Student name: Edvinas Dulskas
Student code: 19040186

source code ***vigenere_cipher.py***

```
import random

from string import ascii_uppercase as letters


# ---------- Functions ---------

# function that cretes 26x26 table (matrix)

def create_table():


        tbl = []                                                    # empty list created

        for i in range(26):                                 # appends emty list in each list item
(26)

                tbl.append([])                          # 26x26 matrix is created


        for row in range(26):

                for col in range(26):

                        if(row + 65) + col > 90:        # if number exceeds 90 [char(90) - Z],
we will write from A again

                                tbl[row].append(chr((row + 65) + col - 26))
        # chr(65) - A  chr(90) - Z

                        else:

                                tbl[row].append(chr(row + 65 + col))


        for row in tbl:

                for col in row:

                        print(col, end=' ')

                print(end='\n')


        return tbl
```

Student name: Edvinas Dulskas
Student code: 19040186

```
# function that generates the key

def vigenere_genkey(n):


        return ''.join(random.choice(letters) for i in range(n))


# function that maps the key with the message (makes the same lenght)

def map_the_key(key, msg):


        key_map = ''                            # mapped key to the message

        counter = 0                                     # counter


        for i in range(len(msg)):               # mapping the key

                if ord(msg[i]) == 32:                   # if message contains space, add space
to key_map

                        key_map += ' '

                else:

                        if counter < len(key):                          # for
e.g.: message is 'hello world' key is 'KEY', than

                                key_map += key[counter]         #
key_map will be 'KEYKE YKEYK'

                                counter += 1

                        else:

                                counter = 0

                                key_map += key[counter]

                                counter +=1


        return key_map


# encryption function
```

Student name: Edvinas Dulskas
Student code: 19040186

```
def vigenere_encrypt(key, msg):

        temp = msg.upper()                          # make every letter uppercase
        ctx = ''                                    # variable for encoded message
        mapped_key = map_the_key(key, temp)         # get mapped key

        for i in range(len(temp)):
                if ord(temp[i]) == 32:              # if it's a space in the message
                        ctx += ' '
                elif ord(temp[i]) < 65 | ord(temp[i]) > 90:   # if it's a special character ',./*-+\..'
                        ctx += temp[i]
                else:
                        row = ord(mapped_key[i]) - 65
                        col = ord(temp[i]) - 65
                        ctx += table[row][col]

        return ctx

# decryption function
def vigenere_decrypt(key, ctx):

        pp = ''                                     # variable for encoded message
        mapped_key = map_the_key(key, ctx)          # get mapped key

        for i in range(len(ctx)):
                if ord(ctx[i]) == 32:               # if it's a space in the message
                        pp += ' '
                elif ord(ctx[i]) < 65 | ord(ctx[i]) > 90:
```

```
                        pp += ctx[i]
                else:
                        idx = ord(ctx[i]) - ord(mapped_key[i])
                        if idx < 0:
                                idx += 26
                        pp += table[0][idx]

        return pp

# --------- Main -------------
table = create_table()
key_lenght = 4
msg = '/-hello.world**'

key = vigenere_genkey(key_lenght)
C = vigenere_encrypt(key, msg)
P = vigenere_decrypt(key, C)

print('Key: ' + key + '\nOriginal message: ' + msg + '\nEncrypted message: |' + C + '|\nDecrypted message: |' + P + '|')
```

*Results of **program**:*

```
Key: BSVI
Original message: /-hello.world**
Encrypted message: |/-CMMDJ.XGMTE**|
Decrypted message: |/-HELLO.WORLD**|
[Finished in 0.4s]
```

Explanation: key lenght is 4 and it's randomly generated. The thing that I did was to ignor the spaces and special characters such as „,./*-+\...“ during the encryption.