**NAME:** Anosha Aziz

**REG. NUMBER:** FA17-BCS-006

**SUBMITTED TO:** SIR KHALID MEHMOOD

**DATE:** 25-11-2020

# Additive Cipher:

text = plainText = input("Enter plain text: ") plainText = plainText.upper()

plainText = plainText.replace(" ", "") cipherText = ""

key = int(input("Enter key: ")) strLength = len(plainText)

table = {

'A': 0,

'B': 1,

'C': 2,

'D': 3,

'E': 4,

'F': 5,

'G': 6,

'H': 7,

'I': 8,

'G': 9,

'K': 10,

'L': 11,

'M': 12,

'N': 13,

'O': 14,

'P': 15,

'Q': 16,

'R': 17,

'S': 18,

'T': 19,

'U': 20,

'V': 21,

'W': 22,

'X': 23,

'Y': 24,

'Z': 25

}

for i in range(strLength): for letter in table:

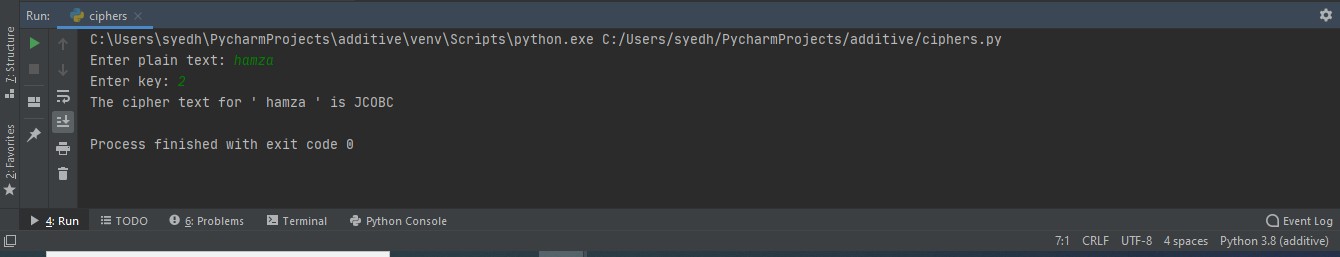
if letter == plainText[i]:

result = (table[letter] + key) % 26 for outputLetter in table:

if result == table[outputLetter]: cipherText = cipherText + outputLetter

print("The cipher text for \'", text, "\' is", cipherText)

**Output:**



1. **Affine cipher:**

text = plainText = input("Enter plain text: ") plainText = plainText.upper()

plainText = plainText.replace(" ", "") cipherText = ""

key1 = int(input("Enter the 1st key: ")) key2 = int(input("Enter the 2nd key: "))

strLength = len(plainText) table = {

'A': 0,

'B': 1,

'C': 2,

'D': 3,

'E': 4,

'F': 5,

'G': 6,

'H': 7,

'I': 8,

'J': 9,

'K': 10,

'L': 11,

'M': 12,

'N': 13,

'O': 14,

'P': 15,

'Q': 16,

'R': 17,

'S': 18,

'T': 19,

'U': 20,

'V': 21,

'W': 22,

'X': 23,

'Y': 24,

'Z': 25

}

for i in range(strLength): for letter in table:

if letter == plainText[i]:

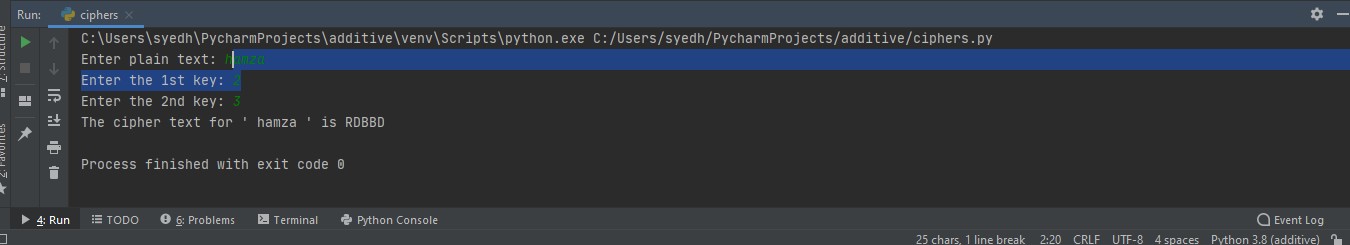
T = (table[letter] \* key1) % 26

result = (T + key2) % 26 for outputLetter in table:

if result == table[outputLetter]: cipherText = cipherText + outputLetter

print("The cipher text for \'", text, "\' is", cipherText)

**Output:**



1. **Auto-Key Cipher:**

text = plainText = input("Enter plain text: ") plainText = plainText.upper()

plainText = plainText.replace(" ", "") cipherText = ""

key = int(input("Enter key: "))

strLength = len(plainText) table = {

'A': 0,

'B': 1,

'C': 2,

'D': 3,

'E': 4,

'F': 5,

'G': 6,

'H': 7,

'I': 8,

'J': 9,

'K': 10,

'L': 11,

'M': 12,

'N': 13,

'O': 14,

'P': 15,

'Q': 16,

'R': 17,

'S': 18,

'T': 19,

'U': 20,

'V': 21,

'W': 22,

'X': 23,

'Y': 24,

'Z': 25

}

for letter in table:

if letter == plainText[0]:

result = (table[letter] + key) % 26 prev = plainText[0]

for outputLetter in table:

if result == table[outputLetter]: cipherText = cipherText + outputLetter

for i in range(1, strLength): for letter in table:

if letter == plainText[i]:

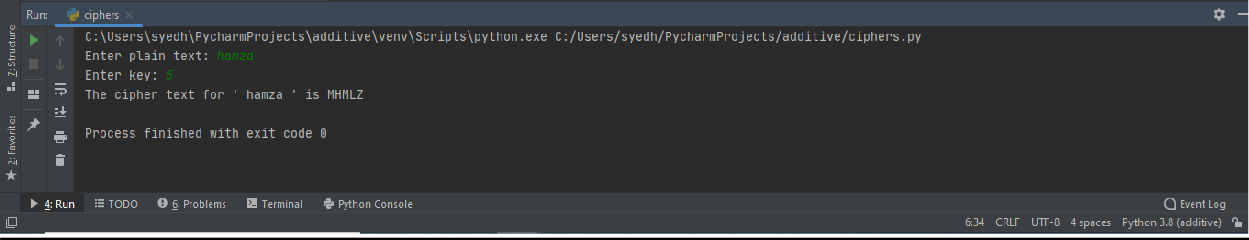
result = (table[letter] + table[prev]) % 26 prev = plainText[i]

for outputLetter in table:

if result == table[outputLetter]: cipherText = cipherText + outputLetter

print("The cipher text for \'", text, "\' is", cipherText)

**Output:**



1. **Multiplicative Cipher:**

text = plainText = input("Enter plain text: ") plainText = plainText.upper()

plainText = plainText.replace(" ", "") cipherText = ""

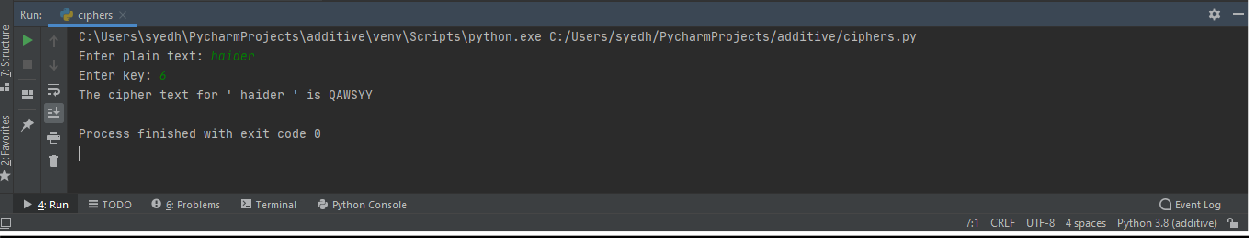
key = int(input("Enter key: "))

strLength = len(plainText) table = {

'A': 0,

|  |  |  |
| --- | --- | --- |
|  | 'B': | 1, |
| 'C': | 2, |
| 'D': | 3, |
| 'E': | 4, |
| 'F': | 5, |
| 'G': | 6, |
| 'H': | 7, |
| 'I': | 8, |
| 'J': | 9, |
| 'K': | 10, |
| 'L': | 11, |
| 'M': | 12, |
| 'N': | 13, |
| 'O': | 14, |
| 'P': | 15, |
| 'Q': | 16, |
| 'R': | 17, |
| 'S': | 18, |
| 'T': | 19, |
| 'U': | 20, |
| 'V': | 21, |
| 'W': | 22, |
| 'X': | 23, |
| 'Y': | 24, |
| } | 'Z': | 25 |
| for | i in | range(strLength): |
| for letter in table:  if letter == plainText[i]:  result = (table[letter] \* key) % 26 for outputLetter in table:  if result == table[outputLetter]: cipherText = cipherText + outputLetter  print("The cipher text for \'", text, "\' is", cipherText) | | |

**Output:**



1. **Playfair cipher:**

secretKey = [

["L", "G", "D", "B", "A"],

["Q", "M", "H", "E", "C"],

["U", "R", "N", "I", "F"],

["X", "V", "S", "O", "K"],

["Z", "Y", "W", "T", "P"]

]

text = plainText = input("Enter plain text: ") cipherText = ""

# Converting the string to uppercase and removing whitespaces plainText = plainText.upper()

plainText = plainText.replace(" ", "") strLength = len(plainText)

# Inserting "X" if two letters are consecutive for i in range(strLength-1):

if plainText[i] == plainText[i+1]:

plainText = plainText[:i+1] + "X" + plainText[i+1:]

# Inserting "X" at the end if string length is odd strLength = len(plainText)

if strLength % 2 == 1: plainText = plainText + "X"

strLength = len(plainText)

for i in range(0, strLength-1, 2):

# Finding the positions of pair of letters

# First letter of the pair for j in range(5):

for k in range(5):

if plainText[i] == secretKey[j][k]: pos1 = j

pos2 = k

# Second letter of the pair for j in range(5):

for k in range(5):

if plainText[i+1] == secretKey[j][k]: pos3 = j

pos4 = k

# Check if the pair is in same row if pos1 == pos3:

if pos2 == 4:

pos1 = -1 elif pos4 == 4:

pos3 = -1

cipherText = cipherText+secretKey[pos1][pos2+1] cipherText = cipherText+secretKey[pos3][pos4+1]

# Check if the pair is in same column elif pos2 == pos4:

if pos1 == 4:

pos1 = -1 elif pos3 == 4:

pos3 = -1

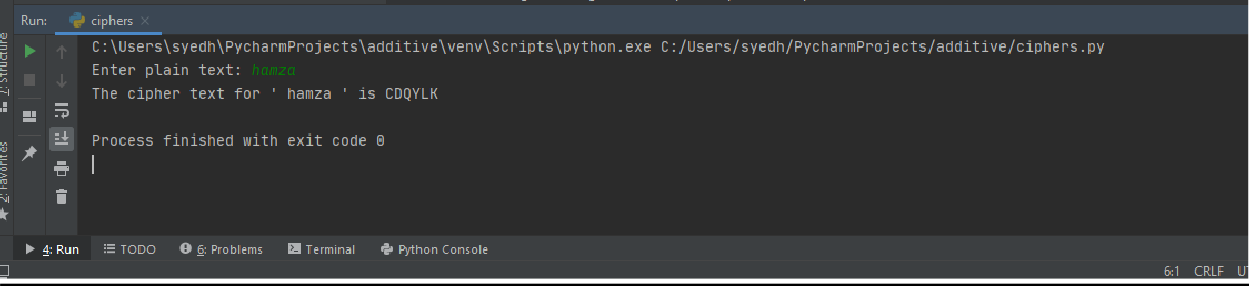
cipherText = cipherText+secretKey[pos1+1][pos2] cipherText = cipherText+secretKey[pos3+1][pos4]

# Check if the pair is in different row and column else:

cipherText = cipherText+secretKey[pos1][pos4] cipherText = cipherText+secretKey[pos3][pos2]

print("The cipher text for \'", text, "\' is", cipherText)

# Output:



1. **Bruteforce:**

cipherText = input("Enter cipher text: ") plainText = ""

strLength = len(cipherText)

table = {

'A': 0,

'B': 1,

'C': 2,

'D': 3,

'E': 4,

'F': 5,

'G': 6,

'H': 7,

'I': 8,

'J': 9,

'K': 10,

'L': 11,

'M': 12,

'N': 13,

'O': 14,

'P': 15,

'Q': 16,

'R': 17,

'S': 18,

'T': 19,

'U': 20,

'V': 21,

'W': 22,

'X': 23,

'Y': 24,

'Z': 25

}

for key in range(1, 27):

for i in range(strLength): for letter in table:

if letter == cipherText[i]:

result = (table[letter] - key) % 26 for outputLetter in table:

if result == table[outputLetter]: plainText = plainText + outputLetter

print("With key =", key, ", the plain text is", plainText) plainText = ""

**Output:**

