T.C.
İSTANBUL ÜNİVERSİTESİ
YÖK-TEBİP ÜSTÜN BAŞARILILAR PROGRAMI

Lisans Bitirme Tezi

PYTHON İLE VERİ ŞİFRELEME VE DEŞİFRELEME

Altuğ BEYHAN

Matematik Anabilim Dalı
Matematik Programı

DANIŞMAN
Prof. Dr. Gülçin ÇİVİ BİLİR, İ.T.Ü.

Mayıs 2023
İSTANBUL

ÖNSÖZ

Yaklaşık bir buçuk yıl önce değerli danışman hocam Prof. Dr. Gülçin ÇİVİ BİLİR sayesinde başlayan kriptoloji yolculuğum hayatıma anlam kazandırdı. Bununla birlikte bilgisayar bilimlerine olan merakım kendimi programlama alanında geliştirmemi sağladı. Böylece çok sevdiğim ve hayatım boyunca çalışmaktan sıkılmayacağım ilgi alanlarımı bulmuş oldum. Bu alanları birleştirecek bir çalışma yapmak istemem üzerine danışman hocamın yönlendirmesiyle bitirme tezimin konusu ve içeriği ortaya çıktı.

Başta beni sürekli yönlendiren, çalışmaya teşvik eden, çalışmalarımı takdir eden ve hayatıma anlam kazandıran danışman hocam Prof. Dr. Gülçin ÇİVİ BİLİR olmak üzere, beni hep destekleyen aileme ve her türlü zorlukta yanımda olan sevgili Berra Beray BEK'e teşekkürlerimi sunarım.

Bitirme tezimin hayallerime giden yolun kapılarını bana açacağına içtenlikle inanıyorum.

Altuğ BEYHAN

Mayıs 2023

ÖZET

Bu tezde, dünya çapında popüler bir programlama dili olan Python'ın kriptoloji uygulamalarıyla birlikte tanıtılması amaçlanmaktadır. Bu amaç doğrultusunda birinci bölümde Python programlama dilinin kısa bir tarihi, dünya genelindeki kullanım istatistikleri ve en çok kullanılan bazı tümleşik geliştirme ortamları hakkında bilgi verilmektedir. İkinci bölümde temel Python programlama bilgisi çeşitli kodlama uygulamaları ile ele alınmaktadır. Bu uygulamalar için yazılan kodlar, bir GitHub deposuna dahil edilerek tezde paylaşılmıştır. Üçüncü bölümde ise kriptolojiye ilişkin temel kavramlar açıklanmış ve örneklendirilmiştir. Ardından, Python programlama dili için tanımlanmış başlıca modül ve paketler tanıtılmış ve bu araçlar kullanılarak bazı şifreleme algoritmalarının Python ile uygulaması yapılmıştır. Tezin Ek

bölümünde ise Türkçe alfabe için tanımlanmış ve AES gibi modern kriptoloji algoritmalarının Türkçe metinlere uygulanmasında kullanılan ASCII Code Page-857 tablosuna yer verilmiştir.

Anahtar kelimeler: Python, Kriptoloji, Sezar Algoritması, Gelişmiş Şifreleme Standardı (AES), NumPy, PyCryptodome, Cryptography, Hashlib, Güvenli Özet Algoritması (SHA).

SUMMARY

This thesis aims to introduce Python, a globally popular programming language, along with its cryptography applications. In line with this objective, the first chapter provides information about the brief history of the Python programming language, usage statistics worldwide, and some commonly used integrated development environments. The second chapter covers basic Python programming knowledge through various coding applications. The codes written for these applications is included in a GitHub repository and shared in the thesis. The third chapter explains and exemplifies fundamental concepts related to cryptography. Subsequently, the main modules and packages defined for the Python programming language are introduced, and the implementation of some encryption algorithms using Python with these tools is demonstrated. In the thesis's Appendix section, the ASCII Code Page-857 Turkish Language table, which is used for applying modern cryptography algorithms such as AES to Turkish texts is presented.

Keywords: Python, Cryptology, Caesar Algorithm, Advanced Encryption Standard (AES), NumPy, PyCryptodome, Cryptography, Hashlib, Secure Hash Algorithm (SHA).

TARTIŞMA VE SONUÇ

Bu çalışmada ana hedef , dünya genelinde popüler bir programlama dili haline gelen Python programlama dilinin şifre bilimi alanındaki uygulamalarına odaklanmak ve bu sahaya giriş yapmak isteyen kişilere temel bir kaynak hazırlamak olmuştur. Bu bağlamda öncelikle, Python programlama dilinin kriptoloji için bir araç olarak kullanılmasına yönelik çeşitli yöntemler incelenmiş ve gerekli temel programlama bilgileri ele alınmıştır. Python'ın kolay ve anlaşılır bir söz dizimine sahip olması, kullanıcıları projelerinde çeşitli amaçlara hizmet etmesi için Python'ı kullanmaya teşvik etmektedir. Özel olarak, kriptoloji alanında halihazırda Python için geliştirilmiş birçok modül, paket ve kütüphane bulunmaktadır ve gün geçtikçe hem bu araçlar geliştirilmekte hem de bu araçların yanına ek olarak yeni araçlar eklenmeye devam etmektedir. Bu nedenle, kriptoloji projelerinde Python programlama dilini kullanmak birçok avantajı beraberinde getirmektedir. Tez kapsamında ele alınan konular, yeni bir kriptoloji algoritmasının Python programlama diliyle geliştirilebilmesi için de gereken temel bilgileri sağlamaktadır. Söz konusu, bu tezin hazırlanış sürecinde Fibonacci Polinomları ile yeni bir kriptoloji algoritması geliştirilmeye başlanmıştır. Bu algoritma, Python kodları ile yakın tarihte bir dergiye gönderilmek üzere yayına hazırlanmaktadır. Geliştirilen yeni şifreleme yönteminin literatüre önemli bir katkı sağlayacağına ve yeni çalışmalara referans olacağına inanılmaktadır.

KAYNAKLAR

Ascii Codes: Code Page 857 (Turkish Language). https://www.ascii-codes.com/cp857.html (Erişim Tarihi: 19.05.2023).

Brookshear, J.G., Brylow, D., "Bilgisayar Bilimine Giriş", 12. Basımdan Çeviri, Nobel Akademik Yayıncılık, Ankara, 2018.

Cryptography Documentation. https://cryptography.io/en/latest (Erişim Tarihi: 31.05.2023).

Çivi Bilir, G., "Sezar'ın Anahtarı: Geçmişten Günümüze Klasik Şifreleme Yöntemleri", İTÜ Yayınevi, Baskıda.

FIPS Publication 197, NIST, 2001. https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.197.pdf (Erişim Tarihi: 19.05.2023).

Lutz, M., "Programming Python", 1st Edition, O'Reilly & Associates, Inc., United States of America, 1996.

NumPy Documentation. https://numpy.org/doc/stable/index.html (Erişim Tarihi: 28.05.2023).

PyCryptodome Documentation. https://www.pycryptodome.org (Erişim Tarihi: 28.05.2023).

Python Developer's Guide. https://devguide.python.org/versions (Erişim Tarihi: 21.04.2023).

Python Documentation. https://docs.python.org/3 (Erişim Tarihi: 31.05.2023). Samancıoğlu, A., "Python Sıfırdan Uzmanlığa Programlama", 6. Baskı, Unikod Yayıncılık, İstanbul, 2023.

TIOBE Index. https://www.tiobe.com/tiobe-index (Erişim Tarihi: 31.05.2023).

Trappe, W., Washington, L. C., "Introduction to Cryptography with Coding Theory", 2nd Edition, Pearson Education, Inc., United States of America, 2006.

Ural, N., Örenç, Ö., "Uygulamalı Şifreleme ve Şifre Çözme Yöntemleri", 4. Baskı, Pusula Yayıncılık, İstanbul, 2020.

EKLER

ASCII-CODE PAGE 857 TABLOSU

ASCII, Bilgi Değişimi için Amerikan Standart Kodlama Sistemi (American Standard Code for Information Interchange) ifadesinin kısaltmasıdır. Orijinal ASCII tablosu İngilizce alfabe için hazırlanmış olup 128 karaktere sahip olmakla birlikte farklı alfabeler için daha fazla karaktere sahip olması adına 256 karaktere genişletilmiş çeşitli tablolar bulunmaktadır. Türkçe alfabe için Code Page 857 tablosu belirlenmiştir (Ascii Codes: CP-857).

Not: Code Page 857 tablosunda onluk tabanda 32'ye "boşluk", 240'a "yumuşak tire" ve 255'e "bölünemez boşluk" karakterleri karşılık gelmiştir. Ayrıca onluk tabanda 213, 231 ve 242'ye karşılık karakter atamaları yapılmamıştır.

TABLOYA BURADAN ULAŞABİLİRSİNİZ: https://github.com/altugbeyhan/ASCII-CP857-TURKISH-ALPHABET-TABLE

BU TEZDE YER ALAN PROGRAMLAR AŞAĞIDA VERİLMİŞTİR.

PROGRAMLAR

```
1 print("Merhaba Dünya!")
```

```
Merhaba Dünya!
```

```
1 #Merhaba Dünya Programı
2 print("Merhaba Dünya!") #print: çıktı veren fonksiyondur
3 #print("Bu yazı ekranda gözükmeyecek")
4 print("Benim Adım Altuğ Beyhan.")
```

```
Merhaba Dünya!
Benim Adım Altuğ Beyhan.
```

```
1 """
2 Altuğ Beyhan
3 2023
4 www.altugbeyhan.com
5 """
6 print("Merhaba Dünya!")
```

```
Merhaba Dünya!
```

```
1 degisken = "Merhaba Dünya!"
2 print(degisken)
```

```
Merhaba Dünya!
```

```
1 degisken = "Merhaba Dünya!"
2 print(degisken)
```

```
Merhaba Dünya!
```

```
help("keywords")
```

```
Here is a list of the Python keywords.  Enter any keyword to get more help.

False               class               from                or
None                continue            global              pass
True                def                 if                  raise
and                 del                 import              return
as                  elif                in                  try
assert              else                is                  while
async               except              lambda              with
await               finally             nonlocal            yield
break               for                 not
```

```
print(type("Merhaba Dünya!"))
```

```
<class 'str'>
```

```
print("Merhaba Dünya!", type("Merhaba Dünya!"))
print(17, type(17))
print(3.14, type(3.14))
print(3+4j, type(3+4j))
print(["altuğ", "beyhan", 22], type(["altuğ", "beyhan", 22]))
print((8,15,17), type((8,15,17)))
print({"elma":"apple", "kiraz":"cherry"}, type({"elma":"apple", "kiraz":"cherry"}))
print({1, 2, 3, 4, 5}, type({1, 2, 3, 4, 5}))
print(True, type(True))
```

```
Merhaba Dünya! <class 'str'>
17 <class 'int'>
3.14 <class 'float'>
(3+4j) <class 'complex'>
['altuğ', 'beyhan', 22] <class 'list'>
(8, 15, 17) <class 'tuple'>
{'elma': 'apple', 'kiraz': 'cherry'} <class 'dict'>
{1, 2, 3, 4, 5} <class 'set'>
True <class 'bool'>
```

```
isim_soyisim = "Altuğ Beyhan"
kisilik_ozellikleri = ["Zeki","Çalışkan","Yardımsever"]
kisisel_bilgiler = ("Tekirdağlı","22 Yaş", "Erkek", "Kriptograf")

print(isim_soyisim[0]) # 0. indis = 1. karakter
print(isim_soyisim[-1]) # -1. indis = sonuncu karakter

print(kisilik_ozellikleri[1]) # 1. indis = 2. eleman
print(kisilik_ozellikleri[-2]) # -2. indis = sondan 2. eleman

print(kisisel_bilgiler[2]) # 2. indis = 3. eleman
print(kisisel_bilgiler[-3]) # -3. indis = sondan 3. eleman
```

```
A
n
Çalışkan
Çalışkan
Erkek
22 Yaş
```

```
isim = "altuğ beyhan"
sayilar = [1,2,3,4,5,6,7,8,9,10]
unluler = ("a","e","ı","i","o","ö","u","ü")

print(isim[1:10:1]) # 1, 2, ..., 9. indisler (10. indis hariç)
print(sayilar[0:9:2]) # 0, 2, 4, 6, 8. indisler
print(unluler[1:8:3]) # 1, 4, 7. indisler
```

```
ltuğ beyh
[1, 3, 5, 7, 9]
('e', 'o', 'ü')
```

```
isim = "altuğ beyhan"
sayilar = [1,2,3,4,5,6,7,8,9,10]
unluler = ("a","e","ı","i","o","ö","u","ü")

print(isim[::-1])
```

```
print(sayilar[::-1])
print(unluler[::-1])
```

```
nahyeb ğutla
[10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
('ü', 'u', 'ö', 'o', 'i', 'ı', 'e', 'a')
```

```
listem = [1, 2, 3]
listem.append(4)
print(listem)
```

```
[1, 2, 3, 4]
```

```
def merhaba():
    isim = input("Adınızı giriniz: ")
    print(f"Merhaba {isim}!")

merhaba()
```

```
Adınızı giriniz: Altuğ Beyhan
Merhaba Altuğ Beyhan!
```

```
def bilgi_ver(isim, soyisim, yas):
    print("KİŞİSEL BİLGİLER")
    print("İsim:", isim)
    print("Soyisim:", soyisim)
    print("Yaş:", yas)

bilgi_ver("Altuğ", "Beyhan", 22)
```

```
KİŞİSEL BİLGİLER
İsim: Altuğ
Soyisim: Beyhan
Yaş: 22
```

```
x = "altuğ BEYHAN"

print(x.capitalize())

print(x.count("a"))

print(x.index("t"))

print(x.lower())

print(x.replace("a", "A"))

print(x.split((" ")))

print(x.upper())
```

```
Altuğ beyhan
1
2
altuğ beyhan
Altuğ BEYHAN
['altuğ', 'BEYHAN']
ALTUĞ BEYHAN
```

```
l = ["a", "b", "c", "a", "b", "c"]
l.append("d")
print(l)

l = ["a", "b", "c", "a", "b", "c"]
l.clear()
print(l)

l = ["a", "b", "c", "a", "b", "c"]
l2 = l.copy()
print(l2)

l = ["a", "b", "c", "a", "b", "c"]
print(l.count("a"))

l = ["a", "b", "c", "a", "b", "c"]
l.extend(("d", "e"))
print(l)
```

```
19
20 l = ["a", "b", "c", "a", "b", "c"]
21 print(l.index("c"))
22
23 l = ["a", "b", "c", "a", "b", "c"]
24 l.insert(2, "d")
25 print(l)
26
27 l = ["a", "b", "c", "a", "b", "c"]
28 l.pop(0)
29 print(l)
30
31 l = ["a", "b", "c", "a", "b", "c"]
32 l.remove("b")
33 print(l)
34
35 l = ["a", "b", "c", "a", "b", "c"]
36 l.reverse()
37 print(l)
38
39 l = ["a", "b", "c", "a", "b", "c"]
40 l.sort(reverse=False)
41 print(l)
```

```
['a', 'b', 'c', 'a', 'b', 'c', 'd']
[]
['a', 'b', 'c', 'a', 'b', 'c']
2
['a', 'b', 'c', 'a', 'b', 'c', 'd', 'e']
2
['a', 'b', 'd', 'c', 'a', 'b', 'c']
['b', 'c', 'a', 'b', 'c']
['a', 'c', 'a', 'b', 'c']
['c', 'b', 'a', 'c', 'b', 'a']
['a', 'a', 'b', 'b', 'c', 'c']
```

```
1 liste = [1,2,3]
2 liste[0] = "a" # listenin elemanları değiştirilebilir
3 print(liste)
4
5 demet = (1,2,3)
6 demet[0] = "a" # ancak demetin elemanları değiştirilemez
7 print(demet)
```

```
['a', 2, 3]
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-19-f36c2960f61a> in <cell line: 6>()
      4
      5 demet = (1,2,3)
----> 6 demet[0] = "a" # ancak demetin elemanları değiştirilemez
      7 print(demet)

TypeError: 'tuple' object does not support item assignment
```

SEARCH STACK OVERFLOW

```
1 demet = ("a","b","c","d","b","d","e","f")
2 print(demet.count("d"))
3 print(demet.index("e"))
```

```
2
6
```

```
1 sozluk = {"apple":"elma", "banana":"muz"}
2
3 print(sozluk)
4 print(sozluk["apple"])
5 print(sozluk["banana"])
```

```
{'apple': 'elma', 'banana': 'muz'}
elma
muz
```

```
1 s = {"apple":"elma", "banana":"muz"}
2
3 print(s.get("c", "yok")) #print(s["cherry"]) -> hata verir
4 print(s.items())
```

```
print(s.keys())
print(s.values())
```

```
yok
dict_items([('apple', 'elma'), ('banana', 'muz')])
dict_keys(['apple', 'banana'])
dict_values(['elma', 'muz'])
```

```
kume1 = {1,2,3,2,3,4,5}
liste = ["a","b","b","c","a","d"]
kume2 = set(liste)
print(kume1)
print(liste)
print(kume2)
```

```
{1, 2, 3, 4, 5}
['a', 'b', 'b', 'c', 'a', 'd']
{'d', 'c', 'b', 'a'}
```

```
k1 = {1,2,3,4}
k2 = {3,4,5,6}

print(k1.difference(k2))
print(k1.intersection(k2))
print(k1.symmetric_difference(k2))
print(k1.union(k2))
```

```
{1, 2}
{3, 4}
{1, 2, 5, 6}
{1, 2, 3, 4, 5, 6}
```

```
print(5>3)
print(5<3)
```

```
True
False
```

```
print(20+3)
print(20-3)
print(20*3)
print(20/3)
print(20**3)
print(20//3)
print(20%3)
```

```
23
17
60
6.666666666666667
8000
6
2
```

```
print(20<3)
print(20==3)
print(20!=3)
print(20>3)
print(20<=3)
print(20>=3)
```

```
False
False
True
True
False
True
```

```
# not (değil/olumsuz)
print(not True)

# and (ve)
print(True and False)

# or (veya)
print(True or False)
```

```
False
False
True
```

```
import numpy as np

x = 14 # 00001110
y = 21 # 00010101

print ("~x :", ~x)
# Bitsel değil: 11110001 (işaretli)
print ("~x :", np.uint8(np.array(~x)))
# Bitsel değil: 11110001 (işaretsiz)
print ("x & y: ", x & y)
# Bitsel ve: 00000100
print ("x | y: ", x | y)
# Bitsel veya: 00011111
print ("x ^ y: ", x ^ y)
# Özel veya (XOR): 00011011
print ("x << 2: ", x << 2)
# Sola (2) kaydırma: 00111000
print ("x >> 1: ", x >> 1)
# Sağa (1) kaydırma: 00000111
```

```
~x : -15
~x : 241
x & y:  4
x | y:  31
x ^ y:  27
x << 2:  56
x >> 1:  7
```

```
x = 10
x = x+5
print(x)

y = 20
y += 7
print(y)
```

```
15
27
```

```
def f(x):

    if x<2:
        return f"f({x}) = {-x**2}"

    elif 2<=x<10:
        return f"f({x}) = {x-2}"

    else:
        return f"f({x}) = {(x+6) ** (1/2)}"


print(f(-4),f(2),f(6),f(10),f(15),sep="\n")
```

```
f(-4) = -16
f(2) = 0
f(6) = 4
f(10) = 4.0
f(15) = 4.58257569495584
```

```
isim_listesi = ["Altuğ", "Beray", "Bayram", "Limon"]
for isim in isim_listesi:
    print(f"Merhaba {isim}!")
```

```
Merhaba Altuğ!
Merhaba Beray!
Merhaba Bayram!
Merhaba Limon!
```

```
sinif_no = 1
while sinif_no<=12:
    print(f"{sinif_no}. sınıf")
    sinif_no += 1
```

```
1. sınıf
2. sınıf
3. sınıf
4. sınıf
5. sınıf
6. sınıf
7. sınıf
8. sınıf
9. sınıf
10. sınıf
11. sınıf
12. sınıf
```

```
import numpy as np

anahtar = np.array([[1,2,3],[4,5,6],[7,8,9]])
print(anahtar)
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

```
import numpy as np

dizi1 = np.array([1,3,5], dtype="complex")
dizi2 = np.array([[1,2,3],[4,5,6],[7,8,9]])

print("Dizi 1:\n",dizi1)
print("Dizi 1'in 2. girdisi:",dizi1[1])
print("\nDizi 2:\n",dizi2)
print("Dizi 2'nin 3. satır-2. sütundaki girdisi:",dizi2[2][1])
```

```
Dizi 1:
 [1.+0.j 3.+0.j 5.+0.j]
Dizi 1'in 2. girdisi: (3+0j)

Dizi 2:
 [[1 2 3]
 [4 5 6]
 [7 8 9]]
Dizi 2'nin 3. satır-2. sütundaki girdisi: 8
```

```
import numpy as np

dizi1 = np.array([1,2,3])
dizi2 = np.array([4,5,6])

# ARİTMETİK OPERATÖRLER

print(np.add(dizi1, dizi2)) # karşılıklı toplama (+)

print(np.subtract(dizi1, dizi2)) # karşılıklı çıkarma (-)

print(np.multiply(dizi1, dizi2)) # karşılıklı çarpma (*)

print(np.divide(dizi1, dizi2)) # karşılıklı bölme (/)

print(np.exp(dizi1)) # elemanların e üssü

print(np.sqrt(dizi2)) # elemanların karekökü

print(np.sin(dizi1)) # elemanların sinüsü

print(np.log(dizi2)) # elemanların doğal logaritması (ln)
```

```
[5 7 9]
[-3 -3 -3]
[ 4 10 18]
[0.25 0.4  0.5 ]
[ 2.71828183  7.3890561  20.08553692]
[2.         2.23606798 2.44948974]
[0.84147098 0.90929743 0.14112001]
[1.38629436 1.60943791 1.79175947]
```

```
import numpy as np

matris1 = np.array([[1,2],[3,4]])
matris2 = np.array([[5,6],[10,12]])
```

```
# TEMEL MATRİSLER VE İŞLEMLER

print(np.zeros((2,3))) # sıfır matris

print(np.ones((1,4))) # birler matrisi

print(np.eye(2)) # birim matris

print(np.matmul(matris1,matris2)) # matris çarpımı

print(np.linalg.inv(matris1)) # matris tersi

print(np.linalg.matrix_power(matris1,3)) # matris kuvvet alma

print(np.linalg.matrix_rank(matris2)) # matris rankı

print(np.linalg.det(matris2)) # matris determinantı
```

```
[[0. 0. 0.]
 [0. 0. 0.]]
[[1. 1. 1. 1.]]
[[1. 0.]
 [0. 1.]]
[[25 30]
 [55 66]]
[[-2.   1. ]
 [ 1.5 -0.5]]
[[ 37  54]
 [ 81 118]]
1
0.0
```

```
import numpy as np

# MANTIK OPERATÖRLERİ

# True <=> 1, False <=> 0 yerine kullanılabilir


# np.logical_and -> Ve
print(np.logical_and([True, False], [1, 1]))


# np.logical_or -> Veya
print(np.logical_or([True, False], [1, 1]))


# np.logical_not -> Değil/Olumsuz
print(np.logical_not([True, False, 1, 0]))


# np.logical_xor -> Özel Veya (XOR)
print(np.logical_xor([True, False], [1, 1]))
```

```
[ True False]
[ True  True]
[False  True False  True]
[False  True]
```

```
import numpy as np

# TABAN ÇEVİRME

# 2 tabanına çevirme
print(np.binary_repr(17, width=8))

# Herhangi bir tabana çevirme
print(np.base_repr(1012, base=16))

# BİTSEL OPERATÖRLER

print(np.bitwise_and(13, 17))
#00001101 ve 00010001 = 00000001 -> 1

print(np.bitwise_or(13, 17))
#00001101 veya 00010001 = 00011101 -> 29

print(np.bitwise_not(13))
```

```
#00001101 -> 11110010 -> -14 (işaretli)

print(np.uint8(np.bitwise_not(13)))
#00001101 -> 11110010 -> 242 (işaretsiz)

print(np.bitwise_xor(13, 17))
#00001101 ve 00010001 = 00011100 -> 28
```

```
00010001
3F4
1
29
-14
242
28
```

```
def tekliyerinekoyma():

    alfabe = "ABCÇDEFGĞHIİJKLMNOÖPRSŞTUÜVYZ"
    sayac = 1

    while True:
        print("İŞLEMLER: Şifreleme (Ş)/Deşifreleme (D)/Kaba Kuvvet Saldırısı(K)/Çıkış (Ç)")
        karar = input(f"{sayac} numaralı işlemi seçiniz: ")

        if karar.upper() == "Ş":  # Şifreleme işlemi
            duz_metin = input("Şifrelenecek metni giriniz: ").replace("i", "İ").upper()
            try:
                anahtar = int(input("Anahtar sayısını giriniz: ")) % len(alfabe)
            except:
                print("Lütfen geçerli bir sayı giriniz.\n")
                continue
            sifreli_metin = ""
            for i in range(len(duz_metin)):
                if duz_metin[i] in alfabe:
                    sifreli_indis = alfabe.index(duz_metin[i]) + anahtar
                    sifreli_metin += alfabe[sifreli_indis % len(alfabe)]
                else:
                    sifreli_metin += duz_metin[i]
            print(f"Şifreli Metin: {sifreli_metin}\n")
            sayac += 1

        elif karar.upper() == "D":  # Deşifreleme işlemi (anahtar biliniyor ise)
            sifreli_metin = input("Deşifrelenecek metni giriniz: ").replace("i", "İ").upper()
            try:
                anahtar = int(input("Anahtar sayısını giriniz: ")) % len(alfabe)
            except:
                print("Lütfen geçerli bir sayı giriniz.\n")
                continue
            duz_metin = ""
            for i in range(len(sifreli_metin)):
                if sifreli_metin[i] in alfabe:
                    duz_indis = alfabe.index(sifreli_metin[i]) - anahtar
                    duz_metin += alfabe[duz_indis % len(alfabe)]
                else:
                    duz_metin += sifreli_metin[i]
            print(f"Düz Metin: {duz_metin}\n")
            sayac += 1

        elif karar.upper() == "K":  # Kaba kuvvet saldırısı (anahtar bilinmiyor ise)
            sifreli_metin = input("Deşifrelenecek metni giriniz: ").replace("i", "İ").upper()
            for anahtar in range(1, len(alfabe)):
                duz_metin = ""
                for i in range(len(sifreli_metin)):
                    if sifreli_metin[i] in alfabe:
                        duz_indis = alfabe.index(sifreli_metin[i]) - anahtar
                        duz_metin += alfabe[duz_indis % len(alfabe)]
                    else:
                        duz_metin += sifreli_metin[i]
                print(f"Anahtar = {anahtar} --> Düz Metin: {duz_metin}")
            print()
            sayac += 1

        elif karar.upper() == "Ç":  # Çıkış
            print("Program sonlandırıldı.")
            break

        else:  # Geçersiz işlem
```

```
            print("Lütfen geçerli bir işlem giriniz.\n")
            continue

tekliyerinekoyma()
```

```
İŞLEMLER: Şifreleme (Ş)/Deşifreleme (D)/Kaba Kuvvet Saldırısı(K)/Çıkış (Ç)
1 numaralı işlemi seçiniz: Ş
Şifrelenecek metni giriniz: ALTUĞ
Anahtar sayısını giriniz: 7
Şifreli Metin: GSBCM

İŞLEMLER: Şifreleme (Ş)/Deşifreleme (D)/Kaba Kuvvet Saldırısı(K)/Çıkış (Ç)
2 numaralı işlemi seçiniz: D
Deşifrelenecek metni giriniz: GSBCM
Anahtar sayısını giriniz: 7
Düz Metin: ALTUĞ

İŞLEMLER: Şifreleme (Ş)/Deşifreleme (D)/Kaba Kuvvet Saldırısı(K)/Çıkış (Ç)
3 numaralı işlemi seçiniz: K
Deşifrelenecek metni giriniz: GSBCM
Anahtar = 1 --> Düz Metin: FRABL
Anahtar = 2 --> Düz Metin: EPZAK
Anahtar = 3 --> Düz Metin: DÖYZJ
Anahtar = 4 --> Düz Metin: ÇOVYİ
Anahtar = 5 --> Düz Metin: CNÜVI
Anahtar = 6 --> Düz Metin: BMUÜH
Anahtar = 7 --> Düz Metin: ALTUĞ
Anahtar = 8 --> Düz Metin: ZKŞTG
Anahtar = 9 --> Düz Metin: YJSŞF
Anahtar = 10 --> Düz Metin: VİRSE
Anahtar = 11 --> Düz Metin: ÜIPRD
Anahtar = 12 --> Düz Metin: UHÖPÇ
Anahtar = 13 --> Düz Metin: TĞOÖC
Anahtar = 14 --> Düz Metin: ŞGNOB
Anahtar = 15 --> Düz Metin: SFMNA
Anahtar = 16 --> Düz Metin: RELMZ
Anahtar = 17 --> Düz Metin: PDKLY
Anahtar = 18 --> Düz Metin: ÖÇJKV
Anahtar = 19 --> Düz Metin: OCİJÜ
Anahtar = 20 --> Düz Metin: NBIİU
Anahtar = 21 --> Düz Metin: MAHIT
Anahtar = 22 --> Düz Metin: LZĞHŞ
Anahtar = 23 --> Düz Metin: KYGĞS
Anahtar = 24 --> Düz Metin: JVFGR
Anahtar = 25 --> Düz Metin: İÜEFP
Anahtar = 26 --> Düz Metin: IUDEÖ
Anahtar = 27 --> Düz Metin: HTÇDO
Anahtar = 28 --> Düz Metin: ĞŞCÇN

İŞLEMLER: Şifreleme (Ş)/Deşifreleme (D)/Kaba Kuvvet Saldırısı(K)/Çıkış (Ç)
4 numaralı işlemi seçiniz: Ç
Program sonlandırıldı.
```

```
pip install PyCryptodome
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting PyCryptodome
  Downloading pycryptodome-3.18.0-cp35-abi3-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (2.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.1/2.1 MB 22.6 MB/s eta 0:00:00
Installing collected packages: PyCryptodome
Successfully installed PyCryptodome-3.18.0
```

```
from Crypto.Cipher import AES

duz_metin = "M. Kemal Atatürk".encode("cp857")
anahtar = "jsezarınanahtarı".encode("cp857")

aes = AES.new(anahtar, AES.MODE_ECB)
sifreli_metin = aes.encrypt(duz_metin)
desifreli_metin = aes.decrypt(sifreli_metin)

print("Şifreli Metin: ", sifreli_metin.decode("cp857"))
print("Düz Metin: ", desifreli_metin.decode("cp857"))
```

```
Şifreli Metin:  ï ê╡ 9İ[ãıÑ· e
┘
Düz Metin:  M. Kemal Atatürk
```

```
 import hashlib

 duz_metin = "Altuğ Beyhan"
 ozet1 = hashlib.sha1(duz_metin.encode())
```

```
ozet2 = hashlib.sha224(duz_metin.encode())
ozet3 = hashlib.sha256(duz_metin.encode())
ozet4 = hashlib.sha384(duz_metin.encode())
ozet5 = hashlib.sha512(duz_metin.encode())
ozet6 = hashlib.md5(duz_metin.encode())

print(duz_metin,"-> SHA1 ->", ozet1.hexdigest())
print(duz_metin,"-> SHA224 ->", ozet2.hexdigest())
print(duz_metin,"-> SHA256 ->", ozet3.hexdigest())
print(duz_metin,"-> SHA384 ->", ozet4.hexdigest())
print(duz_metin,"-> SHA512 ->", ozet5.hexdigest())
print(duz_metin,"-> MD5 ->", ozet6.hexdigest())
```

```
Altuğ Beyhan -> SHA1 -> 3ce8a8f9e752ba458729329cf4b8707072fed4c7
Altuğ Beyhan -> SHA224 -> 41594e0f5456f8a2ee0dcc15b98b7c37f972231393cf91e6f04094e9
Altuğ Beyhan -> SHA256 -> edeae2889975fe75376d4bddd4ad04788ac166a684b9483c65d9fe0c5741a393
Altuğ Beyhan -> SHA384 -> 776bc434574906593d3e547e76122dfa9ee0dba0407986da66c474a9eeee6c640b8d352630242e31e30c8e4
Altuğ Beyhan -> SHA512 -> 1128ea742fdd14ce0e63cb6cabe5ed928eb265e43fc31317df60306f647c8fbea95fb6d43868b4488d401c5
Altuğ Beyhan -> MD5 -> 3619d2d6e428cc34562e9390bf4acb57
```

✓ 0 sn. tamamlanma zamanı: 15:17