School-based Assignment

Test & Evaluation Report

SHIFT CIPHER DECRYPTER

SKH Lam Kau Mow Secondary School | 2025 ICT | Law Wilson

CONTENTS

Introduction

This report outlines the development process of the SHIFT CIPHER DECRYPTER program, the development of the program will be illustrated below.

Access the file at https://github.com/wilsonlaw2299/SHIFT-CIPHER-DECRYPTER.git

Program

Main function

```
base = ord('A')                     # Base Unicode value for uppercase letters
default_long_text = 200             # Default minimum length for long text
default_common = "E"                # Default most common character

mode = int(input("Select mode by entering the number ONLY: \n[1]: Encrypt
                    \n[2]: Decrypt \n[-1]: End\n"))
while mode != -1:
    check_upper = False

    while not check_upper:
        original_message = str(input("\nInput the string: "))
        check_upper = True

        for char in original_message:
            if char.isalpha():
                if char.islower():
                    check_upper = False
                    break

        if not check_upper:
            print("The character(s) is not in all uppercase")
            opt_invalid_input = str(input("Enter the letter:
                                      \n[R]: Enter the string again
                                      \n[U]: Convert all character(s) into uppercase \n"))

            if opt_invalid_input == "U":
               # Convert all characters to uppercase
                original_message = original_message.upper()

    if mode == 1:
        shift = int(input("Shift number: "))
        print("\nEncrypted Message: ", shift_encrypt(original_message, shift))

    elif mode == 2:
        if len(original_message.split(" ")) < default_long_text:
            if str(input("The message is not long enough (>200 words),
                         \n The result may be inaccurate.
                         \n Continue[Y/N]")) == "Y":
                print("\nDecrypted Message: ", shift_decrypt(original_message))
        else:
            print("\nDecrypted Message: ", shift_decrypt(original_message))

    mode = int(input("\nSelect mode: \n[1]: Encrypt \n[2]: Decrypt \n[-1]: End "))
```

Encrypt

```
def shift_encrypt(message, shift):
    encrypted_message = ""      # Initialize an empty string to store the encrypted message

    for char in message:
        if char.isalpha():      # Check if the character is an alphabet

            # Encrypt the character by shifting its Unicode value
            encrypted_char = chr((ord(char) - base + shift) % 26 + base)

            # Add the encrypted character to the encrypted message
            encrypted_message += encrypted_char
        else:
            # If the character is not an alphabet,
              add it to the encrypted message without encryption
            encrypted_message += char

    return encrypted_message    # Return the encrypted message
```

Decrypt

```
def shift_decrypt(message):
    decrypted_message = ""      # Initialize an empty string to store the decrypted message

    # Find the most common character in the encrypted message and determine the shift value
    shift = find_most_common(message)

    for char in message:
        if char.isalpha():      # Check if the character is an alphabet

            # Decrypt the character by shifting its Unicode value
            decrypted_char = chr((ord(char) - base - shift) % 26 + base)

            # Add the decrypted character to the decrypted message
            decrypted_message += decrypted_char
        else:
            # If the character is not an alphabet,
              add it to the decrypted message without decryption
            decrypted_message += char

    return decrypted_message    # Return the decrypted message
```

Find the most frequent character(s)

```
def find_most_common(message):
    max_count = 0               # Initialize the maximum count of a character to 0
    # Initialize an empty array to store the most frequent characters
    freq_letter_array = []

    # Iterate through each character in the message
    for char in message:
        if char.isalpha():      # Check if the character is an alphabet
            count = 0           # Initialize the count of the current character to 0

            # Count the occurrences of the current character in the message
            for char_moving in message:
                if char == char_moving:
                    count += 1

            # If the count is greater than or equal to the maximum count
            if count >= max_count:
                # If the count is strictly greater than the maximum count
                if count > max_count:
                    # Clear the array since there is a new character with a higher count
                    freq_letter_array.clear()
                # Update the maximum count
                max_count = count

                # Add the character to the array of most frequent characters
                freq_letter_array.append(char)

    # Initialize an empty array to store the final unique most frequent characters
    final_array = []

    for char in freq_letter_array:

        # If the character is not already in the final array
        if char not in final_array:
            # Add the character to the final array
            final_array.append(char)

    # If there is more than one final most frequent character
    if len(final_array) > 1:
        # Ask the user to select the most common character
        print(final_array)
        most_common = str(input("Select the most common char: "))

        # Validate the user input
        while most_common not in final_array:
            most_common = str(input("Invalid input \nSelect the most common char: "))
    # If there is only one final most frequent character, assign it directly
    else:
        most_common = final_array[0]
    # Calculate the shift number based on the most common character
    shift_number = cal_shift_number(most_common)

    return shift_number         # Return the calculated shift number
```

Calculate the shift number based on the most common character

```python
def cal_shift_number(most_common):
    # Calculate the shift number based on the difference between the Unicode values
      of the most common character and the default common character
    shift_number = (ord(most_common) - ord(default_common)) % 26

    # Return the calculated shift number
    return shift_number
```

Detailed Explanation on Searching Algorithm

```
def find_most_common(message):
    ...
    for char in message:
        if char.isalpha():
            count = 0
            for char_moving in message:
                if char == char_moving:
                    count +=1
    ...
```

The program executes the idea of linear search, it iterates over each character in the message.

```
            if count >= max_count:
                if count > max_count:
```

After counting the occurrences of the current character, it checks if the count is greater than or equal to the maximum count ***max_count*** found so far.

```
                    freq_letter_array.clear()
```

If it is greater, it clears the ***freq_letter_array***.

```
                max_count = count
                freq_letter_array.append(char)
    ...
```

Then, it updates the ***max_count*** to the new count value and appends the current character to the ***freq_letter_array***.

```
    final_array = []
    for char in freq_letter_array:
        if char not in final_array:
            final_array.append(char)
    ...
```

final_array to store the unique characters with the highest frequency

```
    if len(final_array) > 1:
        print(final_array)
        most_common = str(input("select the
                                most char: "))
        while most_common not in final_array:
            most_common = str(input("Invalid
            input \nselect the most common
            char: "))
    else:
        most_common = final_array[0]
    ...
```

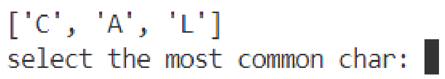
The program allows there are multiple characters with the same frequency,

for example:

CALCULATE

C	2
A	2
L	2

, the user can choose which character to use to calculate the shift number(k),



```
['C', 'A', 'L']
select the most common char: █
```

Pros and Cons

Pros

1. **Clear variable declaration and initialization**

```
base = ord('A')
default_long_text = 200
default_common = "E"
```

2. **Data collection, input**
   a. choose of mode
   b. entering string

```
mode = int(input("select mode: \n[1]:encrypt
        \n[2]:decrypt \n[-1]:end "))
```

3. **System development cycle**
   a. post-test loop
      Allow continuous run of program

```
while mode != -1:
    ...
    original_message = str(input("\nInput the
    string: "))
    ...
    mode = int(input("\nselect mode: \n[1]:encrypt
            \n[2]:decrypt \n[-1]:end "))
...
```

4. **Data validation**
   a. check if the all letters input are all uppercase

```
check_upper = True
...
for char in original_message:
    if char.isalpha():
        if char.islower():
            check_upper = False
            break

def find_most_common(messaga):
        ...
        while most_common not in final_array:
            most_common = str(input("Invalid input
            \nselect the most common char: "))
        ...
```

5. **Subprogram**
   Modularity / Reusability / Portability

```
def shift_encrypt(message,shift): # ...
def shift_decrypt(message): # ...
def find_most_common(messaga): # ...
def cal_shift_number(most_common): # ...
```

6. **Searching algorithm**
   The algorithm iterates through each character in the message and counts the number of occurrences of each alphabetic character.

```
def find_most_common(messaga): # ...
```

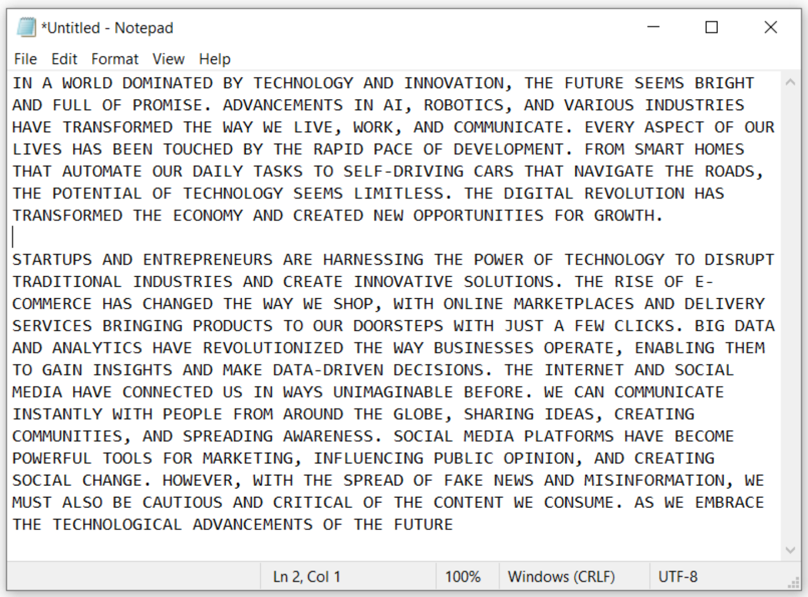
# See pages 2-5 for the complete program

Cons

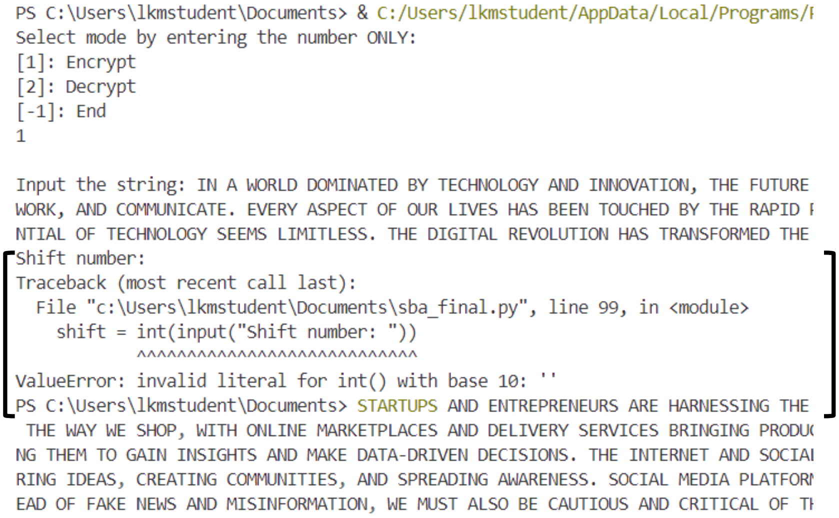
1. Multiple paragraphs are not supported.

If the user enters the newline character "\n" to indicate the completion of their message, the program will proceed to the next step.

“\n”



```
IN A WORLD DOMINATED BY TECHNOLOGY AND INNOVATION, THE FUTURE SEEMS BRIGHT
AND FULL OF PROMISE. ADVANCEMENTS IN AI, ROBOTICS, AND VARIOUS INDUSTRIES
HAVE TRANSFORMED THE WAY WE LIVE, WORK, AND COMMUNICATE. EVERY ASPECT OF OUR
LIVES HAS BEEN TOUCHED BY THE RAPID PACE OF DEVELOPMENT. FROM SMART HOMES
THAT AUTOMATE OUR DAILY TASKS TO SELF-DRIVING CARS THAT NAVIGATE THE ROADS,
THE POTENTIAL OF TECHNOLOGY SEEMS LIMITLESS. THE DIGITAL REVOLUTION HAS
TRANSFORMED THE ECONOMY AND CREATED NEW OPPORTUNITIES FOR GROWTH.

STARTUPS AND ENTREPRENEURS ARE HARNESSING THE POWER OF TECHNOLOGY TO DISRUPT
TRADITIONAL INDUSTRIES AND CREATE INNOVATIVE SOLUTIONS. THE RISE OF E-
COMMERCE HAS CHANGED THE WAY WE SHOP, WITH ONLINE MARKETPLACES AND DELIVERY
SERVICES BRINGING PRODUCTS TO OUR DOORSTEPS WITH JUST A FEW CLICKS. BIG DATA
AND ANALYTICS HAVE REVOLUTIONIZED THE WAY BUSINESSES OPERATE, ENABLING THEM
TO GAIN INSIGHTS AND MAKE DATA-DRIVEN DECISIONS. THE INTERNET AND SOCIAL
MEDIA HAVE CONNECTED US IN WAYS UNIMAGINABLE BEFORE. WE CAN COMMUNICATE
INSTANTLY WITH PEOPLE FROM AROUND THE GLOBE, SHARING IDEAS, CREATING
COMMUNITIES, AND SPREADING AWARENESS. SOCIAL MEDIA PLATFORMS HAVE BECOME
POWERFUL TOOLS FOR MARKETING, INFLUENCING PUBLIC OPINION, AND CREATING
SOCIAL CHANGE. HOWEVER, WITH THE SPREAD OF FAKE NEWS AND MISINFORMATION, WE
MUST ALSO BE CAUTIOUS AND CRITICAL OF THE CONTENT WE CONSUME. AS WE EMBRACE
THE TECHNOLOGICAL ADVANCEMENTS OF THE FUTURE
```



```
PS C:\Users\lkmstudent\Documents> & C:/Users/lkmstudent/AppData/Local/Programs/P
Select mode by entering the number ONLY:
[1]: Encrypt
[2]: Decrypt
[-1]: End
1

Input the string: IN A WORLD DOMINATED BY TECHNOLOGY AND INNOVATION, THE FUTURE
WORK, AND COMMUNICATE. EVERY ASPECT OF OUR LIVES HAS BEEN TOUCHED BY THE RAPID P
NTIAL OF TECHNOLOGY SEEMS LIMITLESS. THE DIGITAL REVOLUTION HAS TRANSFORMED THE
Shift number:
Traceback (most recent call last):
  File "c:\Users\lkmstudent\Documents\sba_final.py", line 99, in <module>
    shift = int(input("Shift number: "))
            ^^^^^^^^^^^^^^^^^^^^^^^^^^^^
ValueError: invalid literal for int() with base 10: ''
PS C:\Users\lkmstudent\Documents> STARTUPS AND ENTREPRENEURS ARE HARNESSING THE
 THE WAY WE SHOP, WITH ONLINE MARKETPLACES AND DELIVERY SERVICES BRINGING PRODUC
NG THEM TO GAIN INSIGHTS AND MAKE DATA-DRIVEN DECISIONS. THE INTERNET AND SOCIAL
RING IDEAS, CREATING COMMUNITIES, AND SPREADING AWARENESS. SOCIAL MEDIA PLATFORM
EAD OF FAKE NEWS AND MISINFORMATION, WE MUST ALSO BE CAUTIOUS AND CRITICAL OF TH
```

← Program restriction

2. Inaccurate decrypted result

The most common character may not be the "E" for short paragraphs, and users may run the risk of producing misspelled strings.

Test Data and Test Cases

Data validation

Input (select mode, string)		*Output*
1	hello	The character(s) is not in all uppercase Enter the letter: [R]: Enter the string again [U]: Convert all character(s) into uppercase
2	hello	
1	Hello	

Encrypt

Test data *(string, shift k)*		*Output* *(Encrypted Message)*
WORK HARD	8	EWZS PIZL
EWZS PIZL	-8	WORK HARD
!!!	8	!!!
123!@#$	8	123!@#$
WORK HARD, PLAY HARD!	10	GYBU RKBN, ZVKI RKBN!
GYBU RKBN, ZVKI RKBN!	-10	WORK HARD, PLAY HARD!
GYBU RKBN, ZVKI RKBN!	16	WORK HARD, PLAY HARD!
IN A WORLD DOMINATED BY TECHNOLOGY AND INNOVATION, THE FUTURE SEEMS BRIGHT AND FULL OF PROMISE. ADVANCEMENTS IN AI, ROBOTICS, AND VARIOUS INDUSTRIES HAVE TRANSFORMED THE WAY WE LIVE, WORK, AND COMMUNICATE. EVERY ASPECT OF OUR LIVES HAS BEEN TOUCHED BY THE RAPID PACE OF DEVELOPMENT. FROM SMART HOMES THAT AUTOMATE OUR DAILY TASKS TO SELF-DRIVING CARS THAT NAVIGATE THE ROADS, THE POTENTIAL OF TECHNOLOGY SEEMS LIMITLESS. THE DIGITAL REVOLUTION HAS TRANSFORMED THE ECONOMY AND CREATED NEW OPPORTUNITIES FOR GROWTH. STARTUPS AND ENTREPRENEURS ARE HARNESSING THE POWER OF TECHNOLOGY TO DISRUPT TRADITIONAL INDUSTRIES AND CREATE INNOVATIVE SOLUTIONS. THE RISE OF E-COMMERCE HAS CHANGED THE WAY WE SHOP, WITH ONLINE MARKETPLACES AND DELIVERY SERVICES BRINGING PRODUCTS TO OUR DOORSTEPS WITH JUST A FEW CLICKS. BIG DATA AND ANALYTICS HAVE REVOLUTIONIZED THE WAY BUSINESSES OPERATE, ENABLING THEM TO GAIN INSIGHTS AND MAKE DATA-DRIVEN DECISIONS. THE INTERNET AND SOCIAL MEDIA HAVE CONNECTED US IN WAYS UNIMAGINABLE BEFORE. WE CAN COMMUNICATE INSTANTLY WITH PEOPLE FROM AROUND THE GLOBE, SHARING IDEAS, CREATING COMMUNITIES, AND SPREADING AWARENESS. SOCIAL MEDIA PLATFORMS HAVE BECOME POWERFUL TOOLS FOR MARKETING, INFLUENCING PUBLIC OPINION, AND CREATING SOCIAL CHANGE. HOWEVER, WITH THE SPREAD OF FAKE NEWS AND MISINFORMATION, WE MUST ALSO BE CAUTIOUS AND CRITICAL OF THE CONTENT WE CONSUME. AS WE EMBRACE THE TECHNOLOGICAL ADVANCEMENTS OF THE FUTURE	8	QV I EWZTL LWUQVIBML JG BMKPVWTWOG IVL QVVWDIBQWV, BPM NCBCZM AMMUA JZQOPB IVL NCTT WN XZWUQAM. ILDIVKMUMVBA QV IQ, ZWJWBQKA, IVL DIZQWCA QVLCABZQMA PIDM BZIVANWZUML BPM EIG EM TQDM, EWZS, IVL KWUUCVQKIBM. MDMZG IAXMKB WN WCZ TQDMA PIA JMMV BWCKPML JG BPM ZIXQL XIKM WN LMDMTWXUMVB. NZWU AUIZB PWUMA BPIB ICBWUIBM WCZ LIQTG BIASA BW AMTN-LZQDQVO KIZA BPIB VIDQOIBM BPM ZWILA, BPM XWBMVBQIT WN BMKPVWTWOG AMMUA TQUQBTMAA. BPM LQOQBIT ZMDWTCBQWV PIA BZIVANWZUML BPM MKWVWUG IVL KZMIBML VME WXXWZBCVQBQMA NWZ OZWEBP. ABIZBCXA IVL MVBZMXZMVMCZA IZM PIZVMAAQVO BPM XWEMZ WN BMKPVWTWOG BW LQAZCXB BZILQBQWVIT QVLCABZQMA IVL KZMIBM QVVWDIBQDM AWTCBQWVA. BPM ZQAM WN M-KWUUMZKM PIA KPIVOML BPM EIG EM APWX, EQBP WVTQVM UIZSMBXTIKMA IVL LMTQDMZG AMZDQKMA JZQVOQVO XZWLCKBA BW WCZ LWWZABMXA EQBP RCAB I NME KTQKSA. JQO LIBI IVL IVITGBQKA PIDM ZMDWTCBQWVQHML BPM EIG JCAQVMAAMA WXMZIBM, MVIJTQVO BPMU BW OIQV QVAQOPBA IVL UISM LIBI-LZQDMV LMKQAQWVA. BPM QVBMZVMB IVL AWKQIT UMLQI PIDM KWVVMKBML CA QV EIGA CVQUIOQVIJTM JMNWZM. EM KIV KWUUCVQKIBM QVABIVBTG EQBP XMWXTM NZWU IZWCVL BPM OTWJM, APIZQVO QLMIA, KZMIBQVO KWUUCVQBQMA, IVL AXZMILQVO IEIZMVMAA. AWKQIT UMLQI XTIBNWZUA PIDM JMKWUM XWEMZNCT BWWTA NWZ UIZSMBQVO, QVNTCMVKQVO XCJTQK WXQVQWV, IVL KZMIBQVO AWKQIT KPIVOM. PWEMDMZ, EQBP BPM AXZMIL WN NISM VMEA IVL UQAQVNWZUIBQWV, EM UCAB ITAW JM KICBQWCA IVL KZQBQKIT WN BPM KWVBMVB EM KWVACUM. IA EM MUJZIKM BPM BMKPVWTWOQKIT ILDIVKMUMVBA WN BPM NCBCZM

Decrypt

Test data *(string, (select most frequent char))*		*Output* *(Decrypted Message)*
EWZS PIZL		JBEX UNEQ
AAABBBCCCDD	A	EEEFFFGGGHH
AAABBBCCCDD	B	DDDEEEFFFGG
QV I EWZTL LWUQVIBML JG BMKPVWTWOG IVL QVVWDIBQWV, BPM NCBCZM AMMUA JZQOPB IVL NCTT WN XZWUQAM. ILDIVKMUMVBA QV IQ, ZWJWBQKA, IVL DIZQWCA QVLCABZQMA PIDM BZIVANWZUML BPM EIG EM TQDM, EWZS, IVL KWUUCVQKIBM. MDMZG IAXMKB WN WCZ TQDMA PIA JMMV BWCKPML JG BPM ZIXQL XIKM WN LMDMTWXUMVB. NZWU AUIZB PWUMA BPIB ICBWUIBM WCZ LIQTG BIASA BW AMTN-LZQDQVO KIZA BPIB VIDQOIBM BPM ZWILA, BPM XWBMVBQIT WN BMKPVWTWOG AMMUA TQUQBTMAA. BPM LQOQBIT ZMDWTCBQWV PIA BZIVANWZUML BPM MKWVWUG IVL KZMIBML VME WXXWZBCVQBQMA NWZ OZWEBP. ABIZBCXA IVL MVBZMXZMVMCZA IZM PIZVMAAQVO BPM XWEMZ WN BMKPVWTWOG BW LQAZCXB BZILQBQWVIT QVLCABZQMA IVL KZMIBM QVVWDIBQDM AWTCBQWVA. BPM ZQAM WN M-KWUUMZKM PIA KPIVOML BPM EIG EM APWX, EQBP WVTQVM UIZSMBXTIKMA IVL LMTQDMZG AMZDQKMA JZQVOQVO XZWLCKBA BW WCZ LWWZABMXA EQBP RCAB I NME KTQKSA. JQO LIBI IVL IVITGBQKA PIDM ZMDWTCBQWVQHML BPM EIG JCAQVMAAMA WXMZIBM, MVIJTQVO BPMU BW OIQV QVAQOPBA IVL UISM LIBI-LZQDMV LMKQAQWVA. BPM QVBMZVMB IVL AWKQIT UMLQI PIDM KWVVMKBML CA QV EIGA CVQUIOQVIJTM JMNWZM. EM KIV KWUUCVQKIBM QVABIVBTG EQBP XMWXTM NZWU IZWCVL BPM OTWJM, APIZQVO QLMIA, KZMIBQVO KWUUCVQBQMA, IVL AXZMILQVO IEIZMVMAA. AWKQIT UMLQI XTIBNWZUA PIDM JMKWUM XWEMZNCT BWWTA NWZ UIZSMBQVO, QVNTCMVKQVO XCJTQK WXQVQWV, IVL KZMIBQVO AWKQIT KPIVOM. PWEMDMZ, EQBP BPM AXZMIL WN NISM VMEA IVL UQAQVNWZUIBQWV, EM UCAB ITAW JM KICBQWCA IVL KZQBQKIT WN BPM KWVBMVB EM KWVACUM. IA EM MUJZIKM BPM BMKPVWTWOQKIT ILDIVKMUMVBA WN BPM NCBCZM		IN A WORLD DOMINATED BY TECHNOLOGY AND INNOVATION, THE FUTURE SEEMS BRIGHT AND FULL OF PROMISE. ADVANCEMENTS IN AI, ROBOTICS, AND VARIOUS INDUSTRIES HAVE TRANSFORMED THE WAY WE LIVE, WORK, AND COMMUNICATE. EVERY ASPECT OF OUR LIVES HAS BEEN TOUCHED BY THE RAPID PACE OF DEVELOPMENT. FROM SMART HOMES THAT AUTOMATE OUR DAILY TASKS TO SELF-DRIVING CARS THAT NAVIGATE THE ROADS, THE POTENTIAL OF TECHNOLOGY SEEMS LIMITLESS. THE DIGITAL REVOLUTION HAS TRANSFORMED THE ECONOMY AND CREATED NEW OPPORTUNITIES FOR GROWTH. STARTUPS AND ENTREPRENEURS ARE HARNESSING THE POWER OF TECHNOLOGY TO DISRUPT TRADITIONAL INDUSTRIES AND CREATE INNOVATIVE SOLUTIONS. THE RISE OF E-COMMERCE HAS CHANGED THE WAY WE SHOP, WITH ONLINE MARKETPLACES AND DELIVERY SERVICES BRINGING PRODUCTS TO OUR DOORSTEPS WITH JUST A FEW CLICKS. BIG DATA AND ANALYTICS HAVE REVOLUTIONIZED THE WAY BUSINESSES OPERATE, ENABLING THEM TO GAIN INSIGHTS AND MAKE DATA-DRIVEN DECISIONS. THE INTERNET AND SOCIAL MEDIA HAVE CONNECTED US IN WAYS UNIMAGINABLE BEFORE. WE CAN COMMUNICATE INSTANTLY WITH PEOPLE FROM AROUND THE GLOBE, SHARING IDEAS, CREATING COMMUNITIES, AND SPREADING AWARENESS. SOCIAL MEDIA PLATFORMS HAVE BECOME POWERFUL TOOLS FOR MARKETING, INFLUENCING PUBLIC OPINION, AND CREATING SOCIAL CHANGE. HOWEVER, WITH THE SPREAD OF FAKE NEWS AND MISINFORMATION, WE MUST ALSO BE CAUTIOUS AND CRITICAL OF THE CONTENT WE CONSUME. AS WE EMBRACE THE TECHNOLOGICAL ADVANCEMENTS OF THE FUTURE

Unit Test

Find the most frequent character(s)

```
def find_most_common(messaga): ...
```

message (str)	*max_count (int)*	*freq_letter_array (array)*	*final_array (array)*	*shift_number (int)* ***[return value]***
DDD	3	['D', 'D', 'D']	['D']	25
HELLO	2	['L', 'L']	['L']	7
CALCULATE	2	['C', 'A', 'L', 'C', 'L', 'A']	['C', 'A', 'L']	*

** : depends on user's choice*

Calculate the shift number based on the most common character

```
def cal_shift_number(most_common): ...
default_common = "E"
```

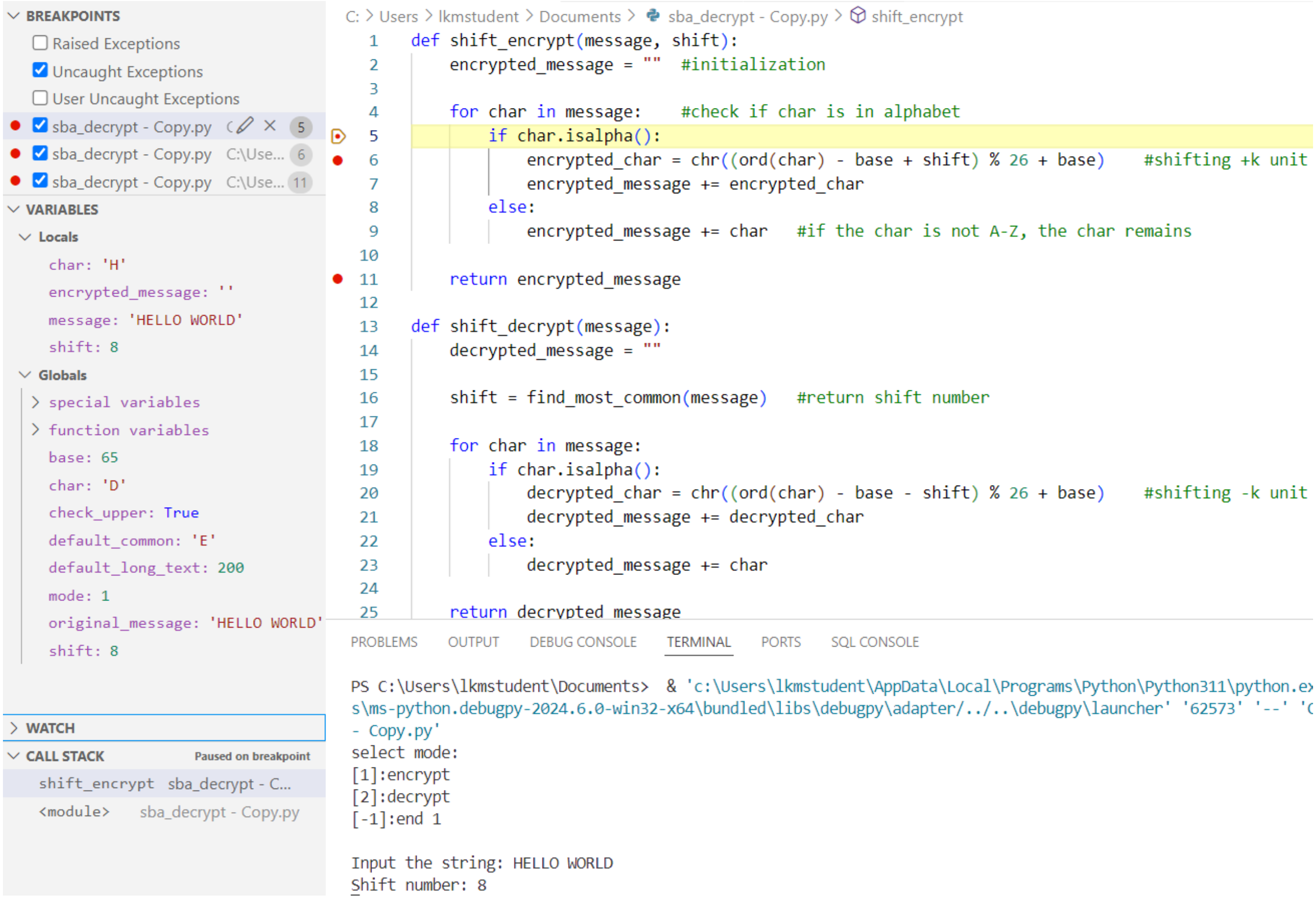
most_common (str)	*shift_number (int)* ***[return value]***
D	25
E	0
F	1

Debugging

1. Printing variable values at various stages to trace the flow of execution.

```
def find_most_common(message):
...
    #print(most_common)        ←
    #print(shift_number)       ←
    return shift_number
```

2 Using breakpoints to pause program execution and inspect intermediate variables.



```
BREAKPOINTS
□ Raised Exceptions
☑ Uncaught Exceptions
□ User Uncaught Exceptions
● ☑ sba_decrypt - Copy.py  C  5
● ☑ sba_decrypt - Copy.py  C:\Use...  6
● ☑ sba_decrypt - Copy.py  C:\Use...  11
VARIABLES
  Locals
    char: 'H'
    encrypted_message: ''
    message: 'HELLO WORLD'
    shift: 8
  Globals
    > special variables
    > function variables
    base: 65
    char: 'D'
    check_upper: True
    default_common: 'E'
    default_long_text: 200
    mode: 1
    original_message: 'HELLO WORLD'
    shift: 8
> WATCH
CALL STACK                Paused on breakpoint
  shift_encrypt   sba_decrypt - C...
  <module>        sba_decrypt - Copy.py
```

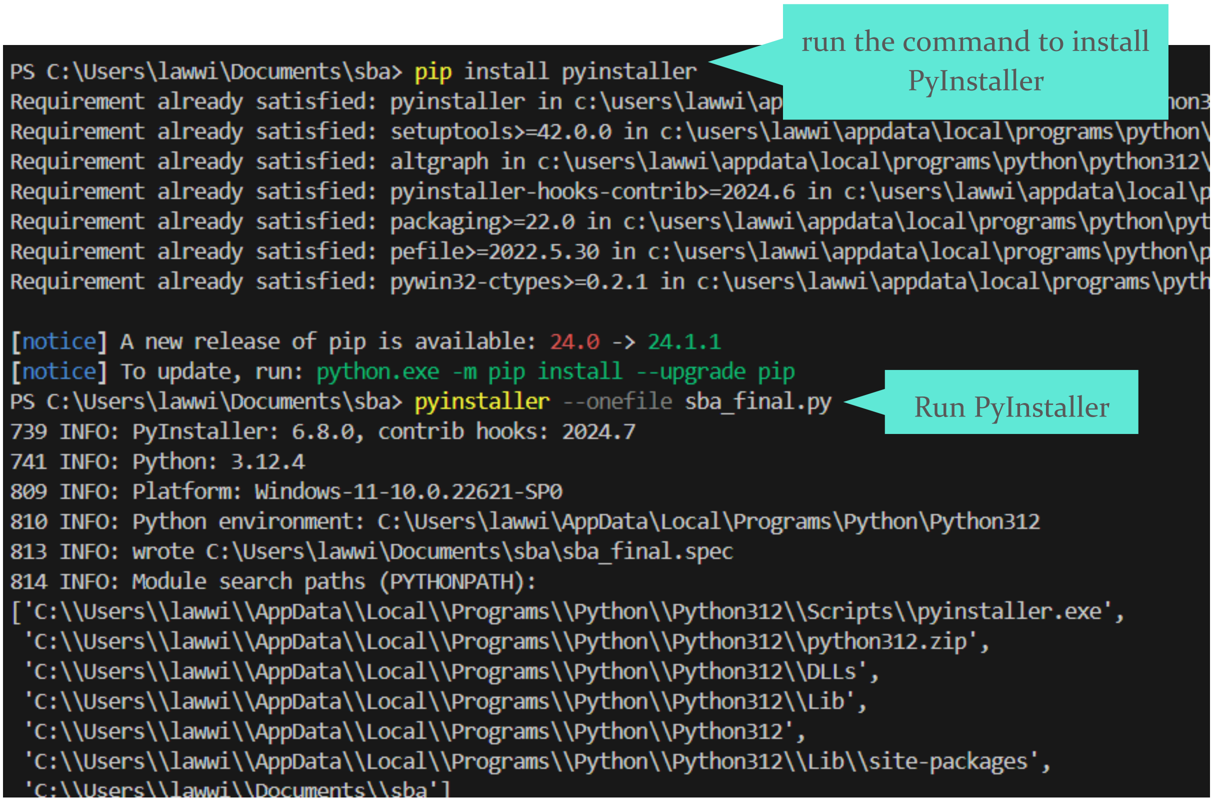
C: > Users > lkmstudent > Documents > sba_decrypt - Copy.py > shift_encrypt

```
def shift_encrypt(message, shift):
    encrypted_message = ""  #initialization

    for char in message:    #check if char is in alphabet
        if char.isalpha():
            encrypted_char = chr((ord(char) - base + shift) % 26 + base)    #shifting +k unit
            encrypted_message += encrypted_char
        else:
            encrypted_message += char   #if the char is not A-Z, the char remains

    return encrypted_message

def shift_decrypt(message):
    decrypted_message = ""

    shift = find_most_common(message)   #return shift number

    for char in message:
        if char.isalpha():
            decrypted_char = chr((ord(char) - base - shift) % 26 + base)    #shifting -k unit
            decrypted_message += decrypted_char
        else:
            decrypted_message += char

    return decrypted message
```

```
PS C:\Users\lkmstudent\Documents>  & 'c:\Users\lkmstudent\AppData\Local\Programs\Python\Python311\python.ex
s\ms-python.debugpy-2024.6.0-win32-x64\bundled\libs\debugpy\adapter/../..\debugpy\launcher' '62573' '--' 'C
- Copy.py'
select mode:
[1]:encrypt
[2]:decrypt
[-1]:end 1

Input the string: HELLO WORLD
Shift number: 8
```

Innovation

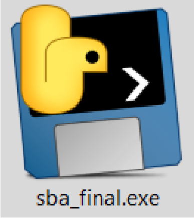
The python file is converted to standalone executable (.exe) files, so that the file can be directly run or executed by computers' operating system. ***PyInstaller***[1] is used to convert .py to .exe

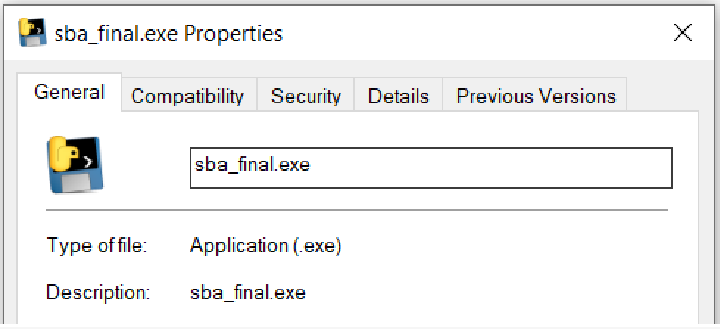


run the command to install PyInstaller

Run PyInstaller

```
PS C:\Users\lawwi\Documents\sba> pip install pyinstaller
Requirement already satisfied: pyinstaller in c:\users\lawwi\ap                                  hon3
Requirement already satisfied: setuptools>=42.0.0 in c:\users\lawwi\appdata\local\programs\python\
Requirement already satisfied: altgraph in c:\users\lawwi\appdata\local\programs\python\python312\
Requirement already satisfied: pyinstaller-hooks-contrib>=2024.6 in c:\users\lawwi\appdata\local\p
Requirement already satisfied: packaging>=22.0 in c:\users\lawwi\appdata\local\programs\python\pyt
Requirement already satisfied: pefile>=2022.5.30 in c:\users\lawwi\appdata\local\programs\python\p
Requirement already satisfied: pywin32-ctypes>=0.2.1 in c:\users\lawwi\appdata\local\programs\pyth

[notice] A new release of pip is available: 24.0 -> 24.1.1
[notice] To update, run: python.exe -m pip install --upgrade pip
PS C:\Users\lawwi\Documents\sba> pyinstaller --onefile sba_final.py
739 INFO: PyInstaller: 6.8.0, contrib hooks: 2024.7
741 INFO: Python: 3.12.4
809 INFO: Platform: Windows-11-10.0.22621-SP0
810 INFO: Python environment: C:\Users\lawwi\AppData\Local\Programs\Python\Python312
813 INFO: wrote C:\Users\lawwi\Documents\sba\sba_final.spec
814 INFO: Module search paths (PYTHONPATH):
['C:\\Users\\lawwi\\AppData\\Local\\Programs\\Python\\Python312\\Scripts\\pyinstaller.exe',
 'C:\\Users\\lawwi\\AppData\\Local\\Programs\\Python\\Python312\\python312.zip',
 'C:\\Users\\lawwi\\AppData\\Local\\Programs\\Python\\Python312\\DLLs',
 'C:\\Users\\lawwi\\AppData\\Local\\Programs\\Python\\Python312\\Lib',
 'C:\\Users\\lawwi\\AppData\\Local\\Programs\\Python\\Python312',
 'C:\\Users\\lawwi\\AppData\\Local\\Programs\\Python\\Python312\\Lib\\site-packages',
 'C:\\Users\\lawwi\\Documents\\sba']
```



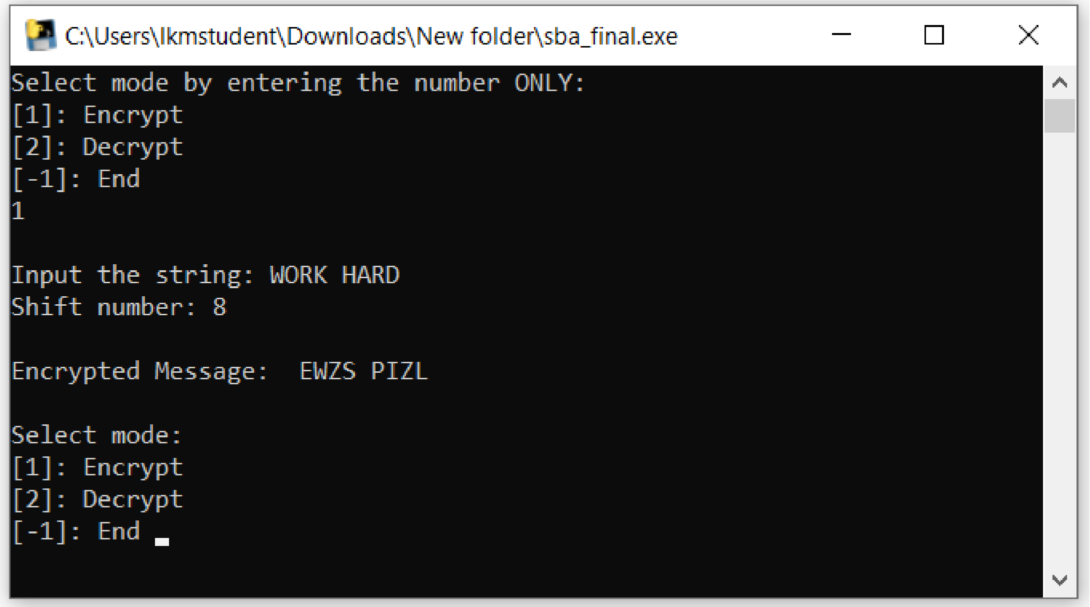


sba_final.exe Properties

sba_final.exe

Type of file: Application (.exe)

Description: sba_final.exe



C:\Users\lkmstudent\Downloads\New folder\sba_final.exe

```
Select mode by entering the number ONLY:
[1]: Encrypt
[2]: Decrypt
[-1]: End
1

Input the string: WORK HARD
Shift number: 8

Encrypted Message:  EWZS PIZL

Select mode:
[1]: Encrypt
[2]: Decrypt
[-1]: End
```

[1] https://pyinstaller.org/en/stable/

Algorithm Optimization

Extend the scope of the program

1. **Allow lowercase Letters**

```
def shift_encrypt(message, shift):
    ...
    for char in message:
        if char.isalpha():
            if char.islower():
                encrypted_char =
                            chr(
                            (ord(char) - ord('a') + shift)
                            % 26 + ord('a')
                            )
            else:
                encrypted_char = chr((ord(char) - ord('A') +
                                      shift)
                % 26 + ord('A'))’
    ...
```

2. **Multiple shift value**

```
shift_values = []
...
if check_upper == True:
    if mode == 1:

        shift = int(input("Shift number: "))

        while shift != -1:
        ...
            print("\nEncrypted Message: ",
                    shift_encrypt(original_message, shift))
            shift = int(input("Shift number: "))
...
```

Conclusion

The development of the program has been an iterative and comprehensive process that involved careful consideration of various aspects. The report also highlights the strengths and areas for improvement in the program.

- - End of TEST AND EVALUATE REPORT - -