> **For the given input, perform Caesar cipher encryption and decryption.**
> **Plain text: “CRYPTOGRAPHY”**
> **Key: 10**

```
def encrypt(plaintext,key):
    ciphertext=''
    for character in plaintext:
        if character.isalpha():
            ciphertext+=chr((ord(character) - 65 + key) % 26 + 65)
        else:
            ciphertext+=character
    return ciphertext
def decrypt(ciphertext,key):
    return encrypt(ciphertext, -key)
plaintext = "CRYPTOGRAPHY"
shift = 10  # Given key
encrypted = encrypt(plaintext, shift)
decrypted = decrypt(encrypted, shift)
print("Plaintext :", plaintext)
print("Encrypted plaintext :", encrypted)
print("Decrypted ciphertext:", decrypted)
```

```python
def encrypt(plaintext,key):
    ciphertext=''
    for character in plaintext:
        if character.isalpha():
            ciphertext+=chr((ord(character) - 65 + key) % 26 + 65)
        else:
            ciphertext+=character
    return ciphertext
def decrypt(ciphertext,key):
    return encrypt(ciphertext, -key)
plaintext = "CRYPTOGRAPHY"
shift = 10 # Given key

encrypted = encrypt(plaintext, shift)
decrypted = decrypt(encrypted, shift)

print("Plaintext :", plaintext)
print("Encrypted plaintext :", encrypted)
print("Decrypted ciphertext:", decrypted)
```

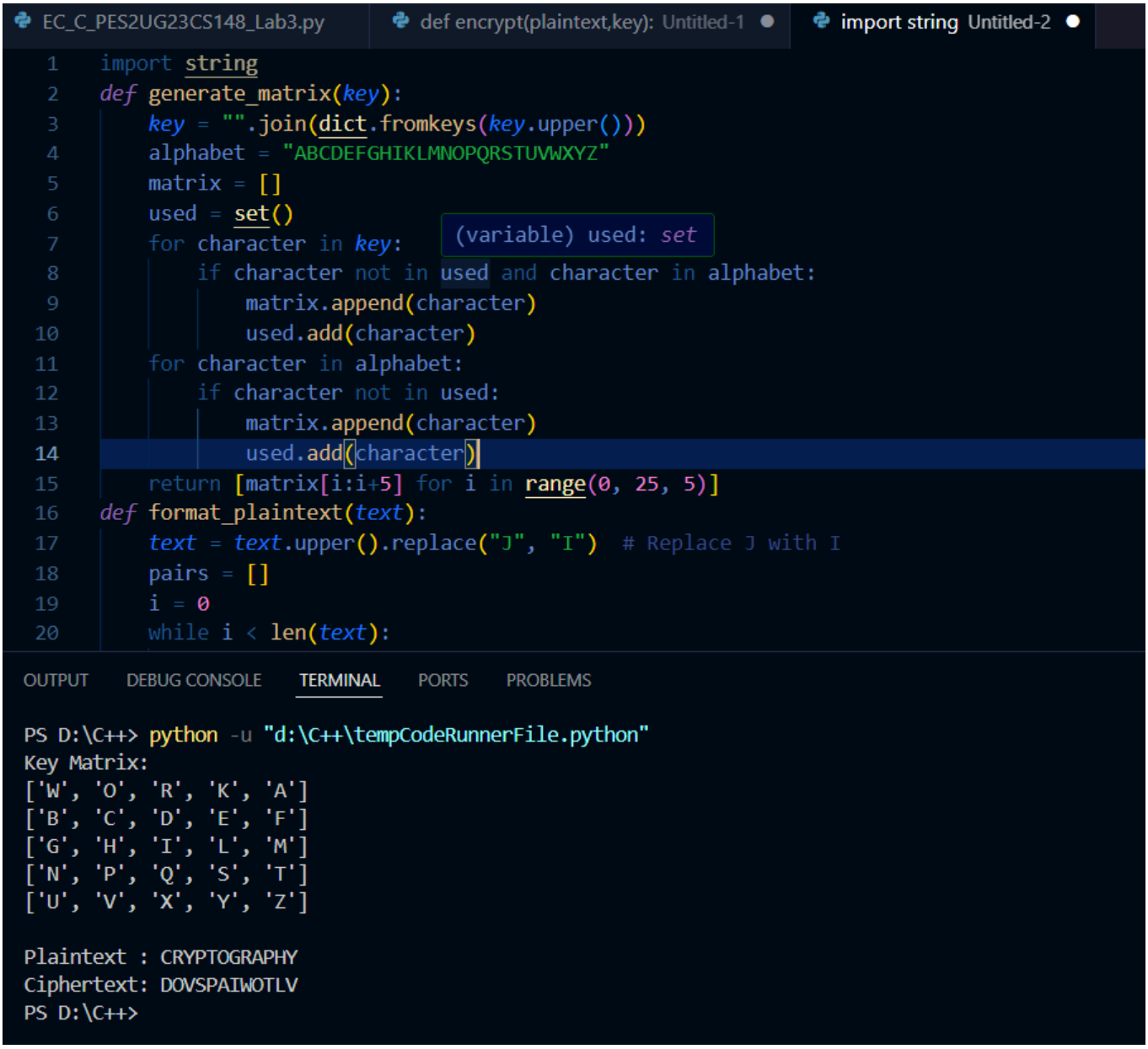
```
OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  PROBLEMS

PS D:\C++> python -u "d:\C++\tempCodeRunnerFile.python"
Plaintext : CRYPTOGRAPHY
Encrypted plaintext : MBIZDYQBKZRI
Decrypted ciphertext: CRYPTOGRAPHY
PS D:\C++>
```

> **Q2 For the plaintext given in question 1, apply Play Fair cipher encryption with key "WORK".**

```
import string
def generate_matrix(key):
    key = "".join(dict.fromkeys(key.upper()))
    alphabet = "ABCDEFGHIKLMNOPQRSTUVWXYZ"
    matrix = []
    used = set()
    for character in key:
        if character not in used and character in alphabet:
            matrix.append(character)
            used.add(character)
    for character in alphabet:
        if character not in used:
```

```
            matrix.append(character)
            used.add(character)
    return [matrix[i:i+5] for i in range(0, 25, 5)]
def format_plaintext(text):
    text = text.upper().replace("J", "I")  # Replace J with I
    pairs = []
    i = 0
    while i < len(text):
        a = text[i]
        b = text[i+1] if i+1 < len(text) else "X"
        if a == b:  # Same letters in pair, insert filler
            pairs.append(a + "X")
            i += 1
        else:
            pairs.append(a + b)
            i += 2
    return pairs
def find_position(matrix, ch):
    for r in range(5):
        for c in range(5):
            if matrix[r][c] == ch:
                return r, c
    return None
def playfair_encrypt(plaintext, key):
    matrix = generate_matrix(key)
    pairs = format_plaintext(plaintext)
    ciphertext = ""
    for a, b in pairs:
        r1, c1 = find_position(matrix, a)
        r2, c2 = find_position(matrix, b)
        if r1 == r2:  # Same row
            ciphertext += matrix[r1][(c1 + 1) % 5]
            ciphertext += matrix[r2][(c2 + 1) % 5]
        elif c1 == c2:  # Same column
            ciphertext += matrix[(r1 + 1) % 5][c1]
            ciphertext += matrix[(r2 + 1) % 5][c2]
        else:  # Rectangle
            ciphertext += matrix[r1][c2]
            ciphertext += matrix[r2][c1]
    return ciphertext
plaintext = "CRYPTOGRAPHY"
key = "WORK"
ciphertext = playfair_encrypt(plaintext, key)
print("Key Matrix:")
for row in generate_matrix(key):
    print(row)
```

```
print("\nPlaintext :", plaintext)
print("Ciphertext:", ciphertext)
```



> **Q3. For the plaintext= “WORK”, apply Hill cipher cipher encryption with key = [1,2; 2;2].**

```
#this function coverts text to numbers
def text_to_numbers(text):
    return [ord(c) - ord('A') for c in text.upper()]
#this function converts numbers to text
def numbers_to_text(nums):
    return ''.join(chr(n + ord('A')) for n in nums)
#this function multiplies a 2x2 matrix with a vector
```

```
#the key given is also a 2x2 matrix
def multiply_matrix_vector(key, vector):
    return [
        (key[0][0]*vector[0] + key[0][1]*vector[1]) % 26,
        (key[1][0]*vector[0] + key[1][1]*vector[1]) % 26
    ]
def hillcipher_encrypt(plaintext, key):
    nums = text_to_numbers(plaintext)
    ciphertext_nums = []
    for i in range(0, len(nums), 2):
        block = nums[i:i+2]
        cipher_block = multiply_matrix_vector(key, block)
        ciphertext_nums.extend(cipher_block)
    return numbers_to_text(ciphertext_nums)
key = [[1, 2],
       [2, 2]]
plaintext = "WORK"
ciphertext = hillcipher_encrypt(plaintext, key)
print("Ciphertext:", ciphertext)
```