

UltraCrypt

A CBSE Grade 12 Project



Aditya Shivaraman

Akshat K

Ananth Bharathwaj t a

barath s

class xii c

# Acknowledgment:

We would like to thank our Computer Science teachers, Ms. Harifa Begum, Ms. Nirmala Devi, for guiding us in making this project. Their valuable inputs and advice have enabled us to make this program efficient and user-friendly.

Contents

[Acknowledgment: 1](#_Toc63753969)

[Synopsis 3](#_Toc63753970)

[User Interface 4](#_Toc63753971)

[Main menu 4](#_Toc63753972)

[Encryption and Decryption Menu: 4](#_Toc63753973)

[File Based Encryption/Decryption 5](#_Toc63753974)

[UI Settings 5](#_Toc63753975)

[Graphing 6](#_Toc63753976)

[Encryption: 6](#_Toc63753977)

[Functions Created 8](#_Toc63753978)

[GUI 8](#_Toc63753979)

[Graphing: 9](#_Toc63753980)

[Encryption 9](#_Toc63753981)

[Project Listing: 12](#_Toc63753982)

[graphing.py 12](#_Toc63753983)

[GUI.py 12](#_Toc63753984)

[encryption.py 28](#_Toc63753985)

[Output: 38](#_Toc63753986)

# Synopsis

UltraCrypt is an encryption and graphing tool made with Python. It has an easy to use User Interface and interactive User Experience. It has a graphing tool which plots the frequency of all characters in a given file. It has an encryption/decryption tool, which can be used on separate strings or entire files. It has six encryption protocols, list as follows:

Morse Code

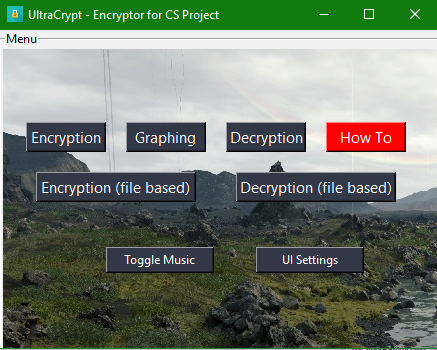
* Caesar Cipher
* Advanced Caesar Cipher
* Homo-Phonic Substitution Cipher
* Four Square Cipher
* At-Bash Cipher

This project was created with:

* Microsoft Windows
* IDLE

## User Interface

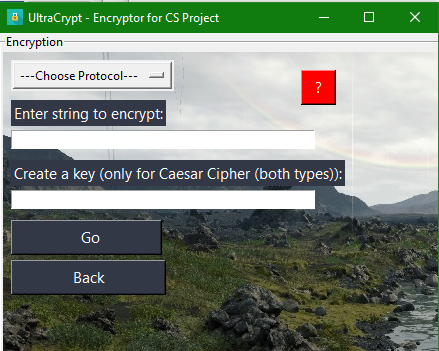
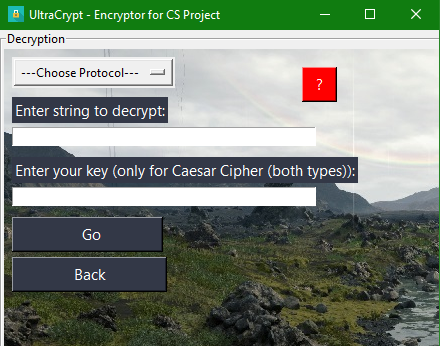
### Main menu



The Main Menu

The main menu provides access to all of the features of the app, including encryption, decryption, graphing, UI settings and How To. Clicking the ‘How To’ button will open a brief text file describing how to use the app. The ‘Toggle Music’ button plays ambient music and clicking it again pauses it.

### Encryption and Decryption Menu:

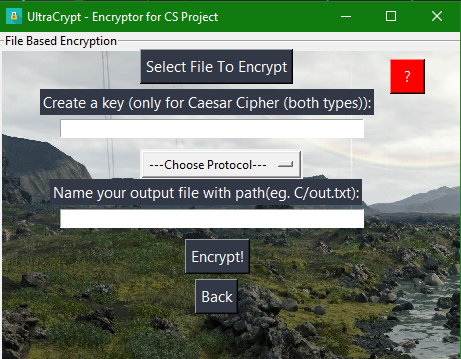
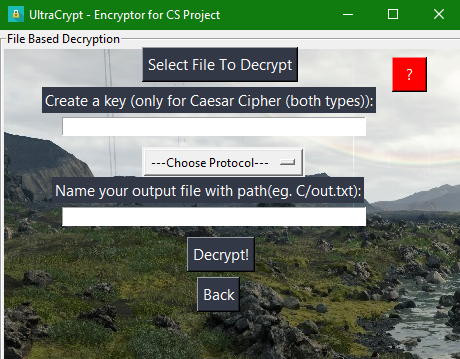
The Decryption Menu

The Encryption Menu

Both Menus are designed with an identical style and share certain traits. The ‘?’ button opens another document on how to use the tool.

The dropdown menu for the protocol contains the choice for six protocols mentioned above. After entering the string to encrypt, and the key if necessary, clicking ‘Go’ will give you the output screen.

### File Based Encryption/Decryption

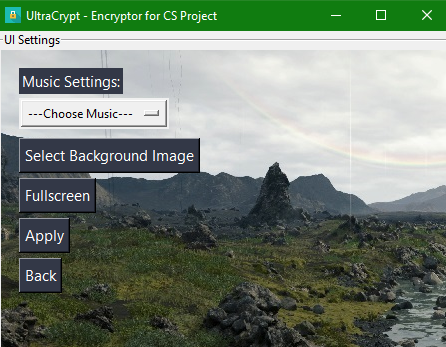
 

The File-Based Encryption menu The File-Based Decryption menu

Both Menus are designed with an identical style and share certain traits. The ‘?’ button opens another document on how to use the tool.

The same protocols as above are available to use. After selecting a file to encrypt/decrypt, one must name the output file with path (example mentioned), one can select the encrypt/decrypt button to generate the output file.

### UI Settings



The UI Settings Menu

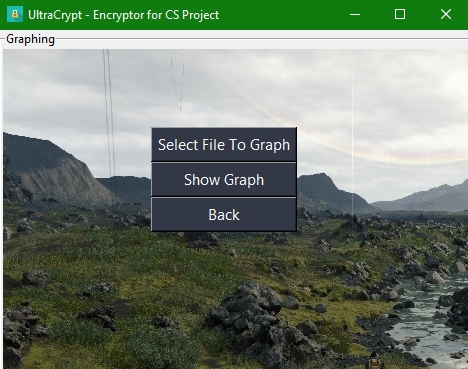
The UI Settings menu lets one configure the ambient music and background image of the app.

The ‘Fullscreen’ button sets the app to full screen. After making your choice, selecting ‘Apply’ will set the changes into effect.

## Graphing

The Graphing feature enables users to see a graph (point graph) of the frequencies of various characters in a text file. After selecting a text file from the menu, a graph is displayed with each character in the file and the corresponding frequencies. The “matplotlib” library has been used to display the graph.

This option can be used to see how strong a particular encryption technique is by comparing the distribution of characters in the encrypted file with the original file.

 Chart, scatter chart

Description automatically generated

The main graphing menu. The graph generated for a test file.

## Encryption:

Morse code Encryption/Decryption:

This encryption methods converts the given string/file to be encrypted into morse code ( Code containing dots and dashes).It has been in use for hundreds of years.

Caesar Cipher Encryption/Decryption:

This is an encryption method which converts the characters of the given string into other characters based on a key determined by the user which changes its ascii value to obtain other characters

Caesar Cipher Advanced/Vigenere Cipher Encryption/Decryption:

This encryption method converts the characters of the given string/file based on a given key which uses polyalphabetical substitution that cycles the character through alphabetical order.

Homophonic Substitution Cipher Encryption/Decryption:

This encryption method converts the characters in the given string/file based on a predefined dictionary containing all keys on a keyboard corresponding to another given value.

Four Square Cipher Encryption/Decryption:

This encryption method employs a four dimensional list containing all the letters of the alphabet , and breaks down the given string / file into pairs of characters and converts the pair of characters into the corresponding pair of characters using the lists.

Atbash Encryption/Decryption:

This encryption method converts alphabets in the given string/file into its corresponding alphabet in the reverse order of the English lexicon

# Functions Created

## GUI

raise\_frame functions: Allows you to switch between frames.

addfile set of functions: Allows the user to select a file for a particular task by using the tkinter filedialogue command askopenfilename

play\_music(): Allows you to play a particular audio file using pygame.

It declares a variable mxstate to check if music is playing or not. Then it uses an if condition to pause or play a selected track.

switch\_music(): Allows you to switch the audio track that is being played. It takes input from a dropdown menu called music\_dd. Then it loads the selected track and plays it.

fullscreen(): Sets the app to full screen mode. A class called FullScreenApp was created to change the geometry of the app window.This class was called using the function.

switch\_bg(): Allows you to switch the background of the app for a specific window . It configures the image present in the current canvas.

apply():Calls the switch\_bg and switch\_music functions

getkey():it accepts a key for encryption/decryption from the user. It derives the input from the input box keyin\_enc.

encryptstr():Accepts a string and encrypts it using the given protocol. It accepts the chosen protocol from the dropdown list dd , the text from input box textin, the key from the getkey().Then it encrypts the string based on the given choice and by calling their respective functions from the encryption module.

decryptstr():Accepts a string and decrypts it using the given protocol. It accepts the chosen protocol from the dropdown list dd , the text from input box textin\_dec, the key from the getkey().Then it encrypts the string based on the given choice and by calling their respective functions from the encryption module.

encrypt\_file():Accepts a file and encrypts it by iteratively calling encryptstr function. . It accepts the chosen protocol from the dropdown list dd , the textfile from the addfile\_enc(). It gets the key from the input box keyin\_fe .Then it encrypts the file based on the given choice and by calling the function file\_encryptstr(A clone of encryptstr()) iteratively for every line of the file. It writes the result into a new textfile whose name is accepted from the input box output\_file\_enc.

decrypt\_file():Accepts a file and decrypts it by iteratively calling decryptstr function. . It accepts the chosen protocol from the dropdown list dd , the textfile from the addfile\_dec(). It gets the key from the input box keyin .Then it decrypts the file based on the given choice and by calling the function file\_decryptstr(A clone of decryptstr()) iteratively for every line of the file. It writes the result into a new textfile whose name is accepted from the input box output\_file\_enc.

graphit():Accepts a file and plots a point graph of the frequency of the characters in it. It gets the file to plot from the function addfilegraph and calls the freq function from the graphing module.

Note:

All accepted data is accepted from the user via dropdown lists input boxes and buttons and these data are globally imported into the functions. In tkinter frames are widgets that can act as menus where buttons can be placed.

## Graphing:

freq: This function reads the input file, calculates the frequency of each character, and returns a dictionary with the characters as keys and respective frequencies as values.

graph: This function takes the dictionary containing frequencies as well as the labels for x and y axes, and plots the point graph with the characters as values on x-axis and the frequencies as values on y-axis, using the matplotlib.pyplot module.

## Encryption

Functions in the “Encryption.py” file:

morse\_enc:

Converts the given character into corresponding character using the keys and values given in the Morse\_code dictionary .

morse\_dec:

Converts the encrypted morse code string back into its original string using inv\_morse dictionary’s keys and values

custom\_enc:

This employs the principle of the Caesar cipher to change the characters by changing its ord values using a given key.

custom\_dec:

This employs the principle of the Caesar cipher to change the characters by changing its ord values using a given key to obtain the original string.

cc2\_enc:

This function converts the characters of the given string based on the vig\_dict dictionary and inputted string key and converts it into the corresponding character by adding the ord value of the corresponding character in the dictionary ,cycling through the English alphabet

cc2\_dec:

This function converts the characters of the given string based on the vig\_dict dictionary and inputted string key and converts it into the corresponding character by subtracting the ord value of the corresponding character in the dictionary ,cycling through the English alphabet

hsc\_enc:

This function converts the given string of characters into its corresponding characters using the hsc\_dict dictionary containing predefined keys and values.

hsc\_dec:

This function converts the given string of characters into its corresponding characters using the inv\_hsc\_dict dictionary containing predefined keys and values to obtain the original string

sch:

This functions converts the given word into pairs of characters and leaves a single character in the end only for words with odd no of letters

four\_square\_enc:

This function uses the output of the previously mentioned sch function and considers both of them as two opposite corners of a square with each characters in list l1 l4 respectively. This function then stores the position of the characters as (i1,j1) and (i2,j2) in a square matrix. The output will be the elements in the position (i1,j2) and (i2,j1) in lists l2 and l3 respectively.

four\_square\_dec:

This function uses the output of the previously mentioned sch function and considers both of them as two opposite corners of a square with each characters in list l2 and l3 respectively. This function then stores the position of the characters as (i1,j2) and (i2,j1) in a square matrix. The output will be the elements in the position (i1,j1) and (i2,j2) in lists l1 and l4 respectively.Thus giving us the original statement.

at\_bash:

This function converts alphabets in the given string into its corresponding value in the given at\_bash\_dict dictionary thus its corresponding alphabet in the reverse order of the English lexicon. This function works for both encrypting and decrypting the string.

# Project Listing:

## graphing.py

import os

import matplotlib.pyplot as plt

def graph(d, x\_label, y\_label):

# graphs the frequencies based on the dictionary

plt.scatter(d.keys(), d.values(), linewidth=0)

plt.xlabel(x\_label)

plt.ylabel(y\_label)

plt.show()

def freq(input\_file):

# reads the input file and gets the frequency of each letter. Returns a dictionary with the frequencies.

if os.path.isfile(input\_file) == False:

print("Error: Input file not found. Please enter a valid file.") # message should be shown in GUI

return {}

with open(input\_file, 'r') as inf:

frequency = {}

s = inf.read().upper()

for char in s:

if char in frequency:

frequency[char] += 1

else:

frequency[char] = 1

return frequency

## GUI.py

from tkinter import \*

from tkinter import filedialog, messagebox

from PIL import ImageTk, Image

import graphing,encryption, os, pygame

root = Tk()

root.title('UltraCrypt - Encryptor for CS Project')

root.iconbitmap('images/icon.ico')

def raise\_frame(frame):

frame.tkraise()

file\_enc = None

def addfile\_enc():

global file\_enc

choice = filedialog.askopenfilename(initialdir = "/",title = "Select file",filetypes = (('Text Files','\*.txt'),("all files","\*.\*")))

file\_enc=choice

file\_dec = None

def addfile\_dec():

global file\_dec

choice = filedialog.askopenfilename(initialdir = "/",title = "Select file",filetypes = (('Text Files','\*.txt'),("all files","\*.\*")))

file\_dec=choice

outfile=''

def addfilegraph():

global outfile

choice = filedialog.askopenfilename(initialdir = "/",title = "Select file",filetypes = (('Text Files','\*.txt'),("all files","\*.\*")))

outfile=choice

#playing music

pygame.init()

mxstate = 0 # music play state

pygame.mixer.music.load('music/bg1.mp3')

def Play\_music():

global mxstate

if mxstate == 0: # music not started

pygame.mixer.music.play()

btn11.configure(text="Pause")

mxstate = 1

return

if mxstate == 1: # music playing

pygame.mixer.music.pause()

btn11.configure(text="Resume")

else: # music paused

pygame.mixer.music.unpause()

btn11.configure(text="Pause")

mxstate = 3 - mxstate # swap pause state

#declaring frames

menu = LabelFrame(root, text = 'Menu')

encrypt = LabelFrame(root, text = 'Encryption')

graph = LabelFrame(root, text = 'Graphing')

decrypt = LabelFrame(root, text = 'Decryption')

easter\_egg = LabelFrame(root, text = 'Secret')

outputframe\_enc = LabelFrame(root, text = 'Output')

outputframe\_dec = LabelFrame(root, text = 'Output')

file\_encrypt = LabelFrame(root, text='File Based Encryption')

file\_decrypt = LabelFrame(root, text='File Based Decryption')

ui\_settings = LabelFrame(root, text='UI Settings')

for frame in (menu,encrypt,decrypt,graph,easter\_egg,outputframe\_enc, outputframe\_dec, file\_encrypt, file\_decrypt, ui\_settings):

frame.grid(row=0, column=0, sticky='news') #row=3, column=15, padx=250, sticky='news'

#background

for frame in (menu,encrypt,decrypt,graph,easter\_egg,outputframe\_enc, outputframe\_dec, file\_encrypt, file\_decrypt, ui\_settings):

im = Image.open('images/newbg.jpg')

w,h=im.size

canvas = Canvas(frame, width=w, height=h)

canvas.pack()

canvas.image = ImageTk.PhotoImage(im)

background = canvas.create\_image(0, 0, image=canvas.image, anchor='nw')

#Menu

Button(menu, text='Encryption',bg='#333847', font=('Segoe UI',11), fg='white', command=lambda:raise\_frame(encrypt)).place(x=25, y=75, height = 30, width = 80)

Button(menu, text='Graphing',bg='#333847', font=('Segoe UI',11), fg='white', command=lambda:raise\_frame(graph)).place(x=125, y=75, height = 30, width = 80)

Button(menu, text='Decryption',bg='#333847', font=('Segoe UI',11), fg='white', command=lambda:raise\_frame(decrypt)).place(x=225, y=75, height = 30, width = 80)

Button(menu, text='How To',bg='red', font=('Segoe UI',11), fg='white', command=lambda:os.startfile('about.txt')).place(x=325, y=75, height = 30, width = 80)

Button(menu, text='Encryption (file based)',bg='#333847', font=('Segoe UI',11), fg='white', command=lambda:raise\_frame(file\_encrypt)).place(x=35, y=125, height = 30, width = 160)

Button(menu, text='Decryption (file based)',bg='#333847', font=('Segoe UI',11), fg='white', command=lambda:raise\_frame(file\_decrypt)).place(x=235, y=125, height = 30, width = 160)

btn11 = Button(menu, text='Toggle Music', width=14, bg='#333847', font=('Segoe UI',9), fg='white', command=Play\_music)

btn11.place(x=105, y=200)

Button(menu, text='UI Settings', width=14, bg='#333847', font=('Segoe UI',9), fg='white', command= lambda:raise\_frame(ui\_settings)).place(x=255, y=200)

#UI Settings

music\_list = ['---Choose Music---', 'Aetheral (default)', 'Cheery 1', 'Cheery 2', 'Cheery 3', 'Cheery 4', 'Dramatic 1', 'Dramatic 2', 'Dramatic 3']

music\_choice = StringVar()

music\_choice.set(music\_list[0])

Label(ui\_settings, text = 'Music Settings:',bg='#333847', font=('Segoe UI',11), fg='white').place(x=20,y=20)

music\_dd = OptionMenu(ui\_settings, music\_choice, \*music\_list)

music\_dd.place(x=20, y=50)

musicpathlist = ['music/bg1.mp3','music/bg2.mp3','music/bg3.mp3','music/bg4.mp3','music/bg5.mp3','music/bg6.mp3','music/bg7.mp3','music/bg8.mp3']

def switch\_music():

music = music\_choice.get()

if music == '---Choose Music---':

x=1

else:

for i in range(len(music\_list)):

if music\_list[i]==music: i1=i-1

print(i1)

mtp = musicpathlist[i1]

pygame.mixer.music.load(mtp)

pygame.mixer.music.play(-1)

newbg=''

class FullScreenApp(object):

def \_\_init\_\_(self, master, \*\*kwargs):

self.master=master

pad=3

self.\_geom='200x200+0+0'

master.geometry("{0}x{1}+0+0".format(

master.winfo\_screenwidth()-pad, master.winfo\_screenheight()-pad))

master.bind('<Escape>',self.toggle\_geom)

def toggle\_geom(self,event):

geom=self.master.winfo\_geometry()

print(geom,self.\_geom)

self.master.geometry(self.\_geom)

self.\_geom=geom

def fullscreen():

FullScreenApp(root)

def addbg():

global newbg

choice = filedialog.askopenfilename(initialdir = "/",title = "Select file",filetypes = (('PNG Files','\*.png'),("all files","\*.\*")))

newbg = choice

def switch\_bg():

global newbg, background, canvas

canvas.delete(background)

im = Image.open(newbg)

w, h = im.size

if w>1360 or h>680: w,h=1360,680

canvas.config(width=w, height=h)

canvas.image = ImageTk.PhotoImage(im)

background = canvas.create\_image(0, 0, image=canvas.image, anchor='nw')

def apply():

if newbg=='':

switch\_music()

else:

switch\_bg()

switch\_music()

Button(ui\_settings, text='Select Background Image (Beta)',bg='#333847', font=('Segoe UI',11), fg='white', command=addbg).place(x=20, y=90)

Button(ui\_settings, text='Fullscreen',bg='#333847', font=('Segoe UI',11), fg='white', command=fullscreen).place(x=20, y=130)

Button(ui\_settings, text='Apply',bg='#333847', font=('Segoe UI',11), fg='white', command=apply).place(x=20, y=170)

Button(ui\_settings, text='Back',bg='#333847', font=('Segoe UI',11), fg='white', command=lambda:raise\_frame(menu)).place(x=20,y=210)

#Encryption/decryption Protocols

protocol\_list = ['---Choose Protocol---','Morse Code','Caesar Cipher','Advanced Caesar Cipher', 'Homo-phonic Substitution Cipher', 'Four Square Cipher', 'At\_bash Cipher']

protocol\_choice=StringVar()

protocol\_choice.set(protocol\_list[0])

#Encryption Menu

dd=OptionMenu(encrypt, protocol\_choice, \*protocol\_list)

dd.place(x = 10, y = 10)

Label(encrypt, text = 'Enter string to encrypt:',bg='#333847', font=('Segoe UI',11), fg='white').place(x=10, y=50)

textin = Entry(encrypt, width = 50)

textin.place(x=10, y=80)

Label(encrypt, text = 'Create a key (only for Caesar Cipher (both types)):',bg='#333847', font=('Segoe UI',11), fg='white').place(x=10,y=110)

keyin\_enc = Entry(encrypt, width = 50)

keyin\_enc.place(x=10,y=140)

def getkey():

ret = keyin\_enc.get()

return ret

def encryptstr():

chosen\_protocol = protocol\_choice.get()

input\_text = textin.get()

key = getkey()

print('\'',key,'\'', type(key))

if chosen\_protocol == '---Choose Protocol---':

messagebox.showerror("Error!", "Please select a valid protocol.")

elif input\_text == '':

messagebox.showerror("Error!", "Please enter a string.")

else:

encrypted\_str = ''

done = True

if chosen\_protocol == 'Morse Code':

encrypted\_str = encryption.morse\_enc(input\_text)

if chosen\_protocol == 'Caesar Cipher':

try:

key = int(key)

print(key)

except:

messagebox.showerror("Error!", "Please enter a vaild key (numeric type within 26).")

done = False

encrypted\_str = encryption.custom\_enc(input\_text, key)

if chosen\_protocol == 'Advanced Caesar Cipher':

if key.isalpha() == False:

messagebox.showerror("Error!", "Please enter a vaild key (alpahbetic).")

done = False

encrypted\_str = encryption.cc2\_enc(input\_text, key)

if chosen\_protocol == 'Homo-phonic Substitution Cipher':

encrypted\_str = encryption.hsc\_enc(input\_text)

if chosen\_protocol == 'Four Square Cipher':

encrypted\_str = encryption.four\_square\_enc(encryption.sch(input\_text))

if chosen\_protocol == 'At\_bash Cipher':

encrypted\_str = encryption.at\_bash(input\_text)

if done == True:

global outputlabel\_enc

print(encrypted\_str)

outputlabel\_enc = Label(outputframe\_enc, text='Your output is:\n'+encrypted\_str,bg='#333847', font=('Segoe UI',11), fg='white')

outputlabel\_enc.place(x=20, y=20)

raise\_frame(outputframe\_enc)

def back\_to\_enc():

outputlabel\_enc.destroy()

raise\_frame(encrypt)

Button(encrypt, text=' Go ',bg='#333847', font=('Segoe UI',11), fg='white', command=encryptstr).place(x=10,y=170)

Button(encrypt, text=' Back ',bg='#333847', font=('Segoe UI',11), fg='white', command=lambda:raise\_frame(menu)).place(x=10,y=210)

Button(outputframe\_enc, text=' Back ',bg='#333847', font=('Segoe UI',11), fg='white', command=back\_to\_enc).pack()

Button(encrypt, bg='red', font=('Segoe UI',11), fg='white',text = ' ? ',command =lambda: os.startfile('about\_encryption.txt')).place(x=300,y=20)

#Decryption Menu

dd=OptionMenu(decrypt, protocol\_choice, \*protocol\_list)

dd.place(x = 10, y = 10)

Label(decrypt, text = 'Enter string to decrypt:',bg='#333847', font=('Segoe UI',11), fg='white').place(x=10, y=50)

textin\_dec = Entry(decrypt, width = 50)

textin\_dec.place(x=10, y=80)

Label(decrypt, text = 'Enter your key (only for Caesar Cipher (both types)):',bg='#333847', font=('Segoe UI',11), fg='white').place(x=10,y=110)

keyin\_dec = Entry(decrypt, width = 50)

keyin\_dec.place(x=10,y=140)

def decryptstr():

chosen\_protocol = protocol\_choice.get()

input\_text = textin\_dec.get()

if chosen\_protocol == '---Choose Protocol---':

messagebox.showerror("Error!", "Please select a valid protocol.")

elif input\_text == '':

messagebox.showerror("Error!", "Please enter a string.")

else:

decrypted\_str = ''

done = True

if chosen\_protocol == 'Morse Code':

try:

decrypted\_str = encryption.morse\_dec(input\_text)

except:

messagebox.showerror("Error!", "Please enter a vaild decryption string (ie dots and dashes).")

done = False

if chosen\_protocol == 'Caesar Cipher':

try:

key = int(keyin\_dec.get())

except:

messagebox.showerror("Error!", "Please enter a vaild key (numeric type within 26).")

done = False

decrypted\_str = encryption.custom\_dec(input\_text, key)

if chosen\_protocol == 'Advanced Caesar Cipher':

key = str(keyin\_dec.get())

if key.isalpha() == False:

messagebox.showerror("Error!", "Please enter a vaild key (alpahbetic).")

done = False

decrypted\_str = encryption.cc2\_dec(input\_text, key)

if chosen\_protocol == 'Homo-phonic Substitution Cipher':

decrypted\_str = encryption.hsc\_dec(input\_text)

if chosen\_protocol == 'Four Square Cipher':

decrypted\_str = encryption.four\_square\_dec(encryption.sch(input\_text.lower()))

if chosen\_protocol == 'At\_bash Cipher':

decrypted\_str = encryption.at\_bash(input\_text)

if done == True:

global outputlabel\_dec

print(decrypted\_str)

outputlabel\_dec = Label(outputframe\_dec, text='Your output is:\n'+decrypted\_str)

outputlabel\_dec.place(x=20, y=20)

raise\_frame(outputframe\_dec)

def back\_to\_dec():

outputlabel\_dec.destroy()

raise\_frame(decrypt)

Button(decrypt, text=' Go ',bg='#333847', font=('Segoe UI',11), fg='white', command=decryptstr).place(x=10,y=170)

Button(decrypt, text=' Back ',bg='#333847', font=('Segoe UI',11), fg='white', command=lambda:raise\_frame(menu)).place(x=10,y=210)

Button(outputframe\_dec, text=' Back ',bg='#333847', font=('Segoe UI',11), fg='white', command=back\_to\_dec).pack()

Button(decrypt, bg='red', font=('Segoe UI',11), fg='white',text = ' ? ',command =lambda: os.startfile('about\_encryption.txt')).place(x=300,y=20)

#File Based Encryption

Button(file\_encrypt, text='Back',bg='#333847', font=('Segoe UI',11), fg='white', command=lambda:raise\_frame(menu)).place(x=195,y=230)

Button(file\_encrypt, text='Select File To Encrypt',bg='#333847', font=('Segoe UI',11), fg='white', command=addfile\_enc).place(x=140, y=0)

Label(file\_encrypt, text = 'Create a key (only for Caesar Cipher (both types)):',bg='#333847', font=('Segoe UI',11), fg='white').place(x=40,y=40)

keyin\_fe = Entry(file\_encrypt, width = 50)

keyin\_fe.place(x=60,y=70)

dd=OptionMenu(file\_encrypt, protocol\_choice, \*protocol\_list)

dd.place(x = 140, y = 100)

Label(file\_encrypt, text = 'Name your output file with path(eg. C/out.txt):',bg='#333847', font=('Segoe UI',11), fg='white').place(x=50,y=130)

output\_file\_enc = Entry(file\_encrypt, width = 50)

output\_file\_enc.place(x=60, y=160)

Button(file\_encrypt, bg='red', font=('Segoe UI',11), fg='white',text = ' ? ',command =lambda: os.startfile('about\_encryption.txt')).place(x=390,y=10)

def getkey\_file():

retfi = keyin\_fe.get()

return retfi

def file\_encryptstr(input\_text):

chosen\_protocol = protocol\_choice.get()

if chosen\_protocol == '---Choose Protocol---':

messagebox.showerror("Error!", "Please select a valid protocol.")

else:

encrypted\_str = ''

done = True

if chosen\_protocol == 'Morse Code':

encrypted\_str = encryption.morse\_enc(input\_text)

if chosen\_protocol == 'Caesar Cipher':

try:

key = int(getkey\_file())

encrypted\_str = encryption.custom\_enc(input\_text, key)

except:

messagebox.showerror("Error!", "Please enter a vaild key (numeric type within 26).")

done = False

if chosen\_protocol == 'Advanced Caesar Cipher':

key = str(getkey\_file())

if key.isalpha() == False:

messagebox.showerror("Error!", "Please enter a vaild key (alpahbetic).")

done = False

encrypted\_str = encryption.cc2\_enc(input\_text, key)

if chosen\_protocol == 'Homo-phonic Substitution Cipher':

encrypted\_str = encryption.hsc\_enc(input\_text)

if chosen\_protocol == 'Four Square Cipher':

encrypted\_str = encryption.four\_square\_enc(encryption.sch(input\_text))

if chosen\_protocol == 'At\_bash Cipher':

encrypted\_str = encryption.at\_bash(input\_text)

if done == True:

return encrypted\_str

def encrypt\_file():

global file\_enc

if file\_enc == None or file\_enc == '':

messagebox.showerror("Error!", "Please select a file.")

file = open(file\_enc, 'r')

outputfile = output\_file\_enc.get()

if outputfile == None or outputfile == '':

messagebox.showerror("Error!", "Please enter a valid file path.")

output = open(outputfile, 'w')

line = file.readline()

while line != '':

x = str(line)[0:-1].lower()

out = file\_encryptstr(x)

print(out)

output.write(out+'\n')

line = file.readline()

file.close()

output.close()

os.startfile(outputfile)

Button(file\_encrypt, text = 'Encrypt!', bg='#333847', font=('Segoe UI',11), fg='white', command= encrypt\_file).place(x=185, y=190)

#File Based Decryption

Button(file\_decrypt, text='Back',bg='#333847', font=('Segoe UI',11), fg='white', command=lambda:raise\_frame(menu)).place(x=195,y=230)

Button(file\_decrypt, text='Select File To Decrypt',bg='#333847', font=('Segoe UI',11), fg='white', command=addfile\_dec).place(x=140, y=0)

Label(file\_decrypt, text = 'Create a key (only for Caesar Cipher (both types)):',bg='#333847', font=('Segoe UI',11), fg='white').place(x=40,y=40)

keyin = Entry(file\_decrypt, width = 50)

keyin.place(x=60,y=70)

dd=OptionMenu(file\_decrypt, protocol\_choice, \*protocol\_list)

dd.place(x = 140, y = 100)

Label(file\_decrypt, text = 'Name your output file with path(eg. C/out.txt):',bg='#333847', font=('Segoe UI',11), fg='white').place(x=50,y=130)

output\_file\_dec = Entry(file\_decrypt, width = 50)

output\_file\_dec.place(x=60, y=160)

Button(file\_decrypt, bg='red', font=('Segoe UI',11), fg='white',text = ' ? ',command =lambda: os.startfile('about\_encryption.txt')).place(x=390,y=10)

def file\_decryptstr(input\_text):

chosen\_protocol = protocol\_choice.get()

if chosen\_protocol == '---Choose Protocol---':

messagebox.showerror("Error!", "Please select a valid protocol.")

else:

decrypted\_str = ''

done = True

if chosen\_protocol == 'Morse Code':

decrypted\_str = encryption.morse\_dec(input\_text)

if chosen\_protocol == 'Caesar Cipher':

try:

key = int(keyin.get())

except:

messagebox.showerror("Error!", "Please enter a vaild key (numeric type within 26).")

done = False

decrypted\_str = encryption.custom\_dec(input\_text, key)

if chosen\_protocol == 'Advanced Caesar Cipher':

key = str(keyin.get())

if key.isalpha() == False:

messagebox.showerror("Error!", "Please enter a vaild key (alpahbetic).")

done = False

decrypted\_str = encryption.cc2\_dec(input\_text, key)

if chosen\_protocol == 'Homo-phonic Substitution Cipher':

decrypted\_str = encryption.hsc\_dec(input\_text)

if chosen\_protocol == 'Four Square Cipher':

decrypted\_str = encryption.four\_square\_dec(encryption.sch(input\_text))

if chosen\_protocol == 'At\_bash Cipher':

decrypted\_str = encryption.at\_bash(input\_text)

if done == True:

return decrypted\_str

def decrypt\_file():

global file\_dec

if file\_dec == None or file\_dec == '':

messagebox.showerror("Error!", "Please select a file.")

file = open(file\_dec, 'r')

outputfile = output\_file\_dec.get()

if outputfile == None or outputfile == '':

messagebox.showerror("Error!", "Please enter a valid file path.")

output = open(outputfile, 'w')

line = file.readline()

while line != '':

x = str(line)[0:-1].lower()

out = file\_decryptstr(x)

print(out)

output.write(out+'\n')

line = file.readline()

file.close()

output.close()

os.startfile(outputfile)

Button(file\_decrypt, text = 'Decrypt!', bg='#333847', font=('Segoe UI',11), fg='white', command= decrypt\_file).place(x=185, y=190)

#Graphing Menu

Button(graph, text='Select File To Graph',bg='#333847', font=('Segoe UI',11), fg='white', command=addfilegraph,width=17,height=1).place(x=150, y=20)

def graphit():

global outfile

if outfile == '':

messagebox.showerror("Error!", "Please select a valid file.")

return

f = graphing.freq(outfile)

graphing.graph(f, "Character", "No. of occurences")

Button(graph, text='Show Graph',bg='#333847', font=('Segoe UI',11), fg='white', command=graphit,width=17,height=1).place(x=150, y=55)

Button(graph, text='Back',bg='#333847', font=('Segoe UI',11), fg='white', command=lambda:raise\_frame(menu),width=17,height=1).place(x=150, y=90)

raise\_frame(menu)

mainloop()

## encryption.py

import sys

Morse\_code = {'A': '.-', 'B': '-...',

'C': '-.-.', 'D': '-..', 'E': '.',

'F': '..-.', 'G': '--.', 'H': '....',

'I': '..', 'J': '.---', 'K': '-.-',

'L': '.-..', 'M': '--', 'N': '-.',

'O': '---', 'P': '.--.', 'Q': '--.-',

'R': '.-.', 'S': '...', 'T': '-',

'U': '..-', 'V': '...-', 'W': '.--',

'X': '-..-', 'Y': '-.--', 'Z': '--..',

'1': '.----', '2': '..---', '3': '...--',

'4': '....-', '5': '.....', '6': '-....',

'7': '--...', '8': '---..', '9': '----.',

'0': '-----', ', ': '--..--', '.': '.-.-.-',

'?': '..--..', '/': '-..-.', '-': '-....-',

'(': '-.--.', ')': '-.--.-', '': '', '!': '!',

'@': '@', '#': '#', '$': '$', '%': '%', '^': '^',

'&': '&', '\*': '\*', '+': '+','=': '=', ',': ',',

'<': '<', '>': '>', ';': ';', ':': ':', '"': '"',

"'": "'", '[': '[', ']': ']', '{': '{', '}': '}'}

vig\_dict = {'a': 1, 'c': 3, 'b': 2, 'e': 5, 'd': 4, 'g': 7, 'f': 6, 'i': 9, 'h': 8, 'k': 11, 'j': 10, 'm': 13, 'l': 12,

'o': 15, 'n': 14, 'q': 17, 'p': 16, 's': 19, 'r': 18, 'u': 21, 't': 20, 'w': 23, 'v': 22, 'y': 25, 'x': 24,

'z': 26, '.': 27, '1': 28, '2': 29, '3': 30, '4': 31, '5': 32, '6': 33, '7': 34, '8': 35, '9': 36,

'!': '!', '@': '@', '#': '#', '$': '$', '%': '%',

'^': '^', '&': '&', '\*': '\*', '(': '(', ')': ')',

'+': '+', '0': '0', '=': '=', ',': ',', '/': '/', '<': '<',

'>': '>', '?': '?', ';': ';', ':': ':', '"': '"',

"'": "'", '[': '[', ']': ']', '{': '{', '}': '}'}

inv\_morse = {v: k for k, v in Morse\_code.items()}

hsc\_dict = {'a': 'q', 'c': 'w', 'b': 'e', 'e': 'r', 'd': 't', 'g': 'y', 'f': 'u', 'i': 'i', 'h': 'o', 'k': '5',

'j': 'p', 'm': '2', 'l': 'a', 'o': 's', 'n': 'd', 'q': 'f', 'p': 'g', 's': 'h', 'r': '0', 'u': 'j',

't': 'k', 'w': 'l',

'v': '1', 'y': '8', 'x': '3', 'z': 'z', '.': 'x', '1': '6', '2': 'c', '3': 'v', '4': '7', '5': 'b',

'6': 'n', '7': '4', '8': 'm', '9': '9', ' ': ' ',

'!': '!', '@': '@', '#': '#', '$': '$', '%': '%',

'^': '^', '&': '&', '\*': '\*', '(': '(', ')': ')',

'+': '+', '=': '=', ',': ',', '/': '/', '<': '<',

'>': '>', '?': '?', ';': ';', ':': ':', '"': '"',

"'": "'", '[': '[', ']': ']', '{': '{', '}': '}'}

inv\_hsc\_dict = {v: k for k, v in hsc\_dict.items()}

at\_bash\_dict = {'a': 'z', 'b': 'y', 'c': 'x', ' ': ' ', 'd': 'w', 'e': 'v', 'f': 'u', 'g': 't', 'h': 's', 'i': 'r',

'j': 'q', 'k': 'p', 'l': 'o', 'm': 'n', '0': '9', '1': '8', '2': '7', '3': '6', '4': '5',

'!': '!', '@': '@', '#': '#', '$': '$', '%': '%',

'^': '^', '&': '&', '\*': '\*', '(': '(', ')': ')',

'+': '+', '5': '5', '6': '6', '7': '7', '8': '8', '9': '9',

'=': '=', ',': ',', '/': '/', '<': '<',

'>': '>', '?': '?', ';': ';', ':': ':', '"': '"',

"'": "'", '[': '[', ']': ']', '{': '{', '}': '}'}

at\_bash\_dict\_inv = {v: k for k, v in at\_bash\_dict.items()}

at\_bash\_dict.update(at\_bash\_dict\_inv)

def mlk(x):

g = ''

for i in x:

g += i

return g

def morse\_enc(string2):

cipher = ''

for i in string2:

if i !=' ':

cipher = cipher + Morse\_code[i.upper()]

cipher = cipher + ' '

else:

cipher = cipher + i

return cipher

def morse\_dec(string1):

cipher = ''

string1 = string1 + ' '

list1 = []

for i in string1:

if i != ' ':

list1 += [i]

else:

s = mlk(list1)

cipher = cipher + inv\_morse[s]

cipher = cipher + ' '

list1 = []

v = ''

c = 0

cipher = cipher + ' '

while c < len(cipher):

if c == len(cipher) - 1:

break

elif cipher[c] != ' ':

v = v + cipher[c]

elif cipher[c] == ' ':

if cipher[c + 1] == ' ':

v = v + ' '

c = c + 1

return v.lower()

'''

def diff\_hellman\_dec(string):

flag = True

while flag == True:

s=0

p = int(input("enter value of prime number p: "))

for i in range(1, p):

if p%i==0:

s+=1

if s == 1:

flag = False

else:

print("enter only prime number")

q = int(input("enter value of generator q"))

a = int(input("enter whole number a lesser than p: "))

b = int(input("enter whole number b lesser than p: "))

a\_star = (q\*\*a) % p

b\_star = (q\*\*b) % p

x1 = ((b\_star)\*\*a) % p

x2 = ((a\_star)\*\*b) % p

print(x1, x2) '''

a = ''

def custom\_enc(string3, gen):

b = ''

for i in string3:

b = b + chr(ord(i) + gen)

b = b[::-1]

return b

def custom\_dec(string4, gen):

b = ''

for i in string4:

b = b + chr(ord(i) - gen)

b = b[::-1]

return b

def cc2\_enc(string5, key1):

enc = ''

f = 0

for i in string5:

if f == len(key1):

f = 0

enc = enc + (chr((ord(i) + vig\_dict[key1[f].lower()])))

f = f + 1

return enc

def cc2\_dec(string6, key2):

enc = ''

f = 0

for i in string6:

if f == len(key2):

f = 0

enc = enc + (chr((ord(i) - vig\_dict[(key2[f]).lower()])))

f = f + 1

return enc

def hsc\_enc(string7):

enc = ''

for i in string7:

enc = enc + hsc\_dict[i]

return enc

def hsc\_dec(string8):

dec = ''

for i in string8:

dec = dec + inv\_hsc\_dict[i]

return dec

def word\_to\_list(string9):

lo = []

f = 0

v = 0

for i in string9:

if f == 0:

lo.append('')

if f < 2:

lo[v] = lo[v] + i

else:

v = v + 1

f = 0

return lo

def four\_square\_enc(j):

l1 = [['a', 'b', 'c', 'd', 'e'], ['f', 'g', 'h', 'i', 'k'], ['l', 'm', 'n', 'o', 'p'], ['q', 'r', 's', 't', 'u'],

['v', 'w', 'x', 'y', 'z']]

l2 = [['p', 'l', 't', 'x', 'u'], ['a', 'i', 'r', 'z', 'y'], ['b', 'e', 'g', 'v', 'h'], ['c', 'k', 'd', 'm', 'w'],

['q', 'f', 'o', 's', 'n']]

l3 = [['c', 'h', 'k', 'o', 's'], ['d', 'g', 'u', 'p', 'n'], ['a', 'e', 'q', 'i', 'f'], ['b', 'm', 'y', 'w', 'x'],

['z', 'u', 't', 'v', 'r']]

l4 = l1

dec = ''

spl\_chr = 'j !@#$%^&\*()+1234567890=,/<>?;:"\'[]{}'

for i in j:

if len(i) == 2 and i[0] not in spl\_chr and i[1] not in spl\_chr:

l1in = []

l4in = []

for x in range(0, 5):

for y in range(0, 5):

if l1[x][y] == i[0]:

l1in = l1in + [x]

l1in = l1in + [y]

for x in range(0, 5):

for y in range(0, 5):

if l4[x][y] == i[1]:

l4in = l4in + [x]

l4in = l4in + [y]

dec = dec + l2[l1in[0]][l4in[1]]

dec = dec + l3[l4in[0]][l1in[1]]

else:

dec = dec + i

return dec

def four\_square\_dec(j):

l1 = [['a', 'b', 'c', 'd', 'e'], ['f', 'g', 'h', 'i', 'k'], ['l', 'm', 'n', 'o', 'p'], ['q', 'r', 's', 't', 'u'],

['v', 'w', 'x', 'y', 'z']]

l2 = [['p', 'l', 't', 'x', 'u'], ['a', 'i', 'r', 'z', 'y'], ['b', 'e', 'g', 'v', 'h'], ['c', 'k', 'd', 'm', 'w'],

['q', 'f', 'o', 's', 'n']]

l3 = [['c', 'h', 'k', 'o', 's'], ['d', 'g', 'u', 'p', 'n'], ['a', 'e', 'q', 'i', 'f'], ['b', 'm', 'y', 'w', 'x'],

['z', 'u', 't', 'v', 'r']]

l4 = l1

dec = ''

spl\_chr = 'j !@#$%^&\*()+1234567890=,/<>?;:"\'[]{}'

for i in j:

if len(i) == 2 and i[0] not in spl\_chr and i[1] not in spl\_chr:

l2in = []

l3in = []

for x in range(0, 5):

for y in range(0, 5):

if l2[x][y] == i[0]:

l2in = l2in + [x]

l2in = l2in + [y]

for x in range(0, 5):

for y in range(0, 5):

if l3[x][y] == i[1]:

l3in = l3in + [x]

l3in = l3in + [y]

dec = dec + l1[l2in[0]][l3in[1]]

dec = dec + l4[l3in[0]][l2in[1]]

else:

dec = dec + i

return dec

def sch(string10):

k = string10.split()

j = []

for i in k:

m = 0

b = 2

while b < (len(i) + 2):

s = ''

s = s + i[int(m): int(b)]

j = j + [s]

m = m + 2

b = b + 2

j = j + [' ']

return j

def at\_bash(string11):

string11 = string11.lower()

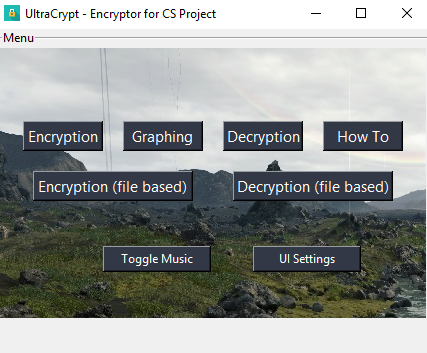
enc = ''

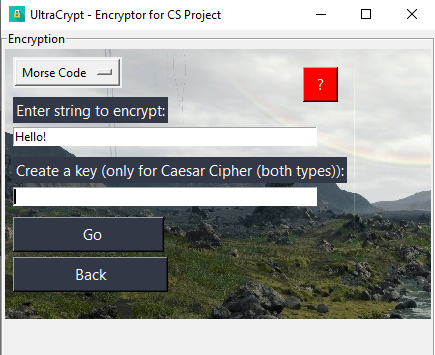
for i in string11:

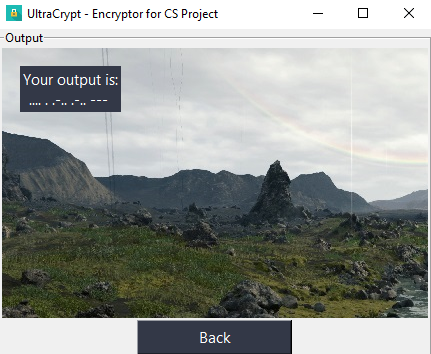
enc = enc + at\_bash\_dict[i]

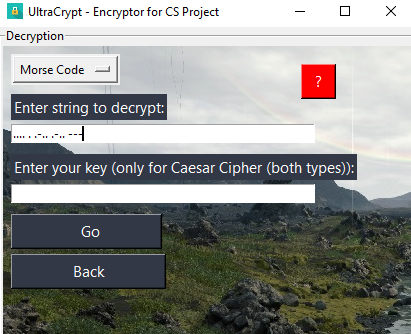
return enc

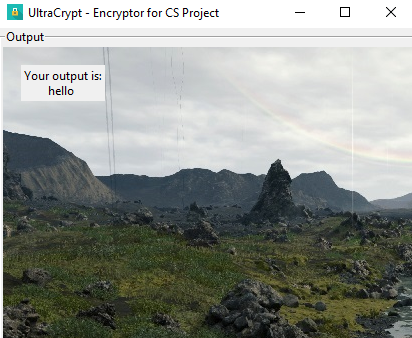
# Output:

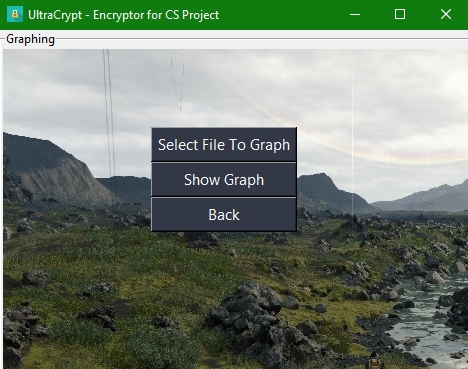










 Chart, scatter chart

Description automatically generated