

Caesar Cipher

```
1 def a2d(text):
2     return [ord(i) for i in text]
3
4
5 def encrypt(text, key):
6     dtext = a2d(text)
7     result = []
8     for i in dtext:
9         if (i ≥ 65 and i ≤ 90):
10            result.append((((i - 65) + key) % 26) + 65)
11        elif(i ≥ 97 and i ≤ 122):
12            result.append((((i - 97) + key) % 26) + 97)
13        else:
14            result.append(i)
15    final = list(map(chr, result))
16    return ''.join(final)
17
18
19 def decrypt(text, key):
20     dtext = a2d(text)
21     result = []
22     for i in dtext:
23         if (i ≥ 65 and i ≤ 90):
24            result.append((((i - 65) - key) % 26) + 65)
25        elif(i ≥ 97 and i ≤ 122):
26            result.append((((i - 97) - key) % 26) + 97)
27        else:
28            result.append(i)
29    final = list(map(chr, result))
30    return ''.join(final)
31
32
33 if __name__ == '__main__':
34     text = input("Before encryption Plain text: ")
35     key = int(input("Key: "))
36
37     cipher = encrypt(text, key)
38     print("Cipher text : {}".format(cipher))
39
40     plain = decrypt(cipher, key)
41     print("After decryption Plain text : {}".format(plain))
```

Output:

```
Before encryption Plain text: ABCDEFGHIJKLMNOPQRSTUVWXYZ
Key: 23
Cipher text : XYZABCDEFGHIJKLMNPOQRSTUVWXYZ
After decryption Plain text : ABCDEFGHIJKLMNOPQRSTUVWXYZ

***Repl Closed***
```

Playfair Cipher

```
playfair_cipher.py
1 import string
2 from collections import OrderedDict
3 import numpy as np
4 from ordered_set import OrderedSet
5
6 def key_text_rule(key):
7     for j in range(len(key)):
8         for i in range(len(key)):
9             if ((i%2==0) and (i+1!=len(key))):
10                 if ((key[i] == (key[i+1]))):
11                     near = i+1
12                     key = key[:near] + 'x' + key[near:]
13                     break
14
15     if (len(key)%2!=0):
16         key = key[:len(key)+1] + 'z'
17     return key
18 else:
19     return key
20
21 def matrix_fill(key):
22     key = "".join(OrderedDict.fromkeys(key)) ## remove the repeated characters in the string
23     str1 = string.ascii_lowercase
24     for i in key:
25         if i in str1:
26             str1 = str1.replace(i, '')
27     str1 = str1.replace('j', 'i')
28     matrix_elements = key + str1
29
30     list1 = []
31     ind = 0
32     for i in range(5):
33         temp = []
34         for j in range(5):
35             temp.append(matrix_elements[ind])
36             ind+=1
37         list1.append(temp)
38
39     return list1
40
41 #####
42
43 ## encryption
44
45 def same_row_encrypt(ind1,ind2,ind3,ind4,matrix): ## Same 1st index(i.e i)
46
47     ## loop
48     if (ind2==4 or ind4==4):
49         if (ind2==4):
50             ind2 = 0
51             print(matrix[ind1][ind2])
52             print(matrix[ind3][ind4+1])
53
54         if (ind4==4):
55             ind4 = 0
56             print(matrix[ind1][ind2+1])
57             print(matrix[ind3][ind4])
58
59     ## not a loop
60     else:
61         print(matrix[ind1][ind2+1])
62         print(matrix[ind3][ind4+1])
63
64 def same_col_encrypt(ind1,ind2,ind3,ind4,matrix): ## Same 2nd index(i.e j)
65
66     ## loop
67     if (ind1==4 or ind3==4):
68         if (ind1==4):
69             ind1 = 0
70             print(matrix[ind1][ind2])
71             print(matrix[ind3+1][ind4])
72
73         if (ind3==4):
74             ind3 = 0
75             print(matrix[ind1+1][ind2])
76             print(matrix[ind3][ind4])
77
78     ## not a loop
79     else:
80         print(matrix[ind1+1][ind2])
81         print(matrix[ind3+1][ind4])
82
83 def diff(ind1,ind2,ind3,ind4,matrix): ## Not in same row and same column
84     print(matrix[ind1][ind4])
85     print(matrix[ind3][ind2])
86
87 def encrypt(i_index, j_index, matrix):
88     for ind in range(len(i_index)):
89         if ((ind%2==0) and ind!=len(i_index)):
90
91             if (i_index[ind]==i_index[ind+1]): ## same i-value
92                 same_row_encrypt(i_index[ind],j_index[ind],i_index[ind+1],j_index[ind+1],matrix)
93
```

```

94         elif (j_index[ind]==j_index[ind+1]): ## same j-value
95             same_col_encrypt(i_index[ind],j_index[ind],i_index[ind+1],j_index[ind+1],matrix)
96
97         else:
98             diff(i_index[ind],j_index[ind],i_index[ind+1],j_index[ind+1],matrix)
99
100 ## decryption
101
102 def same_row_decrypt(ind1,ind2,ind3,ind4,matrix): ## Same 1st index(i.e i)
103
104     print(end='')
105
106     ## loop
107     if (ind2==0 or ind4==0):
108         if (ind2==0):
109             ind2 = 4
110             print(matrix[ind1][ind2])
111             print(matrix[ind3][ind4-1])
112
113         if (ind4==0):
114             ind4 = 4
115             print(matrix[ind1][ind2-1])
116             print(matrix[ind3][ind4])
117
118     ## not a loop
119     else:
120         print(matrix[ind1][ind2-1])
121         print(matrix[ind3][ind4-1])
122
123 def same_col_decrypt(ind1,ind2,ind3,ind4,matrix): ## Same 2nd index(i.e j)
124     print(end='')
125
126     ## loop
127     if (ind1==0 or ind3==0):
128         if (ind1==0):
129             ind1 = 4
130             print(matrix[ind1][ind2])
131             print(matrix[ind3-1][ind4])
132
133         if (ind3==0):
134             ind3 = 4
135             print(matrix[ind1-1][ind2])
136             print(matrix[ind3][ind4])
137
138     ## not a loop
139     else:
140         print(matrix[ind1-1][ind2])
141         print(matrix[ind3-1][ind4])
142
143 def decrypt(i_index, j_index, matrix):
144     for ind in range(len(i_index)):
145         if ((ind%2==0) and ind!=len(i_index)):
146
147             if (i_index[ind]==i_index[ind+1]): ## same i-value
148                 same_row_decrypt(i_index[ind],j_index[ind],i_index[ind+1],j_index[ind+1],matrix)
149
150             elif (j_index[ind]==j_index[ind+1]): ## same j-value
151                 same_col_decrypt(i_index[ind],j_index[ind],i_index[ind+1],j_index[ind+1],matrix)
152
153             else:
154                 diff(i_index[ind],j_index[ind],i_index[ind+1],j_index[ind+1],matrix)
155
156 #####3
157
158 def index_fill(plain_text, matrix,mode):
159
160     i_index = []
161     j_index = []
162
163     for k in range(len(plain_text)):
164         if (k%2==0) and (k+1!=len(plain_text)):
165             word_1 = plain_text[k]
166             word_2 = plain_text[k+1]
167
168             ## fiding the letters in the matrix
169             for i in range(5):
170                 for j in range(5):
171                     if ((word_1==matrix[i][j])):
172                         i_index.append(i)
173                         j_index.append(j)
174
175             for i in range(5):
176                 for j in range(5):
177                     if ((word_2==matrix[i][j])):
178                         i_index.append(i)
179                         j_index.append(j)
180
181     if mode=='encryption':
182         print("Cipher text")
183         encrypt(i_index, j_index,matrix)
184
185     elif mode=='decryption':
186         print("Plain text")
187         decrypt(i_index, j_index,matrix)
188

```

```

190
191     ## encryption
192     key = key_text_rule('monarchy')
193     plain_text = key_text_rule('instruments')
194     matrix = matrix_fill(key)
195
196     index_fill(plain_text, matrix, 'encryption')
197
198     ## decryption
199     cipher_text = key_text_rule('gatlmzclrqtx')
200     index_fill(cipher_text, matrix, 'decryption')

```

```

190
191     ## encryption
192     key = key_text_rule('monarchy')
193     plain_text = key_text_rule('instruments')
194     matrix = matrix_fill(key)
195
196     index_fill(plain_text, matrix, 'encryption')
197
198     ## decryption
199     cipher_text = key_text_rule('gatlmzclrqtx')
200     index_fill(cipher_text, matrix, 'decryption')

```

Output

playfair_cipher.py
REPL [python]

Cipher text
g
a
t
l
m
z
c
l
r
q
t
x
Plain text
i
n
s
t
r
u
m
e
n
t
s
z

Repl Closed

Git-hub link : https://github.com/PrashanthSingaravelan/winter_semester-2022/tree/main/CSI3002%20Applied%20Cryptography%20and%20Network%20Security/lab%20exercises