Nama : Muhammad Choeruman Syah

Kelas : TI. 20. A. 1

NIM : 312010031

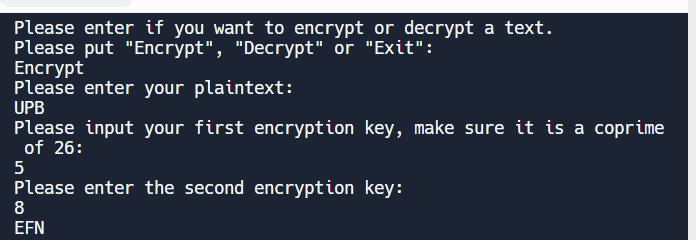
GitHUB : <https://github.com/MuhammadChoerumanSyah/AffineCipher.git>

Replit : https://replit.com/@MuhammadChoerum/AffineCipher#main.py

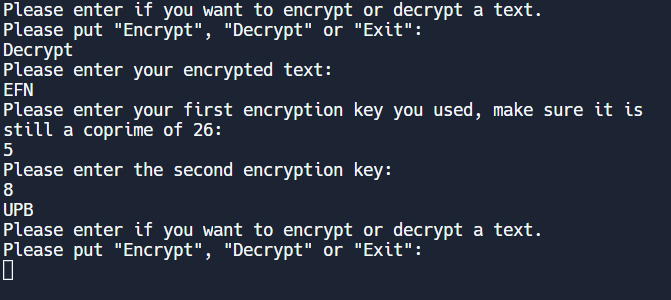
**AFFINE CIPHER**

Output :

Encrypt



Decrypt



Codingan :

import math;

#Import math module to use the math.gcd() command

def encryption():

#Function which handles the encryption process

encryptText = '';

counter = 0;

plainText = input("Please enter your plaintext: \n");

plainText = plainText.upper();

#Takes input and changes to capital letters.

print('Please input your first encryption key, make sure it is a coprime of 26: ');

inputType = 'a';

a = int(validateType(inputType));

#Get variable "a" from validateType function, which takes inputType as the argument.

#Used inputType as triger in validateType if it needs to be coprime or not.

#See rest at def validateType();

print('Please enter the second encryption key: ');

inputType = 'b';

b = int(validateType(inputType));

#Same situation, but with inputType as b.

length = len(plainText);

#Get length of the inputed text for use in for loop.

for x in range(length):

plainNum = ord(plainText[x]);

#Change the character to the Unicode number of it.

if plainNum >= 65 and plainNum <= 90:

#From 65 to 90 are the capital letters. Those are put into the encryption algorythm.

encryptNum = ((plainNum - 65) \* a + b) % 26;

encryptText += chr(encryptNum + 65);

#Adds the encrypted letter to the end of the encrypted text after getting turned back to character.

counter += 1;

#This is for testing in case of errors.

elif plainNum == 32:

#32 is whitespace, it gets converted back to character and added to end of string.

encryptText += chr(plainNum);

counter += 1;

#Ignores all other characters not specified.

return encryptText;

#Returns the encrypted text to main function

#Similar workflow of code to encryption.

def decryption():

plainText = '';

counter = 0;

encryptText = input('Please enter your encrypted text: \n');

encryptText = encryptText.upper();

print('Please enter your first encryption key you used, make sure it is still a coprime of 26: ');

inputType = 'a';

a = int(validateType(inputType));

print('Please enter the second encryption key: ');

inputType = 'b';

b = int(validateType(inputType));

length = len(encryptText);

aInverse = int(inverse(a, 26));

#Takes integer "a" and inputs it into inverse function with 26 as the other argument.

for x in range(length):

encryptNum = ord(encryptText[x]);

if encryptNum >= 65 and encryptNum <= 90:

#Decryption algorythm.

plainNum = (((encryptNum - 65) - b) \* aInverse) % 26;

plainText += chr(plainNum + 65);

counter += 1;

elif encryptNum == 32:

plainText += chr(encryptNum);

counter += 1;

return plainText;

def validateType(inputType):

a = input('');

#Takes input

while a.isdigit() == False:

#.isdigit() checks if variable before it is digit or not, returns False if not, therefor triggers while loop.

a = input('That is not a valid number. Please try again: \n');

if inputType == 'a':

#Checks the argument variable inputType. If it is "a", goes into validateCoprime with argument "a".

validateCoprime(a);

return a;

#Once function runs out, returns "a" to functions.

def validateCoprime(a):

inputType = 'a';

#Assign inputType again, it has gone out of scope.

testA = math.gcd(int(a), 26);

#Uses math function, which finds the greatest common divisor.

while testA != 1:

#If gcd was not 1, it goes back to number validation.

print('That number is not a coprime of 26. Please try again: ');

validateType(inputType);

break;

#Once its run through, it breaks out of the loops.

def inverse(a, m):

a1 = 1;

a2 = a;

b1 = 0;

b2 = m;

while b2 != 0:

#Keeps looping until b2 (remainder) is 0.

x = a2 // b2;

b1, b2, a1, a2 = (a1 - x \* b1), (a2 - x \* b2), b1, b2;

#All on one line, so it all gets changed at same time, to the old value, not the updated one yet.

return a1 % m;

#A could be negative so we take the remainder of it, which will be positive to return.

def main():

choice = '';

while choice != 'Exit':

#This while loop doesnt actually do anything, it would still loop forever so I simplified it.

#while 1 != 2:

#Infinite loop, only way to break out of is by inputting "Exit" (or CTRL + C).

choice = input('Please enter if you want to encrypt or decrypt a text. \nPlease put "Encrypt", "Decrypt" or "Exit": \n');

if choice == 'Encrypt':

print(encryption());

#Prints out the returned value from encryption function.

elif choice == 'Decrypt':

print(decryption());

elif choice == 'Exit':

break;

#Once input is "Exit", break out of the loop to the end of it.

else:

print('You have entered incorrect choice.');

choice = main();

#If not one of the options is inputted, starts again from the start of the main function and asks again.

return choice;

#This is used to not loop through extra times if the else gets triggered.

main();

#This can also be triggered manually in command line, but starts the main function, starting off the whole process.

#test = input("Here: ");

#Test if it prints out Exit at end. If it is run normally, it will not, only in shell.