```python

# *Version: Tlock 1.7

# *This product is from 'AOSP'

# *pip install speechrecognition cryptography gtts auto_py_to_exe ttkbootstrap PyAudio

from base64 import (b16decode, b16encode, b32decode, b32encode, b32hexdecode,
                    b32hexencode, b64decode, b64encode, b85decode, b85encode,
                    urlsafe_b64decode, urlsafe_b64encode)
from multiprocessing import Process
from os import getcwd, path, system
from threading import Thread
from tkinter import Label, filedialog, messagebox, scrolledtext, simpledialog
from webbrowser import open as open_link

import speech_recognition as sr
from cryptography.fernet import Fernet
from gtts import gTTS
from ttkbootstrap import DISABLED, END, Button, Menu, Window


def help_func():
    open_link("ahmed-omar-software-projects.mydurable.com")


CWD = getcwd()
running_py_path = path.join(CWD, "temp.py")
running_js_path = path.join(CWD, "temp.js")


def main():

    # * Alphabet And Keys
    CHARACTERS = (
        "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",
        "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",
            "!",
            '"',
            "#",
            "$",
            "%",
            "&",
            "'",
            "(",
            ")",
            "*",
            "+",
            ",",
            "-",
            ".",
            "/",
            ":",
            ";",
            "<",
            "=",
            ">",
            "?",
            "@",
            "[",
            "\\",
            "]",
            "^",
            "_",
            "`",
            "{",
            "|",
            "}",
            "~",
            "0",
            "1",
            "2",
            "3",
            "4",
            "5",
            "6",
            "7",
            "8",
            "9",
        )
        KEY_1 = (
            "E",
            "|",
            "9",
```

```
135         "~",
136         ")",
137         "s",
138         "C",
139         "l",
140         "/",
141         "!",
142         "o",
143         "<",
144         "$",
145         "]",
146         "M",
147         "z",
148         "X",
149         "\\",
150         "Z",
151         "5",
152         "Q",
153         "S",
154         "h",
155         "t",
156         "e",
157         "7",
158         "}",
159         "?",
160         "O",
161         "'",
162         "i",
163         "Y",
164         "j",
165         "G",
166         "-",
167         "v",
168         "V",
169         "%",
170         "P",
171         "R",
172         "[",
173         "p",
174         "r",
175         "g",
176         "U",
177         "@",
178         "(",
179         "=",
180         "W",
181         "A",
182         ">",
183         "f",
184         "B",
185         "d",
186         "H",
187         "q",
188         '"',
189         "#",
190         "{",
191         "u",
192         "I",
193         "+",
194         "3",
195         "4",
196         "`",
197         "N",
198         "L",
199         ";",
200         "J",
201         "8",
```

```
            "m",
            "w",
            "6",
            "k",
            "_",
            "-",
            ".",
            "K",
            "b",
            "D",
            "&",
            "0",
            "x",
            ",",
            ":",
            "c",
            "T",
            "^",
            "2",
            "1",
            "F",
            "y",
            "*",
            "a",
            "n",
        )
        KEY_2 = (
            ",",
            "u",
            "i",
            "5",
            ">",
            "6",
            "3",
            "7",
            "L",
            "<",
            "=",
            "H",
            "]",
            "`",
            "E",
            "R",
            "8",
            "\\",
            "4",
            "B",
            "m",
            "0",
            "!",
            "Z",
            "|",
            "^",
            "y",
            "X",
            "O",
            "T",
            "f",
            "M",
            "c",
            "W",
            "N",
            "G",
            ":",
            "p",
            "w",
            "1",
            "Q",
```

```
            "(",
            "q",
            "U",
            "S",
            "C",
            ".",
            "d",
            "&",
            "_",
            "2",
            "s",
            "9",
            "g",
            ")",
            "k",
            "-",
            "A",
            "K",
            "[",
            "e",
            "/",
            "*",
            "?",
            "I",
            "~",
            "l",
            "z",
            "$",
            "j",
            "#",
            "v",
            "D",
            "@",
            "h",
            '"',
            "+",
            "b",
            "J",
            "F",
            "r",
            "V",
            "{",
            "}",
            ";",
            "a",
            "t",
            "P",
            "o",
            "%",
            "x",
            "'",
            "Y",
            "n",
        )
        KEY_3 = (
            "6",
            "i",
            "W",
            "=",
            "F",
            "$",
            "T",
            "8",
            "C",
            "+",
            "!",
            "l",
```

```
336         "G",
337         "<",
338         "a",
339         "(",
340         "J",
341         "b",
342         "K",
343         "2",
344         "Z",
345         "7",
346         "Q",
347         "M",
348         "f",
349         "_",
350         "o",
351         "P",
352         "5",
353         "j",
354         "r",
355         "@",
356         "m",
357         "-",
358         "p",
359         ";",
360         "X",
361         "x",
362         ".",
363         "R",
364         "[",
365         "A",
366         ":",
367         "S",
368         "?",
369         "O",
370         "V",
371         "U",
372         "|",
373         "w",
374         "Y",
375         ">",
376         "c",
377         "N",
378         "z",
379         "e",
380         "D",
381         "}",
382         "*",
383         ")",
384         "v",
385         "s",
386         "H",
387         "`",
388         "%",
389         "L",
390         "/",
391         "y",
392         "3",
393         "q",
394         "I",
395         "n",
396         "9",
397         "^",
398         "#",
399         "d",
400         "]",
401         "\\",
402         "0",
```

```
            '"',
            "g",
            "t",
            "1",
            "u",
            "E",
            "B",
            "4",
            "'",
            "k",
            "&",
            "h",
            ",",
            "~",
            "{",
        )
        KEY_4 = (
            "1",
            ".",
            "&",
            "q",
            "Q",
            "l",
            "m",
            "Z",
            "Y",
            ":",
            "'",
            "C",
            "|",
            "i",
            "A",
            "r",
            "O",
            "}",
            "5",
            "x",
            "J",
            ">",
            "d",
            "K",
            "!",
            ")",
            "]",
            "F",
            "V",
            "j",
            "@",
            "t",
            "4",
            "E",
            "[",
            "p",
            "7",
            "$",
            "h",
            "c",
            "M",
            "6",
            "/",
            '"',
            ",",
            "`",
            "9",
            "\\",
            "3",
            "s",
```

```
            "D",
            "S",
            "<",
            "=",
            "b",
            ";",
            "*",
            "k",
            "2",
            "N",
            "L",
            "I",
            "e",
            "H",
            "X",
            "8",
            "{",
            "w",
            "z",
            "W",
            "P",
            "U",
            "0",
            "?",
            "%",
            "v",
            "g",
            "n",
            "R",
            "o",
            "-",
            "T",
            "_",
            "G",
            "^",
            "~",
            "a",
            "+",
            "u",
            "Y",
            "(",
            "B",
            "#",
            "f",
        )
        KEY_5 = (
            "O",
            "p",
            "/",
            "1",
            "X",
            "3",
            ">",
            "5",
            "Q",
            "n",
            "T",
            "}",
            "@",
            "'",
            "#",
            "&",
            "P",
            "=",
            "g",
            "b",
            "d",
```

```
        "C",
        "f",
        "U",
        "A",
        "D",
        "%",
        "q",
        "G",
        "K",
        "\\",
        "`",
        "!",
        "|",
        "k",
        "S",
        "R",
        "(",
        "y",
        "6",
        "^",
        "Y",
        ")",
        "*",
        "j",
        "z",
        "l",
        "J",
        "N",
        "c",
        "W",
        "-",
        "e",
        "M",
        "2",
        "<",
        "h",
        "]",
        "?",
        ":",
        '"',
        "7",
        "9",
        "H",
        "F",
        "w",
        "L",
        "Z",
        "s",
        "$",
        "x",
        "B",
        "0",
        "[",
        "I",
        "{",
        "m",
        "_",
        "r",
        "E",
        ".",
        "t",
        "~",
        "a",
        "8",
        "i",
        "+",
        "u",
```

```
            "v",
            "o",
            ",",
            "V",
            "4",
            ";",
        )
        KEY_6 = (
            ">",
            "S",
            "<",
            "B",
            "O",
            "[",
            "Y",
            "7",
            "0",
            "?",
            '"',
            "*",
            "X",
            "W",
            "i",
            "D",
            "8",
            "6",
            "4",
            "'",
            "r",
            "j",
            "f",
            "+",
            "c",
            "v",
            "P",
            "_",
            "V",
            "H",
            "d",
            "q",
            "T",
            "R",
            "w",
            "h",
            "M",
            "Z",
            "J",
            "y",
            ".",
            ",",
            "K",
            "u",
            "9",
            "&",
            "`",
            "#",
            "s",
            "(",
            ":",
            "e",
            "p",
            "\\",
            "^",
            "U",
            "n",
            "C",
            "}",
```

```
671             "G",
672             "-",
673             "g",
674             "z",
675             "Q",
676             "=",
677             "N",
678             "$",
679             ";",
680             "2",
681             "]",
682             "|",
683             "E",
684             "~",
685             "k",
686             "L",
687             "!",
688             "3",
689             "1",
690             "b",
691             "/",
692             "x",
693             "t",
694             "5",
695             "l",
696             ")",
697             "%",
698             "F",
699             "A",
700             "o",
701             "a",
702             "{",
703             "@",
704             "m",
705             "I",
706         )
707         KEY_7 = (
708             "y",
709             ":",
710             "q",
711             "S",
712             "0",
713             "+",
714             "W",
715             "w",
716             "|",
717             "m",
718             "e",
719             "%",
720             "F",
721             ";",
722             "\\",
723             "R",
724             "J",
725             "I",
726             "a",
727             "i",
728             "G",
729             "/",
730             "U",
731             "8",
732             "1",
733             "x",
734             "~",
735             "B",
736             "v",
737             "#",
```

```
            "Z",
            "?",
            "A",
            "^",
            "@",
            "X",
            ",",
            "7",
            "o",
            "s",
            '"',
            ".",
            "<",
            "5",
            "C",
            "[",
            "*",
            "$",
            "&",
            "g",
            "j",
            ">",
            "4",
            "b",
            "-",
            "H",
            "_",
            "r",
            "z",
            "E",
            "6",
            "T",
            "f",
            "d",
            "9",
            "(",
            "{",
            "h",
            "D",
            ")",
            "Q",
            "O",
            "`",
            "N",
            "2",
            "p",
            "l",
            "V",
            "K",
            "k",
            "t",
            "c",
            "]",
            "}",
            "=",
            "'",
            "n",
            "L",
            "Y",
            "M",
            "P",
            "3",
            "!",
            "u",
        )
        KEY_8 = (
            ">",
```

```
        '"',
        "w",
        "+",
        ")",
        "&",
        "Y",
        "D",
        "(",
        "M",
        "Z",
        ".",
        "v",
        "r",
        "*",
        "T",
        "[",
        "s",
        "e",
        "n",
        "K",
        "0",
        "C",
        "5",
        "f",
        "L",
        "/",
        "I",
        "l",
        "`",
        "?",
        "W",
        "m",
        "x",
        "p",
        "3",
        "H",
        ",",
        "$",
        "A",
        "B",
        "<",
        "S",
        "X",
        "{",
        "]",
        "!",
        ":",
        "4",
        "R",
        "6",
        "O",
        "a",
        "J",
        "|",
        "q",
        "@",
        "7",
        "}",
        "g",
        "_",
        "^",
        "%",
        "N",
        "\\",
        "Y",
        "1",
        "k",
```

```
            "E",
            "'",
            "c",
            "-",
            "P",
            "o",
            "Q",
            "~",
            "F",
            "2",
            "#",
            "z",
            "G",
            "h",
            ";",
            "9",
            "d",
            "U",
            "u",
            "t",
            "j",
            "i",
            "8",
            "b",
            "V",
            "=",
        )
        KEY_9 = (
            "g",
            "9",
            "z",
            "n",
            "w",
            "p",
            "[",
            "=",
            "J",
            "a",
            "}",
            "t",
            "Q",
            "Y",
            "V",
            "_",
            "G",
            "^",
            "2",
            "{",
            "k",
            "u",
            "S",
            ";",
            "0",
            "L",
            "A",
            "-",
            "8",
            "1",
            "%",
            "|",
            "\\",
            "i",
            "'",
            "U",
            "f",
            "+",
            "D",
```

```
            "(",
            "<",
            "O",
            "r",
            "C",
            "P",
            "o",
            "R",
            "H",
            "c",
            "#",
            "6",
            "4",
            "x",
            '"',
            "h",
            ":",
            "3",
            "5",
            ",",
            "!",
            "&",
            "N",
            "j",
            "$",
            "@",
            "I",
            "F",
            "m",
            "q",
            ")",
            "B",
            ".",
            "s",
            "e",
            "]",
            "E",
            "d",
            "X",
            "Z",
            "/",
            "M",
            "K",
            "W",
            "?",
            "T",
            "y",
            "l",
            ">",
            "v",
            "7",
            "`",
            "b",
            "*",
            "~",
        )
        KEY_10 = (
            "n",
            "f",
            ">",
            "p",
            "3",
            "F",
            "\\",
            ":",
            "D",
            "h",
```

```
1006          "`",
1007          ")",
1008          "(",
1009          "<",
1010          "O",
1011          "R",
1012          "P",
1013          "u",
1014          "y",
1015          "8",
1016          "I",
1017          "!",
1018          "2",
1019          "*",
1020          "t",
1021          "V",
1022          "Y",
1023          "&",
1024          "/",
1025          "S",
1026          "L",
1027          "+",
1028          "5",
1029          ".",
1030          "k",
1031          "m",
1032          "G",
1033          "{",
1034          "Z",
1035          "w",
1036          "0",
1037          "A",
1038          "e",
1039          ";",
1040          "E",
1041          "U",
1042          "q",
1043          "[",
1044          "N",
1045          "s",
1046          "^",
1047          "x",
1048          "C",
1049          "=",
1050          "|",
1051          "g",
1052          "J",
1053          "c",
1054          "~",
1055          "b",
1056          ",",
1057          "$",
1058          "T",
1059          "]",
1060          "9",
1061          "Q",
1062          "a",
1063          "r",
1064          "4",
1065          "H",
1066          "v",
1067          "@",
1068          "-",
1069          "j",
1070          "i",
1071          "X",
1072          "6",
```

```
            "7",
            "}",
            "B",
            "?",
            "#",
            "M",
            "z",
            '"',
            "d",
            "W",
            "'",
            "_",
            "K",
            "o",
            "%",
            "l",
            "1",
        )

        def reverser(text):
            return text[::-1]

        def base16_encode(text):
            return b16encode(text.encode("utf-8")).decode("utf-8")

        def base16_decode(text):
            return b16decode(text.encode("utf-8")).decode("utf-8")

        def base32_encode(text):
            return b32encode(text.encode("utf-8")).decode("utf-8")

        def base32_decode(text):
            return b32decode(text.encode("utf-8")).decode("utf-8")

        def base64_encode(text):
            return b64encode(text.encode("utf-8")).decode("utf-8")

        def base64_decode(text):
            return b64decode(text.encode("utf-8")).decode("utf-8")

        def base85_encode(text):
            return b85encode(text.encode("utf-8")).decode("utf-8")

        def base85_decode(text):
            return b85decode(text.encode("utf-8")).decode("utf-8")

        def b32hex_encode(text):
            return b32hexencode(text.encode("utf-8")).decode("utf-8")

        def b32hex_decode(text):
            return b32hexdecode(text.encode("utf-8")).decode("utf-8")

        def b64_urlsafe_encode(text):
            return urlsafe_b64encode(text.encode("utf-8")).decode("utf-8")

        def b64_urlsafe_decode(text):
            return urlsafe_b64decode(text.encode("utf-8")).decode("utf-8")

        def mac1_encode(text):
            result = []
            for char in text:
                if char in CHARACTERS:
                    index = CHARACTERS.index(char)
                    result.append(KEY_1[index])
                else:
                    result.append(char)
            return "".join(result)
```

```
    def mac1_decode(text):
        result = []
        for char in text:
            if char in KEY_1:
                index = KEY_1.index(char)
                result.append(CHARACTERS[index])
            else:
                result.append(char)
        return "".join(result)

    def mac2_encode(text):
        result = []
        for char in text:
            if char in CHARACTERS:
                index = CHARACTERS.index(char)
                result.append(KEY_2[index])
            else:
                result.append(char)
        return "".join(result)

    def mac2_decode(text):
        result = []
        for char in text:
            if char in KEY_2:
                index = KEY_2.index(char)
                result.append(CHARACTERS[index])
            else:
                result.append(char)
        return "".join(result)

    def mac3_encode(text):
        result = []
        for char in text:
            if char in CHARACTERS:
                index = CHARACTERS.index(char)
                result.append(KEY_3[index])
            else:
                result.append(char)
        return "".join(result)

    def mac3_decode(text):
        result = []
        for char in text:
            if char in KEY_3:
                index = KEY_3.index(char)
                result.append(CHARACTERS[index])
            else:
                result.append(char)
        return "".join(result)

    def mac4_encode(text):
        result = []
        for char in text:
            if char in CHARACTERS:
                index = CHARACTERS.index(char)
                result.append(KEY_4[index])
            else:
                result.append(char)
        return "".join(result)

    def mac4_decode(text):
        result = []
        for char in text:
            if char in KEY_4:
                index = KEY_4.index(char)
                result.append(CHARACTERS[index])
```

```
            else:
                result.append(char)
        return "".join(result)

    def mac5_encode(text):
        result = []
        for char in text:
            if char in CHARACTERS:
                index = CHARACTERS.index(char)
                result.append(KEY_5[index])
            else:
                result.append(char)
        return "".join(result)

    def mac5_decode(text):
        result = []
        for char in text:
            if char in KEY_5:
                index = KEY_5.index(char)
                result.append(CHARACTERS[index])
            else:
                result.append(char)
        return "".join(result)

    def mac6_encode(text):
        result = []
        for char in text:
            if char in CHARACTERS:
                index = CHARACTERS.index(char)
                result.append(KEY_6[index])
            else:
                result.append(char)
        return "".join(result)

    def mac6_decode(text):
        result = []
        for char in text:
            if char in KEY_6:
                index = KEY_6.index(char)
                result.append(CHARACTERS[index])
            else:
                result.append(char)
        return "".join(result)

    def mac7_encode(text):
        result = []
        for char in text:
            if char in CHARACTERS:
                index = CHARACTERS.index(char)
                result.append(KEY_7[index])
            else:
                result.append(char)
        return "".join(result)

    def mac7_decode(text):
        result = []
        for char in text:
            if char in KEY_7:
                index = KEY_7.index(char)
                result.append(CHARACTERS[index])
            else:
                result.append(char)
        return "".join(result)

    def mac8_encode(text):
        result = []
        for char in text:
```

```
            if char in CHARACTERS:
                index = CHARACTERS.index(char)
                result.append(KEY_8[index])
            else:
                result.append(char)
        return "".join(result)

    def mac8_decode(text):
        result = []
        for char in text:
            if char in KEY_8:
                index = KEY_8.index(char)
                result.append(CHARACTERS[index])
            else:
                result.append(char)
        return "".join(result)

    def mac9_encode(text):
        result = []
        for char in text:
            if char in CHARACTERS:
                index = CHARACTERS.index(char)
                result.append(KEY_9[index])
            else:
                result.append(char)
        return "".join(result)

    def mac9_decode(text):
        result = []
        for char in text:
            if char in KEY_9:
                index = KEY_9.index(char)
                result.append(CHARACTERS[index])
            else:
                result.append(char)
        return "".join(result)

    def mac10_encode(text):
        result = []
        for char in text:
            if char in CHARACTERS:
                index = CHARACTERS.index(char)
                result.append(KEY_10[index])
            else:
                result.append(char)
        return "".join(result)

    def mac10_decode(text):
        result = []
        for char in text:
            if char in KEY_10:
                index = KEY_10.index(char)
                result.append(CHARACTERS[index])
            else:
                result.append(char)
        return "".join(result)

    def first_main_encryption(text: str):
        layer1 = mac5_encode(text)
        layer2 = reverser(layer1)
        layer3 = mac4_encode(layer2)
        layer4 = mac2_encode(layer3)
        layer5 = mac1_encode(layer4)
        layer6 = mac3_encode(layer5)
        layer7 = mac8_encode(layer6)
        layer8 = mac7_encode(layer7)
        layer9 = mac9_encode(layer8)
```

```
            layer10 = mac6_encode(layer9)
            layer11 = mac8_encode(layer10)
            return str(mac10_encode(layer11))

        def first_main_decryption(text: str):
            layer1 = mac10_decode(text)
            layer2 = mac8_decode(layer1)
            layer3 = mac6_decode(layer2)
            layer4 = mac9_decode(layer3)
            layer5 = mac7_decode(layer4)
            layer6 = mac8_decode(layer5)
            layer7 = mac3_decode(layer6)
            layer8 = mac1_decode(layer7)
            layer9 = mac2_decode(layer8)
            layer10 = mac4_decode(layer9)
            layer11 = reverser(layer10)
            return str(mac5_decode(layer11))

        def second_main_encryption(text: str):
            layer1 = mac5_encode(text)
            layer2 = reverser(layer1)
            layer3 = mac4_encode(layer2)
            layer4 = mac2_encode(layer3)
            layer5 = mac1_encode(layer4)
            layer6 = mac3_encode(layer5)
            layer7 = base16_encode(layer6)
            layer8 = base85_encode(layer7)
            layer9 = base64_encode(layer8)
            layer10 = b64_urlsafe_encode(layer9)
            layer11 = base32_encode(layer10)
            return str(b32hex_encode(layer11))

        def second_main_decryption(text: str):
            layer1 = b32hex_decode(text)
            layer2 = base32_decode(layer1)
            layer3 = b64_urlsafe_decode(layer2)
            layer4 = base64_decode(layer3)
            layer5 = base85_decode(layer4)
            layer6 = base16_decode(layer5)
            layer7 = mac3_decode(layer6)
            layer8 = mac1_decode(layer7)
            layer9 = mac2_decode(layer8)
            layer10 = mac4_decode(layer9)
            layer11 = reverser(layer10)
            return str(mac5_decode(layer11))

        def encrypt(text):
            return str(
                first_main_encryption(
                    first_main_encryption(
                        first_main_encryption(second_main_encryption(text))
                    )
                )
            )

        def decrypt(text):
            return str(
                second_main_decryption(
                    first_main_decryption(
                        first_main_decryption(first_main_decryption(text))
                    )
                )
            )

        # *File management func.

        def save_encrypted_text():
```

```
        password = simpledialog.askstring(
            "Password", "Please, type the password for the encryption:", show="*"
        )
        if password:
            while True:
                tlide_path = filedialog.asksaveasfilename(
                    title="Save the Tlock encrypted text file",
                    initialdir=CWD,
                    filetypes=(
                        ("Tlock encrypted text files", "*.tetf"),
                        ("All files", "*.*"),
                    ),
                )
                if not tlide_path:
                    messagebox.showwarning(
                        "Warning", "No file selected. Operation cancelled."
                    )
                    break
                content = text_input.get("1.0", END)

                encrypted_content = encrypt(content)
                final_path = tlide_path + ".tetf"
                with open(final_path, "w", encoding="utf-8") as wrapper:
                    wrapper.write(mac1_encode(password) + "\n")
                    wrapper.write(encrypted_content)
                    messagebox.showinfo(
                        "Info", "The file is created successfully.")
                    break
        if not password:
            messagebox.showwarning(
                "Warning", "No password entered. Operation cancelled.")

        if __name__ == "__tlock__":
            create_process = Process(target=save_encrypted_text)
            create_process.start()
            create_process.join()

            create_thread = Thread(target=save_encrypted_text)
            create_thread.start()
            create_thread.join()

    def open_decrypted_text():
        try:
            tlide_path = filedialog.askopenfilename(
                title="Open a Tlock encrypted text file",
                initialdir=CWD,
                filetypes=(
                    ("Tlock encrypted text files", "*.tetf"),
                    ("All files", "*.*"),
                ),
            )
            if not tlide_path:
                messagebox.showwarning(
                    "Warning", "No file selected. Operation cancelled."
                )
            password = simpledialog.askstring(
                "Password", "Enter the password for the decryption:", show="*"
            ).strip()
            with open(tlide_path, "r", encoding="utf-8") as wrapper:
                file_pass = wrapper.readline().strip()
                if mac1_decode(file_pass) != password:
                    messagebox.showerror(
                        "Error", "The password is incorrect. Operation cancelled."
                    )
                if mac1_decode(file_pass) == password:

                    encrypted_content = wrapper.read()
```

```
                    decrypted_content = decrypt(encrypted_content)
                    decryption_displaying_screen = Window(
                        themename="superhero")
                    decryption_displaying_screen.title("Decrypted Content")
                    decryption_displaying_screen.state("zoomed")

                    open_button = Button(
                        decryption_displaying_screen,
                        text="Text to Speech",
                        width=188,
                        command=lambda: read(decrypted_content),
                    )
                    open_button.place(x=5, y=5)

                    text_widget = scrolledtext.ScrolledText(
                        decryption_displaying_screen, font=("Arial", 11)
                    )
                    text_widget.place(x=5, y=50)

                    text_widget.insert("1.0", decrypted_content)
                    text_widget.config(
                        state=DISABLED,
                        font=("default", 11),
                        width=155,
                        height=36,
                    )

                    decryption_displaying_screen.mainloop()
            except FileNotFoundError:
                pass
            if __name__ == "__tlock__":
                open_process = Process(target=open_decrypted_text)
                open_process.start()
                open_process.join()

                open_thread = Thread(target=open_decrypted_text)
                open_thread.start()
                open_thread.join()

        def save_encrypted_image():
            KEY = Fernet.generate_key()
            cipher = Fernet(KEY)

            def encrypt_file(file):
                return bytes(
                    cipher.encrypt(
                        cipher.encrypt(
                            cipher.encrypt(
                                cipher.encrypt(
                                    cipher.encrypt(
                                        cipher.encrypt(
                                            cipher.encrypt(
                                                cipher.encrypt(
                                                    cipher.encrypt(
                                                        cipher.encrypt(
                                                            file)
                                                    )
                                                )
                                            )
                                        )
                                    )
                                )
                            )
                        )
                    )
```

```
                    )
                )
            )

        while True:
            image_path = filedialog.askopenfilename(
                title="Open an image",
                filetypes=(
                    ("JPG files", "*.jpg"),
                    ("PNG files", "*.png"),
                    ("All files", "*.*"),
                ),
                initialdir=CWD,
            )
            if not image_path:
                messagebox.showwarning(
                    "Warning", "No file selected. Operation cancelled."
                )
                break
            encrypted_path = filedialog.asksaveasfilename(
                title="Save encrypted image",
                filetypes=(
                    ("Tlock Encrypted Image File", "*.teif"),
                    ("All files", "*.*"),
                ),
                initialdir=CWD,
            )

            if image_path and encrypted_path:
                with open(image_path, "rb") as file:
                    date = file.read()
                    encrypted_data = encrypt_file(date)
                with open(encrypted_path + ".teif", "wb") as file:
                    file.write(encrypted_data)
                    messagebox.showinfo(
                        "Info", "The file is encrypted successfully.")
                    messagebox.showinfo(
                        "Info", f"The key for decryption: {KEY.decode("utf-8")}"
                    )

                    window.clipboard_clear()
                    window.clipboard_append(KEY.decode("utf-8"))
                    break

    def open_decrypted_image():
        while True:
            key = simpledialog.askstring(
                "Key", "Please, the key for the decryption:", show="*"
            )
            if key:
                cipher = Fernet(key)

                def decrypt_file(file):
                    return bytes(
                        cipher.decrypt(
                            cipher.decrypt(
                                cipher.decrypt(
                                    cipher.decrypt(
                                        cipher.decrypt(
                                            cipher.decrypt(
                                                cipher.decrypt(
                                                    cipher.decrypt(
                                                        cipher.decrypt(
                                                            cipher.decrypt(
                                                                file)
```

```
                                                                    )
                                                                )
                                                            )
                                                        )
                                                    )
                                                )
                                            )
                                        )
                                    )
                                )
                            )

                        saving_path = filedialog.askopenfilename(
                            title="Open encrypted image",
                            filetypes=(
                                ("Tlock Encrypted Image File", "*.teif"),
                                ("All files", "*.*"),
                            ),
                            initialdir=CWD,
                        )
                        if not saving_path:
                            messagebox.showwarning(
                                "Warning", "No file selected. Operation cancelled."
                            )
                            break
                        if saving_path:
                            with open(saving_path, "rb") as file:
                                encrypted_data = file.read()

                                decrypted_data = decrypt_file(encrypted_data)

                                with open(saving_path + ".jpg", "wb") as wrapper:
                                    wrapper.write(decrypted_data)
                                    messagebox.showinfo(
                                        "Info", "The file is decrypted successfully."
                                    )
                                    break
                    if not key:
                        messagebox.showwarning(
                            "Warning", "No key entered. Operation cancelled."
                        )
                        break

    def save_encrypted_audio():
        KEY = Fernet.generate_key()
        cipher = Fernet(KEY)

        def encrypt_file(file):
            return bytes(
                cipher.encrypt(
                    cipher.encrypt(
                        cipher.encrypt(
                            cipher.encrypt(
                                cipher.encrypt(
                                    cipher.encrypt(
                                        cipher.encrypt(
                                            cipher.encrypt(
                                                cipher.encrypt(
                                                    cipher.encrypt(
                                                        file)
                                                )
                                            )
                                        )
                                    )
                                )
                            )
                        )
                    )
                )
            )
```

```
                                        )
                                    )
                                )
                            )
                        )
                    )
                )
            )

        while True:
            audio_path = filedialog.askopenfilename(
                title="Open an Audio",
                filetypes=(
                    ("Mp3 files", "*.mp3"),
                    ("All files", "*.*"),
                ),
                initialdir=CWD,
            )
            if not audio_path:
                messagebox.showwarning(
                    "Warning", "No file selected. Operation cancelled."
                )
                break
            encrypted_path = filedialog.asksaveasfilename(
                title="Save encrypted Audio",
                filetypes=(
                    ("Tlock Encrypted Audio File", "*.teaf"),
                    ("All files", "*.*"),
                ),
                initialdir=CWD,
            )

            if audio_path and encrypted_path:
                with open(audio_path, "rb") as file:
                    date = file.read()
                    encrypted_data = encrypt_file(date)
                with open(encrypted_path + ".teaf", "wb") as file:
                    file.write(encrypted_data)
                    messagebox.showinfo(
                        "Info", "The file is encrypted successfully.")
                    messagebox.showinfo(
                        "Info",
                        f"The key for decryption: {KEY.decode("utf-8")}. The key is
                        copied to the clipboard.",
                    )

                    window.clipboard_clear()
                    window.clipboard_append(KEY.decode("utf-8"))
                    break

    def open_decrypted_audio():
        while True:
            key = simpledialog.askstring(
                "Key", "Please, the key for the decryption:", show="*"
            )
            if key:
                cipher = Fernet(key)

                def decrypt_file(file):
                    return bytes(
                        cipher.decrypt(
                            cipher.decrypt(
                                cipher.decrypt(
                                    cipher.decrypt(
                                        cipher.decrypt(
                                            cipher.decrypt(
```

```
                                                    cipher.decrypt(
                                                        cipher.decrypt(
                                                            cipher.decrypt(
                                                                cipher.decrypt(
                                                                    cipher.decrypt(
                                                                        file)
                                                                )
                                                            )
                                                        )
                                                    )
                                                )
                                            )
                                        )
                                    )
                                )
                            )
                        )

                    saving_path = filedialog.askopenfilename(
                        title="Open encrypted audio",
                        filetypes=(
                            ("Tlock Encrypted Audio File", "*.teaf"),
                            ("All files", "*.*"),
                        ),
                        initialdir=CWD,
                    )
                    if not saving_path:
                        messagebox.showwarning(
                            "Warning", "No file selected. Operation cancelled."
                        )
                        break
                    if saving_path:
                        with open(saving_path, "rb") as file:
                            encrypted_data = file.read()

                            decrypted_data = decrypt_file(encrypted_data)

                            with open(saving_path + ".mp3", "wb") as wrapper:
                                wrapper.write(decrypted_data)
                                messagebox.showinfo(
                                    "Info", "The file is decrypted successfully."
                                )
                                break
                if not key:
                    messagebox.showwarning(
                        "Warning", "No key entered. Operation cancelled."
                    )
                    break

    def save_encrypted_video():
        KEY = Fernet.generate_key()
        cipher = Fernet(KEY)

        def encrypt_file(file):
            return bytes(
                cipher.encrypt(
                    cipher.encrypt(
                        cipher.encrypt(
                            cipher.encrypt(
                                cipher.encrypt(
                                    cipher.encrypt(
                                        cipher.encrypt(
                                            cipher.encrypt(
                                                cipher.encrypt(
                                                    cipher.encrypt(
                                                        cipher.encrypt(
```

```
                                                                cipher.encrypt(
                                                                    file)
                                                            )
                                                        )
                                                    )
                                                )
                                            )
                                        )
                                    )
                                )
                            )
                        )
                    )

            while True:
                video_path = filedialog.askopenfilename(
                    title="Open a Video",
                    filetypes=(
                        ("Mp4 files", "*.Mp4"),
                        ("All files", "*.*"),
                    ),
                    initialdir=CWD,
                )
                if not video_path:
                    messagebox.showwarning(
                        "Warning", "No file selected. Operation cancelled."
                    )
                    break
                encrypted_path = filedialog.asksaveasfilename(
                    title="Save encrypted video",
                    filetypes=(
                        ("Tlock Encrypted Video File", "*.tevf"),
                        ("All files", "*.*"),
                    ),
                    initialdir=CWD,
                )

                if video_path and encrypted_path:
                    with open(video_path, "rb") as file:
                        date = file.read()
                        encrypted_data = encrypt_file(date)
                    with open(encrypted_path + ".teif", "wb") as file:
                        file.write(encrypted_data)
                        messagebox.showinfo(
                            "Info", "The file is encrypted successfully.")
                        messagebox.showinfo(
                            "Info",
                            f"The key for decryption: {KEY.decode("utf-8")}. The key is
                            copied to the clipboard.",
                        )

                        window.clipboard_clear()
                        window.clipboard_append(KEY.decode("utf-8"))
                        break

        def open_decrypted_video():
            while True:
                key = simpledialog.askstring(
                    "Key", "Please, the key for the decryption:", show="*"
                )
                if key:
                    cipher = Fernet(key)

                    def decrypt_file(file):
                        return bytes(
                            cipher.decrypt(
```

```
                                cipher.decrypt(
                                    cipher.decrypt(
                                        cipher.decrypt(
                                            cipher.decrypt(
                                                cipher.decrypt(
                                                    cipher.decrypt(
                                                        cipher.decrypt(
                                                            cipher.decrypt(
                                                                cipher.decrypt(
                                                                    cipher.decrypt(
                                                                        file)
                                                                )
                                                            )
                                                        )
                                                    )
                                                )
                                            )
                                        )
                                    )
                                )
                            )
                        )

                    saving_path = filedialog.askopenfilename(
                        title="Open encrypted video",
                        filetypes=(
                            ("Tlock Encrypted Video File", "*.tevf"),
                            ("All files", "*.*"),
                        ),
                        initialdir=CWD,
                    )
                    if not saving_path:
                        messagebox.showwarning(
                            "Warning", "No file selected. Operation cancelled."
                        )
                        break
                    if saving_path:
                        with open(saving_path, "rb") as file:
                            encrypted_data = file.read()

                            decrypted_data = decrypt_file(encrypted_data)

                            with open(saving_path + ".mp4", "wb") as wrapper:
                                wrapper.write(decrypted_data)
                                messagebox.showinfo(
                                    "Info", "The file is decrypted successfully."
                                )
                                break
                if not key:
                    messagebox.showwarning(
                        "Warning", "No key entered. Operation cancelled."
                    )
                    break

        def save_encrypted_docs():
            KEY = Fernet.generate_key()
            cipher = Fernet(KEY)

            def encrypt_file(file):
                return bytes(
                    cipher.encrypt(
                        cipher.encrypt(
                            cipher.encrypt(
                                cipher.encrypt(
                                    cipher.encrypt(
```

```
                                        cipher.encrypt(
                                            cipher.encrypt(
                                                cipher.encrypt(
                                                    cipher.encrypt(
                                                        cipher.encrypt(
                                                            cipher.encrypt(
                                                                cipher.encrypt(
                                                                    file)
                                                            )
                                                        )
                                                    )
                                                )
                                            )
                                        )
                                    )
                                )
                            )
                        )
                    )

        while True:
            doc_path = filedialog.askopenfilename(
                title="Open a Document",
                filetypes=(
                    ("Word Document", "*.docx"),
                    ("Word 97-2003 Document", "*.doc"),
                    ("All files", "*.*"),
                ),
                initialdir=CWD,
            )
            if not doc_path:
                messagebox.showwarning(
                    "Warning", "No file selected. Operation cancelled."
                )
                break
            encrypted_path = filedialog.asksaveasfilename(
                title="Save encrypted document",
                filetypes=(
                    ("Tlock Encrypted Document File", "*.tedf"),
                    ("All files", "*.*"),
                ),
                initialdir=CWD,
            )

            if doc_path and encrypted_path:
                with open(doc_path, "rb") as file:
                    date = file.read()
                    encrypted_data = encrypt_file(date)
                with open(encrypted_path + ".tedf", "wb") as file:
                    file.write(encrypted_data)
                    messagebox.showinfo(
                        "Info", "The file is encrypted successfully.")
                    messagebox.showinfo(
                        "Info", f"The key for decryption: {KEY.decode("utf-8")}"
                    )

                    window.clipboard_clear()
                    window.clipboard_append(KEY.decode("utf-8"))
                    break

    def open_decrypted_docs():
        while True:
            key = simpledialog.askstring(
                "Key", "Please, the key for the decryption:", show="*"
            )
            if key:
```

```
                cipher = Fernet(key)

                def decrypt_file(file):
                    return bytes(
                        cipher.decrypt(
                            cipher.decrypt(
                                cipher.decrypt(
                                    cipher.decrypt(
                                        cipher.decrypt(
                                            cipher.decrypt(
                                                cipher.decrypt(
                                                    cipher.decrypt(
                                                        cipher.decrypt(
                                                            cipher.decrypt(
                                                                cipher.decrypt(
                                                                    file)
                                                                )
                                                            )
                                                        )
                                                    )
                                                )
                                            )
                                        )
                                    )
                                )
                            )
                        )

                saving_path = filedialog.askopenfilename(
                    title="Open encrypted document",
                    filetypes=(
                        ("Tlock Encrypted Document File", "*.tedf"),
                        ("All files", "*.*"),
                    ),
                    initialdir=CWD,
                )
                if not saving_path:
                    messagebox.showwarning(
                        "Warning", "No file selected. Operation cancelled."
                    )
                    break
                if saving_path:
                    with open(saving_path, "rb") as file:
                        encrypted_data = file.read()

                        decrypted_data = decrypt_file(encrypted_data)

                        with open(saving_path + ".docx", "wb") as wrapper:
                            wrapper.write(decrypted_data)
                            messagebox.showinfo(
                                "Info", "The file is decrypted successfully."
                            )
                            break
            if not key:
                messagebox.showwarning(
                    "Warning", "No key entered. Operation cancelled."
                )
                break

    def save_encrypted_pptx():
        KEY = Fernet.generate_key()
        cipher = Fernet(KEY)

        def encrypt_file(file):
            return bytes(
```

```
                cipher.encrypt(
                    cipher.encrypt(
                        cipher.encrypt(
                            cipher.encrypt(
                                cipher.encrypt(
                                    cipher.encrypt(
                                        cipher.encrypt(
                                            cipher.encrypt(
                                                cipher.encrypt(
                                                    cipher.encrypt(
                                                        cipher.encrypt(
                                                            file)
                                                    )
                                                )
                                            )
                                        )
                                    )
                                )
                            )
                        )
                    )
                )

        while True:
            pptx_path = filedialog.askopenfilename(
                title="Open a Presentation",
                filetypes=(
                    ("Powerpoint Presentation", "*.pptx"),
                    ("PowerPoint 97-2003 Presentation", "*.ppt"),
                    ("All files", "*.*"),
                ),
                initialdir=CWD,
            )
            if not pptx_path:
                messagebox.showwarning(
                    "Warning", "No file selected. Operation cancelled."
                )
                break
            encrypted_path = filedialog.asksaveasfilename(
                title="Save encrypted Presentation",
                filetypes=(
                    ("Tlock Encrypted Powerpoint Presentation File", "*.teppf"),
                    ("All files", "*.*"),
                ),
                initialdir=CWD,
            )

            if pptx_path and encrypted_path:
                with open(pptx_path, "rb") as file:
                    date = file.read()
                    encrypted_data = encrypt_file(date)
                with open(encrypted_path + ".teppf", "wb") as file:
                    file.write(encrypted_data)
                    messagebox.showinfo(
                        "Info", "The file is encrypted successfully.")
                    messagebox.showinfo(
                        "Info",
                        f"The key for decryption: {KEY.decode("utf-8")}. The key is
                        copied to the clipboard.",
                    )

                    window.clipboard_clear()
                    window.clipboard_append(KEY.decode("utf-8"))
                    break
```

```
2142
2143        def open_decrypted_pptx():
2144            while True:
2145                key = simpledialog.askstring(
2146                    "Key", "Please, the key for the decryption:", show="*"
2147                )
2148                if key:
2149                    cipher = Fernet(key)
2150
2151                    def decrypt_file(file):
2152                        return bytes(
2153                            cipher.decrypt(
2154                                cipher.decrypt(
2155                                    cipher.decrypt(
2156                                        cipher.decrypt(
2157                                            cipher.decrypt(
2158                                                cipher.decrypt(
2159                                                    cipher.decrypt(
2160                                                        cipher.decrypt(
2161                                                            cipher.decrypt(
2162                                                                cipher.decrypt(
2163                                                                    cipher.decrypt(
2164                                                                        cipher.decrypt(
2165                                                                            file)
2166                                                                    )
2167                                                                )
2168                                                            )
2169                                                        )
2170                                                    )
2171                                                )
2172                                            )
2173                                        )
2174                                    )
2175                                )
2176                            )
2177                        )
2178
2179                    saving_path = filedialog.askopenfilename(
2180                        title="Open encrypted Presentation",
2181                        filetypes=(
2182                            ("Tlock Encrypted Powerpoint Presentation File", "*.teppf"),
2183                            ("All files", "*.*"),
2184                        ),
2185                        initialdir=CWD,
2186                    )
2187                    if not saving_path:
2188                        messagebox.showwarning(
2189                            "Warning", "No file selected. Operation cancelled."
2190                        )
2191                        break
2192                    if saving_path:
2193                        with open(saving_path, "rb") as file:
2194                            encrypted_data = file.read()
2195
2196                            decrypted_data = decrypt_file(encrypted_data)
2197
2198                            with open(saving_path + ".teppf", "wb") as wrapper:
2199                                wrapper.write(decrypted_data)
2200                                messagebox.showinfo(
2201                                    "Info", "The file is decrypted successfully."
2202                                )
2203                            break
2204                if not key:
2205                    messagebox.showwarning(
2206                        "Warning", "No key entered. Operation cancelled."
2207                    )
2208                    break
```

```
2209
2210        def save_encrypted_xlsx():
2211            KEY = Fernet.generate_key()
2212            cipher = Fernet(KEY)
2213
2214            def encrypt_file(file):
2215                return bytes(
2216                    cipher.encrypt(
2217                        cipher.encrypt(
2218                            cipher.encrypt(
2219                                cipher.encrypt(
2220                                    cipher.encrypt(
2221                                        cipher.encrypt(
2222                                            cipher.encrypt(
2223                                                cipher.encrypt(
2224                                                    cipher.encrypt(
2225                                                        cipher.encrypt(
2226                                                            cipher.encrypt(
2227                                                                cipher.encrypt(
2228                                                                    file)
2229                                                            )
2230                                                        )
2231                                                    )
2232                                                )
2233                                            )
2234                                        )
2235                                    )
2236                                )
2237                            )
2238                        )
2239                    )
2240                )
2241
2242            while True:
2243                sheet_path = filedialog.askopenfilename(
2244                    title="Open a Spreadsheet",
2245                    filetypes=(
2246                        ("Excel Spreadsheet", "*.xlsx"),
2247                        ("Excel 97-2003 Spreadsheet", "*.xls"),
2248                        ("All files", "*.*"),
2249                    ),
2250                    initialdir=CWD,
2251                )
2252                if not sheet_path:
2253                    messagebox.showwarning(
2254                        "Warning", "No file selected. Operation cancelled."
2255                    )
2256                    break
2257                encrypted_path = filedialog.asksaveasfilename(
2258                    title="Save encrypted Spreadsheet",
2259                    filetypes=(
2260                        ("Tlock Encrypted Excel File", "*.teef"),
2261                        ("All files", "*.*"),
2262                    ),
2263                    initialdir=CWD,
2264                )
2265
2266                if sheet_path and encrypted_path:
2267                    with open(sheet_path, "rb") as file:
2268                        date = file.read()
2269                        encrypted_data = encrypt_file(date)
2270                    with open(encrypted_path + ".teef", "wb") as file:
2271                        file.write(encrypted_data)
2272                        messagebox.showinfo(
2273                            "Info", "The file is encrypted successfully.")
2274                        messagebox.showinfo(
2275                            "Info",
```

```
                        f"The key for decryption: {KEY.decode("utf-8")}. The key is
                        copied to the clipboard.",
                    )

                    window.clipboard_clear()
                    window.clipboard_append(KEY.decode("utf-8"))
                    break

    def open_decrypted_xlsx():
        while True:
            key = simpledialog.askstring(
                "Key", "Please, the key for the decryption:", show="*"
            )
            if key:
                cipher = Fernet(key)

                def decrypt_file(file):
                    return bytes(
                        cipher.decrypt(
                            cipher.decrypt(
                                cipher.decrypt(
                                    cipher.decrypt(
                                        cipher.decrypt(
                                            cipher.decrypt(
                                                cipher.decrypt(
                                                    cipher.decrypt(
                                                        cipher.decrypt(
                                                            cipher.decrypt(
                                                                cipher.decrypt(
                                                                    file)
                                                            )
                                                        )
                                                    )
                                                )
                                            )
                                        )
                                    )
                                )
                            )
                        )
                    )

                saving_path = filedialog.askopenfilename(
                    title="Open encrypted Spreadsheet",
                    filetypes=(
                        ("Tlock Encrypted Excel File", "*.teef"),
                        ("All files", "*.*"),
                    ),
                    initialdir=CWD,
                )
                if not saving_path:
                    messagebox.showwarning(
                        "Warning", "No file selected. Operation cancelled."
                    )
                    break
                if saving_path:
                    with open(saving_path, "rb") as file:
                        encrypted_data = file.read()

                        decrypted_data = decrypt_file(encrypted_data)

                        with open(saving_path + ".xlsx", "wb") as wrapper:
                            wrapper.write(decrypted_data)
                            messagebox.showinfo(
                                "Info", "The file is decrypted successfully."
```

```
                            )
                        break
            if not key:
                messagebox.showwarning(
                    "Warning", "No key entered. Operation cancelled."
                )
                break

    def save_encrypted_accdb():
        KEY = Fernet.generate_key()
        cipher = Fernet(KEY)

        def encrypt_file(file):
            return bytes(
                cipher.encrypt(
                    cipher.encrypt(
                        cipher.encrypt(
                            cipher.encrypt(
                                cipher.encrypt(
                                    cipher.encrypt(
                                        cipher.encrypt(
                                            cipher.encrypt(
                                                cipher.encrypt(
                                                    cipher.encrypt(
                                                        file)
                                                )
                                            )
                                        )
                                    )
                                )
                            )
                        )
                    )
                )
            )

        while True:
            accdb_path = filedialog.askopenfilename(
                title="Open a Database",
                filetypes=(
                    ("Access Database", "*.accdb"),
                    ("Access 2002-2003 Database", "*.mdb"),
                    ("All files", "*.*"),
                ),
                initialdir=CWD,
            )
            if not accdb_path:
                messagebox.showwarning(
                    "Warning", "No file selected. Operation cancelled."
                )
                break
            encrypted_path = filedialog.asksaveasfilename(
                title="Save encrypted Database",
                filetypes=(
                    ("Tlock Encrypted Access Database File", "*.teadf"),
                    ("All files", "*.*"),
                ),
                initialdir=CWD,
            )

            if accdb_path and encrypted_path:
                with open(accdb_path, "rb") as file:
                    date = file.read()
```

```
                    encrypted_data = encrypt_file(date)
                with open(encrypted_path + ".teadf", "wb") as file:
                    file.write(encrypted_data)
                    messagebox.showinfo(
                        "Info", "The file is encrypted successfully.")
                    messagebox.showinfo(
                        "Info",
                        f"The key for decryption: {KEY.decode("utf-8")}. The key is
                        copied to the clipboard.",
                    )

                    window.clipboard_clear()
                    window.clipboard_append(KEY.decode("utf-8"))
                    break

    def open_decrypted_accdb():
        while True:
            key = simpledialog.askstring(
                "Key", "Please, the key for the decryption:", show="*"
            )
            if key:
                cipher = Fernet(key)

                def decrypt_file(file):
                    return bytes(
                        cipher.decrypt(
                            cipher.decrypt(
                                cipher.decrypt(
                                    cipher.decrypt(
                                        cipher.decrypt(
                                            cipher.decrypt(
                                                cipher.decrypt(
                                                    cipher.decrypt(
                                                        cipher.decrypt(
                                                            cipher.decrypt(
                                                                file)
                                                        )
                                                    )
                                                )
                                            )
                                        )
                                    )
                                )
                            )
                        )
                    )

                saving_path = filedialog.askopenfilename(
                    title="Open encrypted Database",
                    filetypes=(
                        ("Tlock Encrypted Access Database File", "*.teadf"),
                        ("All files", "*.*"),
                    ),
                    initialdir=CWD,
                )
                if not saving_path:
                    messagebox.showwarning(
                        "Warning", "No file selected. Operation cancelled."
                    )
                    break
                if saving_path:
                    with open(saving_path, "rb") as file:
                        encrypted_data = file.read()
```

```
                        decrypted_data = decrypt_file(encrypted_data)

                        with open(saving_path + ".accdb", "wb") as wrapper:
                            wrapper.write(decrypted_data)
                            messagebox.showinfo(
                                "Info", "The file is decrypted successfully."
                            )
                        break
            if not key:
                messagebox.showwarning(
                    "Warning", "No key entered. Operation cancelled."
                )
                break

    def save_encrypted_pub():
        KEY = Fernet.generate_key()
        cipher = Fernet(KEY)

        def encrypt_file(file):
            return bytes(
                cipher.encrypt(
                    cipher.encrypt(
                        cipher.encrypt(
                            cipher.encrypt(
                                cipher.encrypt(
                                    cipher.encrypt(
                                        cipher.encrypt(
                                            cipher.encrypt(
                                                cipher.encrypt(
                                                    cipher.encrypt(
                                                        file)
                                                )
                                            )
                                        )
                                    )
                                )
                            )
                        )
                    )
                )
            )

        while True:
            pub_path = filedialog.askopenfilename(
                title="Publisher File",
                filetypes=(
                    ("Publisher File", "*.pub"),
                    ("All files", "*.*"),
                ),
                initialdir=CWD,
            )
            if not pub_path:
                messagebox.showwarning(
                    "Warning", "No file selected. Operation cancelled."
                )
                break
            encrypted_path = filedialog.asksaveasfilename(
                title="Save encrypted Publisher File",
                filetypes=(
                    ("Tlock Encrypted Microsoft Publisher File", "*.tempf"),
                    ("All files", "*.*"),
                ),
```

```
                initialdir=CWD,
            )

            if pub_path and encrypted_path:
                with open(pub_path, "rb") as file:
                    date = file.read()
                    encrypted_data = encrypt_file(date)
                with open(encrypted_path + ".tempf", "wb") as file:
                    file.write(encrypted_data)
                    messagebox.showinfo(
                        "Info", "The file is encrypted successfully.")
                    messagebox.showinfo(
                        "Info",
                        f"The key for decryption: {KEY.decode("utf-8")}. The key is
                        copied to the clipboard.",
                    )

                    window.clipboard_clear()
                    window.clipboard_append(KEY.decode("utf-8"))
                    break

    def open_decrypted_pub():
        while True:
            key = simpledialog.askstring(
                "Key", "Please, the key for the decryption:", show="*"
            )
            if key:
                cipher = Fernet(key)

                def decrypt_file(file):
                    return bytes(
                        cipher.decrypt(
                            cipher.decrypt(
                                cipher.decrypt(
                                    cipher.decrypt(
                                        cipher.decrypt(
                                            cipher.decrypt(
                                                cipher.decrypt(
                                                    cipher.decrypt(
                                                        cipher.decrypt(
                                                            cipher.decrypt(
                                                                file)
                                                        )
                                                    )
                                                )
                                            )
                                        )
                                    )
                                )
                            )
                        )
                    )

                saving_path = filedialog.askopenfilename(
                    title="Open encrypted Publisher File",
                    filetypes=(
                        ("Tlock Encrypted Publisher File", "*.tempf"),
                        ("All files", "*.*"),
                    ),
                    initialdir=CWD,
                )
                if not saving_path:
                    messagebox.showwarning(
```

```
                        "Warning", "No file selected. Operation cancelled."
                    )
                    break
                if saving_path:
                    with open(saving_path, "rb") as file:
                        encrypted_data = file.read()

                        decrypted_data = decrypt_file(encrypted_data)

                        with open(saving_path + ".pub", "wb") as wrapper:
                            wrapper.write(decrypted_data)
                            messagebox.showinfo(
                                "Info", "The file is decrypted successfully."
                            )
                        break
            if not key:
                messagebox.showwarning(
                    "Warning", "No key entered. Operation cancelled."
                )
                break

    def save_encrypted_note():
        KEY = Fernet.generate_key()
        cipher = Fernet(KEY)

        def encrypt_file(file):
            return bytes(
                cipher.encrypt(
                    cipher.encrypt(
                        cipher.encrypt(
                            cipher.encrypt(
                                cipher.encrypt(
                                    cipher.encrypt(
                                        cipher.encrypt(
                                            cipher.encrypt(
                                                cipher.encrypt(
                                                    cipher.encrypt(
                                                        file)
                                                )
                                            )
                                        )
                                    )
                                )
                            )
                        )
                    )
                )
            )

        while True:
            pub_path = filedialog.askopenfilename(
                title="OneNote File",
                filetypes=(
                    ("OneNote File", "*.one"),
                    ("All files", "*.*"),
                ),
                initialdir=CWD,
            )
            if not pub_path:
                messagebox.showwarning(
                    "Warning", "No file selected. Operation cancelled."
                )
                break
```

```
            encrypted_path = filedialog.asksaveasfilename(
                title="Save encrypted OneNote File",
                filetypes=(
                    ("Tlock Encrypted Microsoft OneNote File", "*.temof"),
                    ("All files", "*.*"),
                ),
                initialdir=CWD,
            )

            if pub_path and encrypted_path:
                with open(pub_path, "rb") as file:
                    date = file.read()
                    encrypted_data = encrypt_file(date)
                with open(encrypted_path + ".temof", "wb") as file:
                    file.write(encrypted_data)
                    messagebox.showinfo(
                        "Info", "The file is encrypted successfully.")
                    messagebox.showinfo(
                        "Info",
                        f"The key for decryption: {KEY.decode("utf-8")}. The key is
                        copied to the clipboard.",
                    )

                    window.clipboard_clear()
                    window.clipboard_append(KEY.decode("utf-8"))
                    break

    def open_decrypted_note():
        while True:
            key = simpledialog.askstring(
                "Key", "Please, the key for the decryption:", show="*"
            )
            if key:
                cipher = Fernet(key)

                def decrypt_file(file):
                    return bytes(
                        cipher.decrypt(
                            cipher.decrypt(
                                cipher.decrypt(
                                    cipher.decrypt(
                                        cipher.decrypt(
                                            cipher.decrypt(
                                                cipher.decrypt(
                                                    cipher.decrypt(
                                                        cipher.decrypt(
                                                            cipher.decrypt(
                                                                file)
                                                        )
                                                    )
                                                )
                                            )
                                        )
                                    )
                                )
                            )
                        )
                    )

                saving_path = filedialog.askopenfilename(
                    title="Open encrypted OneNote File",
                    filetypes=(
                        ("Tlock Encrypted OneNote File", "*.temof"),
```

```
                        ("All files", "*.*"),
                    ),
                    initialdir=CWD,
                )
                if not saving_path:
                    messagebox.showwarning(
                        "Warning", "No file selected. Operation cancelled."
                    )
                    break
                if saving_path:
                    with open(saving_path, "rb") as file:
                        encrypted_data = file.read()

                        decrypted_data = decrypt_file(encrypted_data)

                        with open(saving_path + ".one", "wb") as wrapper:
                            wrapper.write(decrypted_data)
                            messagebox.showinfo(
                                "Info", "The file is decrypted successfully."
                            )
                            break
            if not key:
                messagebox.showwarning(
                    "Warning", "No key entered. Operation cancelled."
                )
                break

    def save_encrypted_pdf():
        KEY = Fernet.generate_key()
        cipher = Fernet(KEY)

        def encrypt_file(file):
            return bytes(
                cipher.encrypt(
                    cipher.encrypt(
                        cipher.encrypt(
                            cipher.encrypt(
                                cipher.encrypt(
                                    cipher.encrypt(
                                        cipher.encrypt(
                                            cipher.encrypt(
                                                cipher.encrypt(
                                                    cipher.encrypt(
                                                        file)
                                                )
                                            )
                                        )
                                    )
                                )
                            )
                        )
                    )
                )
            )

        while True:
            image_path = filedialog.askopenfilename(
                title="Open a PDF",
                filetypes=(
                    ("PDF files", "*.pdf"),
                    ("All files", "*.*"),
                ),
                initialdir=CWD,
```

```
            )
            if not image_path:
                messagebox.showwarning(
                    "Warning", "No file selected. Operation cancelled."
                )
                break
            encrypted_path = filedialog.asksaveasfilename(
                title="Save encrypted PDF",
                filetypes=(
                    ("Tlock Encrypted PDF File", "*.tepf"),
                    ("All files", "*.*"),
                ),
                initialdir=CWD,
            )

            if image_path and encrypted_path:
                with open(image_path, "rb") as file:
                    date = file.read()
                    encrypted_data = encrypt_file(date)
                with open(encrypted_path + ".tepf", "wb") as file:
                    file.write(encrypted_data)
                    messagebox.showinfo(
                        "Info", "The file is encrypted successfully.")
                    messagebox.showinfo(
                        "Info",
                        f"The key for decryption: {KEY.decode("utf-8")}. The key is
                        copied to the clipboard.",
                    )

                    window.clipboard_clear()
                    window.clipboard_append(KEY.decode("utf-8"))
                    break

    def open_decrypted_pdf():
        while True:
            key = simpledialog.askstring(
                "Key", "Please, the key for the decryption:", show="*"
            )
            if key:
                cipher = Fernet(key)

                def decrypt_file(file):
                    return bytes(
                        cipher.decrypt(
                            cipher.decrypt(
                                cipher.decrypt(
                                    cipher.decrypt(
                                        cipher.decrypt(
                                            cipher.decrypt(
                                                cipher.decrypt(
                                                    cipher.decrypt(
                                                        cipher.decrypt(
                                                            cipher.decrypt(
                                                                file)
                                                        )
                                                    )
                                                )
                                            )
                                        )
                                    )
                                )
                            )
                        )
                    )
```

```
                    )

                saving_path = filedialog.askopenfilename(
                    title="Open encrypted PDF",
                    filetypes=(
                        ("Tlock Encrypted PDF File", "*.tepf"),
                        ("All files", "*.*"),
                    ),
                    initialdir=CWD,
                )
                if not saving_path:
                    messagebox.showwarning(
                        "Warning", "No file selected. Operation cancelled."
                    )
                    break
                if saving_path:
                    with open(saving_path, "rb") as file:
                        encrypted_data = file.read()

                        decrypted_data = decrypt_file(encrypted_data)

                        with open(saving_path + ".pdf", "wb") as wrapper:
                            wrapper.write(decrypted_data)
                            messagebox.showinfo(
                                "Info", "The file is decrypted successfully."
                            )
                            break
            if not key:
                messagebox.showwarning(
                    "Warning", "No key entered. Operation cancelled."
                )
                break

    def save_encrypted_zip():
        KEY = Fernet.generate_key()
        cipher = Fernet(KEY)

        def encrypt_file(file):
            return bytes(
                cipher.encrypt(
                    cipher.encrypt(
                        cipher.encrypt(
                            cipher.encrypt(
                                cipher.encrypt(
                                    cipher.encrypt(
                                        cipher.encrypt(
                                            cipher.encrypt(
                                                cipher.encrypt(
                                                    cipher.encrypt(
                                                        file)
                                                    )
                                                )
                                            )
                                        )
                                    )
                                )
                            )
                        )
                    )
                )

        while True:
            image_path = filedialog.askopenfilename(
```

```
                title="Open a zip file",
                filetypes=(
                    ("Zip files", "*.zip"),
                    ("All files", "*.*"),
                ),
                initialdir=CWD,
            )
            if not image_path:
                messagebox.showwarning(
                    "Warning", "No file selected. Operation cancelled."
                )
                break
            encrypted_path = filedialog.asksaveasfilename(
                title="Save a zip file",
                filetypes=(
                    ("Tlock Encrypted Zip File", "*.tezf"),
                    ("All files", "*.*"),
                ),
                initialdir=CWD,
            )

            if image_path and encrypted_path:
                with open(image_path, "rb") as file:
                    date = file.read()
                    encrypted_data = encrypt_file(date)
                with open(encrypted_path + ".tezf", "wb") as file:
                    file.write(encrypted_data)
                    messagebox.showinfo(
                        "Info", "The file is encrypted successfully.")
                    messagebox.showinfo(
                        "Info",
                        f"The key for decryption: {KEY.decode("utf-8")}. The key is
                        copied to the clipboard.",
                    )

                    window.clipboard_clear()
                    window.clipboard_append(KEY.decode("utf-8"))
                    break

    def open_decrypted_zip():
        while True:
            key = simpledialog.askstring(
                "Key", "Please, the key for the decryption:", show="*"
            )
            if key:
                cipher = Fernet(key)

                def decrypt_file(file):
                    return bytes(
                        cipher.decrypt(
                            cipher.decrypt(
                                cipher.decrypt(
                                    cipher.decrypt(
                                        cipher.decrypt(
                                            cipher.decrypt(
                                                cipher.decrypt(
                                                    cipher.decrypt(
                                                        cipher.decrypt(
                                                            cipher.decrypt(
                                                                file)
                                                        )
                                                    )
                                                )
                                            )
                                        )
                                    )
                                )
                            )
                        )
```

```
                                        )
                                    )
                                )
                            )
                        )
                    )

                saving_path = filedialog.askopenfilename(
                    title="Open encrypted zip file",
                    filetypes=(
                        ("Tlock Encrypted Zip File", "*.tezf"),
                        ("All files", "*.*"),
                    ),
                    initialdir=CWD,
                )
                if not saving_path:
                    messagebox.showwarning(
                        "Warning", "No file selected. Operation cancelled."
                    )
                    break
                if saving_path:
                    with open(saving_path, "rb") as file:
                        encrypted_data = file.read()

                        decrypted_data = decrypt_file(encrypted_data)

                        with open(saving_path + ".zip", "wb") as wrapper:
                            wrapper.write(decrypted_data)
                            messagebox.showinfo(
                                "Info", "The file is decrypted successfully."
                            )
                            break
            if not key:
                messagebox.showwarning(
                    "Warning", "No key entered. Operation cancelled."
                )
                break

    def python_ide():
        running_py_path = path.join(CWD, "temp.py")

        def run_code():
            with open(running_py_path, "w+", encoding="utf-8") as temp:
                temp.write(code_input.get("1.0", END))
            system(f'python "{running_py_path}"')

        def run_py():
            Thread(target=run_code).start()

        def save_tpy():
            password = simpledialog.askstring(
                "Password", "Please, type the password for the encryption:"
            )
            if password:
                code = code_input.get(index1="1.0", index2=END)
                while True:
                    tlide_path = filedialog.asksaveasfilename(
                        title="Save the Python file",
                        initialdir=CWD,
                        filetypes=(
                            ("Tlock Python files", "*.tpy"),
                            ("All files", "*.*"),
                        ),
                    )
                    if not tlide_path:
                        messagebox.showwarning(
```

```
                            "Warning", "No file selected. Operation cancelled."
                        )
                        break
                    final_path = tlide_path + ".tpy"
                    with open(final_path, "w", encoding="utf-8") as wrapper:
                        wrapper.write(mac1_encode(password) + "\n")
                        wrapper.write(encrypt(code))
                        messagebox.showinfo(
                            "Info", "The file is saved successfully.")
                        break

        def open_tpy():
            try:
                tlide_path = filedialog.askopenfilename(
                    title="Open a Python file",
                    initialdir=CWD,
                    filetypes=(
                        ("Tlock Python files", "*.tpy"),
                        ("All files", "*.*"),
                    ),
                )
                if not tlide_path:
                    messagebox.showwarning(
                        "Warning", "No file selected. Operation cancelled."
                    )
                password = simpledialog.askstring(
                    "Password", "Enter the password for the decryption:"
                ).strip()
                with open(tlide_path, "r", encoding="utf-8") as wrapper:
                    file_pass = wrapper.readline().strip()
                    if mac1_decode(file_pass) != password:
                        messagebox.showerror(
                            "Error", "The password is incorrect. Operation cancelled."
                        )
                    if mac1_decode(file_pass) == password:
                        read_code = decrypt(wrapper.read())
                        code_input.delete("1.0", END)
                        code_input.insert("1.0", read_code)
            except FileNotFoundError:
                pass

        # *GUI CONFIG.

        python_window = Window(themename="superhero")
        python_window.title("Python IDE")
        python_window.geometry("1920x1080")
        python_window.state("zoomed")
        # *WIDGETS

        tlide_help_button = Button(
            python_window,
            text="Help",
            command=help_func,
            width=235,
        )
        tlide_help_button.place(x=5, y=5)

        run_button = Button(
            python_window,
            text="Run Python Code",
            command=run_py,
            width=235,

        )
        run_button.place(x=5, y=44)

        save_button = Button(
```

```
            python_window,
            text="Save Python Code",
            width=235,
            command=save_tpy,

        )
        save_button.place(x=5, y=84)

        open_button = Button(
            python_window,
            text="Open Python Code",
            width=235,
            command=open_tpy,
        )
        open_button.place(x=5, y=125)
        code_label = Label(python_window, text="Python Code:")
        code_label.place(x=5, y=163)

        code_input = scrolledtext.ScrolledText(
            python_window, width=169, height=33, font=("Consolas", 12)
        )
        code_input.place(x=5, y=200)

        # *MAIN LOOP

        python_window.mainloop()

    def javascript_ide():
        running_js_path = path.join(CWD, "temp.js")

        def run_js_code():
            with open(running_js_path, "w+", encoding="utf-8") as temp:
                temp.write(code_input.get("1.0", END))
            system(f'node "{running_js_path}"')

        def run_js():
            Thread(target=run_js_code).start()

        def save_tjs():
            password = simpledialog.askstring(
                "Password", "Please, type the password for the encryption:"
            )
            code = code_input.get(index1="1.0", index2=END)
            while True:
                tlide_path = filedialog.asksaveasfilename(
                    title="Save the Javascript file",
                    initialdir=CWD,
                    filetypes=(
                        ("Tlock Javascript files", "*.tjs"),
                        ("All files", "*.*"),
                    ),
                )
                if not tlide_path:
                    messagebox.showwarning(
                        "Warning", "No file selected. Operation cancelled."
                    )
                    break

                final_path = tlide_path + ".tjs"
                with open(final_path, "w", encoding="utf-8") as wrapper:
                    wrapper.write(mac1_encode(password) + "\n")
                    wrapper.write(encrypt(code))
                    messagebox.showinfo(
                        "Info", "The file is saved successfully.")
                    break

        def open_tjs():
```

```
3208
3209                try:
3210                    tlide_path = filedialog.askopenfilename(
3211                        title="Open a Javascript file",
3212                        initialdir=CWD,
3213                        filetypes=(
3214                            ("Tlock Javascript files", "*.tjs"),
3215                            ("All files", "*.*"),
3216                        ),
3217                    )
3218                    if not tlide_path:
3219                        messagebox.showwarning(
3220                            "Warning", "No file selected. Operation cancelled."
3221                        )
3222                    password = simpledialog.askstring(
3223                        "Password", "Enter the password for the decryption:"
3224                    ).strip()
3225                    with open(tlide_path, "r", encoding="utf-8") as wrapper:
3226                        file_pass = wrapper.readline().strip()
3227                        if mac1_decode(file_pass) != password:
3228                            messagebox.showerror(
3229                                "Error", "The password is incorrect. Operation cancelled."
3230                            )
3231                        if mac1_decode(file_pass) == password:
3232                            read_code = decrypt(wrapper.read())
3233                            code_input.delete("1.0", END)
3234                            code_input.insert("1.0", read_code)
3235                except FileNotFoundError:
3236                    pass
3237
3238            # *GUI CONFIG.
3239
3240            js_window = Window(themename="superhero")
3241            js_window.title("Javascript IDE")
3242            js_window.geometry("1920x1080")
3243            js_window.state("zoomed")
3244            # *WIDGETS
3245
3246            tlide_help_button = Button(
3247                js_window,
3248                text="Help",
3249                command=help_func,
3250                width=235,
3251            )
3252            tlide_help_button.place(x=5, y=5)
3253
3254            run_button = Button(
3255                js_window,
3256                text="Run Javascript Code",
3257                command=run_js,
3258                width=235,
3259
3260            )
3261            run_button.place(x=5, y=44)
3262
3263            save_button = Button(
3264                js_window,
3265                text="Save Javascript Code",
3266                width=235,
3267                command=save_tjs,
3268
3269            )
3270            save_button.place(x=5, y=84)
3271
3272            open_button = Button(
3273                js_window,
3274                text="Open Javascript Code",
```

```
            width=235,
            command=open_tjs,
        )
        open_button.place(x=5, y=125)
        code_label = Label(js_window, text="Javascript Code:")
        code_label.place(x=5, y=163)

        code_input = scrolledtext.ScrolledText(
            js_window, width=169, height=33, font=("Consolas", 12)
        )
        code_input.place(x=5, y=200)

        # *MAIN LOOP

        js_window.mainloop()

    def listen():
        confirm = messagebox.askyesno(
            "Confirm", "Are you sure to start recording?")
        if confirm:
            try:
                if confirm:
                    recognizer = sr.Recognizer()

                    with sr.Microphone() as source:
                        audio = recognizer.listen(source)
                        # type: ignore
                        text = str(recognizer.recognize_google(audio))

                        text_input.insert("1.0", text)
                        return text
            except sr.UnknownValueError:
                messagebox.showerror("Error", "Could not understand audio")
        if not confirm:
            messagebox.showinfo("Info", "Recording Canceled")

    def listen_click():
        listen()
        text_input.insert("52", f"{listen}")

    def read(text):
        try:
            ts = gTTS(text, lang="en")
            ts.save("temp.mp3")
            system("start temp.mp3")
        except AssertionError:
            messagebox.showerror(
                "Error", "No text to read, Operation cancelled.")

    # *GUI CONFIG.

    window = Window(themename="superhero")
    window.title("TLock")
    window.geometry("1920x1080")
    window.state("zoomed")
    menubar = Menu(window)
    window.config(menu=menubar)

    accessibility_menu = Menu(menubar, tearoff=False)
    accessibility_menu.add_command(label="Help", command=help_func)
    accessibility_menu.add_command(
        label="Microphone Writer", command=listen_click)
    accessibility_menu.add_command(
        label="Text to Speech", command=lambda: read(text_input.get("1.0", END))
    )
    accessibility_menu.add_command(
        label="Exit", command=lambda: window.destroy())
```

```
3342        menubar.add_cascade(label="Accessibility", menu=accessibility_menu)
3343
3344        text_menu = Menu(menubar, tearoff=False)
3345        text_menu.add_command(label="Save a Tetf", command=save_encrypted_text)
3346        text_menu.add_command(label="Open a Tetf", command=open_decrypted_text)
3347        text_menu.add_command(label="Open Notepad",
3348                              command=lambda: system("start notepad"))
3349        menubar.add_cascade(label="Text Files", menu=text_menu)
3350
3351        image_menu = Menu(menubar, tearoff=False)
3352        image_menu.add_command(label="Create a Teif", command=save_encrypted_image)
3353        image_menu.add_command(label="Open a Teif", command=open_decrypted_image)
3354        image_menu.add_command(
3355            label="Open Camera",
3356            command=lambda: system("start microsoft.windows.camera:"),
3357        )
3358        menubar.add_cascade(label="Image Files", menu=image_menu)
3359
3360        audio_menu = Menu(menubar, tearoff=False)
3361        audio_menu.add_command(label="Create a Teaf", command=save_encrypted_audio)
3362        audio_menu.add_command(label="Open a Teaf", command=open_decrypted_audio)
3363        audio_menu.add_command(
3364            label="Open Sound Recorder",
3365            command=lambda: system(
3366                r"explorer.exe
                    shell:appsFolder\Microsoft.WindowsSoundRecorder_8wekyb3d8bbwe!App"
3367            ),
3368        )
3369        menubar.add_cascade(label="Audio Files", menu=audio_menu)
3370
3371        video_menu = Menu(menubar, tearoff=0)
3372        video_menu.add_command(label="Create a Tevf", command=save_encrypted_video)
3373        video_menu.add_command(label="Open a Tevf", command=open_decrypted_video)
3374        video_menu.add_command(
3375            label="Open Camera",
3376            command=lambda: system("start microsoft.windows.camera:"),
3377        )
3378        menubar.add_cascade(label="Video Files", menu=video_menu)
3379
3380        doc_menu = Menu(menubar, tearoff=0)
3381        doc_menu.add_command(label="Create a Tedf", command=save_encrypted_docs)
3382        doc_menu.add_command(label="Open a Tedf", command=open_decrypted_docs)
3383        doc_menu.add_command(
3384            label="Open Word", command=lambda: system("start winword"))
3385        menubar.add_cascade(label="Word Files", menu=doc_menu)
3386
3387        pptx_menu = Menu(menubar, tearoff=0)
3388        pptx_menu.add_command(label="Create a Teppf", command=save_encrypted_pptx)
3389        pptx_menu.add_command(label="Open a Teppf", command=open_decrypted_pptx)
3390        pptx_menu.add_command(
3391            label="Open Powerpoint", command=lambda: system("start powerpnt")
3392        )
3393        menubar.add_cascade(label="Powerpoint Files", menu=pptx_menu)
3394
3395        xlsx_menu = Menu(menubar, tearoff=0)
3396        xlsx_menu.add_command(label="Create a Teef", command=save_encrypted_xlsx)
3397        xlsx_menu.add_command(label="Open a Teef", command=open_decrypted_xlsx)
3398        xlsx_menu.add_command(label="Open Excel",
3399                              command=lambda: system("start excel"))
3400        menubar.add_cascade(label="Excel Files", menu=xlsx_menu)
3401
3402        accdb_menu = Menu(menubar, tearoff=0)
3403        accdb_menu.add_command(label="Create a teadf",
3404                               command=save_encrypted_accdb)
3405        accdb_menu.add_command(label="Open a teadf", command=open_decrypted_accdb)
3406        accdb_menu.add_command(label="Open Access",
3407                               command=lambda: system("start MSACCESS.EXE"))
```

```
3408        menubar.add_cascade(label="Access Files", menu=accdb_menu)
3409
3410        pub_menu = Menu(menubar, tearoff=0)
3411        pub_menu.add_command(label="Create a Tempf", command=save_encrypted_pub)
3412        pub_menu.add_command(label="Open a Tempf", command=open_decrypted_pub)
3413        pub_menu.add_command(label="Open Publisher",
3414                             command=lambda: system("start MSPUB.EXE"))
3415        menubar.add_cascade(label="Publisher Files", menu=pub_menu)
3416
3417        one_menu = Menu(menubar, tearoff=0)
3418        one_menu.add_command(label="Create a Temof", command=save_encrypted_note)
3419        one_menu.add_command(label="Open a Temof", command=open_decrypted_note)
3420        one_menu.add_command(label="Open OneNote",
3421                             command=lambda: system("start onenote"))
3422        menubar.add_cascade(label="OneNote Files", menu=one_menu)
3423
3424        pdf_menu = Menu(menubar, tearoff=0)
3425        pdf_menu.add_command(label="Create a Tepf", command=save_encrypted_pdf)
3426        pdf_menu.add_command(label="Open a Tepf", command=open_decrypted_pdf)
3427        menubar.add_cascade(label="PDF Files", menu=pdf_menu)
3428
3429        zip_menu = Menu(menubar, tearoff=0)
3430        zip_menu.add_command(label="Create a Tezf", command=save_encrypted_zip)
3431        zip_menu.add_command(label="Open a Tezf", command=open_decrypted_zip)
3432        menubar.add_cascade(label="Zip Files", menu=zip_menu)
3433
3434        code_menu = Menu(menubar, tearoff=0)
3435        code_menu.add_command(label="Python IDE", command=python_ide)
3436        code_menu.add_command(label="Javascript IDE", command=javascript_ide)
3437        menubar.add_cascade(label="Code", menu=code_menu)
3438
3439        text_label = Label(window, text="Text:")
3440        text_label.place(x=10, y=2)
3441
3442        text_input = scrolledtext.ScrolledText(
3443            window, width=169, height=39, bg="gray80", font=("default", 12)
3444        )
3445        text_input.place(x=12, y=32)
3446
3447        # *MAIN LOOP
3448
3449        window.mainloop()
3450
3451
3452    multithreading = Thread(target=main)
3453    multithreading.start()
3454    multithreading.join()
3455
3456    if __name__ == "__tlock__":
3457        main_process = Process(target=main)
3458        main_process.start()
3459        main_process.join()
3460
```