Cryptography Project 2

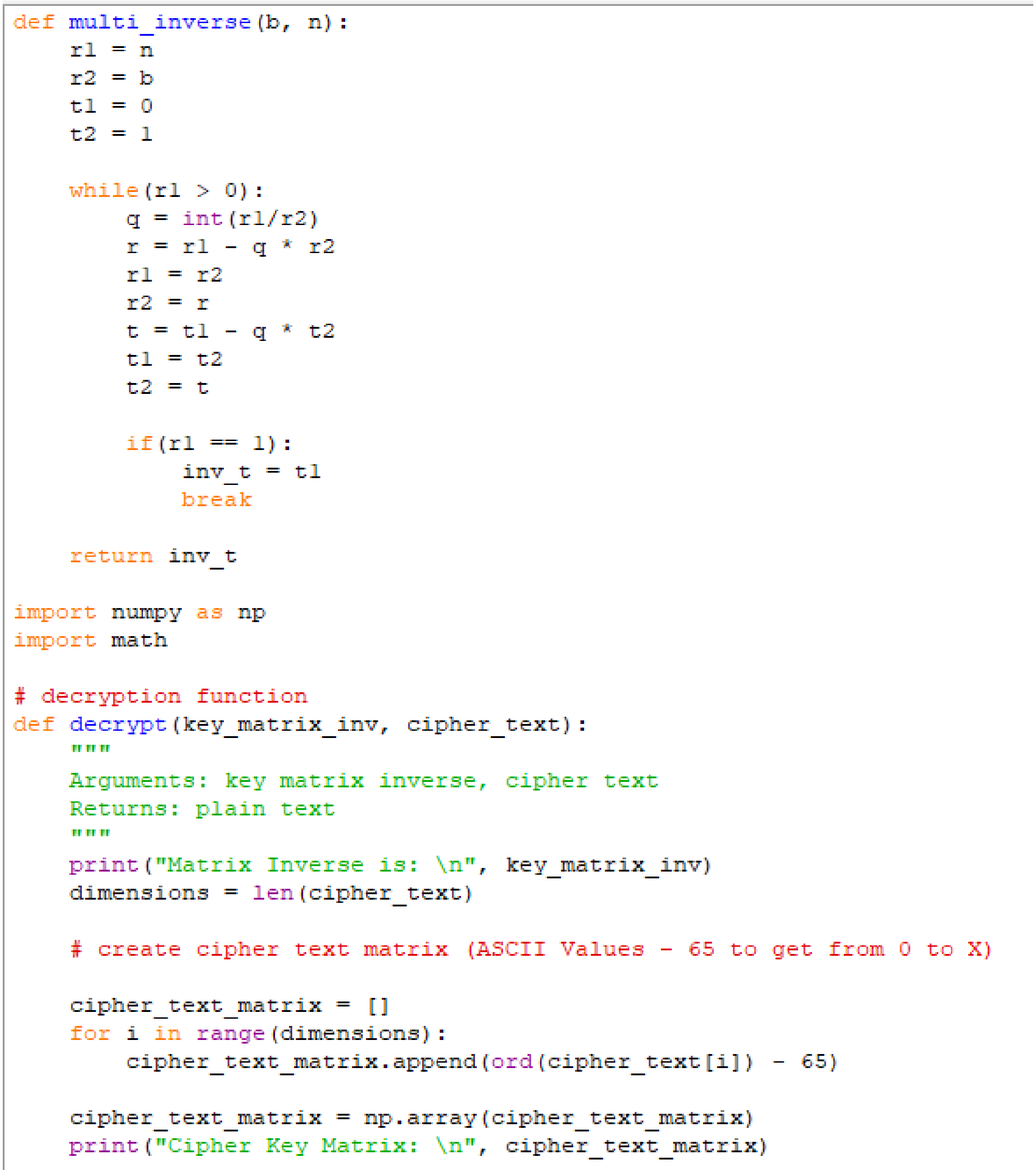
Project 3: - *Hill Cipher*

ADITI ADHIKARY

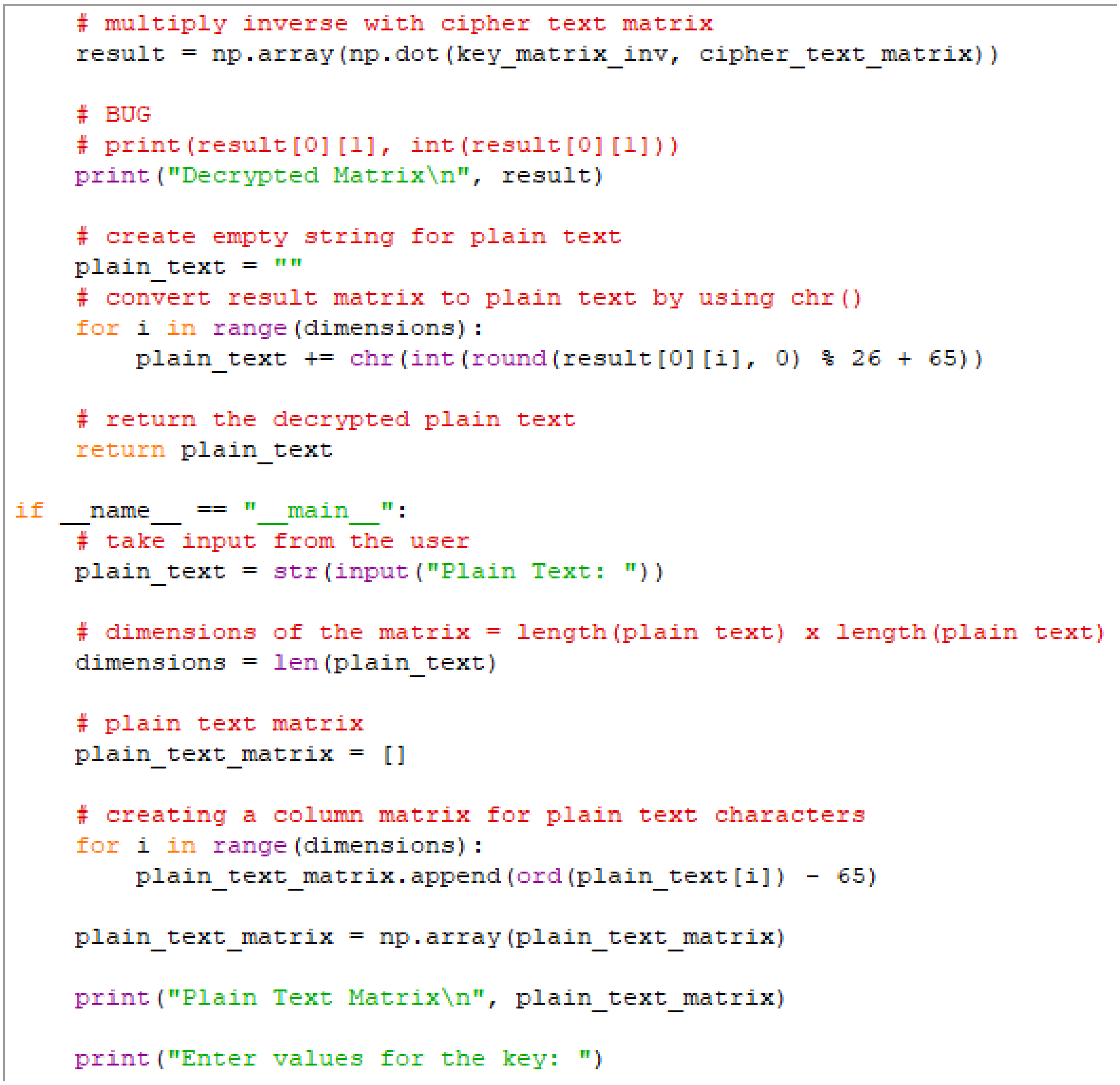
PES1UG19EC013

6th Semester "A"

CODE: -



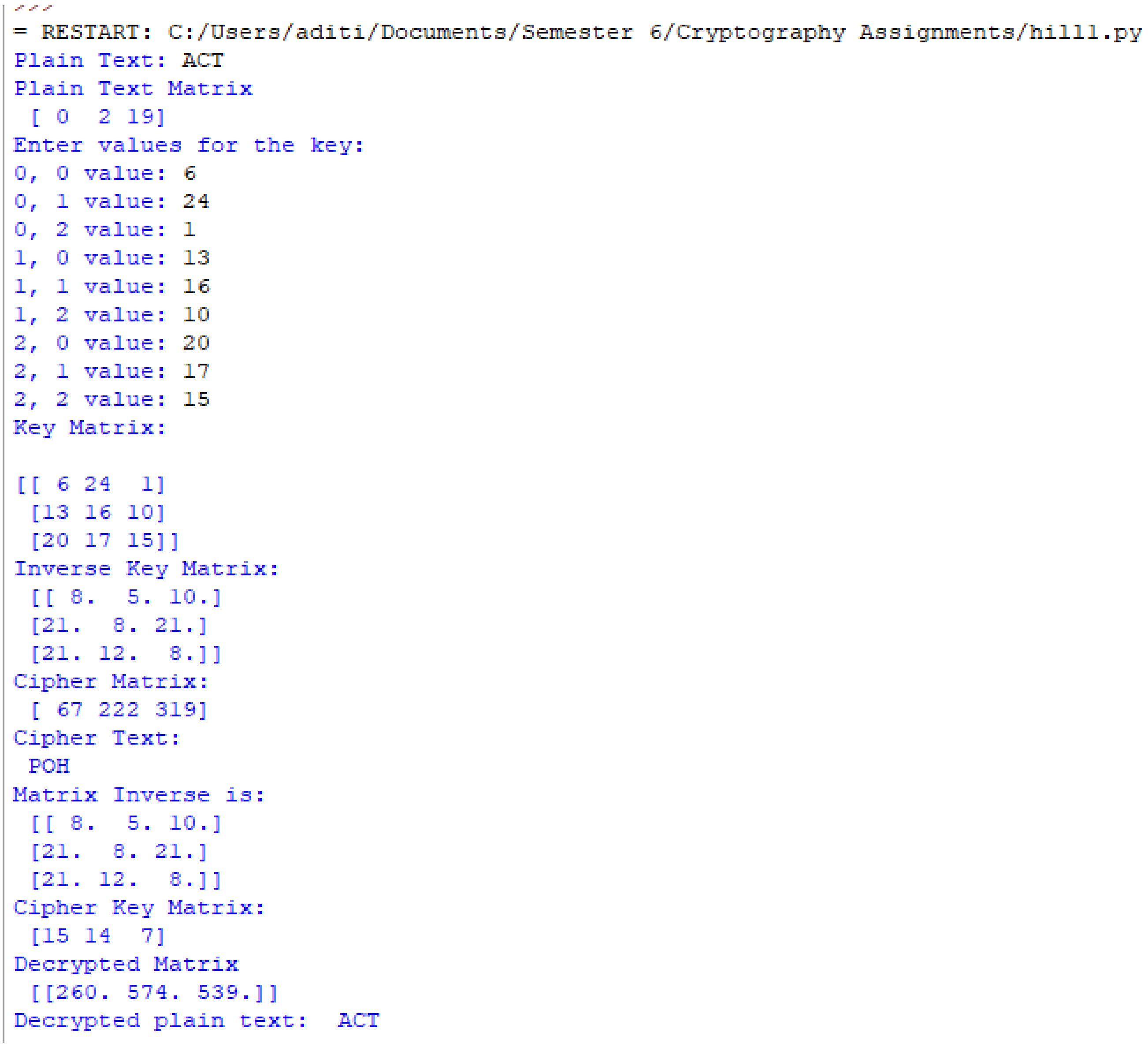
```
def multi_inverse(b, n):
    r1 = n
    r2 = b
    t1 = 0
    t2 = 1

    while(r1 > 0):
        q = int(r1/r2)
        r = r1 - q * r2
        r1 = r2
        r2 = r
        t = t1 - q * t2
        t1 = t2
        t2 = t

        if(r1 == 1):
            inv_t = t1
            break

    return inv_t

import numpy as np
import math

# decryption function
def decrypt(key_matrix_inv, cipher_text):
    """
    Arguments: key matrix inverse, cipher text
    Returns: plain text
    """
    print("Matrix Inverse is: \n", key_matrix_inv)
    dimensions = len(cipher_text)

    # create cipher text matrix (ASCII Values - 65 to get from 0 to X)

    cipher_text_matrix = []
    for i in range(dimensions):
        cipher_text_matrix.append(ord(cipher_text[i]) - 65)

    cipher_text_matrix = np.array(cipher_text_matrix)
    print("Cipher Key Matrix: \n", cipher_text_matrix)
```



```
    # multiply inverse with cipher text matrix
    result = np.array(np.dot(key_matrix_inv, cipher_text_matrix))

    # BUG
    # print(result[0][1], int(result[0][1]))
    print("Decrypted Matrix\n", result)

    # create empty string for plain text
    plain_text = ""
    # convert result matrix to plain text by using chr()
    for i in range(dimensions):
        plain_text += chr(int(round(result[0][i], 0) % 26 + 65))

    # return the decrypted plain text
    return plain_text

if __name__ == "__main__":
    # take input from the user
    plain_text = str(input("Plain Text: "))

    # dimensions of the matrix = length(plain text) x length(plain text)
    dimensions = len(plain_text)

    # plain text matrix
    plain_text_matrix = []

    # creating a column matrix for plain text characters
    for i in range(dimensions):
        plain_text_matrix.append(ord(plain_text[i]) - 65)

    plain_text_matrix = np.array(plain_text_matrix)

    print("Plain Text Matrix\n", plain_text_matrix)

    print("Enter values for the key: ")
```

```
# take values for the key matrix
key_matrix = []
for i in range(dimensions):
    row_ = []
    for j in range(dimensions):
        value = int(input(str(i) + ", " + str(j) + " value: "))
        row_.append(value)
    key_matrix.append(row_)

print("Key Matrix: \n")

# for encryption
key_matrix = np.array(key_matrix)
# for decryption
# key_matrix_inv = (np.linalg.inv(np.matrix(key_matrix)) % 26)
# key_matrix_inv = utils.multi_inverse(np.linalg.det(key_matrix), 26) * \
#        np.matrix(key_matrix).getH()
# print(np.matrix(key_matrix).getH())

# calculate key matrix inverse using modulo multiplicative inverse
key_matrix_inv = np.linalg.inv(np.matrix(key_matrix)) * \
        np.linalg.det(key_matrix) * \
        multi_inverse(np.linalg.det(key_matrix), 26) % 26

print(key_matrix)

print("Inverse Key Matrix: \n", key_matrix_inv)

result = key_matrix.dot(plain_text_matrix)

print("Cipher Matrix: \n", result)

cipher_text = ""

for i in range(dimensions):
    cipher_text += chr(result[i] % 26 + 65)

print("Cipher Text: \n", cipher_text)
decrypted_plain_text = decrypt(key_matrix_inv, cipher_text)
print("Decrypted plain text: ", decrypted_plain_text)
```

OUTPUT :-



```
= RESTART: C:/Users/aditi/Documents/Semester 6/Cryptography Assignments/hill1.py
Plain Text: ACT
Plain Text Matrix
 [ 0  2 19]
Enter values for the key:
0, 0 value: 6
0, 1 value: 24
0, 2 value: 1
1, 0 value: 13
1, 1 value: 16
1, 2 value: 10
2, 0 value: 20
2, 1 value: 17
2, 2 value: 15
Key Matrix:

[[ 6 24  1]
 [13 16 10]
 [20 17 15]]
Inverse Key Matrix:
 [[ 8.   5. 10.]
 [21.   8. 21.]
 [21. 12.   8.]]
Cipher Matrix:
 [ 67 222 319]
Cipher Text:
 POH
Matrix Inverse is:
 [[ 8.   5. 10.]
 [21.   8. 21.]
 [21. 12.   8.]]
Cipher Key Matrix:
 [15 14  7]
Decrypted Matrix
 [[260. 574. 539.]]
Decrypted plain text:  ACT
```