# Pynigma

Cédric Laubacher, Tao Grenkowski

4e, Kantonsschule Alpenquai Luzern

July 4, 2017

**Abstract.** Pynigma is an easily comprehensible Enigma Emulator written in Python 3
Source code is available on Github under `https://github.com/Starbix/pynigma`
Website: `https://starbix.github.io/pynigma`
License: GNU General Public License v3.0

# Table of Contents

## 1 Introduction

Our goal was to write a script or program that allows us to simulate the encryption of an Enigma M3 (and also M4 with certain settings). We decided on doing this, because we didn't want to write an only mathematical program and because cryptography is a very interesting and important topic. We got our inspiration from the movies "Enigma (2001)" and "The Imitation Game (2014)" because we recently watched them and made us realize how important cryptography and cracking its code can be.

Encryption has also never been more relevant. Because a huge amount of data gets transferred through the internet, people try to steal that data. That's why encryption is being used.

## 2 Theory

The Enigma machine is an electro-mechanical device. It is mechanically operated, with an electric signal passed through wires and various mechanical parts. The easiest way to explain the mechanics is to follow the journey of a single letter from keyboard to lampboard.

The diagram below (Figure 1) shows the path the signal takes from pressing the letter 'T' on the keyboard to the 'G' lamp lighting up.

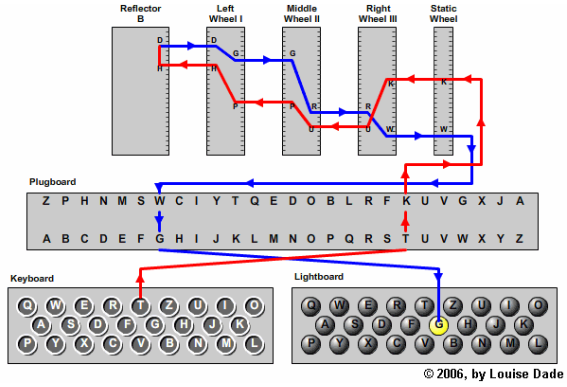


**Fig. 1.** [1] How one letter is changed into another letter at each stage as it passes through an Enigma machine.

When the the letter 'T' is pressed on the keyboard it creates an electric signal that goes through the Enigma machine wiring that will end with a lamp flashing on the lampboard.

First, the current flows through the plugboard. Here the signal is connected to the 'T' input on the plugboard. Some of the letters on the plugboard will be wired up to other letters (the plugs). If the 'T' input is not plugged to another

letter then our signal will pass straight to the 'T output. Here though the 'T' is plugged to the 'K', so the signal comes out of the 'K' output.

The plugboard isn't implemented in pynigma at the moment, however it can always be added. The source code is located on Github on `https://github.com/Starbix/pynigma`

Then the signal pass through three rotors (there are five available). Each of them is wired differently, so the output will be different. The first one gets rotated every letter, which will also scramble the signal even more. The others also rotate, but based on notches. It works just like a clock.

The reflector takes the input and reflects back the electrical signal for its return through the rotors. The reflector takes the letter and converts it to a different one, just like the plugboard. There are 5 different reflectors.

The reflector is important because it just wouldn't work if it got reflected the same way as no current could flow. (This is also the Enigma's biggest flaw.)

The reflected signal now passes back through the rotors, which work in exactly the same way in reverse.

The final stop is the lampboard, where the plugboard output is connected to the corresponding lamp for that letter. In our example, the letter 'G' lights up meaning the original letter 'T' is encrypted as 'G'.

The Enigma machine operator notes down the output letter and then enters the next letter in the message, and so on for every letter in the message.

### 2.1 Numbers

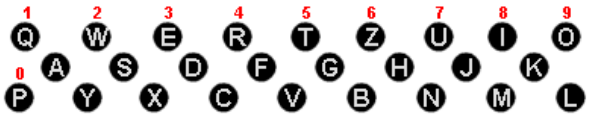


**Fig. 2.** [3] An Enigma keyboard showing the letters used as numbers.

To include numbers in your message, you first need to indicate that you are about to use a number by entering the letter "Y" before each number.The numbers itself are replaced with letters just like in the picture above (Figure 2). So, to encode the number 5 in your message you would type in "YT", the number 6 would be "YZ", and 42 would be "YRW".

### 2.2 Strength

The enigma was extremely secure for it's time. Yet it's code still got cracked. That's because of some flaws the enigma has. One of them is that a letter can't get encrypted to itself, one option less may not sound like much, but it is. Another flaw was that the Nazis had the rule that no same rotor order should be the same

in one month, this lead to worse security. Another flaw is that encrypting works the exact same ways as decrypting. [2] and [5]

## 3 Implementation

The program is easily readable and understandable, so to understand how it works, just take a look at the source code.

### 3.1 Initialization

So first we import some needed libraries. Then we define a class based on the OS. This is needed because Windows isn't POSIX compliant and it wouldn't look correct otherwise. Then we need to define all of the rotors and reflectors. [3] Then the script gets all of the inputs from the user and check if they are valid and reacts accordingly. The program then makes the input uppercase and converts it into a list. Some functions then are defined and the rotors get shifted into the initial position given by the user.

### 3.2 Main program

Now comes the real program. The for-loop runs once for every letter in the input.

```
for i in range(0, len(textArray)):
    firstRotor = shift(firstRotor, 1)
```

The first rotor gets shifted for every key "pressed".

Then it checks whether the turnover notch position is reached or not and shifts the next rotor if it is. Then the reverse rotors get generated.

Here's the most important part of this script.

```
print(colors.GREEN + colors.BOLD +
firstRotorReversed[ABC.index(
secondRotorReversed[ABC.index(
thirdRotorReversed[ABC.index(
reflector[ABC.index(
thirdRotor[ABC.index(
secondRotor[ABC.index(
firstRotor[ABC.index(
textArray[i])])])])])])])]
+ colors.ENDC, end="")
```

It takes the current letter that needs to get converted and then gets the position of it in the alphabet, for example for the letter 'C' it's 2 (lists start counting at 0). Then that position becomes the new letter because the third letter of the first rotor is for example 'M'. Then that letter becomes a new letter because of the second rotor and so on. In the end we get one letter for each print command. The ‘, end=""‘ is used because each character would be printed on a new line.

# 4 Results

## 4.1 How to get the code

First, you need to install git, this is preinstalled on most OSs, if not just head to this website `https://git-scm.com` and install it.

To get the code, enter the following commands:

```
git clone https://github.com/Starbix/pynigma
```

Now there should be a directory called pynigma which contains the script.

## 4.2 How to execute the code

To execute the program simply enter the directory and execute main.py

```
cd pynigma
./main.py
```

You will now see the following output where you can input the settings just like in the following example.

```
Choose the second rotor (each rotor can only be used once)
↪  [I/II/III/IV/V/VI/VII/VIII] VII
Choose the third rotor (each rotor can only be used once)
↪  [I/II/III/IV/V/VI/VII/VIII] V
Choose the position of the first rotor [1-26] 8
Choose the position of the second rotor [1-26] 24
Choose the position of the third rotor [1-26] 1
Choose reflector [A/B/C/B_thin/C_thin] A
Please type your text you want to decode or encode. Use 'X' as
↪  space or a stop.
ALLEXXFOLGENDESISTSOFORTBEKANNTZUGEBENXXICHHABEFOLGELNDEBEFEHLERHALTENXXJANSTELLE
↪  DESBISHERIGENREICHSMARSCHALLSJGOERINGJSETZTDERFUEHRERSIEYHERRGROSSADMIRAL
YALSSEINENNACHFOLGEREINXSCHRIFTLICHEVOLLMACHTUNTERWEGSXABSOFORTSOLLENSIES
AEMTLICHEMASSNAHMENVERFUEGENYDIESICHAUSDERGEGENWAERTIGENLAGEERGEBENXGEZX
REICHSLEITEIRKTULPEKKJBORMANNJ

UDHQYWALZPPEMKXCAMLAUSNSKVTWOMXCICDKQLBGKQLFDKCTXBTTZZKGICTSTNQKYKXJGAJNYHLXGBC
HDCIISNVNRADILWICLAUCQWWTWAUKQOOHUFYMVKFKZOAXZMSWOYZLGHJROUGNVIAWFKQJGIZXRWUSGH
CKYRKXPUSLNSDIOCAMTBTUWWEBOGTKFGLDGHYYHDYWCEQCACUPJYOZATZDRQAWXBLHULGSQBQQZZWRB
LDBPPCKTQKHECZCJNCAUOFKSBPZXOLYFMWYSZIGRBNFSSVVPBQZXSORGHSCKBJDUOSWUXTCPVBOMUCQ
KYHAZDSNQDEJQ

$ python3 main.py
Choose the first rotor (each rotor can only be used once)
↪  [I/II/III/IV/V/VI/VII/VIII] II
```

```
Choose the second rotor (each rotor can only be used once)
↪  [I/II/III/IV/V/VI/VII/VIII] VII
Choose the third rotor (each rotor can only be used once)
↪  [I/II/III/IV/V/VI/VII/VIII] V
Choose the position of the first rotor [1-26] 8
Choose the position of the second rotor [1-26] 24
Choose the position of the third rotor [1-26] 1
Choose reflector [A/B/C/B_thin/C_thin] A
Please type your text you want to decode or encode. Use 'X' as
↪  space or a stop.
UDHQYWALZPPEMKXCAMLAUSNSKVTWOMXCICDKQLBGKQLFDKCTXBTTZZKGICTSTNQKYKXJGAJNYHLXGBC
HDCIISNVNRADILWICLAUCQWWTWAUKQOOHUFYMVKFKZOAXZMSWOYZLGHJROUGNVIAWFKQJGIZXRWUSGH
CKYRKXPUSLNSDIOCAMTBTUWWEBOGTKFGLDGHYYHDYWCEQCACUPJYOZATZDRQAWXBLHULGSQBQQZZWRB
LDBPPCKTQKHECZCJNCAUOFKSBPZXOLYFMWYSZIGRBNFSSVVPBQZXSORGHSCKBJDUOSWUXTCPVBOMUCQ
KYHAZDSNQDEJQ

ALLEXXFOLGENDESISTSOFORTBEKANNTZUGEBENXXICHHABEFOLGELNDEBEFEHLERHALTENXXJANSTELLE
↪  DESBISHERIGENREICHSMARSCHALLSJGOERINGJSETZTDERFUEHRERSIEYHERRGROSSADMIRAL
YALSSEINENNACHFOLGEREINXSCHRIFTLICHEVOLLMACHTUNTERWEGSXABSOFORTSOLLENSIES
AEMTLICHEMASSNAHMENVERFUEGENYDIESICHAUSDERGEGENWAERTIGENLAGEERGEBENXGEZX
REICHSLEITEIRKTULPEKKJBORMANNJ
```

So the output is this again:

```
An Alle:

   Folgendes ist sofort bekanntzugeben:
   Ich habe folgende Befehl erhalten:
   'Anstelle des bisherigen Reichsmarschalls 'Göring' setzt der Führer
   Sie, Herr Großadmiral, als seinen Nachfolger ein.
   Schriftlische Vollmacht unterwegs.
   Ab sofort sollen Sie sämtliche Maßnahmen verfügen,
   die sich aus die gegenwärtigen Lage ergeben.
   Gez. Reichsleiter (Tulpe) 'Bormann'
```

As you can see the program works as expected and encrypts and decrypts correctly.

## 5 Discussion

### 5.1 What we learned

This project was quite fun and we learned a lot. For example was this a great opportunity to learn git (`https://git-scm.com`), a distributed version control system , and Github (`https://github.com`), a hosting site for said version control system and more e.g. Issues (bug reports, feature requests) and branching for easy development.

But not only that, we also setup a simple website which is available under `https://starbix.github.io/pynigma`.

We also took a look at different lincenses such as MIT and GPL 3.0 (which we use in this project) and pep8 which is a style guide for Python code (our code is pep8 compliant).

And we also learned a bit of LaTeX as one can probably see.

### 5.2 Initial problems

The biggest problem in the beginning was the question how we want to represent the input and how we can easily convert it. Fortunately we decided on using lists for that, which saved us a lot of hassle in later development. Also fully understanding how the Enigma works was challenging, but after some really helpful websites and videos [1] and [5] it was much clearer how we need to write the program.

### 5.3 Application

This program can be used for encrypting and decrypting text, not for exchanging nuclear launch codes of course. We actually also wanted to make this program able to directly take a file's content and encrypt it to another file using the redirect (>) function of UNIX OSs. Unfortunately we didn't have time for that, but maybe we add this functionality to Github in the future if we feel like it.

# Bibliography

[5] 158,962,555,217,826,360,000 (enigma machine) - numberphile. https://www.youtube.com/watch?v=G2_Q9FoD-oQ.

[4] Colors and more in print for python. https://stackoverflow.com/questions/287871/print-in-terminal-with-colors-using-python.

[1] How enigma machines work. http://enigma.louisedade.co.uk/howitworks.html, Accessed: 2017-06-19.

[2] How secure the enigma is. http://www.ostfalia.de/cms/de/pws/seutter/kryptologie/enigma/Sicherheit/Sicherheit/sicherheit2.html, Accessed: 2017-06-19.

[3] Numbers in enigma. http://enigma.louisedade.co.uk/help.html, Accessed: 2017-06-19.

## 6 Attachment

```
#!/usr/bin/env python3

import sys
import string
import os

# use colors class for better input and output, while still
↪  supporting NT based OSs
if os.name == "posix":
    class colors:
        HEADER = '\033[95m'
        BLUE = '\033[94m'
        GREEN = '\033[92m'
        WARNING = '\033[93m'
        RED = '\033[91m'
        ENDC = '\033[0m'
        BOLD = '\033[1m'
        UNDERLINE = '\033[4m'
else:
    class colors:
        HEADER = ''
        BLUE = ''
        GREEN = ''
        WARNING = ''
        RED = ''
        ENDC = ''
```

```
26          BOLD = ''
27          UNDERLINE = ''
28
29  # Alphabet list for converting input letters to numbers
30  ABC = list(string.ascii_uppercase)
31
32  # define the Walzen
33  I = ['E', 'K', 'M', 'F', 'L', 'G', 'D', 'Q', 'V', 'Z', 'N', 'T',
    ↪  'O', 'W', 'Y', 'H', 'X', 'U', 'S', 'P', 'A', 'I', 'B', 'R',
    ↪  'C', 'J']
34  II = ['A', 'J', 'D', 'K', 'S', 'I', 'R', 'U', 'X', 'B', 'L', 'H',
    ↪  'W', 'T', 'M', 'C', 'Q', 'G', 'Z', 'N', 'P', 'Y', 'F', 'V',
    ↪  'O', 'E']
35  III = ['B', 'D', 'F', 'H', 'J', 'L', 'C', 'P', 'R', 'T', 'X', 'V',
    ↪  'Z', 'N', 'Y', 'E', 'I', 'W', 'G', 'A', 'K', 'M', 'U', 'S',
    ↪  'Q', 'O']
36  IV = ['E', 'S', 'O', 'V', 'P', 'Z', 'J', 'A', 'Y', 'Q', 'U', 'I',
    ↪  'R', 'H', 'X', 'L', 'N', 'F', 'T', 'G', 'K', 'D', 'C', 'M',
    ↪  'W', 'B']
37  V = ['V', 'Z', 'B', 'R', 'G', 'I', 'T', 'Y', 'U', 'P', 'S', 'D',
    ↪  'N', 'H', 'L', 'X', 'A', 'W', 'M', 'J', 'Q', 'O', 'F', 'E',
    ↪  'C', 'K']
38  VI = ['J', 'P', 'G', 'V', 'O', 'U', 'M', 'F', 'Y', 'Q', 'B', 'E',
    ↪  'N', 'H', 'Z', 'R', 'D', 'K', 'A', 'S', 'X', 'L', 'I', 'C',
    ↪  'T', 'W']
39  VII = ['N', 'Z', 'J', 'H', 'G', 'R', 'C', 'X', 'M', 'Y', 'S', 'W',
    ↪  'B', 'O', 'U', 'F', 'A', 'I', 'V', 'L', 'P', 'E', 'K', 'Q',
    ↪  'D', 'T']
40  VIII = ['F', 'K', 'Q', 'H', 'T', 'L', 'X', 'O', 'C', 'B', 'J',
    ↪  'S', 'P', 'D', 'Z', 'R', 'A', 'M', 'E', 'W', 'N', 'I', 'U',
    ↪  'Y', 'G', 'V']
41  # beta and gamma were used in the Enigma M4
42  beta = ['L', 'E', 'Y', 'J', 'V', 'C', 'N', 'I', 'X', 'W', 'P',
    ↪  'B', 'Q', 'M', 'D', 'R', 'T', 'A', 'K', 'Z', 'G', 'F', 'U',
    ↪  'H', 'O', 'S']
43  gamma = ['F', 'S', 'O', 'K', 'A', 'N', 'U', 'E', 'R', 'H', 'M',
    ↪  'B', 'T', 'I', 'Y', 'C', 'W', 'L', 'Q', 'P', 'Z', 'X', 'V',
    ↪  'G', 'J', 'D']
44  A = ['E', 'J', 'M', 'Z', 'A', 'L', 'Y', 'X', 'V', 'B', 'W', 'F',
    ↪  'C', 'R', 'Q', 'U', 'O', 'N', 'T', 'S', 'P', 'I', 'K', 'H',
    ↪  'G', 'D']
45  B = ['Y', 'R', 'U', 'H', 'Q', 'S', 'L', 'D', 'P', 'X', 'N', 'G',
    ↪  'O', 'K', 'M', 'I', 'E', 'B', 'F', 'Z', 'C', 'W', 'V', 'J',
    ↪  'A', 'T']
```

```
46  C = ['F', 'V', 'P', 'J', 'I', 'A', 'O', 'Y', 'E', 'D', 'R', 'Z',
    ↪  'X', 'W', 'G', 'C', 'T', 'K', 'U', 'Q', 'S', 'B', 'N', 'M',
    ↪  'H', 'L']
47  B_thin = ['E', 'N', 'K', 'Q', 'A', 'U', 'Y', 'W', 'J', 'I', 'C',
    ↪  'O', 'P', 'B', 'L', 'M', 'D', 'X', 'Z', 'V', 'F', 'T', 'H',
    ↪  'R', 'G', 'S']
48  C_thin = ['R', 'D', 'O', 'B', 'J', 'N', 'T', 'K', 'V', 'E', 'H',
    ↪  'M', 'L', 'F', 'C', 'W', 'Z', 'A', 'X', 'G', 'Y', 'I', 'P',
    ↪  'S', 'U', 'Q']
49
50  # get input from user
51  firstRotor = eval(input("Choose the " + colors.BOLD + "first " +
    ↪  colors.ENDC + "rotor (each rotor can only be used once)
    ↪  [I/II/III/IV/V/VI/VII/VIII] "))
52  # use static rotor for correct turning
53  firstRotorStatic = firstRotor
54  secondRotor = eval(input("Choose the " + colors.BOLD + "second " +
    ↪  colors.ENDC + "rotor (each rotor can only be used once)
    ↪  [I/II/III/IV/V/VI/VII/VIII] "))
55  secondRotorStatic = secondRotor
56  thirdRotor = eval(input("Choose the " + colors.BOLD + "third " +
    ↪  colors.ENDC + "rotor (each rotor can only be used once)
    ↪  [I/II/III/IV/V/VI/VII/VIII] "))
57  # fourthRotor for Enigma M4
58  # check if every Rotor is only used once
59  if firstRotor == secondRotor or firstRotor == thirdRotor or
    ↪  secondRotor == thirdRotor:
60      print(colors.RED + "Each rotor can only be used once" +
        ↪  colors.ENDC)
61      sys.exit(1)
62  # get the initial positions of the rotors
63  firstRotorPosition = eval(input("Choose the position of the " +
    ↪  colors.BOLD + "first " + colors.ENDC + "rotor [1-26] ")) - 1
64  if 0 > firstRotorPosition or firstRotorPosition > 25:
65      print(colors.WARNING + "Choose a value between 1 and 26" +
        ↪  colors.ENDC)
66      firstRotorPosition = eval(input("Choose the position of the "
        ↪  + colors.BOLD + "first " + colors.ENDC + "rotor [1-26] "))
        ↪  - 1
67      if 0 > firstRotorPosition or firstRotorPosition > 25:
68          print(colors.RED + "Choose a value between 1 and 26" +
            ↪  colors.ENDC)
69          sys.exit(1)
70  secondRotorPosition = eval(input("Choose the position of the " +
    ↪  colors.BOLD + "second " + colors.ENDC + "rotor [1-26] ")) - 1
```

```
71  if 0 > secondRotorPosition or secondRotorPosition > 25:
72      print(colors.WARNING + "Choose a value between 1 and 26" +
        ↪  colors.ENDC)
73      secondRotorPosition = eval(input("Choose the position of the "
        ↪  + colors.BOLD + "second " + colors.ENDC + "rotor [1-26]
        ↪  ")) - 1
74      if 0 > secondRotorPosition or secondRotorPosition > 25:
75          print(colors.RED + "Choose a value between 1 and 26" +
            ↪  colors.ENDC)
76          sys.exit(1)
77  thirdRotorPosition = eval(input("Choose the position of the " +
    ↪  colors.BOLD + "third " + colors.ENDC + "rotor [1-26] ")) - 1
78  if 0 > thirdRotorPosition or thirdRotorPosition > 25:
79      print(colors.WARNING + "Choose a value between 1 and 26" +
        ↪  colors.ENDC)
80      thirdRotorPosition = eval(input("Choose the position of the "
        ↪  + colors.BOLD + "third " + colors.ENDC + "rotor [1-26] "))
        ↪  - 1
81      if 0 > thirdRotorPosition or thirdRotorPosition > 25:
82          print(colors.RED + "Choose a value between 1 and 26" +
            ↪  colors.ENDC)
83          sys.exit(1)
84  # let the user choose the reflector
85  reflector = eval(input("Choose reflector [A/B/C/B_thin/C_thin] "))
86
87  # input text from user
88  input = input("Please type your text you want to decode or encode.
    ↪  Use 'X' as space or a stop. ")
89
90  # convert text to only upper case letters
91  input = input.upper()
92  # check for numbers in input
93  if len(set(string.digits).intersection(input)) > 0:
94      print(colors.RED + colors.BOLD + "Refer to the README. Don't
        ↪  use digits")
95      if (' ' in input):
96          print("and no space!")
97      print(colors.ENDC)
98      sys.exit(1)
99  # convert input to list
100 textArray = list(input)
101
102
103 # define reverse function for the way "back"
104 def reverse(array):
```

```
105     reverseRotor = []
106     for s in range(0, 26):
107         reverseRotor.append(ABC[array.index(ABC[s])])
108     return reverseRotor
109 
110 
111 # define shift function for shifting the rotors
112 def shift(array, int):
113     return array[int:] + array[:int]
114 
115 
116 # shift to initial positions
117 firstRotor = shift(firstRotor, firstRotorPosition)
118 secondRotor = shift(secondRotor, secondRotorPosition)
119 thirdRotor = shift(thirdRotor, thirdRotorPosition)
120 
121 # run code once for every letter in the input
122 for i in range(0, len(textArray)):
123 
124     # initial shift
125     firstRotor = shift(firstRotor, 1)
126     # check if any rotor is at the turnover notch position
127     if firstRotorStatic == I:
128         if firstRotor[0] == 'U':
129             secondRotor = shift(secondRotor, 1)
130     elif firstRotorStatic == II:
131         if firstRotor[0] == 'I':
132             secondRotor = shift(secondRotor, 1)
133     elif firstRotorStatic == III:
134         if firstRotor[0] == 'U':
135             secondRotor = shift(secondRotor, 1)
136     elif firstRotorStatic == IV:
137         if firstRotor[0] == 'U':
138             secondRotor = shift(secondRotor, 1)
139     elif firstRotorStatic == V:
140         if firstRotor[0] == 'V':
141             secondRotor = shift(secondRotor, 1)
142 # some rotors have two turnover positions
143     elif firstRotorStatic == VI:
144         if firstRotor[0] == 'J' or firstRotor[0] == 'H':
145             secondRotor = shift(secondRotor, 1)
146     elif firstRotorStatic == VII:
147         if firstRotor[0] == 'N' or firstRotor[0] == 'O':
148             secondRotor = shift(secondRotor, 1)
149     elif firstRotorStatic == VIII:
```

```
150             if firstRotor[0] == 'F' or firstRotor[0] == 'D':
151                 secondRotor = shift(secondRotor, 1)
152         elif firstRotorStatic == beta:
153             if firstRotor[0] == 'P':
154                 secondRotor = shift(secondRotor, 1)
155         elif firstRotorStatic == gamma:
156             if firstRotor[0] == 'P':
157                 secondRotor = shift(secondRotor, 1)
158         elif secondRotorStatic == I:
159             if secondRotor[0] == 'U':
160                 thirdRotor = shift(thirdRotor, 1)
161         elif secondRotorStatic == II:
162             if secondRotor[0] == 'I':
163                 thirdRotor = shift(thirdRotor, 1)
164         elif secondRotorStatic == III:
165             if secondRotor[0] == 'U':
166                 thirdRotor = shift(thirdRotor, 1)
167         elif secondRotorStatic == IV:
168             if secondRotor[0] == 'U':
169                 thirdRotor = shift(thirdRotor, 1)
170         elif secondRotorStatic == V:
171             if secondRotor[0] == 'V':
172                 thirdRotor = shift(thirdRotor, 1)
173         elif secondRotorStatic == VI:
174             if secondRotor[0] == 'J' or secondRotor[0] == 'H':
175                 thirdRotor = shift(thirdRotor, 1)
176         elif secondRotorStatic == VII:
177             if secondRotor[0] == 'N' or secondRotor[0] == 'O':
178                 thirdRotor = shift(thirdRotor, 1)
179         elif secondRotorStatic == VIII:
180             if secondRotor[0] == 'F' or secondRotor[0] == 'D':
181                 thirdRotor = shift(thirdRotor, 1)
182         elif secondRotorStatic == beta:
183             if secondRotor[0] == 'P':
184                 thirdRotor = shift(thirdRotor, 1)
185     # reverse the rotors
186         firstRotorReversed = reverse(firstRotor)
187         secondRotorReversed = reverse(secondRotor)
188         thirdRotorReversed = reverse(thirdRotor)
189 
190     # most important piece of the script
191     # it basically takes the position of the letter in the list which
        ↪  then gets passed through the rotors, reflector and reverse
        ↪  rotors
```

```
192         print(colors.GREEN + colors.BOLD +
            ↪  firstRotorReversed[ABC.index(secondRotorReversed[ABC.index(
            ↪  thirdRotorReversed[ABC.index(reflector[ABC.index(
            ↪  thirdRotor[ABC.index(secondRotor[ABC.index(firstRotor[ABC.index
            ↪  (textArray[i])])])])])])])] + colors.ENDC, end="")
193     # was needed for better looks
194     print('')
```