Student name: Edvinas Dulskas
Student code: 19040186

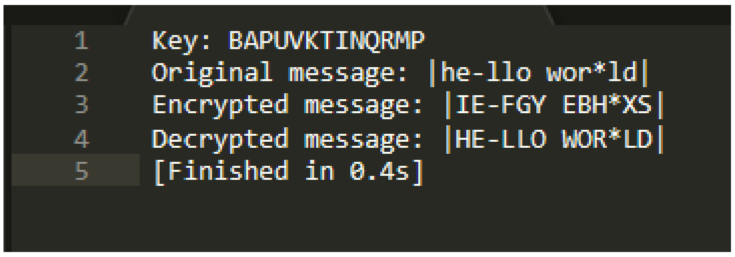
source code ***lfsr_cipher.py***

```
from string import ascii_uppercase as upper


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

# function that generated the key by the message lenght

def lfsr_genkey(n):

# I'm using x^4 key. xn=(xn-1 + xn-3 + xn-4)mod26

key = [1, 0, 15, 20]                                         # B, A, P, U - primary key (seed) You can always change
it. range[0-25]


if len(key) >= n:                      # if key is longer or the same lenght as a message, return key

return key

else:

for i in range(n-len(key)):                                                                             # loop for key
generation

            xn = (key[len(key)-1] + key[len(key)-3] + key[len(key)-4]) % 26              # formula: xn=(xn-1 +
xn-3 + xn-4)mod26

            key.append(xn)


return key


# encryption function

def lfsr_encrypt(key, msg):


temp = msg.upper()

ctx = ''                                     # emty string for ciphertext


for i in range(len(temp)):

if ord(temp[i]) == 32:                       # ignore space ' '

            ctx += ' '

elif ord(temp[i]) < 65 | ord(temp[i]) > 90:          # ignore special characters

            ctx += temp[i]

else:

            new_idx = ((ord(temp[i]) - 65) + key[i]) % 26                    # add indexes of key and original msg
and make new idx with mod26

            ctx += chr(new_idx+65)
```

```
return ctx


# decryption function
def lfsr_decrypt(key, ctx):


pp = ''                          # empty string for decrypted message


for i in range(len(ctx)):
if ord(ctx[i]) == 32:                         # ignore space
            pp += ' '
elif ord(ctx[i]) < 65 | ord(ctx[i]) > 90:                 #ignore special characters
            pp += ctx[i]
else:
            new_idx = (ord(ctx[i])-65) - key[i]                           # find original index of
the letter
            if new_idx < 0:
                        new_idx += 26
            pp += chr(new_idx + 65)


return pp


# --------- Main -------------
msg = 'he-llo wor*ld'                                                      # message
key = lfsr_genkey(len(msg))                              # key


C = lfsr_encrypt(key, msg)
P = lfsr_decrypt(key, C)


print('Key: ' + ''.join(chr(x+65) for x in key) + '\nOriginal message: |' + msg + '|\nEncrypted message: |' + C +
'|\nDecrypted message: |' + P + '|')
```

Student name: Edvinas Dulskas
Student code: 19040186

*Results of **program**:*



```
Key: BAPUVKTINQRMP
Original message: |he-llo wor*ld|
Encrypted message: |IE-FGY EBH*XS|
Decrypted message: |HE-LLO WOR*LD|
[Finished in 0.4s]
```

<u>Explanation:</u> key lenght is the same as message lenght and it is generated by the formula. The thing that I did was to ignore the spaces and special characters such as „,./*-+\...“ during the encryption