

COMP 1043

Problem Solving and Programming

**Programming Assignment 2**

UniSA STEM

The University of South Australia

March, 2022

Contents

Introduction

Assignment Overview

Graduate Qualities

Part II Specification

* Practical Requirements (Part II)
* Stages (Part II)

Submission Details

Extensions and Late Submissions

Academic Misconduct

Marking Criteria

Sample Output – Part II

Introduction

This document describes the second assignment for Problem Solving and Programming.

The assignment is intended to provide you with the opportunity to put into practice what you have learnt in the course by applying your knowledge and skills to the implementation of **simple encryption and decryption on strings entered by the user.**

This assignment is an **individual task** that will require an **individual submission**. You will be required to submit your work via learnonline before **Monday Week 7, 12pm (Adelaide time).**

This document is a kind of specification of the required end product that will be generated by implementing the assignment. Like many specifications, it is written in English and hence will contain some imperfectly specified parts. Please make sure you seek clarification if you are not clear on any aspect of this assignment.

Assignment Overview

**Part II: Encryption**

You are required to write a Python program that performs simple encryption and decryption on strings entered by the user. To do so, you are to use one of the simplest and most widely known encryption techniques known as the Caesar Cipher.

The Caesar Cipher is named after Julius Caesar who used it for private correspondence. Further information on the Caesar Cipher can be found here: <https://en.wikipedia.org/wiki/Caesar_cipher>

***Please ensure that you read sections titled ‘Part II Specification’ below for further details.***

Graduate Qualities

By undertaking this assessment, you will progress in developing the qualities of a University of South Australia graduate.

The Graduate qualities being assessed by this assignment are:

* The ability to demonstrate and apply a body of knowledge (GQ1) gained from the lectures, workshops, practicals and readings. This is demonstrated in your ability to apply problem solving and programming theory to a practical situation.
* The development of skills required for lifelong learning (GQ2), by searching for information and learning to use and understand the resources provided (Python standard library, lectures, workshops, practical exercises, etc); in order to complete a programming exercise.
* The ability to effectively problem solve (GQ3) using Python to complete the programming problem. Effective problem solving is demonstrated by the ability to understand what is required, utilise the relevant information from lectures, workshops and practical work, write Python code, and evaluate the effectiveness of the code by testing it.
* The ability to work autonomously (GQ4) in order to complete the task.
* The use of communication skills (GQ6) by producing code that has been properly formatted; and writing adequate, concise and clear comments.
* The application of international standards (GQ7) by making sure your solution conforms to the standards presented in the Python Style Guide slides (available on the course website).

[](https://www.google.com.au/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=0ahUKEwi79s2C5ILVAhUCvbwKHeiNDOgQjRwIBw&url=https://it.pinterest.com/explore/caesar-cipher/&psig=AFQjCNE5Gv0wXTuMkpZfsFItAYnwQIA3hw&ust=1499914743871036)Part II Specification – Encryption

A simple way to encrypt data is attributed to Julius Caesar, the Roman Emperor. (If you are interested, you may like to read the following… <http://en.wikipedia.org/wiki/Caesar_cipher>).

This method takes each character in a message and replaces it with one which is a certain distance (offset) along the alphabet from it (moving right).

Write a **menu driven program** called *yourEmailId\_encryptor.py* that will allow the user to enter commands and process these commands until the quit command is entered.

Your program will accept entry of a *string* and an *offset value* from the keyboard and encrypt or decrypt it as requested.

*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 🡪

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. In this case the word DIG encrypts to GLJ. In order to decrypt the word/string, simply offset by the same amount moving in the opposite direction (i.e. moving left).

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

The following commands should be allowed:

1. **Encrypt string:**

Prompt for and read (from the keyboard) a string to be encrypted. The program should then ask for the offset value (limited to a range of 1 to 94 inclusive). Display the encrypted string to the screen.

1. **Decrypt string:**

Decrypt an encrypted message by prompting for and reading (from the keyboard) a string to be decrypted. The program should then ask for the offset value (limited to the range of 1 to 94 inclusive). Display the decrypted string to the screen.

1. **Brute force decryption:**

If the offset is not known, we can implement a brute force decryption algorithm that tries all of the 94 possible Caesar offsets in order to decrypt the encrypted text. Prompt for and read the string to be decrypted and display all 94 possible decrypted strings to the screen. (If you are interested, you may like to read the following… <http://en.wikipedia.org/wiki/Brute-force_attack>).

1. **Quit:**

Quits the program displaying a goodbye message to the screen.

***Note:***

*Your program must 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 beyond 126 it should wrap around to the beginning of the set.*

*For example:*

*If the offset is 4 and character is ‘}’ (ASCII 125) then it will encrypt to ASCII 129. This is beyond 126 so wrap back to the beginning by subtracting the total number of characters (95). This gives character 34. Similarly, when decrypting, if the subtracted offset results in a number less than 32 then add 95 to the result.*

Practical Requirements (Part II)

It is recommended that you develop this part of the assignment in the suggested stages.

**It is expected that your solution will include the use of:**

* Your solution in a file called *yourEmailId*\_encryptor.py.
* Appropriate and well constructed while and/or for loops (as necessary).
* Appropriate if, if-else, if-elif-else statements (as necessary).
* The use of the ord() and chr() functions (as necessary).
* The following three functions (refer to stage 6 for description):
  + display\_details()
  + get\_menu\_choice()
  + get\_offset()
* Output that **strictly** adheres to the assignment specifications. If you are not sure about these details, you should check with the ‘Sample Output – Part II’ provided at the end of this document.
* Good programming practice:
  + Consistent commenting, layout and indentation. You are to provide comments to describe: your details, program description, all variable definitions, and significant sections of code.
  + Meaningful variable names.

Your solutions **MAY** make use of the following:

* Built-in functions int(), input(), print(), range(), ord(),and chr().
* Concatenation (+) operator to create/build new strings.
* Access the individual elements in a string with an index (one element only). i.e. string\_name[index].
* Your own (user-defined) functions (in addition to the necessary four functions listed above).

Your solutions **MUST NOT** use:

* Built-in functions (other than the int(), input(), print(), range(), pow(), len() and str() functions).
* The built-in functions int() and/or bin() in order to convert between number systems, i.e., to perform the conversion from binary to decimal and vice versa.
* Slice expressions to select a range of elements from a string or list. i.e. name[start:end].
* String or list methods (other than those used for input validation, i.e. isdigit() and the append() method. i.e. *list\_name*.append(item)).
* break, or continue statements in your solution. **Do not** use the quit() or exit() functions or the break or return statements (or any other techniques) as a way to break out of loops. Doing so will result in a significant mark deduction.

**PLEASE NOTE: You are reminded that you should ensure that all input and output conform to the specifications listed here; if you are not sure about these details you should check with the sample output provided at the end of this document or post a message to the discussion forum in order to seek clarification.**

Please ensure that you use Python 3.9.2 or a later version (i.e. the latest version) in order to complete your assignments. Your programs **MUST** run using Python 3.9.2 (or latest version).

Stages (Part II)

It is recommended that you develop this part of the assignment in the suggested stages. Many problems in later stages are due to errors in early stages. **Make sure you have finished and thoroughly tested each stage before continuing.**

The following stages of development are recommended:

**Stage 1**

* Implement the interactive mode (prompt for and read menu commands).
* Set up a loop to obtain and process commands.
* Test to ensure that this is working correctly before moving onto stage 2.
* You need not perform any encryption/decryption of text at this point, you may simply display an appropriate message to the screen.

*Sample output:*

\*\*\* Menu \*\*\*

1. Encrypt string

2. Decrypt string

3. Brute force decryption

4. Quit

*Note:*

*When developing software, you should always have fixed points in your development where you know your software is bug free and runs correctly.*

What would you like to do [1,2,3,4]? **1**

In command 1 - encrypt string

\*\*\* Menu \*\*\*

1. Encrypt string

2. Decrypt string

3. Brute force decryption

4. Quit

What would you like to do [1,2,3,4]? **2**

In command 2 - decrypt string

\*\*\* Menu \*\*\*

1. Encrypt string

2. Decrypt string

3. Brute force decryption

4. Quit

What would you like to do [1,2,3,4]? **3**

In command 3 - brute force

\*\*\* Menu \*\*\*

1. Encrypt string

2. Decrypt string

3. Brute force decryption

4. Quit

What would you like to do [1,2,3,4]? **4**

Goodbye.

*Make sure the program is running correctly before continuing. Once you have that working, back up your program.*

**Stage 2**

* Add code to implement command 1.
* Encrypt string.
* Prompt for and read a string to be encrypted.
* Prompt for and read the offset value (limited to a range of 1 to 94 inclusive).
* Display the encrypted string to the screen.

*Note:*

*Your program must 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 beyond 126 it should wrap around to the beginning of the set.*

*For example:*

*If the offset is 4 and character is ‘}’ (ASCII 125) then it will encrypt to ASCII 129. This is beyond 126 so wrap back to the beginning by subtracting the total number of characters (95). This gives character 34. Similarly, when decrypting, if the subtracted offset results in a number less than 32 then add 95 to the result.*

*Sample output:*

\*\*\* Menu \*\*\*

1. Encrypt string

2. Decrypt string

3. Brute force decryption

4. Quit

What would you like to do [1,2,3,4]? 1

Please enter string to encrypt: secret squirrel

Please enter offset value (1 to 94): 6

Encrypted string:

ykixkz&yw{oxxkr

\*\*\* Menu \*\*\*

1. Encrypt string

2. Decrypt string

3. Brute force decryption

4. Quit

What would you like to do [1,2,3,4]? 4

Goodbye.

*Make sure you read the section located at the end of this document:*

*“Useful Built-in Python Functions”*

**Stage 3**

* Add code to implement command 2.
* Decrypt string.
* Prompt for and read a string to be decrypted.
* Prompt for and read the offset value (limited to a range of 1 to 94 inclusive).
* Display the decrypted string to the screen.

*Note:*

*Your program must 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 beyond 126 it should wrap around to the beginning of the set.*

*For example:*

*If the offset is 4 and character is ‘}’ (ASCII 125) then it will encrypt to ASCII 129. This is beyond 126 so wrap back to the beginning by subtracting the total number of characters (95). This gives character 34. Similarly, when decrypting, if the subtracted offset results in a number less than 32 then add 95 to the result.*

*Sample output:*

\*\*\* Menu \*\*\*

1. Encrypt string

2. Decrypt string

3. Brute force decryption

4. Quit

What would you like to do [1,2,3,4]? 2

Please enter string to decrypt: ykixkz&yw{oxxkr

Please enter offset value (1 to 94): 6

Decrypted string:

secret squirrel

\*\*\* Menu \*\*\*

1. Encrypt string

2. Decrypt string

3. Brute force decryption

4. Quit

What would you like to do [1,2,3,4]? 4

Goodbye.

**Stage 4**

* Add code to implement command 3. Brute force decryption.
* Prompt for and read a string to be decrypted.
* Display all 94 possible decrypted strings to the screen.

*Sample output:*

*Note:*

*Your program must 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 beyond 126 it should wrap around to the beginning of the set.*

*For example:*

*If the offset is 4 and character is ‘}’ (ASCII 125) then it will encrypt to ASCII 129. This is beyond 126 so wrap back to the beginning by subtracting the total number of characters (95). This gives character 34. Similarly, when decrypting, if the subtracted offset results in a number less than 32 then add 95 to the result.*

\*\*\* Menu \*\*\*

1. Encrypt string

2. Decrypt string

3. Brute force decryption

4. Quit

What would you like to do [1,2,3,4]? 3

Please enter string to decrypt: ykixkz&yw{oxxkr

Offset: 1 = Decrypted string: xjhwjy%xvznwwjq

Offset: 2 = Decrypted string: wigvix$wuymvvip

Offset: 3 = Decrypted string: vhfuhw#vtxluuho

Offset: 4 = Decrypted string: ugetgv"uswkttgn

Offset: 5 = Decrypted string: tfdsfu!trvjssfm

Offset: 6 = Decrypted string: secret squirrel

Offset: 7 = Decrypted string: rdbqds~rpthqqdk

Offset: 8 = Decrypted string: qcapcr}qosgppcj

Offset: 9 = Decrypted string: pb`obq|pnrfoobi

Offset: 10 = Decrypted string: oa\_nap{omqennah

:

: *you get the idea… :)*

:

Offset: 87 = Decrypted string: "sq!s#." $w!!sz

Offset: 88 = Decrypted string: !rp r"-!~#v ry

Offset: 89 = Decrypted string: qo~q!, }"u~~qx

Offset: 90 = Decrypted string: ~pn}p +~|!t}}pw

Offset: 91 = Decrypted string: }om|o~\*}{ s||ov

Offset: 92 = Decrypted string: |nl{n})|z~r{{nu

Offset: 93 = Decrypted string: {mkzm|({y}qzzmt

Offset: 94 = Decrypted string: zljyl{'zx|pyyls

\*\*\* Menu \*\*\*

1. Encrypt string

2. Decrypt string

3. Brute force decryption

4. Quit

What would you like to do [1,2,3,4]? 4

Goodbye.

**Stage 5**

Add code to validate the following user input only:

* What would you like to do [1,2,3,4]?

*Sample output:*

What would you like to do [1,2,3,4]? 7

Invalid choice, please enter either 1, 2, 3 or 4.

What would you like to do [1,2,3,4]?

*Remember:*

*defining a function does not execute the function – you will need to call the function from the appropriate place in the program.*

* Please enter offset value (1 to 94):

*Sample output:*

Please enter offset value (1 to 94): 101

Please enter offset value (1 to 94): -1

Please enter offset value (1 to 94): 3

**Stage 6**

Modify your code to include and make use of the following functions:

* Write a function called display\_details that will display your details to the screen.

The function takes no parameters and does not return any values. Your function should produce the following output (with your details).

**Output:**

File : wayby001\_encryptor.py

Author : Batman

Stud ID : 0123456X

Email ID : wayby001

Description: Programming Assignment 2 - Caesar Cipher

This is my own work as defined by the University's Academic Misconduct Policy.

* Write a function called get\_menu\_choice() that displays the menu to the screen, prompts for, reads and validates the menu command entered by the user. The function takes no parameters and returns the menu command entered by the user.
* Write a function called get\_offset() that prompts for, reads and validates the offset entered by the user. The function takes no parameters and returns the offset entered by the user.

**Stage 7**

Finally, check the sample output (see section titled ‘Sample Output – Part B’ towards the end of this document) and if necessary, modify your code so that:

* The output produced by your program **EXACTLY** adheres to the sample output provided.

Your program behaves as described in these specs and the sample output provided.

Submission Details

All students must follow the submission instructions below:

**Ensure that your files are named correctly (as per instructions outlined in this document).**

Ensure that the following files are included in your submission:

* yourEmailId\_encryptor.py

For example (if your name is James Bond, your submission files would be as follows):

* bonjy007\_ encryptor.py

All files that you submit must include the following comments.

#

# File: fileName.py

# Author: your name

# Email Id: your email id

# Description: Assignment 2 – place assignment description here…

# This is my own work as defined by the University's

# Academic Misconduct policy.

#

Assignments that do not contain these details may not be marked.

You must submit your program **before the online due date**. Work that has not been correctly submitted to learnonline will not be marked.

**It is expected that students will make copies of all assignments and be able to provide these if required.**

Extensions and Late Submissions

There will be **no** extensions/late submissions for this course without one of the following exceptions:

1. A medical certificate is provided that has the timing and duration of the illness and an opinion on how much the student’s ability to perform has been compromised by the illness. **Please note** if this information is not provided the medical certificate WILL NOT BE ACCEPTED. Late assessment items will not be accepted unless a medical certificate is presented to the Course Coordinator. The certificate must be produced as soon as possible and must cover the dates during which the assessment was to be attempted. In the case where you have a valid medical certificate, the due date will be extended by the number of days stated on the certificate up to five working days.
2. A Learning and Teaching Unit councillor contacts the Course Coordinator on your behalf requesting an extension. Normally you would use this if you have events outside your control adversely affecting your course work.
3. Unexpected work commitments. In this case, you will need to attach a letter from your work supervisor with your application stating the impact on your ability to complete your assessment.
4. Military obligations with proof.

Applications for extensions must be lodged via learnonline before the due date of the assignment.

Note: Equipment failure, loss of data, ‘Heavy work commitments’ or late starting of the course are not sufficient grounds for an extension.

Academic Misconduct

### Academic Misconduct

Students are reminded that they should be aware of the academic misconduct guidelines available from the University of South Australia website.

Deliberate academic misconduct such as plagiarism is subject to penalties. Information about Academic integrity can be found in Section 9 of the *Assessment policies and procedures manual* at:

<http://www.unisa.edu.au/policies/manual/>

Marking Criteria

*See separate feedback form.*

*Other possible deductions:*

* *Programming style:* Things to watch for are poor or no commenting, poor variable names, etc.
* *Submitted incorrectly:* -10 marks if assignment is submitted incorrectly (i.e. not adhering to the specs).

Sample Output – Part II

**Sample output 1:**

File : wayby001\_encryptor.py

Author : Batman

Stud ID : 0123456X

Email ID : wayby001

Description: Programming Assignment 2 - Caesar Cipher

This is my own work as defined by the University's Academic Misconduct Policy.

\*\*\* Menu \*\*\*

1. Encrypt string

2. Decrypt string

3. Brute force decryption

4. Quit

What would you like to do [1,2,3,4]? 1

Please enter string to encrypt: Elvis has left the building!

Please enter offset value (1 to 94): 3

Encrypted string:

Hoylv#kdv#ohiw#wkh#exloglqj$

\*\*\* Menu \*\*\*

1. Encrypt string

2. Decrypt string

3. Brute force decryption

4. Quit

What would you like to do [1,2,3,4]? 2

Please enter string to decrypt: Hoylv#kdv#ohiw#wkh#exloglqj$

Please enter offset value (1 to 94): 3

Decrypted string:

Elvis has left the building!

\*\*\* Menu \*\*\*

1. Encrypt string

2. Decrypt string

3. Brute force decryption

4. Quit

What would you like to do [1,2,3,4]? 3

Please enter string to decrypt: Hoylv#kdv#ohiw#wkh#exloglqj$

Offset: 1 = Decrypted string: Gnxku"jcu"nghv"vjg"dwknfkpi#

Offset: 2 = Decrypted string: Fmwjt!ibt!mfgu!uif!cvjmejoh"

Offset: 3 = Decrypted string: Elvis has left the building!

Offset: 4 = Decrypted string: Dkuhr~g`r~kdes~sgd~athkchmf

Offset: 5 = Decrypted string: Cjtgq}f\_q}jcdr}rfc}`sgjbgle~

Offset: 6 = Decrypted string: Bisfp|e^p|ibcq|qeb|\_rfiafkd}

Offset: 7 = Decrypted string: Ahreo{d]o{habp{pda{^qeh`ejc|

Offset: 8 = Decrypted string: @gqdnzc\nzg`aozoc`z]pdg\_dib{

Offset: 9 = Decrypted string: ?fpcmyb[myf\_`nynb\_y\ocf^chaz

Offset: 10 = Decrypted string: >eoblxaZlxe^\_mxma^x[nbe]bg`y

Offset: 11 = Decrypted string: =dnakw`Ykwd]^lwl`]wZmad\af\_x

Offset: 12 = Decrypted string: <cm`jv\_Xjvc\]kvk\_\vYl`c[`e^w

Offset: 13 = Decrypted string: ;bl\_iu^Wiub[\juj^[uXk\_bZ\_d]v

Offset: 14 = Decrypted string: :ak^ht]VhtaZ[iti]ZtWj^aY^c\u

Offset: 15 = Decrypted string: 9`j]gs\Ugs`YZhsh\YsVi]`X]b[t

Offset: 16 = Decrypted string: 8\_i\fr[Tfr\_XYgrg[XrUh\\_W\aZs

Offset: 17 = Decrypted string: 7^h[eqZSeq^WXfqfZWqTg[^V[`Yr

Offset: 18 = Decrypted string: 6]gZdpYRdp]VWepeYVpSfZ]UZ\_Xq

Offset: 19 = Decrypted string: 5\fYcoXQco\UVdodXUoReY\TY^Wp

Offset: 20 = Decrypted string: 4[eXbnWPbn[TUcncWTnQdX[SX]Vo

Offset: 21 = Decrypted string: 3ZdWamVOamZSTbmbVSmPcWZRW\Un

Offset: 22 = Decrypted string: 2YcV`lUN`lYRSalaURlObVYQV[Tm

Offset: 23 = Decrypted string: 1XbU\_kTM\_kXQR`k`TQkNaUXPUZSl

Offset: 24 = Decrypted string: 0WaT^jSL^jWPQ\_j\_SPjM`TWOTYRk

Offset: 25 = Decrypted string: /V`S]iRK]iVOP^i^ROiL\_SVNSXQj

Offset: 26 = Decrypted string: .U\_R\hQJ\hUNO]h]QNhK^RUMRWPi

Offset: 27 = Decrypted string: -T^Q[gPI[gTMN\g\PMgJ]QTLQVOh

Offset: 28 = Decrypted string: ,S]PZfOHZfSLM[f[OLfI\PSKPUNg

Offset: 29 = Decrypted string: +R\OYeNGYeRKLZeZNKeH[ORJOTMf

Offset: 30 = Decrypted string: \*Q[NXdMFXdQJKYdYMJdGZNQINSLe

Offset: 31 = Decrypted string: )PZMWcLEWcPIJXcXLIcFYMPHMRKd

Offset: 32 = Decrypted string: (OYLVbKDVbOHIWbWKHbEXLOGLQJc

Offset: 33 = Decrypted string: 'NXKUaJCUaNGHVaVJGaDWKNFKPIb

Offset: 34 = Decrypted string: &MWJT`IBT`MFGU`UIF`CVJMEJOHa

Offset: 35 = Decrypted string: %LVIS\_HAS\_LEFT\_THE\_BUILDING`

Offset: 36 = Decrypted string: $KUHR^G@R^KDES^SGD^ATHKCHMF\_

Offset: 37 = Decrypted string: #JTGQ]F?Q]JCDR]RFC]@SGJBGLE^

Offset: 38 = Decrypted string: "ISFP\E>P\IBCQ\QEB\?RFIAFKD]

Offset: 39 = Decrypted string: !HREO[D=O[HABP[PDA[>QEH@EJC\

Offset: 40 = Decrypted string: GQDNZC<NZG@AOZOC@Z=PDG?DIB[

Offset: 41 = Decrypted string: ~FPCMYB;MYF?@NYNB?Y<OCF>CHAZ

Offset: 42 = Decrypted string: }EOBLXA:LXE>?MXMA>X;NBE=BG@Y

Offset: 43 = Decrypted string: |DNAKW@9KWD=>LWL@=W:MAD<AF?X

Offset: 44 = Decrypted string: {CM@JV?8JVC<=KVK?<V9L@C;@E>W

Offset: 45 = Decrypted string: zBL?IU>7IUB;<JUJ>;U8K?B:?D=V

Offset: 46 = Decrypted string: yAK>HT=6HTA:;ITI=:T7J>A9>C<U

Offset: 47 = Decrypted string: x@J=GS<5GS@9:HSH<9S6I=@8=B;T

Offset: 48 = Decrypted string: w?I<FR;4FR?89GRG;8R5H<?7<A:S

Offset: 49 = Decrypted string: v>H;EQ:3EQ>78FQF:7Q4G;>6;@9R

Offset: 50 = Decrypted string: u=G:DP92DP=67EPE96P3F:=5:?8Q

Offset: 51 = Decrypted string: t<F9CO81CO<56DOD85O2E9<49>7P

Offset: 52 = Decrypted string: s;E8BN70BN;45CNC74N1D8;38=6O

Offset: 53 = Decrypted string: r:D7AM6/AM:34BMB63M0C7:27<5N

Offset: 54 = Decrypted string: q9C6@L5.@L923ALA52L/B6916;4M

Offset: 55 = Decrypted string: p8B5?K4-?K812@K@41K.A5805:3L

Offset: 56 = Decrypted string: o7A4>J3,>J701?J?30J-@47/492K

Offset: 57 = Decrypted string: n6@3=I2+=I6/0>I>2/I,?36.381J

Offset: 58 = Decrypted string: m5?2<H1\*<H5./=H=1.H+>25-270I

Offset: 59 = Decrypted string: l4>1;G0);G4-.<G<0-G\*=14,16/H

Offset: 60 = Decrypted string: k3=0:F/(:F3,-;F;/,F)<03+05.G

Offset: 61 = Decrypted string: j2</9E.'9E2+,:E:.+E(;/2\*/4-F

Offset: 62 = Decrypted string: i1;.8D-&8D1\*+9D9-\*D':.1).3,E

Offset: 63 = Decrypted string: h0:-7C,%7C0)\*8C8,)C&9-0(-2+D

Offset: 64 = Decrypted string: g/9,6B+$6B/()7B7+(B%8,/',1\*C

Offset: 65 = Decrypted string: f.8+5A\*#5A.'(6A6\*'A$7+.&+0)B

Offset: 66 = Decrypted string: e-7\*4@)"4@-&'5@5)&@#6\*-%\*/(A

Offset: 67 = Decrypted string: d,6)3?(!3?,%&4?4(%?"5),$).'@

Offset: 68 = Decrypted string: c+5(2>' 2>+$%3>3'$>!4(+#(-&?

Offset: 69 = Decrypted string: b\*4'1=&~1=\*#$2=2&#= 3'\*"',%>

Offset: 70 = Decrypted string: a)3&0<%}0<)"#1<1%"<~2&)!&+$=

Offset: 71 = Decrypted string: `(2%/;$|/;(!"0;0$!;}1%( %\*#<

Offset: 72 = Decrypted string: \_'1$.:#{.:' !/:/# :|0$'~$)";

Offset: 73 = Decrypted string: ^&0#-9"z-9&~ .9."~9{/#&}#(!:

Offset: 74 = Decrypted string: ]%/",8!y,8%}~-8-!}8z."%|"' 9

Offset: 75 = Decrypted string: \$.!+7 x+7$|},7, |7y-!${!&~8

Offset: 76 = Decrypted string: [#- \*6~w\*6#{|+6+~{6x, #z %}7

Offset: 77 = Decrypted string: Z",~)5}v)5"z{\*5\*}z5w+~"y~$|6

Offset: 78 = Decrypted string: Y!+}(4|u(4!yz)4)|y4v\*}!x}#{5

Offset: 79 = Decrypted string: X \*|'3{t'3 xy(3({x3u)| w|"z4

Offset: 80 = Decrypted string: W~){&2zs&2~wx'2'zw2t({~v{!y3

Offset: 81 = Decrypted string: V}(z%1yr%1}vw&1&yv1s'z}uz x2

Offset: 82 = Decrypted string: U|'y$0xq$0|uv%0%xu0r&y|ty~w1

Offset: 83 = Decrypted string: T{&x#/wp#/{tu$/$wt/q%x{sx}v0

Offset: 84 = Decrypted string: Sz%w".vo".zst#.#vs.p$wzrw|u/

Offset: 85 = Decrypted string: Ry$v!-un!-yrs"-"ur-o#vyqv{t.

Offset: 86 = Decrypted string: Qx#u ,tm ,xqr!,!tq,n"uxpuzs-

Offset: 87 = Decrypted string: Pw"t~+sl~+wpq + sp+m!twotyr,

Offset: 88 = Decrypted string: Ov!s}\*rk}\*vop~\*~ro\*l svnsxq+

Offset: 89 = Decrypted string: Nu r|)qj|)uno})}qn)k~rumrwp\*

Offset: 90 = Decrypted string: Mt~q{(pi{(tmn|(|pm(j}qtlqvo)

Offset: 91 = Decrypted string: Ls}pz'ohz'slm{'{ol'i|pskpun(

Offset: 92 = Decrypted string: Kr|oy&ngy&rklz&znk&h{orjotm'

Offset: 93 = Decrypted string: Jq{nx%mfx%qjky%ymj%gznqinsl&

Offset: 94 = Decrypted string: Ipzmw$lew$pijx$xli$fymphmrk%

\*\*\* Menu \*\*\*

1. Encrypt string

2. Decrypt string

3. Brute force decryption

4. Quit

What would you like to do [1,2,3,4]? 4

Goodbye.

**Sample output 2:**

File : wayby001\_encryptor.py

Author : Batman

Stud ID : 0123456X

Email ID : wayby001

Description: Programming Assignment 2 - Caesar Cipher

This is my own work as defined by the University's Academic Misconduct Policy.

\*\*\* Menu \*\*\*

1. Encrypt string

2. Decrypt string

3. Brute force decryption

4. Quit

What would you like to do [1,2,3,4]? 1

Please enter string to encrypt: Top Secret

Please enter offset value (1 to 94): 98

Please enter offset value (1 to 94): -3

Please enter offset value (1 to 94): 20

Encrypted string:

h$%4gyw'y)

\*\*\* Menu \*\*\*

1. Encrypt string

2. Decrypt string

3. Brute force decryption

4. Quit

What would you like to do [1,2,3,4]? 4

Goodbye.

**Sample output 3:**

File : wayby001\_encryptor.py

Author : Batman

Stud ID : 0123456X

Email ID : wayby001

Description: Programming Assignment 2 - Caesar Cipher

This is my own work as defined by the University's Academic Misconduct Policy.

\*\*\* Menu \*\*\*

1. Encrypt string

2. Decrypt string

3. Brute force decryption

4. Quit

What would you like to do [1,2,3,4]? 2

Please enter string to decrypt: h$%4gyw'y)

Please enter offset value (1 to 94): -2

Please enter offset value (1 to 94): 150

Please enter offset value (1 to 94): 0

Please enter offset value (1 to 94): 95

Please enter offset value (1 to 94): 20

Decrypted string:

Top Secret

\*\*\* Menu \*\*\*

1. Encrypt string

2. Decrypt string

3. Brute force decryption

4. Quit

What would you like to do [1,2,3,4]? 4

Goodbye.

**Sample output 4:**

File : wayby001\_encryptor.py

Author : Batman

Stud ID : 0123456X

Email ID : wayby001

Description: Programming Assignment 2 - Caesar Cipher

This is my own work as defined by the University's Academic Misconduct Policy.

\*\*\* Menu \*\*\*

1. Encrypt string

2. Decrypt string

3. Brute force decryption

4. Quit

What would you like to do [1,2,3,4]? 9

Invalid choice, please enter either 1, 2, 3 or 4.

What would you like to do [1,2,3,4]? 1

Please enter string to encrypt: one more time

Please enter offset value (1 to 94): 7

Encrypted string:

vul'tvyl'{ptl

\*\*\* Menu \*\*\*

1. Encrypt string

2. Decrypt string

3. Brute force decryption

4. Quit

What would you like to do [1,2,3,4]? 4

Goodbye.

Useful Built-in Python Functions – Useful for Assignment 2

**ord() and chr() Functions (useful for part II):**

**ord(c)**

Given c, a string of length one ord(c), returns an integer (ASCII value) representing the value of the string.

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

**chr(i)**

Returns a string of one character whose ASCII value is the integer i.

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