

1MPR14_Simona_Blinova sb24037

1.uzdevums

Programma, kas pārraksta informāciju no vienas datnes uz otro, pārveidojot burtus par uppercase.

Kods:

```
import os
import sys

# jāpievieno pareizas cēļš līdz datnei
datnes_cels1 = 'C:/Users/Simona/Desktop/lu/programmesana un datori
I/2sem/1MPR14/Simona_Blinova_1MPR14_programmas_un_datnes/uzd1 test1.txt'
if not os.path.isfile(datnes_cels1):
    print(f'Kļūda: Datne "{datnes_cels1}" neeksistē.')
    sys.exit(1)

# jāpievieno pareizas cēļš līdz datnei
datnes_cels2 = 'C:/Users/Simona/Desktop/lu/programmesana un datori
I/2sem/1MPR14/Simona_Blinova_1MPR14_programmas_un_datnes/uzd1 test2.txt'
if not os.path.isfile(datnes_cels2):
    print(f'Kļūda: Datne "{datnes_cels2}" neeksistē.')
    sys.exit(1)

with open('uzd1 test1.txt', 'w', encoding='utf-8') as datne:
    while True:
        rinda = input('Ievadiet teksta rindu --> ')
        datne.write(rinda + '\n')
        print("")

        paz = input('Vai vēlaties ievadīt vēl rindu? (n - nē) --> ')
        print("")

        if paz == 'n':
            break

    datne.close()

with open('uzd1 test1.txt', 'r', encoding='utf-8') as datne1, \
    open('uzd1 test2.txt', 'w', encoding='utf-8') as datne2:
    saturs = datne1.read()
```

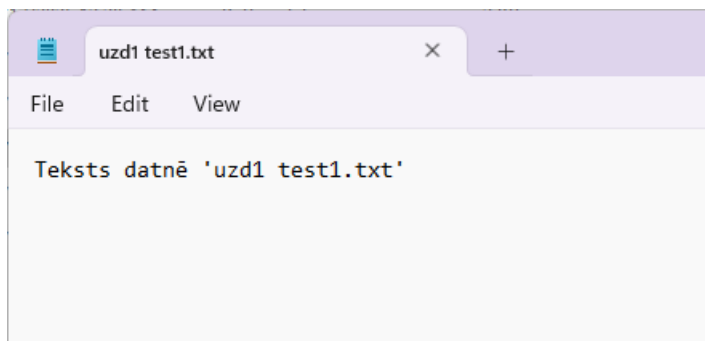
```
#print(saturs)

for b in saturs:
    #print(b)
    try:
        b = b.upper()
        datne2.write(b)
    except:
        datne2.write(b)
```

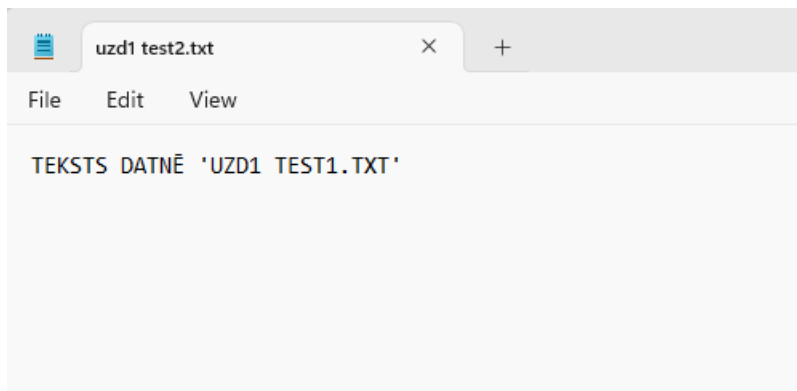
```
datne1.close()
datne2.close()
```

Testa piemērs(1)

Teksta datne 'uzd1 test1.txt':



Teksta datne 'uzd1 test2.txt':

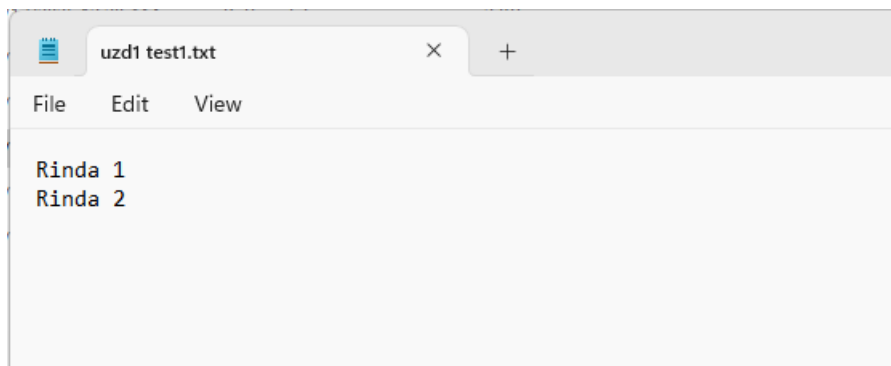


Terminālis:

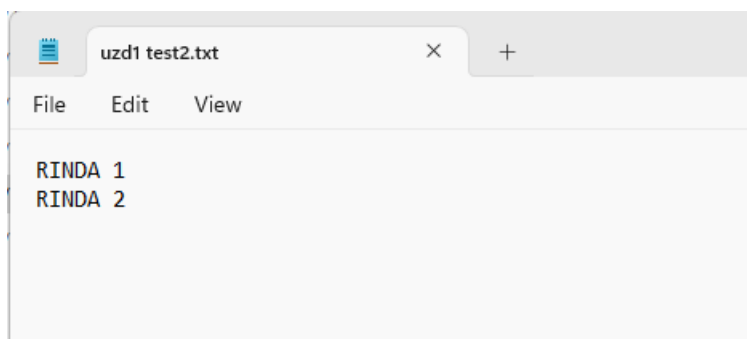
```
Ievadiet teksta rindu --> Teksts datnē 'uzd1 test1.txt'
Vai vēlaties ievadīt vēl rindu? (n - nē) --> n
```

Testa piemērs(2)

Teksta datne 'uzd1 test1.txt':



Teksta datne 'uzd1 test2.txt':

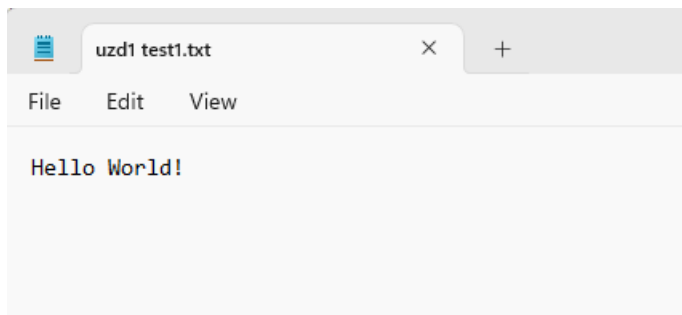


Terminālis:

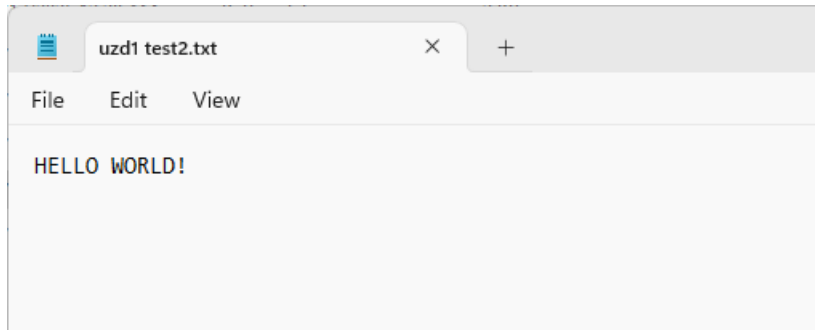
```
Ievadiet teksta rindu --> Rinda 1
Vai vēlaties ievadīt vēl rindu? (n - nē) --> j
Ievadiet teksta rindu --> Rinda 2
Vai vēlaties ievadīt vēl rindu? (n - nē) --> n
```

Testa piemērs(3)

Teksta datne 'uzd1 test1.txt':



Teksta datne 'uzd1 test2.txt':



Terminālis:

```
Ievadiet teksta rindu --> Hello World!  
Vai vēlaties ievadīt vēl rindu? (n - nē) --> n
```

2.uzdevums

Programma, kas teksta datnē saņem katra burtā biežumu.

Kods:

```
import os  
import sys  
  
# jāpievieno pareizas ceļš līdz datnei  
datnes_cels = 'C:/Users/Simona/Desktop/lu/programmesana un datori  
I/2sem/1MPR14/Simona_Blinova_1MPR14_programmas_un_datnes/uzd2 test1.txt'  
if not os.path.isfile(datnes_cels):  
    print(f'Kļūda: Datne "{datnes_cels}" neeksistē.')  
    sys.exit(1)  
  
with open('uzd2 test1.txt', 'w', encoding='utf-8') as datne:  
    while True:  
        rinda = input('Ievadiet teksta rindu --> ')  
        datne.write(rinda + '\n')  
        print("")  
  
        paz = input('Vai vēlaties ievadīt vēl rindu? (n - nē) --> ')  
        print("")  
  
        if paz == 'n':  
            break
```

```
datne.close()
```

```
vardnica = {}
```

```
with open('uzd2 test1.txt', 'r', encoding='utf-8') as datne:
```

```
    simbols = datne.read(1)
```

```
    while simbols != "":
```

```
        while simbols != '\n' and simbols != "":
```

```
            simbols = simbols.upper()
```

```
            if 64 < ord(simbols) < 91:
```

```
                if simbols in vardnica:
```

```
                    vardnica[simbols] += 1
```

```
            else:
```

```
                vardnica[simbols] = 1
```

```
            simbols = datne.read(1)
```

```
        if simbols == '\n':
```

```
            simbols = datne.read(1)
```

```
#print(vardnica)
```

```
burti = list(vardnica.keys())
```

```
#print(burti)
```

```
skaiti = list(vardnica.values())
```

```
#print(skaiti)
```

```
garums = len(skaiti)
```

```
atkartojumi = garums - 1
```

```
pazime = True
```

```
while pazime:
```

```
    pazime = False
```

```
    for j in range(0, atkartojumi):
```

```
        if skaiti[j] < skaiti[j+1]:
```

```
            pazime = True
```

```
        b = skaiti[j]
```

```
        skaiti[j] = skaiti[j+1]
```

```
        skaiti[j+1] = b
```

```
    b1 = burti[j]
```

```
    burti[j] = burti[j+1]
```

```
burti[j+1] = b1
```

```
atkartojumi -= 1
```

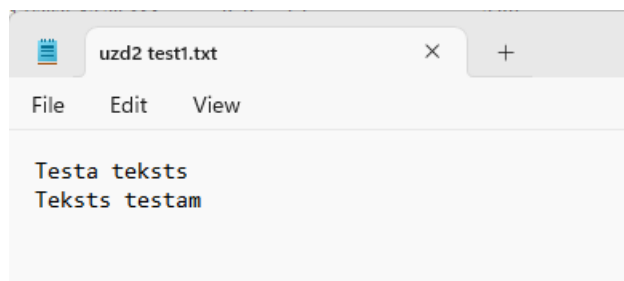
```
#print(burti)
```

```
#print(skaiti)
```

```
for i in range(len(burti)):
    print(burti[i], '->', skaiti[i])
```

Testa piemērs(1)

Teksta datne 'uzd2 test1.txt':



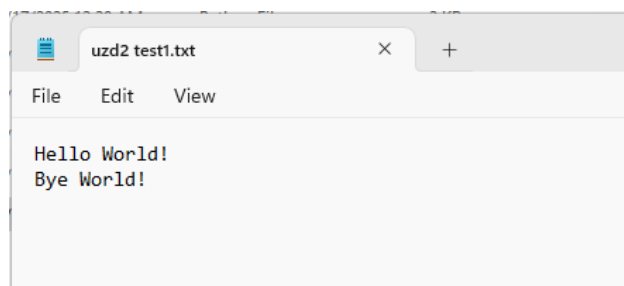
Terminālis:

```
Ievadiet teksta rindu --> Testa teksts
Vai vēlaties ievadīt vēl rindu? (n - nē) --> j
Ievadiet teksta rindu --> Teksts testam
Vai vēlaties ievadīt vēl rindu? (n - nē) --> n

T -> 8
S -> 6
E -> 4
A -> 2
K -> 2
M -> 1
```

Testa piemērs(2)

Teksta datne 'uzd2 test1.txt':



Terminālis:

```
Ievadiet teksta rindu --> Hello World!

Vai vēlaties ievadīt vēl rindu? (n - nē) --> j

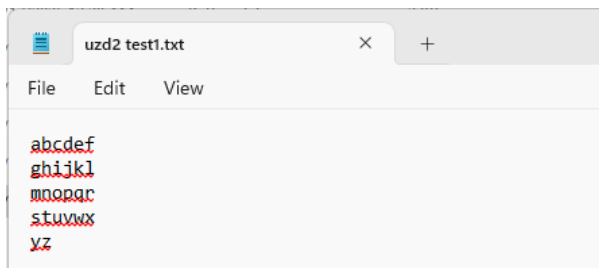
Ievadiet teksta rindu --> Bye World!

Vai vēlaties ievadīt vēl rindu? (n - nē) --> n

L -> 4
O -> 3
E -> 2
W -> 2
R -> 2
D -> 2
H -> 1
B -> 1
Y -> 1
```

Testa piemērs(3)

Teksta datne 'uzd2 test1.txt':



The screenshot shows a text editor window with the title 'uzd2 test1.txt'. The menu bar includes 'File', 'Edit', and 'View'. The text content of the file is as follows:

```
abcdef
ghijkl
mnopqr
stuvwx
yz
```

Terminālis:

```
Ievadiet teksta rindu --> abcdef

Vai vēlaties ievadīt vēl rindu? (n - nē) --> j

Ievadiet teksta rindu --> ghijkl

Vai vēlaties ievadīt vēl rindu? (n - nē) --> j

Ievadiet teksta rindu --> mnopqr

Vai vēlaties ievadīt vēl rindu? (n - nē) --> j

Ievadiet teksta rindu --> stuvwx

Vai vēlaties ievadīt vēl rindu? (n - nē) --> j

Ievadiet teksta rindu --> yz

Vai vēlaties ievadīt vēl rindu? (n - nē) --> n

A -> 1
B -> 1
C -> 1
D -> 1
E -> 1
F -> 1
G -> 1
H -> 1
I -> 1
J -> 1
K -> 1
L -> 1
M -> 1
N -> 1
O -> 1
P -> 1
Q -> 1
R -> 1
S -> 1
T -> 1
U -> 1
V -> 1
W -> 1
X -> 1
Y -> 1
Z -> 1
```

3.uzdevums

Programma, kas veic teksta šifrēšanu un atšifrēšanu ar Cēzara algoritmu.

Kods:

```
teksts = input('Ievadiet tekstu --> ')
darb = input('Ko vēlaties darīt? (c - šifrēt, d - atšifrēt) --> ')
solis = int(input('Ievadiet šifrēšanas soli --> '))
```

```
teksts = teksts.upper()
```

```
simboli = []
for i in teksts:
    simboli.append(i)
```

```
#print(simboli)
```

```
if darb == 'c':
    solis = -solis
```

```
for i in range(len(simboli)):
    #print(i)
    burts = simboli[i]
    #print(burts)
```

```
if 64 < ord(burts) < 91:
    jaun_burts = ord(burts) + solis
    if jaun_burts <= 64:
        plus_solis = 65 - jaun_burts
        jaun_burts = 91 - plus_solis
```

```
if jaun_burts >= 91:
    plus_solis = jaun_burts - 90
    jaun_burts = 64 + plus_solis
```

```
simboli[i] = chr(jaun_burts)
```

```
#print(simboli)
```

```
jaun_teksts = ""
for elem in simboli:
```



```
jaun_teksts += elem
```

```
print(jaun_teksts)
```

Testa piemērs(1)

```
Ievadiet tekstu --> Hello world!  
Ko vēlaties darīt? (c - šifrēt, d - atšifrēt) --> c  
Ievadiet šifrēšanas soli --> 4  
DAHKK SKNHZ!
```

Testa piemērs(2)

```
Ievadiet tekstu --> DAHKK SKNHZ!  
Ko vēlaties darīt? (c - šifrēt, d - atšifrēt) --> d  
Ievadiet šifrēšanas soli --> 4  
HELLO WORLD!
```

Testa piemērs(3)

```
Ievadiet tekstu --> abcdef  
Ko vēlaties darīt? (c - šifrēt, d - atšifrēt) --> c  
Ievadiet šifrēšanas soli --> 3  
XYZABC
```

PU1

Programma, kas teksta datnē saņem katra latviešu burtu biežumu.

Kods:

```
import os
```

```
import sys
```

```
def ievietosanas_metode(a, b):
```

```
    garums = len(a)
```

```
    for i in range(1, garums):
```

```
        x = a[i]
```

```
        xb = b[i]
```

```
        j = i
```

```
        while j > 0 and a[j-1] < x:
```

```
            a[j] = a[j-1]
```

```
            b[j] = b[j-1]
```

```
            j -= 1
```

```
        a[j] = x
```

```
        b[j] = xb
```

```
    return a, b
```

```
datnes_cels = 'C:/Users/Simona/Desktop/lu/programmesana un datori  
I/2sem/1MPR14/Simona_Blinova_1MPR14_programmas_un_datnes/PU1 test.txt'  
if not os.path.isfile(datnes_cels):  
    print(f'Kļūda: Datne "{datnes_cels}" neeksistē.')  
    sys.exit(1)
```

```
with open('PU1 test.txt', 'w', encoding='utf-8') as datne:
```

```
    while True:
```

```
        rinda = input('Ievadiet teksta rindu --> ')
```

```
        datne.write(rinda + '\n')
```

```
        print("")
```

```
        paz = input('Vai vēlaties ievadīt vēl rindu? (n - nē) --> ')
```

```
        print("")
```

```
        if paz == 'n':
```

```
            break
```

```
    datne.close()
```

```
vardnica = {}
```

```
lat_burti = [256, 268, 274, 286, 298, 310, 315, 325, 352, 362, 381]
```

```
with open('PU1 test.txt', 'r', encoding='utf-8') as datne:
```

```
    simbols = datne.read(1)
```

```
    while simbols != "":
```

```
        while simbols != '\n' and simbols != "":
```

```
            simbols = simbols.upper()
```

```
            if 64 < ord(simbols) < 81 or 81 < ord(simbols) < 87 or ord(simbols) == 90 or ord(simbols) in
```

```
lat_burti:
```

```
                if simbols in vardnica:
```

```
                    vardnica[simbols] += 1
```

```
                else:
```

```
                    vardnica[simbols] = 1
```

```
            simbols = datne.read(1)
```

```
        if simbols == '\n':
```

```
            simbols = datne.read(1)
```

```
#print(vardnica)
```

```
burti = list(vardnica.keys())
```

```
#print(burti)
```

```

skaiti = list(vardnica.values())
#print(skaiti)

#print(burti)
#print(skaiti)

skaiti, burti = ievietosanas_metode(skaiti, burti)

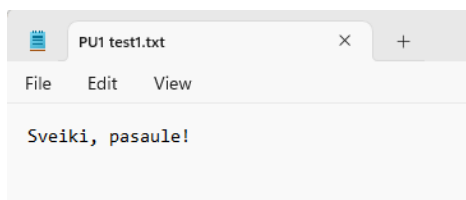
#print(burti)
#print(skaiti)

for i in range(len(burti)):
    print(burti[i], '->', skaiti[i])

```

Testa piemērs(1)

Teksta datne 'PU1 test.txt':



Terminālis:

```

Ievadiet teksta rindu --> Sveiki, pasaule!

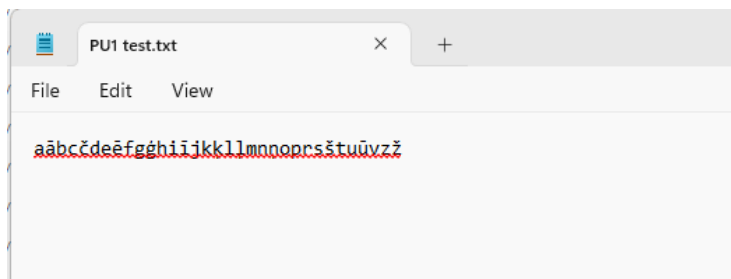
Vai vēlaties ievadīt vēl rindu? (n - nē) --> n

S -> 2
E -> 2
I -> 2
A -> 2
V -> 1
K -> 1
P -> 1
U -> 1
L -> 1

```

Testa piemērs(2)

Teksta datne 'PU1 test.txt':

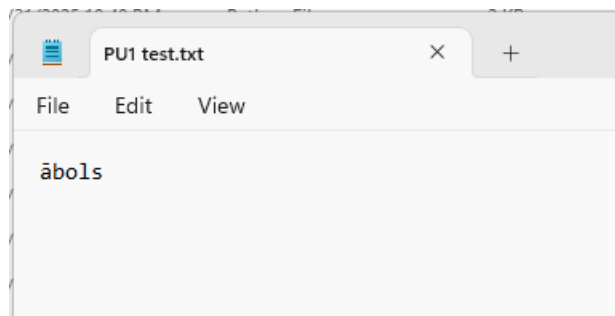


Terminālis:

```
Ievadiet teksta rindu --> aābcċdeēfgġhiījkķlļmnņoprsštuūvzž  
Vai vēlaties ievadīt vēl rindu? (n - nē) --> n  
A -> 1  
Ā -> 1  
B -> 1  
C -> 1  
Č -> 1  
D -> 1  
E -> 1  
Ē -> 1  
F -> 1  
G -> 1  
H -> 1  
I -> 1  
Ī -> 1  
J -> 1  
K -> 1  
Ķ -> 1  
L -> 1  
Ļ -> 1  
M -> 1  
N -> 1  
Ņ -> 1  
O -> 1  
P -> 1  
R -> 1  
S -> 1  
Š -> 1  
T -> 1  
U -> 1  
Ū -> 1  
V -> 1  
Z -> 1  
Ž -> 1
```

Testa piemērs(3)

Teksta datne 'PU1 test.txt':



Terminālis:

```
Ievadiet teksta rindu --> ābols  
Vai vēlaties ievadīt vēl rindu? (n - nē) --> n  
Ā -> 1  
B -> 1  
O -> 1  
L -> 1  
S -> 1
```

PU2

Programma, kas veic teksta šifrēšanu un atšifrēšanu ar Cēzara šifru.

Kods:

```
lat_burti = ['Ā', 'Č', 'Ē', 'Ģ', 'Ī', 'Ķ', 'Ļ', 'Ņ', 'Š', 'Ū', 'Ž']  
alfabets = []
```

```
for i in range(65, 81):  
    alfabets.append(chr(i))
```

```
for i in range(82, 87):  
    alfabets.append(chr(i))
```

```
alfabets.append(chr(90))
```

```
i = 0
```

```
burts = alfabets[i]
```

```
while burts != 'Ž':
```

```
    match burts:
```

```
        case 'A':
```

```
            alfabets.insert(i+1, 'Ā')
```

```
        case 'C':
```

```
            alfabets.insert(i+1, 'Č')
```

```
        case 'E':
```

```
            alfabets.insert(i+1, 'Ē')
```

```
        case 'G':
```

```
            alfabets.insert(i+1, 'Ģ')
```

```
        case 'I':
```

```
            alfabets.insert(i+1, 'Ī')
```

```
        case 'K':
```

```
            alfabets.insert(i+1, 'Ķ')
```

```
        case 'L':
```

```
            alfabets.insert(i+1, 'Ļ')
```

```
        case 'N':
```

```
            alfabets.insert(i+1, 'Ņ')
```

```
        case 'S':
```

```
            alfabets.insert(i+1, 'Š')
```

```
        case 'U':
```

```
            alfabets.insert(i+1, 'Ū')
```

```
        case 'Z':
```

```

        alfabets.insert(i+1, 'Ž')
    i += 1
    burts = alfabets[i]

#print(alfabets)

teksts = input('Ievadiet tekstu --> ')
darb = input('Ko vēlaties darīt? (c - šifrēt, d - atšifrēt) --> ')
solis = int(input('Ievadiet šifrēšanas soli --> '))

teksts = teksts.upper()

simboli = []
for i in teksts:
    simboli.append(i)

if darb == 'c':
    solis = -solis

for i in range(len(simboli)):
    #print(i)
    burts = simboli[i]
    vieta = alfabets.index(burts)
    #print(vieta)

    if -1 < vieta < len(alfabets):
        jaun_vieta = vieta + solis
        #print(jaun_vieta)
        if jaun_vieta <= -1:
            jaun_vieta = len(alfabets) + jaun_vieta
            #print(jaun_vieta)

        if jaun_vieta >= len(alfabets):
            plus_solis = jaun_vieta - len(alfabets)
            #print(plus_solis)
            jaun_vieta = 0 + plus_solis

    simboli[i] = alfabets[jaun_vieta]

#print(simboli)

```

```
jaun_teksts = ""
for elem in simboli:
    jaun_teksts += elem
```

```
print(jaun_teksts)
```

Testa piemērs(1)

```
Ievadiet tekstu --> ābols
Ko vēlaties darīt? (c - šifrēt, d - atšifrēt) --> c
Ievadiet šifrēšanas soli --> 3
ZŽMJO
```

Testa piemērs(2)

```
Ievadiet tekstu --> aābcčdeē
Ko vēlaties darīt? (c - šifrēt, d - atšifrēt) --> c
Ievadiet šifrēšanas soli --> 6
TUŪVZŽĀĀ
```

Testa piemērs(3)

```
Ievadiet tekstu --> TUŪVZŽĀĀ
Ko vēlaties darīt? (c - šifrēt, d - atšifrēt) --> d
Ievadiet šifrēšanas soli --> 6
AĀBCČDEĒ
```