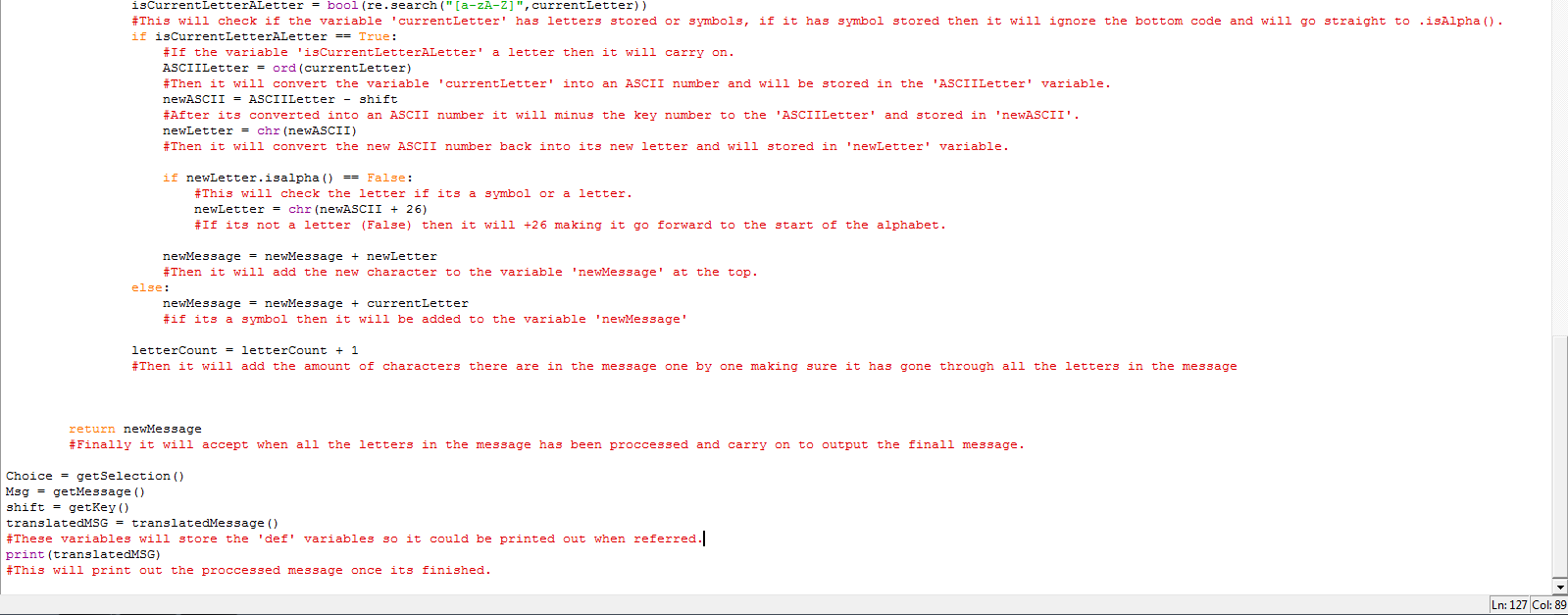
After a few codes I had finally fixed the problem with the above code

**Final Code:**

I have finished my code with comments, which you can observe from the screenshots below



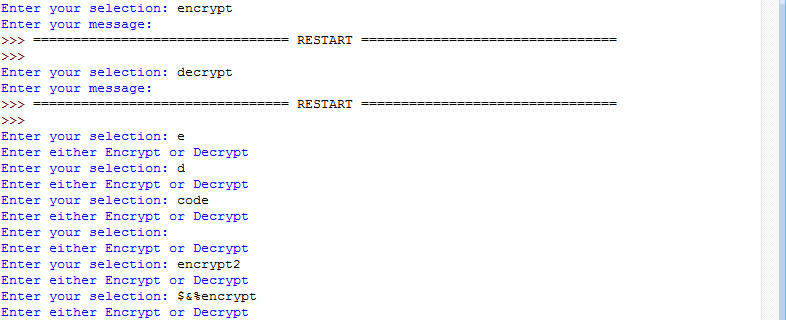




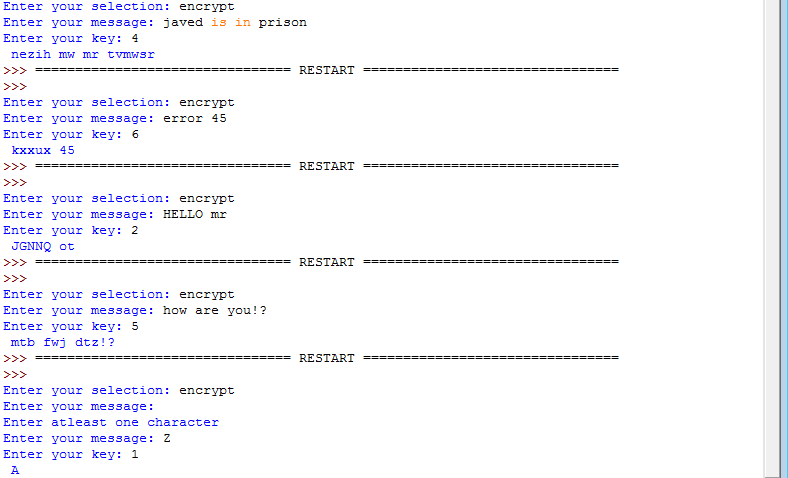
**TESTING:**

|  |  |  |  |
| --- | --- | --- | --- |
| No | Test Description | Test Data | Expected Outcome |
| 1 | Check if the program will accept encrypt or decrypt and not accept symbols numbers or leave it blank | Choice: encrypt  Message:  Key: | Accept and ask for message |
| 2 | Check if the program will accept encrypt or decrypt and not accept symbols numbers or leave it blank | Choice: decrypt  Message:  Key: | Accept and ask for message |
| 3 | Check if the program will accept e or d for encrypt or decrypt and not accept symbols or number or leave it blank | Choice: e  Message:  Key: | Reject the user and ask to enter encrypt or decrypt |
| 4 | Check if the program will accept e or d for encrypt or decrypt and not accept symbols or number or leave it blank | Choice: d  Message:  Key: | Reject the user and ask to enter encrypt or decrypt |
| 5 | Check if the program will accept encrypt or decrypt and not accept symbols numbers or leave it blank | Choice: code  Message:  Key: | Reject the user and ask to enter encrypt or decrypt |
| 6 | Check if the program will accept encrypt or decrypt and not accept symbols numbers or leave it blank | Choice: [Blank ]  Message:  Key: | Reject the user and ask to enter encrypt or decrypt |
| 7 | Check if the program will accept encrypt or decrypt and not accept symbols numbers or leave it blank | Choice: encrypt2  Message:  Key: | Reject the user and ask to enter encrypt or decrypt without numbers |
| 8 | Check if the program will accept encrypt or decrypt and not accept symbols numbers or leave it blank | Choice: $&%encrypt  Message:  Key: | Reject the user and ask to enter encrypt or decrypt  without symbols |

The above tests which are for selection, I have proved from my code which you can see from below

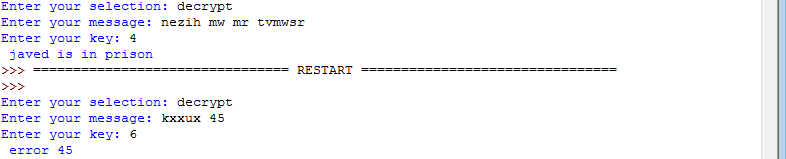


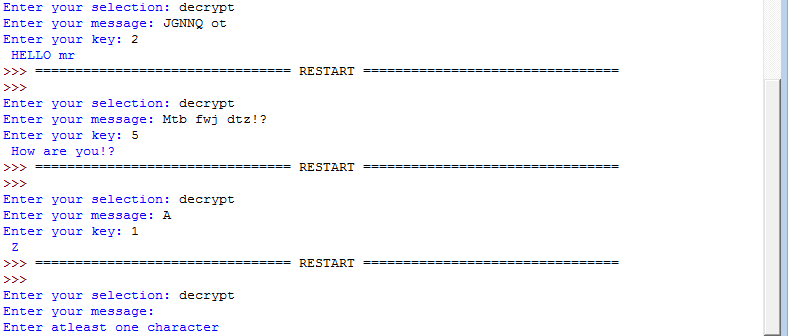
|  |  |  |  |
| --- | --- | --- | --- |
| 9 | Check if the program should ignore symbols and numbers and accept letters or capital letter. | Choice: encrypt  Message: javed is in prison  Key: 4 | “Nezih mw mr tvmwsr” |
| 10 | Check if the program should ignore symbols and numbers and accept letters or capital letter. | Choice: encrypt  Message: error 45  Key: 6 | “kxxux 45” |
| 11 | Check if the program should ignore symbols and numbers and accept letters or capital letter. | Choice: encrypt  Message: HELLO mr  Key: 2 | “JQNNQ ot” |
| 12 | Check if the program should ignore symbols and numbers and accept letters or capital letter. | Choice: encrypt  Message: how are you!?  Key: 5 | “Mtb fwj dtz!?” |
| 13 | Check if the program should ignore symbols and numbers and accept letters or capital letter. | Choice: encrypt  Message: [Blank]  Key: | Reject the user and ask to enter a message. |
| 14 | Check if the program should accept whole numbers between 1 and 26 and reject letters or symbols and A should go back to A vice versa. | Choice: encrypt  Message: Z  Key: 1 | “A” |

The above tests which are for message and encrypt , I have proved from my code tests which you can see from below

|  |  |  |  |
| --- | --- | --- | --- |
| 15 | Check if the program should ignore symbols and numbers and accept letters or capital letter. | Choice: decrypt  Message: nezih mw mr tvmwsr  Key: 4 | “javed is in prison” |
| 16 | Check if the program should ignore symbols and numbers and accept letters or capital letter. | Choice: decrypt  Message: kxxux 45  Key: 6 | “error 645” |
| 17 | Check if the program should ignore symbols and numbers and accept letters or capital letter. | Choice: decrypt  Message: JGNNQ ot  Key: 2 | “HELLO mr” |
| 18 | Check if the program should ignore symbols and numbers and accept letters. | Choice: decrypt  Message: Mtb fwj dtz!?  Key: 5 | “How are you!?” |
| 19 | Check if the program should accept whole numbers between 1 and 26 and reject letters or symbols and A should go back to A vice versa. | Choice: decrypt  Message: A  Key: 1 | “Z” |
| 20 | Check if the program should ignore symbols and numbers and accept letters or capital letter. | Choice: decrypt  Message: [Blank]  Key: | Reject the user and ask to enter a message. |

The above tests which are for message and decrypt , I have proved from my code tests which you can see from below

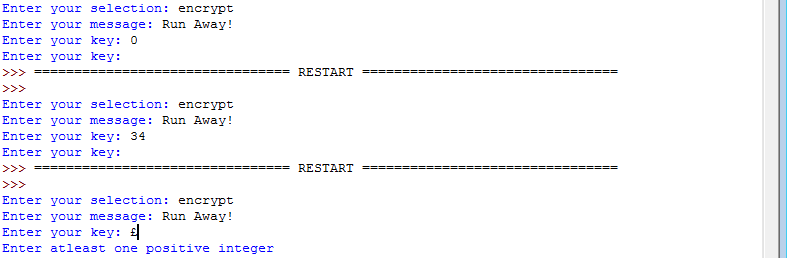


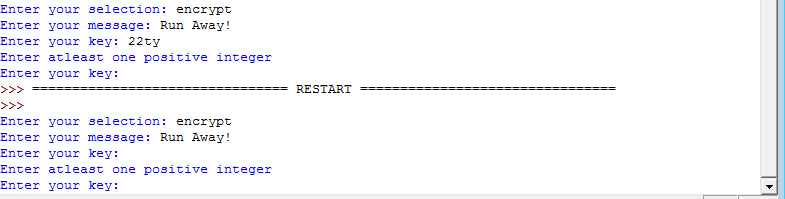


|  |  |  |  |
| --- | --- | --- | --- |
| 21 | Check if the program should accept whole numbers between 1 and 26 and reject letters or symbols or leave it blank | Choice: encrypt  Message: Run Away!  Key: ab | Reject the user and ask to enter a whole positive key number between 1-26 |
| 22 | Check if the program should accept whole numbers between 1 and 26 and reject letters or symbols or leave it blank | Choice: encrypt  Message: Run Away!  Key: -15 | Reject the user and ask to enter a whole positive key number between 1-26 |
| 23 | Check if the program should accept whole numbers between 1 and 26 and reject letters or symbols or leave it blank | Choice: encrypt  Message: Run Away!  Key: 0 | Reject the user and ask to enter a whole positive key number between 1-26 |
| 24 | Check if the program should accept whole numbers between 1 and 26 and reject letters or symbols or leave it blank | Choice: encrypt  Message: Run Away!  Key: 34 | Reject the user and ask to enter a whole positive key number between 1-26 |
| 25 | Check if the program should accept whole numbers between 1 and 26 and reject letters or symbols or leave it blank | Choice: encrypt  Message: Run Away!  Key: £ | Reject the user and ask to enter a whole positive key number between 1-26 |
| 26 | Check if the program should accept whole numbers between 1 and 26 and reject letters or symbols or leave it blank | Choice: encrypt  Message: Run Away!  Key: 22ty | Reject the user and ask to enter a whole positive key number between 1-26 |
| 27 | Check if the program should accept whole numbers between 1 and 26 and reject letters or symbols or leave it blank | Choice: encrypt  Message: Run Away!  Key: [Blank] | Reject the user and ask to enter a whole positive key number between 1-26 |

The above tests which are for key, I have proved from my code tests which you can see from below







**Evaluation:**

My program has been developed according to plan from my problem definition. My program has been thoroughly checked and made efficient as possible, I have regularly screen shot my work where errors have occurred to show my mistakes and screenshots of the fix, also I have tested and checked my output code which you can observe to work from all the above screen shots. These screen shots have been captured to show all my evidence throughout my program, my program is errorless and has been finished according to plan.

To finish my evaluation further I have must refer back to my success criteria and to prove that everything is correct.

1. My selection should only accept “encrypt” and “decrypt”.

-This has been proven by tests 1 – 8 which passed

1. My selection should reject numbers, blanks and symbols.

-This has been proved by tests 1 – 8 which passed

1. My message should accept all symbols and numbers, and it will not be case sensitive and encrypt or decrypt as followed and should contain at least one character.

-This has been proven by tests 9 – 20 which passed

1. My key should only accept positive numbers between 1 and 26 and will reject if there is any symbols, numbers, blanks and letters.

-This can be proven by tests 21 – 27 which passed

1. My output would go back to A if its Z for encrypt.

-This can be proven by test 14 which passed

1. My output would go back to Z if it’s a for decrypt.

-This can be proven by test 19 which passed

**Task 2**

**Problem definition**:

In task 2 I must create a program which will encrypt and decrypt the message by a keyword, a keyword can be a word that will be needed in order for it to encrypt and decrypt the message, the keyword will be repeated enough times until it matches the length of the message. The alphabet value of the message and keyword will be added in order for it to generate an encrypted message, or will be taken away to generate a decrypted message. There are 26 letters in the alphabet and if the final alphabet number is more than 26 then it will loop back.

For example if we use the keyword BENCH and message COMPUTINGRULES, the keyword BENCH will be repeated until it fits the 14 letter length message BENCHBENCHBENC, B represents the 2nd letter of the alphabet, E the 5th letter, N the 14th letter, C the 3rd letter, H the 8th letter.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Message | C | O | M | P | U | T | I | N | G | R | U | L | E | S |
| Alphabet Value | 3 | 15 | 13 | 16 | 21 | 20 | 9 | 14 | 7 | 18 | 21 | 12 | 5 | 17 |
| Keyword | B | E | N | C | H | B | E | N | C | H | B | E | N | C |
| Alphabet value of keyword | 2 | 5 | 14 | 3 | 8 | 2 | 5 | 14 | 3 | 8 | 2 | 5 | 14 | 3 |
| Alphabet value of message +  Alphabet value of keyword | 5 | 20 | 27 | 19 | 29 | 22 | 14 | 28 | 10 | 26 | 23 | 17 | 19 | 20 |
| Encrypted message | E | T | A | S | C | V | N | B | J | Z | W | Q | S | T |

I am using this type of program because it’s more secure and will be harder to crack.

**Success criteria**:

I am using success criteria to show if my program works according to my success criteria and if it’s successful. And this will show if my success criteria have been met. I will be changing a few lines of code of task 1 and will not re-test code from task 1 which have been already shown to work.

* Keyword that will accept any length of word
* The keyword should only accept at least one letter and no symbols or numbers
* The keyword should encrypt and decrypt correctly
* The keyword and message will not be case sensitive

**Test Plan:**

|  |  |  |  |
| --- | --- | --- | --- |
| No | Test Description | Test Data | Expected Outcome |
| 1 | The keyword should only accept letters as the keyword and any length of letter and reject symbols | Choice: Encrypt  Keyword: BENCH  Message: | Accept and ask to input the message |
| 2 | The keyword should only accept letters as the keyword and any length of letter and reject symbols | Choice: Encrypt  Keyword: bench  Message: | Accept and ask to input the message |
| 3 | The keyword should only accept letters as the keyword and any length of letter and reject symbols | Choice: Encrypt  Keyword: BENCH!$  Message: | Reject and ask to input the keyword again |
| 4 | The keyword should only accept letters as the keyword and any length of letter and reject symbols | Choice: Encrypt  Keyword: [Blank]  Message: | Reject and ask to input the keyword again |
| 5 | The keyword should only accept letters as the keyword and any length of letter and reject symbols | Choice: Encrypt  Keyword: BENCH123  Message: | Reject and ask to input the keyword again |
| 6 | The keyword will not allow spaces and will not work | Choice: Encrypt  Keyword: BENCH B U S  Message: | Reject and ask to input the keyword again |
| 7 | The message should only accept at least one letter and ignore symbols and numbers | Choice: Encrypt  Keyword: BENCH  Message: COMPUTING | “ETASCVNBJ” |
| 8 | The message should only accept at least one letter and ignore symbols and numbers | Choice: Encrypt  Keyword: BENCH  Message: cryptography | “ewmsbqlfdxjd” |
| 9 | The message should only accept at least one letter and ignore symbols and numbers | Choice: Encrypt  Keyword: BENCH  Message: COMPUTING!$ | “ETASCVNBJ!$” |
| 10 | The message should only accept at least one letter and ignore symbols and numbers | Choice: Encrypt  Keyword: BENCH  Message: COMPUTING123 | “ETASCVNBJ123” |
| 11 | The message should only accept at least one letter and ignore symbols and numbers | Choice: Encrypt  Keyword: BENCH  Message: T H I S W I L L  H E L P | “V M W V E K Q Z  K M N U” |
| 12 | The message should only accept at least one letter and ignore symbols and numbers | Choice: Decrypt  Keyword: BENCH  Message: ETASCVNBJ | “COMPUTING” |
| 13 | The message should only accept at least one letter and ignore symbols and numbers | Choice: Decrypt  Keyword: BENCH  Message: ewmsbqlfdxjd | “cryptography” |
| 14 | The message should only accept at least one letter and ignore symbols and numbers | Choice: Decrypt  Keyword: BENCH  Message: ETASCVNBJ!$ | “COMPUTING!$” |

|  |  |  |  |
| --- | --- | --- | --- |
| 15 | The message should only accept at least one letter and ignore symbols and numbers | Choice: Decrypt  Keyword: BENCH  Message: ETASCVNBJ123 | “COMPUTING123” |
| 16 | The message should only accept at least one letter and ignore symbols and numbers | Choice: Decrypt  Keyword: BENCH  Message: V M W V E K Q Z K M N U | “T H I S W I L L H E L P” |
| 17 | The keyword can accept any length of letters, the letter can be longer than the message or shorter than the message | Choice: Encrypt  Keyword: B  Message: HELLO | “JGNNQ” |
| 18 | The keyword can accept any length of letters, the letter can be longer than the message or shorter than the message | Choice: Encrypt  Keyword: ABC  Message: HELLO | “IGOMQ” |
| 19 | The keyword can accept any length of letters, the letter can be longer than the message or shorter than the message | Choice: Encrypt  Keyword: CPURAM  Message: BINARY | “EYISSL” |
| 20 | The keyword can accept any length of letters, the letter can be longer than the message or shorter than the message | Choice: Encrypt  Keyword: HEXADECIMAL  Message: BINARY | “JNLBVD” |
| 21 | The keyword can accept any length of letters, the letter can be longer than the message or shorter than the message | Choice: Decrypt  Keyword: HEXADECIMAL  Message: JNLBVD | "BINARY” |

**Proposed solution**:

The program language that I am going to use in my controlled assessment is python. I will be using this to create my encryption and decryption program. My program will start off with an input to ask the user which choice to input, the choice of encrypt or decrypt. As validation I will use iteration, until the user has made a correct selection, which will be a string.

After the selection the program will ask the user to input keyword and once again I will use iteration until the user enters a valid keyword, which should only be letters and can be any length. Finally the user will enter a message for the program to calculate, once again I will use iteration until a valid message is entered, which should only be letters of lowercase or uppercase and should accept symbols and number because it will be ignored. These three information will be stored in three variables (Choice, Keyword, and Message).

The program will perform a selection depending on the users choice of encrypt or decrypt. And after the decision it will be calculated.

In my program, to encrypt and decrypt messages I will be using the python built in functions such as Ord and Chr. Ord is a function that converts letters into their unique ASCII number. The keyword will match the length of the message, then using Ord it will loop through each letter converting the keyword into its ASCII number then It will loop through each letter again converting the message into its ASCII number. If the user has selected encrypt then each letter in the ASCII keyword will be added with each letter of the ASCII message otherwise if the user has selected decrypt then each letter in the ASCII keyword will be subtracted with each letter of the ASCII message, this will be done to show the new ASCII letter either for encrypt or decrypt. When this is done the Chr function will convert the new ASCII letter of the message back into its alphabetic letters.

If the letter has gone over “Z” then it will be symbol because we expect the letter “a = 1” and “b = 2” and “c = 3” and so on, but actually it’s not, the ASCII value of capital “A” is 65 so if it has gone more than “Z” then we would -64 and then -26 for it to show the new ASCII number for the new letter plus the ASCII value for lowercase letter “a” is 97 so if it has gone over “z” then you will have to -96 and then -26. And it’s the same concept for when decrypt is selected but this time if it has gone beyond “A” then it would +64 and +26 or if it’s gone beyond “a” then it would +96.

To make my understanding clear I have constructed a chart.

Uppercase: Encrypt

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Message | Z | E | B | R | A |
| ASCII number of each letter of message | 90 | 69 | 66 | 82 | 65 |
| Keyword | B | E | N | C | H |
| ASCII number of each letter of keyword | 66 | 69 | 78 | 67 | 72 |
| Message + Keyword of each letter | 156 | 138 | 144 | 149 | 137 |
| Message and Keyword added number of each letter, -64 | 66( 156-64-26 Because message letter has gone beyond Z) | 74 | 80 | 85 | 73 |
| Encrypted Message | B | J | P | U | I |

After my program has done calculating, the calculated message will be printed out.

**Pseudo code:**

I have created a pseudo code to show the code of my program that will help me throughout my actual program

**FUNCTION** getSelection()

**WHILE TRUE**:

**INPUT** Selection

**IF** Selection == “Encrypt” **OR** Selection == “Decrypt” **THEN**

**RETURN** Selection

**ELSE**

**OUTPUT** “Enter Encrypt or Decrypt only”

**LOOP**

**END FUNCTION**

**FUNCTION** getKeyword()

**WHILE TRUE**:

**INPUT** Keyword

**IF** Keyword.isalpha() == True:

**RETURN** Keyword

**ELSE** :

**OUTPUT** “Keyword must only contain atleast one letter”

**LOOP**

**END FUNCTION**

**FUNCTION** getMessage()

**WHILE TRUE**:

**INPUT** Message

**IF** Letter in Message **THEN**

**RETURN** Message

**ELSE**  **OUTPUT** “The Message contains no letters”

**LOOP**

**END FUNCTION**

**IF** Selection == Encrypt **THEN**

currentASCII = CapitalA

**WHILE** currentLoop < currentKeyword:

**WHILE** currentASCII = < CapitalA:

currentLetter = CHR(currentASCII)

**OUTPUT** currentASCII + “:” + currentLetter

currentASCII = currentASCII + 1

**LOOP**

currentASCII = currentASCII + 26

currentLoop = currenLoop + 1

**LOOP**

**ELSE**

currentASCII = CapitalZ

**WHILE** currentLoop < currentKeyword:

**WHILE** currentASCII = < CapitalZ:

currentLetter = CHR(currentNumber)

**OUTPUT** currentASCII + “:” + currentLetter

currentASCII = currentNumber + 1

**LOOP**

currentASCII = currentASCII - 26

currentLoop = currenLoop + 1

**LOOP**

**END IF**

keyW1 = getKeyword()

Choice = getSelection()

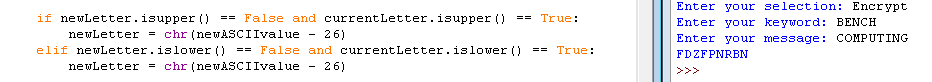
Msg = getMessage()

translatedMSG = translatedMessage(Choice,keyW1,Msg)

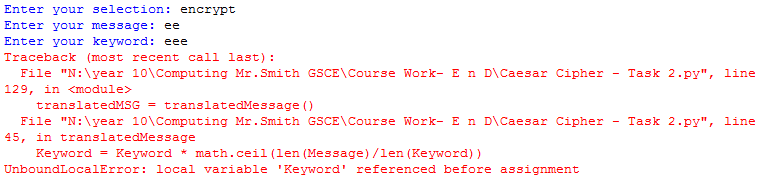
**Development**:

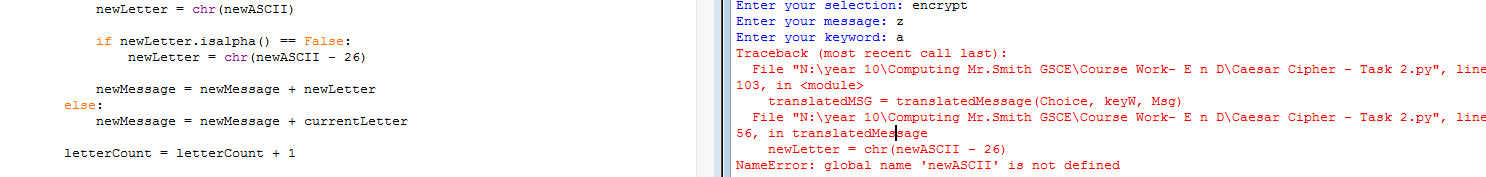
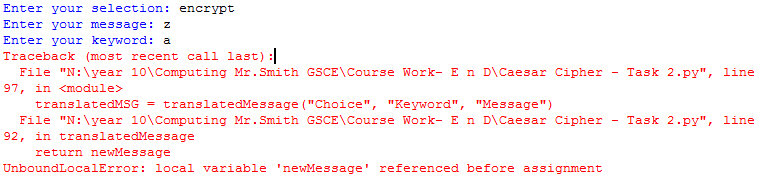


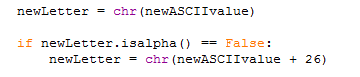
Some letters were given lower case and they were the wrong letters

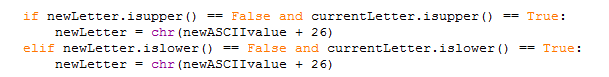


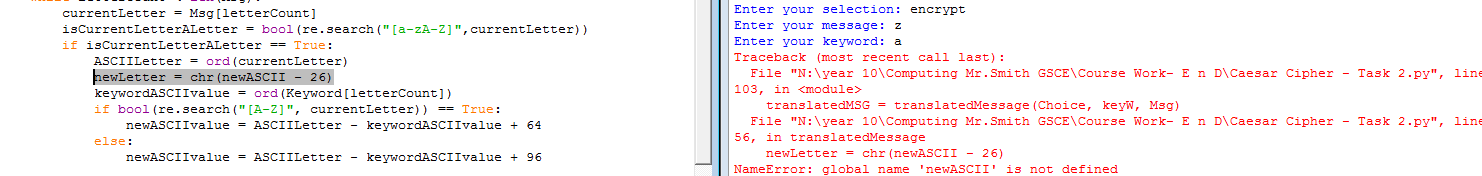
I have fixed this error by putting the above code which will check if the new letter is not upper case and the current letter is uppercase then we would -26 to the new ASCII value. But if the new letter is not lower case and the current letter is lower case then we would -26 to the new ASCII value.



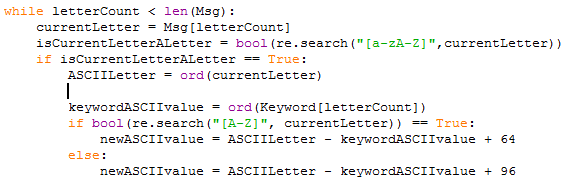
This error had occurred when I started programing task 2 



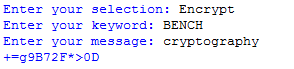
This code did not do any good as it caused errors which then I had modified it so it would work. You can see the modified code below



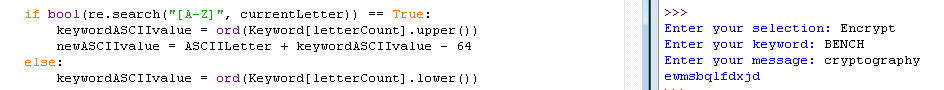
The error also occurred by the above highlighted code



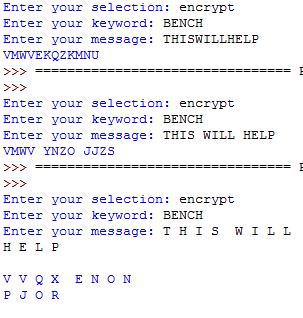
But later on I had deleted the highlighted line of code which fixed the problem as it was not needed



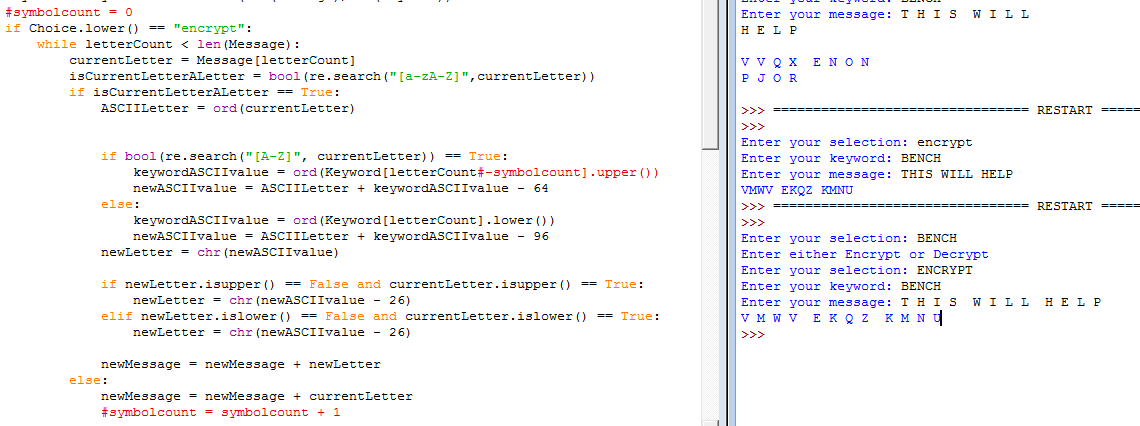
This error was occurred because the keyword was upper case and the message was lower case this caused error



But I fixed the symbol error by adding .upper after [letterCount] on the second line on the above code and .lower after [letterCount] on the fifth line



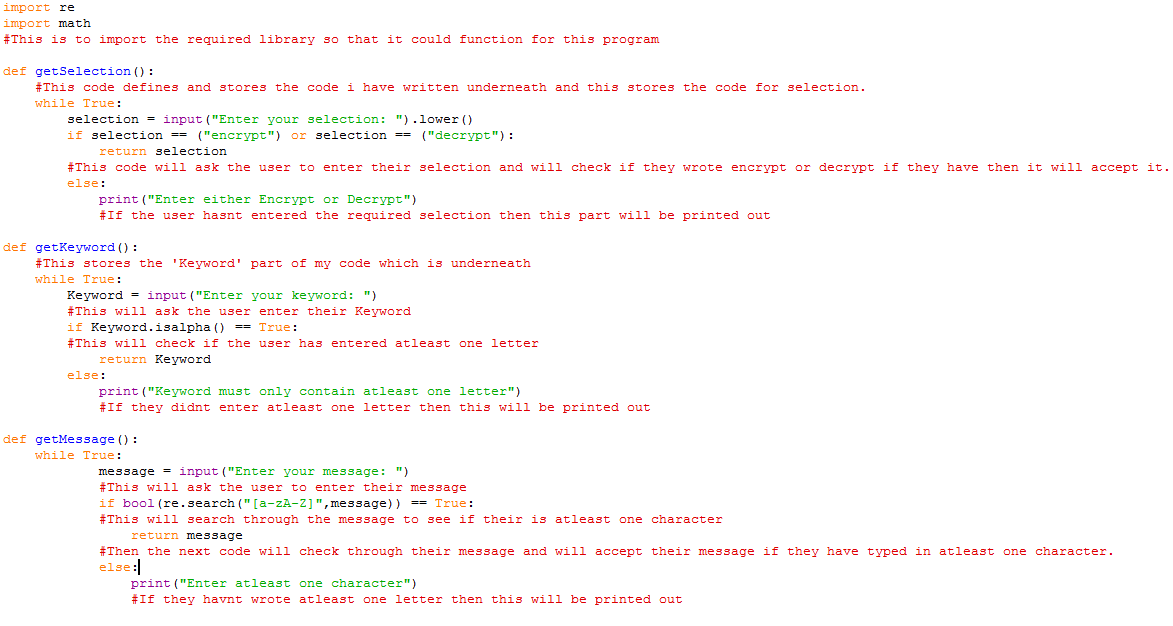
The above error is an error that it would change the output sentence if you put a space because it will count it as a letter.

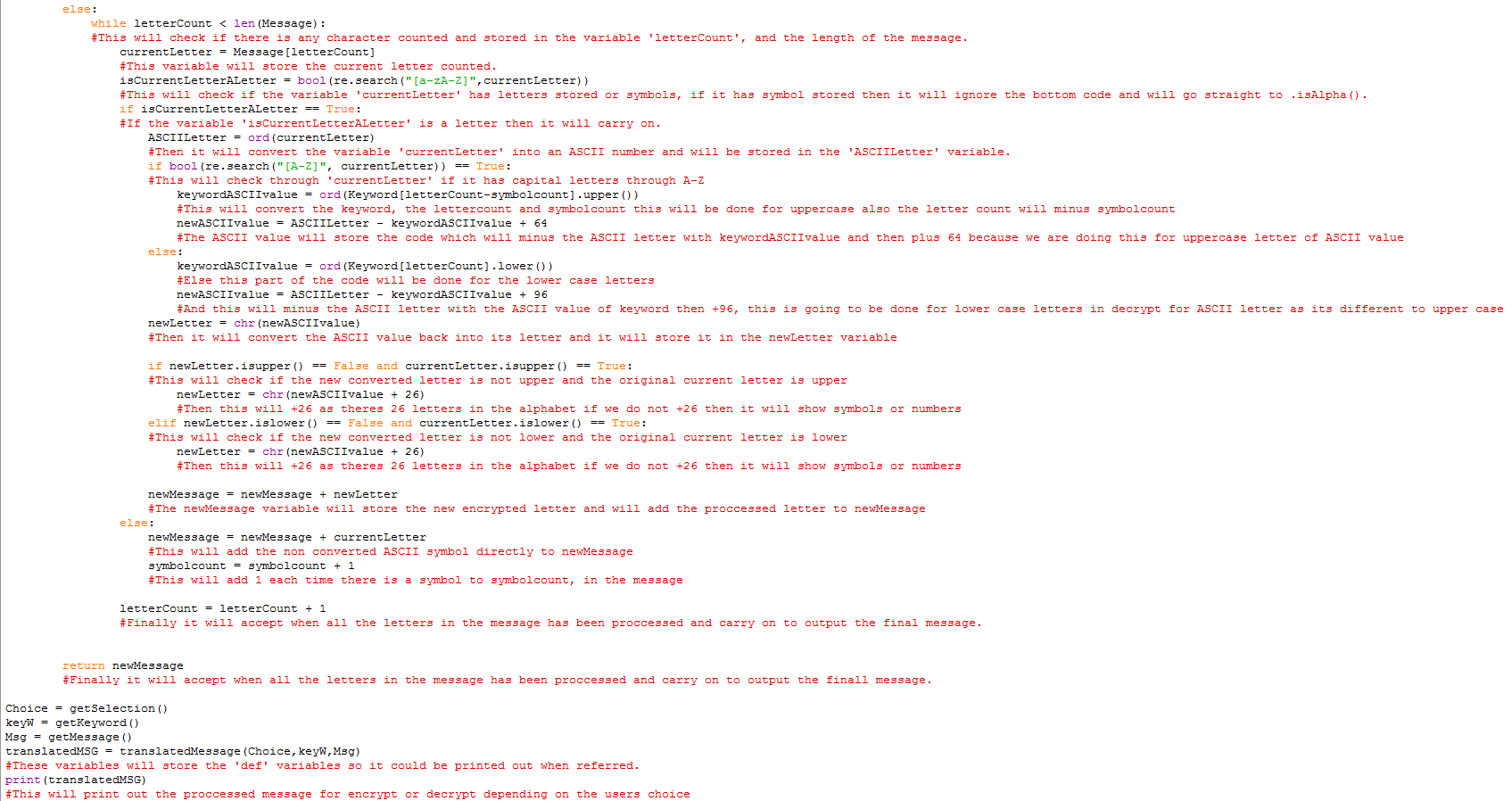
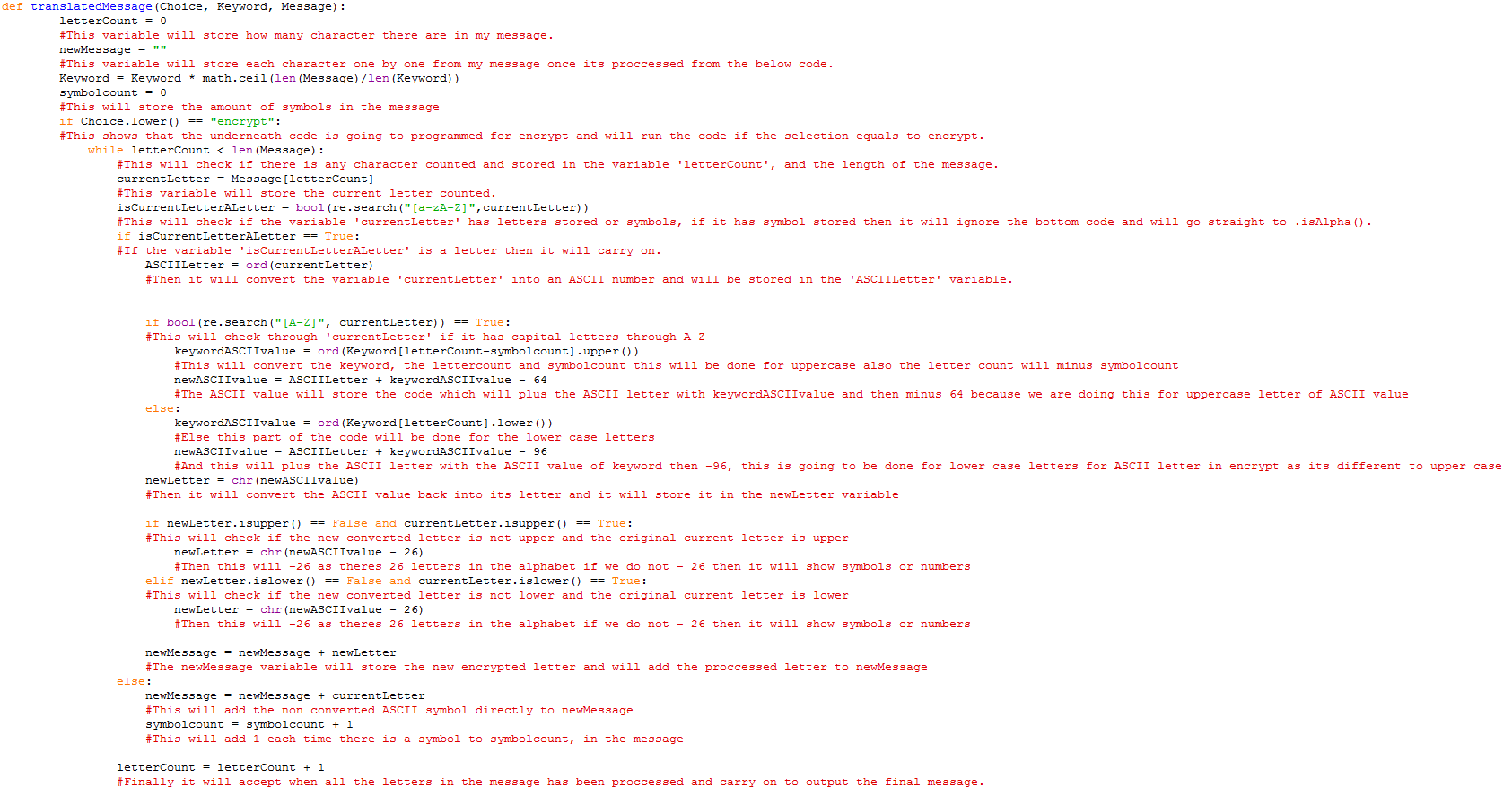


But then I had fixed the error by putting a symbol variable and putting ‘-symbolcount’ which will subtract the symbol or space if there is one, plus the variable will store the processed symbol or space at the end. You can see the fixed code above, the red highlighted codes are the added code which fixed the bug

**Final code:**

I have finished my code with comments, which you can observe from the screenshots below

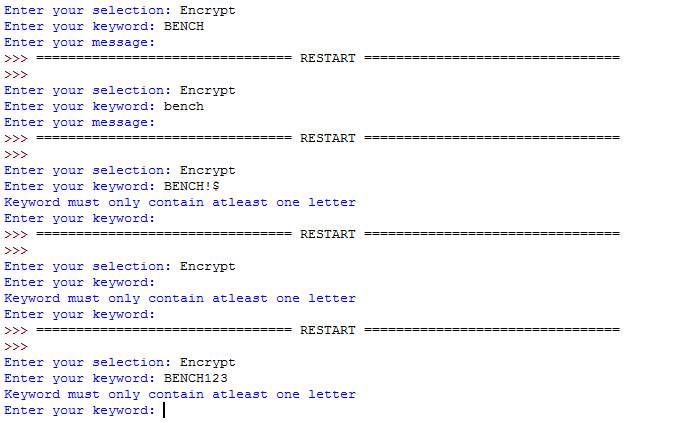




**Testing:**

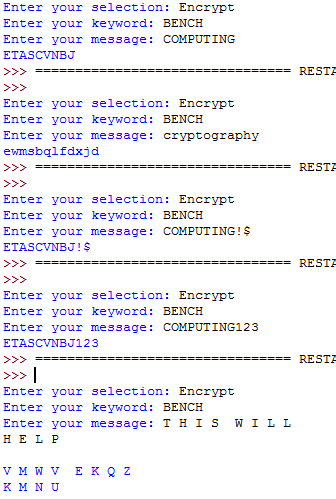
|  |  |  |  |
| --- | --- | --- | --- |
| 1 | The keyword should only accept letters as the keyword and any length of letter and reject symbols | Choice: Encrypt  Keyword: BENCH  Message: | Accept and ask to input the message |
| 2 | The keyword should only accept letters as the keyword and any length of letter and reject symbols | Choice: Encrypt  Keyword: bench  Message: | Accept and ask to input the message |
| 3 | The keyword should only accept letters as the keyword and any length of letter and reject symbols | Choice: Encrypt  Keyword: BENCH!$  Message: | Reject and ask to input the keyword again |
| 4 | The keyword should only accept letters as the keyword and any length of letter and reject symbols | Choice: Encrypt  Keyword: [Blank]  Message: | Reject and ask to input the keyword again |
| 5 | The keyword should only accept letters as the keyword and any length of letter and reject symbols | Choice: Encrypt  Keyword: BENCH123  Message: | Reject and ask to input the keyword again |

Using the test plan above I have tested the keyword, as you can see from the screenshot below



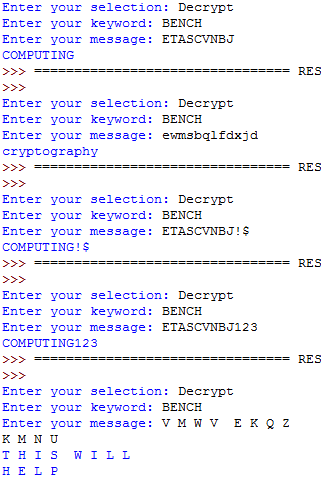
|  |  |  |  |
| --- | --- | --- | --- |
| 6 | The message should only accept at least one letter and ignore symbols and numbers | Choice: Encrypt  Keyword: BENCH  Message: COMPUTING | “ETASCVNBJ” |
| 7 | The message should only accept at least one letter and ignore symbols and numbers | Choice: Encrypt  Keyword: BENCH  Message: cryptography | “ewmsbqlfdxjd” |
| 8 | The message should only accept at least one letter and ignore symbols and numbers | Choice: Encrypt  Keyword: BENCH  Message: COMPUTING!$ | “ETASCVNBJ!$” |
| 9 | The message should only accept at least one letter and ignore symbols and numbers | Choice: Encrypt  Keyword: BENCH  Message: COMPUTING123 | “ETASCVNBJ123” |
| 10 | The message should only accept at least one letter and ignore symbols and numbers | Choice: Encrypt  Keyword: BENCH  Message: T H I S W I L L  H E L P | “V M W V E K Q Z  K M N U” |

Using the test plan above I have tested Encrypt, as you can see from the screenshot below



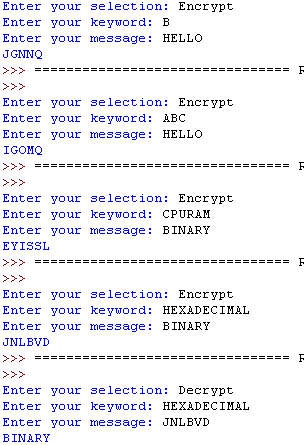
|  |  |  |  |
| --- | --- | --- | --- |
| 11 | The message should only accept at least one letter and ignore symbols and numbers | Choice: Decrypt  Keyword: BENCH  Message: ETASCVNBJ | “COMPUTING” |
| 12 | The message should only accept at least one letter and ignore symbols and numbers | Choice: Decrypt  Keyword: BENCH  Message: ewmsbqlfdxjd | “cryptography” |
| 13 | The message should only accept at least one letter and ignore symbols and numbers | Choice: Decrypt  Keyword: BENCH  Message: ETASCVNBJ!$ | “COMPUTING!$” |
| 14 | The message should only accept at least one letter and ignore symbols and numbers | Choice: Decrypt  Keyword: BENCH  Message: ETASCVNBJ123 | “COMPUTING123” |
| 15 | The message should only accept at least one letter and ignore symbols and numbers | Choice: Decrypt  Keyword: BENCH  Message: V M W V E K Q Z K M N U | “T H I S W I L L H E L P” |

Using the test plan above I have tested Decrypt, as you can see from the screenshot below



|  |  |  |  |
| --- | --- | --- | --- |
| 16 | The keyword can accept any length of letters, the letter can be longer than the message or shorter than the message | Choice: Encrypt  Keyword: B  Message: HELLO | “JGNNQ” |
| 17 | The keyword can accept any length of letters, the letter can be longer than the message or shorter than the message | Choice: Encrypt  Keyword: ABC  Message: HELLO | “IGOMQ” |
| 18 | The keyword can accept any length of letters, the letter can be longer than the message or shorter than the message | Choice: Encrypt  Keyword: CPURAM  Message: BINARY | “EYISSL” |
| 19 | The keyword can accept any length of letters, the letter can be longer than the message or shorter than the message | Choice: Encrypt  Keyword: HEXADECIMAL  Message: BINARY | “JNLBVD” |
| 20 | The keyword can accept any length of letters, the letter can be longer than the message or shorter than the message | Choice: Decrypt  Keyword: HEXADECIMAL  Message: JNLBVD | "BINARY” |

Using the test plan above I have tested the different lengths for keyword, as you can see from the screenshot below



**Evaluation:**

My program has been developed according to plan from my problem definition. My program has been thoroughly checked and made efficient as possible, I have regularly screen shot my work where errors have occurred to show my mistakes and screenshots of the fix, also I have tested and checked my output code which you can observe to work from all the above screen shots. These screen shots have been captured to show all my evidence throughout my program, my program is errorless and has been finished according to plan.

To finish my evaluation further I have must refer back to my success criteria and to prove that everything is correct.

1. My keyword should only accept at least one letter and no symbols or numbers.

-This has been proved by tests 1 – 5 which passed

1. My keyword should encrypt and decrypt correctly.

-This has been proven by tests 6 – 15 which passed

1. My keyword and message will not be case sensitive.

-This can be proven by tests 7 and 12 which passed

1. My keyword that will accept any length of word.

-This can be proven by test 16 – 20 which passed

**Task 3**

**Problem definition**:

In task 3 I must create a program which will encrypt and decrypt the message by two keywords, a keyword can be a word that will be needed in order for it to encrypt and decrypt the message, the first keyword will be repeated enough timed until it matches the length of the message, then each letter of the message will be encrypted or decrypted with the first keyword, then the second keyword will be repeated enough times again until it matches the length of the new message and then the new letter of the new message will be encrypted or decrypted with the second keyword. The alphabet value of the message and keyword will be added in order for it to generate an encrypted message, or will be taken away to generate a decrypted message. There are 26 letters in the alphabet and if the alphabet number is more than 26 then it will loop back.

For example if we use the first keyword as ADDER and message EXPANSIONCARDS, the keyword ADDER will be repeated until it fits the 14 letter length message ADDERADDERADDE, and then encrypted with the message which will be FBTFFTMSSUBVHY. But it will be encrypted again but with the second keyword which is EXABYTEEXABYTE and the final encrypted message will be KYUHDNRXQVDTBC. A from the keyword ADDER represents the 1st letter of the alphabet, D the 4th letter, D the 4th letter, E the 5th letter, R the 18th letter.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Message | E | X | P | A | N | S | I | O | N | C | A | R | D | S |
| Alphabet Value | 5 | 24 | 16 | 1 | 14 | 19 | 9 | 15 | 14 | 3 | 1 | 18 | 4 | 19 |
| Keyword 1 | A | D | D | E | R | A | D | D | E | R | A | D | D | E |
| Alphabet value of keyword 1 | 1 | 4 | 4 | 5 | 18 | 1 | 4 | 4 | 5 | 18 | 1 | 4 | 4 | 5 |
| Keyword 2 | E | X | A | B | Y | T | E | E | X | A | B | Y | T | E |
| Alphabet value of keyword 2 | 5 | 24 | 1 | 2 | 25 | 20 | 5 | 5 | 24 | 1 | 2 | 24 | 20 | 5 |
| Alphabet value of message +  Alphabet value of Keyword1 and Keyword2 | 11 | 52 | 21 | 8 | 57 | 40 | 18 | 24 | 43 | 22 | 4 | 46 | 28 | 29 |
| Encrypted message | K | Y | U | H | D | N | R | X | Q | V | D | T | B | C |

I am using this type of program because it’s more secure and will be harder to crack.

**Success criteria:**

I am using success criteria to show if my program works according to my success criteria and if it’s successful. And this will show if my success criteria have been met. I will be changing a few lines of code of task 2 and will not re-test code from task 1 which has been already shown to work.

* Both Keywords will accept any length of word
* Both keywords should only accept at least one letter and no symbols or numbers
* Both keywords should encrypt and decrypt correctly
* Both keywords and message will not be case sensitive

**Test Plan:**

|  |  |  |  |
| --- | --- | --- | --- |
| No | Test Description | Test Data | Expected Outcome |
| 1 | Both keywords should only accept letters and any length of letter and reject symbols or numbers and should not be case sensitive | Choice: Encrypt  Keyword1: BENCH  Keword2: WHEN  Message: | Accept and ask to input the message |
| 2 | Both keywords should only accept letters and any length of letter and reject symbols or numbers and should not be case sensitive | Choice: Encrypt  Keyword1: BENCH  Keword2: when  Message: | Accept and ask to input the message |
| 3 | Both keywords should only accept letters and any length of letter and reject symbols or numbers and should not be case sensitive | Choice: Encrypt  Keyword1: BENCH!$  Keword2:HELLO  Message: | Reject and ask to input the keyword again |
| 4 | Both keywords should only accept letters and any length of letter and reject symbols or numbers and should not be case sensitive | Choice: Encrypt  Keyword1: WHAT332  Keword2: WHEN42  Message: | Reject and ask to input the keyword again |
| 5 | Both keywords should only accept letters and any length of letter and reject symbols or numbers and should not be case sensitive | Choice: Encrypt  Keyword1: BENCH Keword2: [Blank]  Message: | Reject and ask to input the keyword again |
| 6 | Both keywords should not have spaces and will not work | Choice: Encrypt  Keyword: T H I S W I L L  H E L P  Keword2: JAVA SCRIPT  Message: | Reject and ask to input the keyword again |
| 7 | Both keywords cannot be the same and it will not work | Choice: Encrypt  Keyword: JABER  Keword2: JABER  Message: | Reject and ask to input the keyword again |
| 8 | The message should encrypt from both keywords correctly | Choice: Encrypt  Keyword: BENCH  Keword2: WHEN  Message: FORMAT | “EBKDFD” |
| 9 | The message should encrypt from both keywords correctly | Choice: Encrypt  Keyword: BENCH  Keword2: WHEN  Message: format | “ebkdfd” |
| 10 | The message should encrypt from both keywords correctly | Choice: Encrypt  Keyword: BENCH  Keword2: WHEN  Message: PACKETS!$ | “ONVBJDC!$” |
| 11 | The message should encrypt from both keywords correctly | Choice: Encrypt  Keyword1: BENCH  Keword2: WHEN  Message: NODE123 | “MBWV123” |
| 12 | The message should encrypt from both keywords correctly | Choice: Encrypt  Keyword1: BENCH  Keword2: WHEN  Message: B I G T O W E R | “A V Z K T G O T” |
| 13 | The message should decrypt from both keywords correctly | Choice: Decrypt  Keyword1: BENCH  Keword2: WHEN  Message: EBKDFD | “FORMAT” |
| 14 | The message should decrypt from both keywords correctly | Choice: Decrypt  Keyword1: BENCH  Keword2: WHEN  Message: ebkdfd | “format” |
| 15 | The message should decrypt from both keywords correctly | Choice: Decrypt  Keyword1: BENCH Keword2: WHEN  Message: ONVBJDC!$ | “PACKETS!$” |
| 16 | The message should decrypt from both keywords correctly | Choice: Decrypt  Keyword1: BENCH  Keword2: WHEN  Message: MBWV123 | “NODE123” |
| 17 | The message should decrypt from both keywords correctly | Choice: Decrypt  Keyword1: BENCH  Keword2: WHEN  Message: A V Z K T G O T | “B I G T O W E R” |
| 18 | Both keywords should accept any length of words | Choice: Encrypt  Keyword: A  Keword2: ABC  Message: SERVER | “UHVXHV” |
| 19 | Both keywords should accept any length of words | Choice: Encrypt  Keyword: A  Keword2: HOWARE  Message: SERVER | “BUPXXX” |
| 21 | Both keywords should accept any length of words | Choice: Encrypt  Keyword: ABC  Keword2: HOW  Message: SERVER | “BVREVR” |
| 22 | Both keywords should accept any length of words | Choice: Encrypt  Keyword: PRIVATEVPN  Keword2: PRIV  Message: SERVER | “YOJNVD” |
| 23 | Both keywords should accept any length of words | Choice: Encrypt  Keyword: ABCDEF  Keword2: HOWARE  Message: S | “B” |
| 24 | Both keywords should accept any length of words | Choice: Decrypt  Keyword: PRIVATEVPN  Keword2: PRIV  Message: YOJNVD | “SERVER” |

**Proposed solution:**

The program language that I am going to use in my controlled assessment is python. I will be using this to create my encryption and decryption program. My program will start off with an input to ask the user which choice to input, the choice of encrypt or decrypt. As validation I will use iteration, until the user has made a correct selection, which will be a string.

After the selection the program will ask the user to input two keywords and once again I will use iteration until the user enters both valid keywords, which should only be letters and can be any length. Finally the user will enter a message for the program to calculate, once again I will use iteration until a valid message is entered, which should only be letters of lowercase or uppercase and should accept symbols and number because it will be ignored. These three information will be stored in three variables (Choice, Keyword, and Message).

The program will perform a selection depending on the users choice of encrypt or decrypt. And after the decision it will be calculated.

In my program, to encrypt and decrypt messages I will be using the python built in functions such as Ord and Chr. Ord is a function that converts letters into their unique ASCII number. The keyword will match the length of the message, then using Ord it will loop through each letter converting the keyword into its ASCII number then It will loop through each letter again converting the message into its ASCII number. If the user has selected encrypt then each letter in the ASCII keyword will be added with each letter of the ASCII message otherwise if the user has selected decrypt then each letter in the ASCII keyword will be subtracted with each letter of the ASCII message, the message will be encrypted or decrypted twice as there is two keywords this will be done to show the new ASCII letter either for encrypt or decrypt. When this is done the Chr function will convert the new ASCII letter of the message back into its alphabetic letters.

If the letter has gone over “Z” then it will be symbol because we expect the letter “a = 1” and “b = 2” and “c = 3” and so on, but actually it’s not, the ASCII value of capital “A” is 65 so if it has gone more than “Z” then we would -64 and then -26 for it to show the new ASCII number for the new letter plus the ASCII value for lowercase letter “a” is 97 so if it has gone over “z” then you will have to -96 and then -26. And it’s the same concept for when decrypt is selected but this time if it has gone beyond “A” then it would +64 and +26 or if it’s gone beyond “a” then it would +96. And +26

To make my understanding clear I have constructed a chart with two keywords for task 3.

Uppercase: Encrypt

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Message | Z | E | B | R | A |
| ASCII number of each letter of message | 90 | 69 | 66 | 82 | 65 |
| Keyword 1 | B | E | N | C | H |
| ASCII number of each letter of keyword 1 | 66 | 69 | 78 | 67 | 72 |
| Message + Keyword 1 of each letter | 156 | 138 | 144 | 149 | 137 |
| Message and Keyword added number of each letter, -64 | 66( 156-64-26 Because message letter has gone beyond Z) | 74 | 80 | 85 | 73 |
| Encrypted Message | B | J | P | U | I |
| ASCII number of each letter in Encrypted Message | 66 | 74 | 80 | 85 | 73 |
| Keyword 2 | S | E | R | V | E |
| ASCII number of each letter of keyword 2 | 83 | 69 | 82 | 86 | 69 |
| Encrypted Message + Keyword 2 of each letter | 149 | 143 | 162 | 171 | 142 |
| Encrypted Message and Keyword added number of each letter, -64 | 85 | 79 | 72( 162-64-26 Because message letter has gone beyond Z) | 81( 171-64-26 Because message letter has gone beyond Z) | 78 |
| New Encrypted Message | U | O | H | Q | N |

After my program has done calculating, the calculated message will be printed out.

**Pseudo code:**

I have created a pseudo code to show the code of my program that will help me throughout my actual program

**FUNCTION** getSelection()

**WHILE TRUE**:

**INPUT** Selection

**IF** Selection == “Encrypt” **OR** Selection == “Decrypt” **THEN**

**RETURN** Selection

**ELSE**

**OUTPUT** “Enter Encrypt or Decrypt only”

**LOOP**

**END FUNCTION**

**FUNCTION** getKeyword()

**WHILE TRUE**:

**INPUT** Keyword

**IF** Keyword.isalpha() == True:

**RETURN** Keyword

**ELSE** :

**OUTPUT** “Keyword must only contain atleast one letter”

**LOOP**

**END FUNCTION**

**FUNCTION** getMessage()

**WHILE TRUE**:

**INPUT** Message

**IF** Letter in Message **THEN**

**RETURN** Message

**ELSE**  **OUTPUT** “The Message contains no letters”

**LOOP**

**END FUNCTION**

**IF** Selection == Encrypt **THEN**

currentASCII = CapitalA

**WHILE** currentLoop < currentKey:

**WHILE** currentASCII = < CapitalA:

currentLetter = CHR(currentASCII)

**OUTPUT** currentASCII + “:” + currentLetter

currentASCII = currentASCII + 1

**LOOP**

currentASCII = currentASCII + 26

currentLoop = currenLoop + 1

**LOOP**

**ELSE**

currentASCII = CapitalZ

**WHILE** currentLoop < currentKeyword:

**WHILE** currentASCII = < CapitalZ:

currentLetter = CHR(currentASCII)

**OUTPUT** currentASCII + “:” + currentLetter

currentASCII = currentASCII + 1

**LOOP**

currentASCII = currentNumber - 26

currentLoop = currenLoop + 1

**LOOP**

**END IF**

Choice = getSelection()

keyW1 = getKeyword()

**OUTPUT** "Keyword accepted!"

keyW2 = getKeyword()

**WHILE** keyW2 == keyW1:

**OUTPUT** "Keyword two cannot match keyword one"

keyW2 = getKeyword()

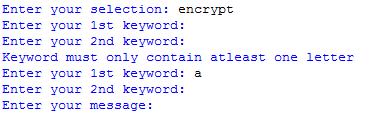
**OUTPUT** "Keyword accepted!"

Msg = getMessage()

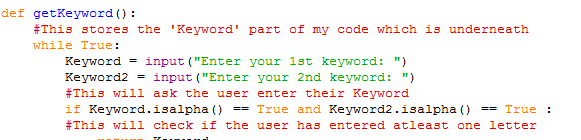
translatedMSG = translatedMessage(Choice,keyW1,Msg)

secondtranslatedMSG = translatedMessage(Choice,keyW2,translatedMSG)

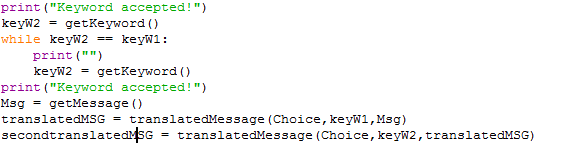
**Development:**

****

This output would warn the user with an error after the 2nd keyword even if the user had made an error in the first keyword.



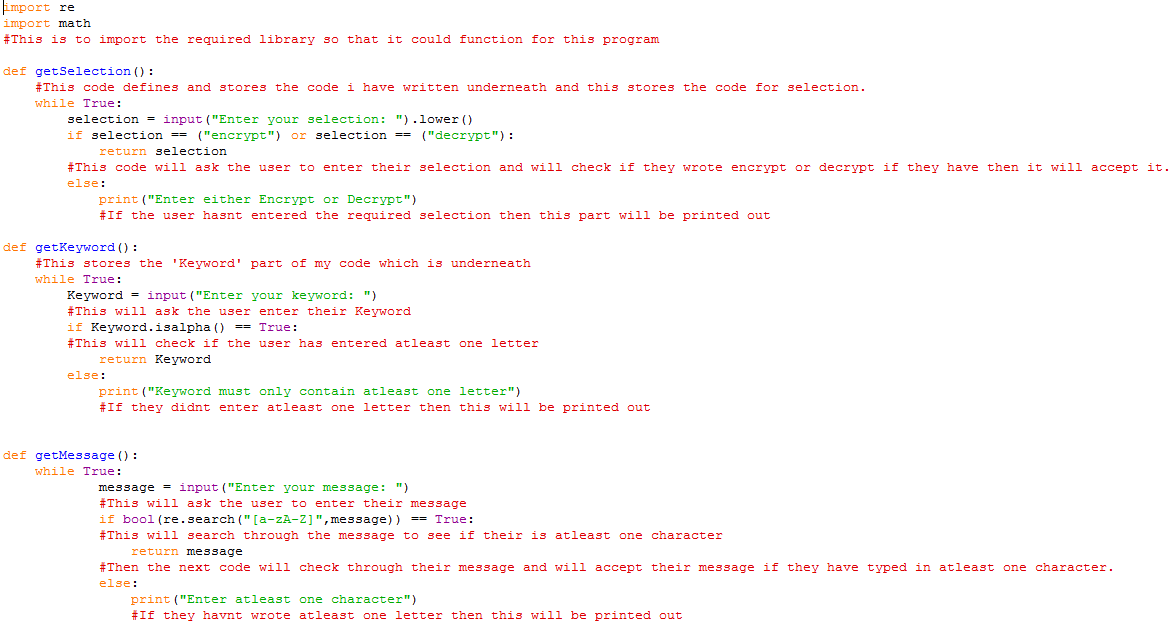
And this was the code that was causing this bug

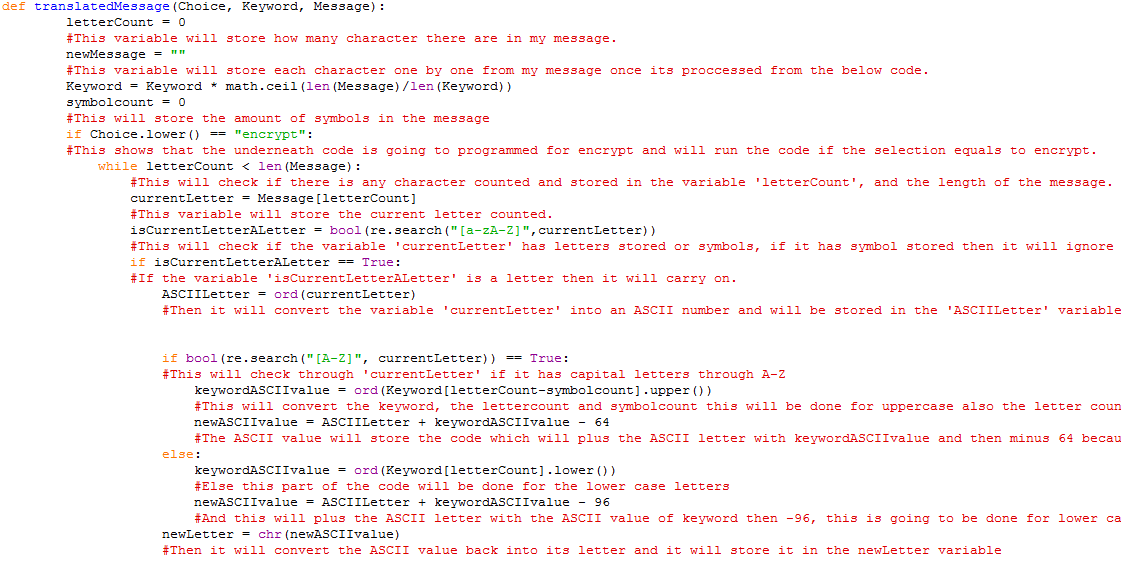


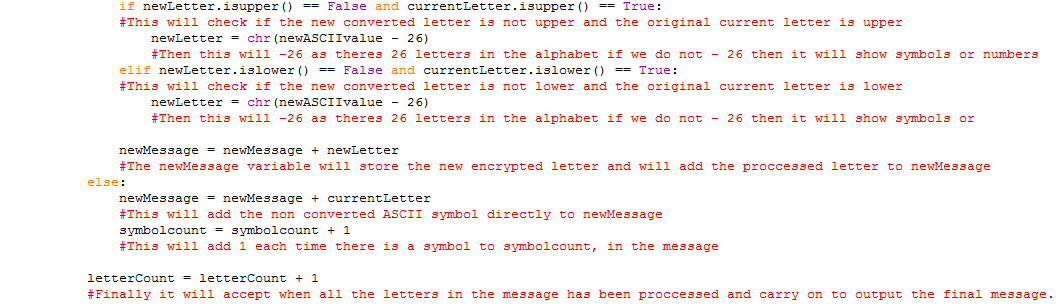
But then I had fixed the error with a simple and efficient code which I have added at the end of the program

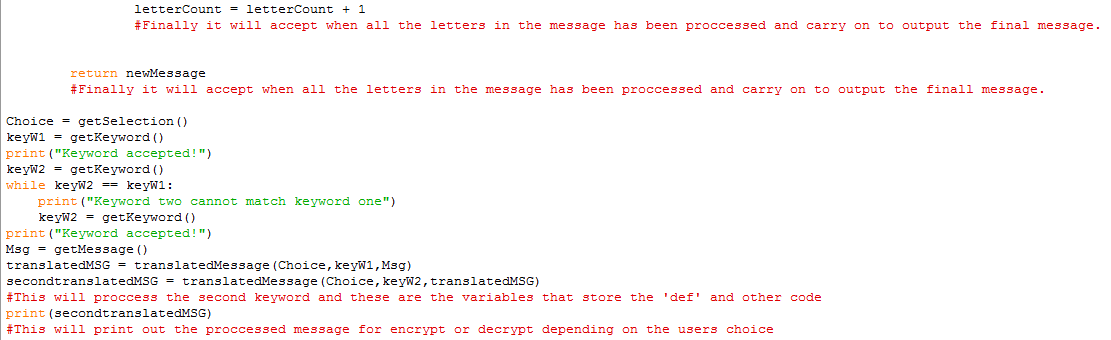
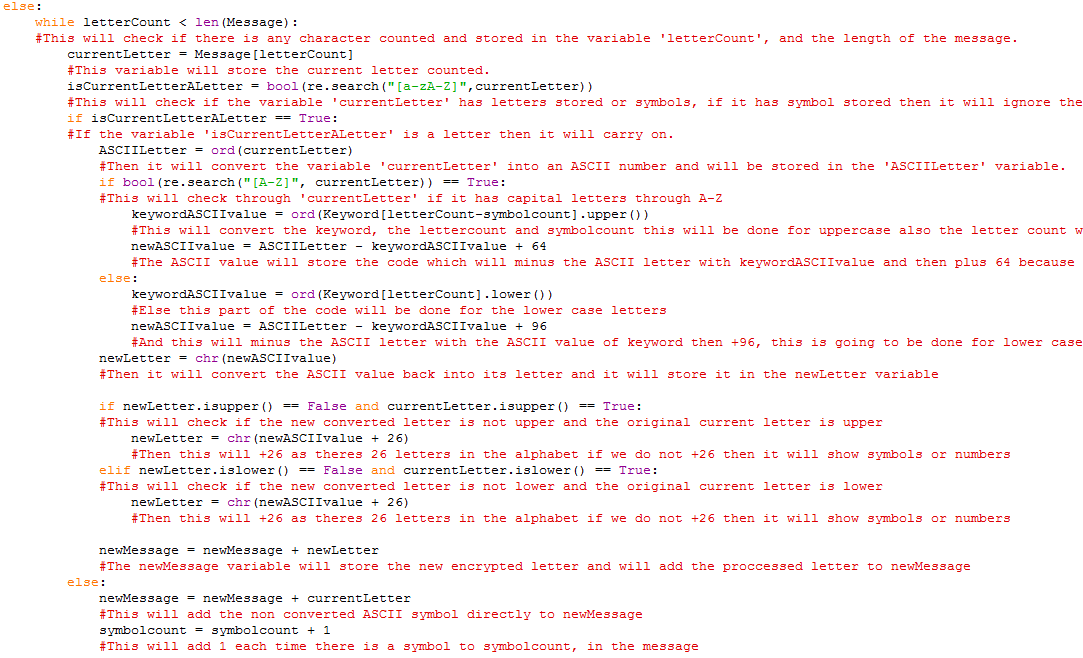
**Final code:**

I have finished my code with comments, which you can observe from the screenshots below



****

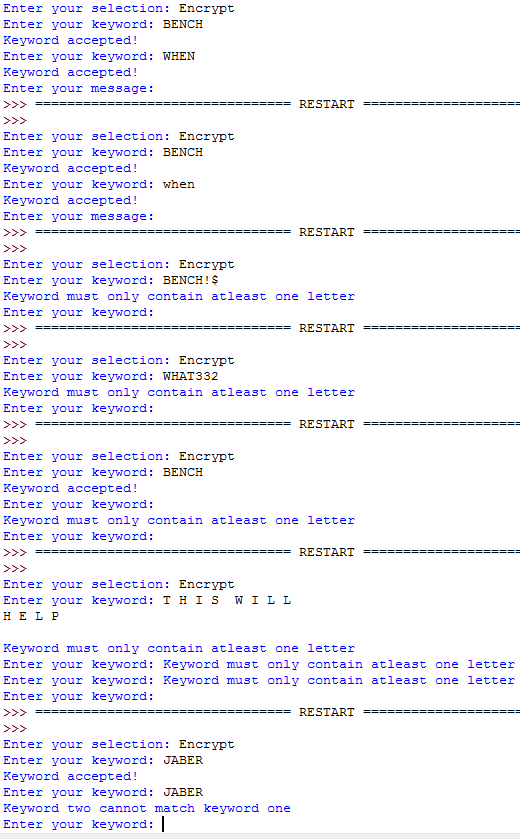




**Testing:**

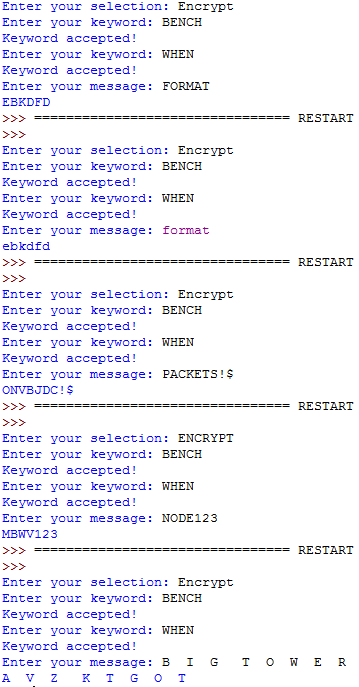
|  |  |  |  |
| --- | --- | --- | --- |
| 1 | Both keywords should only accept letters and any length of letter and reject symbols or numbers and should not be case sensitive | Choice: Encrypt  Keyword1: BENCH  Keword2: WHEN  Message: | Accept and ask to input the message |
| 2 | Both keywords should only accept letters and any length of letter and reject symbols or numbers and should not be case sensitive | Choice: Encrypt  Keyword1: BENCH  Keword2: when  Message: | Accept and ask to input the message |
| 3 | Both keywords should only accept letters and any length of letter and reject symbols or numbers and should not be case sensitive | Choice: Encrypt  Keyword1: BENCH!$  Keword2:  Message: | Reject and ask to input the keyword again |
| 4 | Both keywords should only accept letters and any length of letter and reject symbols or numbers and should not be case sensitive | Choice: Encrypt  Keyword1: WHAT332  Keword2:  Message: | Reject and ask to input the keyword again |
| 5 | Both keywords should only accept letters and any length of letter and reject symbols or numbers and should not be case sensitive | Choice: Encrypt  Keyword1: BENCH Keword2: [Blank]  Message: | Reject and ask to input the keyword again |
| 6 | Both keywords should not have spaces and will not work | Choice: Encrypt  Keyword: T H I S W I L L  H E L P  Keword2:  Message: | Reject and ask to input the keyword again |
| 7 | Both keywords can not be the same and it will not work | Choice: Encrypt  Keyword: JABER  Keword2: JABER  Message: | Reject and ask to input the keyword again |

Using the test plan above I have tested Both of the keywords, as you can see from the screenshot below



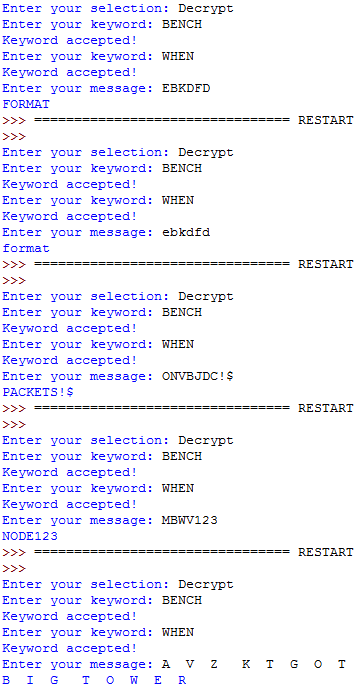
|  |  |  |  |
| --- | --- | --- | --- |
| 8 | The message should encrypt from both keywords correctly | Choice: Encrypt  Keyword: BENCH  Keword2: WHEN  Message: FORMAT | “EBKDFD” |
| 9 | The message should encrypt from both keywords correctly | Choice: Encrypt  Keyword: BENCH  Keword2: WHEN  Message: format | “ebkdfd” |
| 10 | The message should encrypt from both keywords correctly | Choice: Encrypt  Keyword: BENCH  Keword2: WHEN  Message: PACKETS!$ | “ONVBJDC!$” |
| 11 | The message should encrypt from both keywords correctly | Choice: Encrypt  Keyword1: BENCH  Keword2: WHEN  Message: NODE123 | “MBWV123” |
| 12 | The message should encrypt from both keywords correctly | Choice: Encrypt  Keyword1: BENCH  Keword2: WHEN  Message: B I G T O W E R | “A V Z K T G O T” |

Using the test plan above I have tested Encrypt, as you can see from the screenshot below



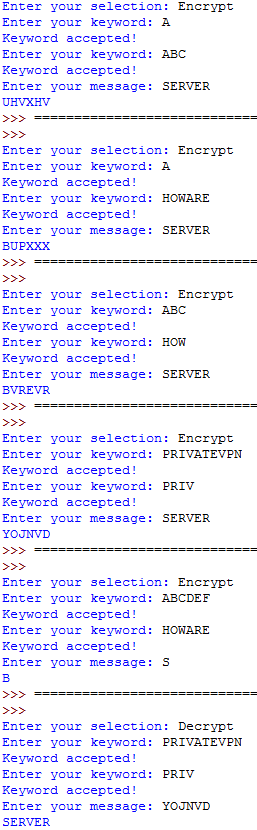
|  |  |  |  |
| --- | --- | --- | --- |
| 13 | The message should decrypt from both keywords correctly | Choice: Decrypt  Keyword1: BENCH  Keword2: WHEN  Message: EBKDFD | “FORMAT” |
| 14 | The message should decrypt from both keywords correctly | Choice: Decrypt  Keyword1: BENCH  Keword2: WHEN  Message: ebkdfd | “format” |
| 15 | The message should decrypt from both keywords correctly | Choice: Decrypt  Keyword1: BENCH Keword2: WHEN  Message: ONVBJDC!$ | “PACKETS!$” |
| 16 | The message should decrypt from both keywords correctly | Choice: Decrypt  Keyword1: BENCH  Keword2: WHEN  Message: MBWV123 | “NODE123” |
| 17 | The message should decrypt from both keywords correctly | Choice: Decrypt  Keyword1: BENCH  Keword2: WHEN  Message: A V Z K T G O T | “B I G T O W E R” |

Using the test plan above I have tested Decrypt, as you can see from the screenshot below



|  |  |  |  |
| --- | --- | --- | --- |
| 18 | Both keywords should accept any length of words | Choice: Encrypt  Keyword: A  Keword2: ABC  Message: SERVER | “UHVXHV” |
| 19 | Both keywords should accept any length of words | Choice: Encrypt  Keyword: A  Keword2: HOWARE  Message: SERVER | “BUPXXX” |
| 21 | Both keywords should accept any length of words | Choice: Encrypt  Keyword: ABC  Keword2: HOW  Message: SERVER | “BVREVR” |
| 22 | Both keywords should accept any length of words | Choice: Encrypt  Keyword: PRIVATEVPN  Keword2: PRIV  Message: SERVER | “YOJNVD” |
| 23 | Both keywords should accept any length of words | Choice: Encrypt  Keyword: ABCDEF  Keword2: HOWARE  Message: S | “B” |
| 24 | Both keywords should accept any length of words | Choice: Decrypt  Keyword: PRIVATEVPN  Keword2: PRIV  Message: YOJNVD | “SERVER” |

Using the test plan above I have tested the different lengths for both keywords, as you can see from the screenshot below



**Evaluation:**

My program has been developed according to plan from my problem definition. My program has been thoroughly checked and made efficient as possible, I have regularly screen shot my work where errors have occurred to show my mistakes and screenshots of the fix, also I have tested and checked my output code which you can observe to work from all the above screen shots. These screen shots have been captured to show all my evidence throughout my program, my program is errorless and has been finished according to plan.

To finish my evaluation further I have must refer back to my success criteria and to prove that everything is correct.

1. Both keywords should only accept at least one letter and no symbols or numbers.

-This has been proved by tests 1 – 7 which passed

1. Both keywords should encrypt and decrypt correctly.

-This has been proven by tests 8 – 19 which passed

1. Both keywords and message will not be case sensitive.

-This can be proven by tests 2, 9 and 14 which passed

1. Both keywords will accept any length of word

-This can be proven by test 20 – 24 which passed