Introduction

I had a couple of ideas regarding creating a cipher. I thought it would be cool if upon entering a piece of text, a musical score would be spit back out. I also thought that converting text into hex color codes would make for something visually appealing. Ultimately, I thought it would be easier to convert letters into hex codes without having to introduce too many libraries. I would not have to worry about defining each and every punctuation mark.

Hurdles

Each character has a hex value of two characters. For example, ‘A’ = 41, ‘O’ = 4F, ‘m’ = 6D, and ‘x’ = 78. Each character has their own unique hex code. Including quotation marks, tildes, carrots, commas, and periods. I had initially thought that I could format the letters like hex codes and fill out the rest of the hex codes with pairs of 00’s if the words weren’t multiples of three. Alternatively, I could also format the hex codes like a bit locker recovery key in pairs of four separated by dashes.

While looking at the ASCII table that depicted decimal, hex, binary, html numbers, and characters, I realized that there was an ASCII label for SP (Space). In which case this made things easier for me. When decrypting the message, I could just do things in reverse.

Some of the other hurdles I faced revolved around me trying to print out the hex codes in their own color. Initially I tried using a library called colorist, however I kept running into errors about invalid hex codes. I looked further into more options and ran across ANSI escape codes. These escape codes only work/show up in the terminal and that is why the function for the implementation of this, is separate from the reading and writing of files.

Functions

```
def encrypt(phrase):
    str = ""
    for i in range(len(phrase)):
        letter = phrase[i]
        # cinvert to ASCII
        hexNum = ord(letter)
        # we convert to hex here and
        # we strip the unneccesary bit here
        hexCode = hex(hexNum).lstrip("0x")
        str += hexCode
    str = formatIntoHexColorCodes(str)
    return str

def decrypt(phrase):
    newPhrase = removeWhiteSpaces(phrase)
    str = ""
    for i in range(0, len(newPhrase), 2):
        # each hex code comes in pairs of 2
        part =  newPhrase[i : i + 2]
        # converting back to characters
        ch = chr(int(part, 16))
        str += ch
    return str
```

Here in our encrypt function, we take each letter or character, and convert it into their ASCII value. After which using the hex function we convert it into a hex code. Contrary to my initial belief, we do not receive a 2 character hex function. Therefore we had to remove the unnecessary parts. Once we get those 2 character hex codes we add it to our str. This string is then formatted so that we get spaced out 6 character hex color codes.

For our decrypt function, we do the reverse. We take out the whitespace and NUL characters we inserted. Instead of going 1 by 1, we go 2 by 2. Because our hex codes are 2 characters in length. Then we convert them back into the original characters they were and add it to our string. This string is then written onto our output file.

```
def printColoredHexCodes(str):
    # here we split up our hex codes
    # ANSI ESCAPE CODES SHOULD ONLY WORK ON THE TERMINAL,
    # WHICH IS WHY WE DONT TRY TO HAVE THEM SHOW UP IN OUR TXT FILES
    hexCodes = str.split()
    for code in hexCodes:
        # we are using regex to check if they are valid hex codes
        # we check that they only have letters and numbers
        # we also check that they are 6 cars long
        if re.match(r'^[0-9A-Fa-f]{6}$', code):

            # here we convert hex into RGB
            rgb = tuple(int(code[i:i+2], 16) for i in (0, 2, 4))
            # ANSI color code
            # \033[XXXm
            # this escape code format used for our color code and rest code
            color_code = f"\033[38;2;{rgb[0]};{rgb[1]};{rgb[2]}m"
            reset_code = "\033[0m"  # ANSI reset code
            # printing to terminal here
            print(color_code + code + reset_code, end=" ")
```

This only prints to the terminal, and this is done right before the string is copied to the encrypted file. This only occurs when choosing the encryption option. We use regular expressions or regex just to check that we have valid hex codes. Hex codes should only include letters and numbers and no other characters. Then we convert our hex code into RGB values. After which we follow a format for ANSI escape codes of \033[XXXm. We use the RGB values we obtained in our color code. Our reset code is a standard one.

Sources

https://stackoverflow.com/questions/4842424/list-of-ansi-color-escape-sequences

https://www.freecodecamp.org/news/ascii-table-hex-to-ascii-value-character-code-chart-2/