***Lab # 3***

***VERNAM CIPHER***

def char\_to\_num(char):

"""Convert character to number where 'a'/'A' = 0, 'b'/'B' = 1, ..., 'z'/'Z' = 25."""

if 'a' <= char <= 'z':

return ord(char) - ord('a')

elif 'A' <= char <= 'Z':

return ord(char) - ord('A')

else:

return None

def num\_to\_char(num):

"""Convert number to character where 0 = 'a', 1 = 'b', ..., 25 = 'z'."""

return chr(num + ord('a'))

def decrypt(ciphertext, key):

"""Decrypt the ciphertext using the Vernam cipher."""

plaintext = ""

print("\n \*\*\*\*\*VERNAM CIPHER DECRYPTION\*\*\*\*\*")

print("\nSubtraction Result (numbers):")

for c, k in zip(ciphertext, key):

ciphertext\_num = char\_to\_num(c)

key\_num = char\_to\_num(k)

if ciphertext\_num is not None and key\_num is not None:

print(f"{ciphertext\_num} (ciphertext) - {key\_num} (key) =", end=" ")

subtracted\_num = (ciphertext\_num - key\_num)

print(subtracted\_num, end=", ")

if subtracted\_num < 0:

subtracted\_num += 26

print(f"(Adjusted by adding 26: {subtracted\_num})", end=", ")

decrypted\_char = num\_to\_char(subtracted\_num)

plaintext += decrypted\_char

print(decrypted\_char)

return plaintext

def main():

print("\n\*\*\*\*\*VERNAM CIPHER ENCRYPTION\*\*\*\*\*")

plaintext = input("Enter the plaintext: ").lower()

# Count non-space characters in plaintext

plaintext\_count = sum(1 for char in plaintext if char != ' ')

while True:

key = input(f"Enter the key ({plaintext\_count} characters): ").lower().replace(" ", "")

if len(key) != plaintext\_count:

print("Error: Key length must match plaintext length (excluding spaces).")

else:

break

print("\nPlaintext (numbers):")

plaintext\_nums = []

for char in plaintext:

if char != ' ':

num = char\_to\_num(char)

if num is not None:

print(f"{char} -> {num}")

plaintext\_nums.append(num)

print("\nKey (numbers):")

key\_nums = []

for char in key:

num = char\_to\_num(char)

if num is not None:

print(f"{char} -> {num}")

key\_nums.append(num)

# Perform addition of corresponding numbers in plaintext and key

print("\nAddition Result (numbers):")

result\_nums = [(p + k) for p, k in zip(plaintext\_nums, key\_nums)]

for num in result\_nums:

print(num)

# Subtract values greater than or equal to 26 by 26

print("\nAdjusted Result (numbers):")

adjusted\_nums = []

for num in result\_nums:

if num >= 26:

adjusted\_num = num - 26

adjusted\_nums.append(adjusted\_num)

print(adjusted\_num)

else:

adjusted\_nums.append(num)

print(num)

# Convert numbers back to characters

print("\nCiphertext (characters):")

result\_chars = [num\_to\_char(num) for num in adjusted\_nums]

for char in result\_chars:

print(char)

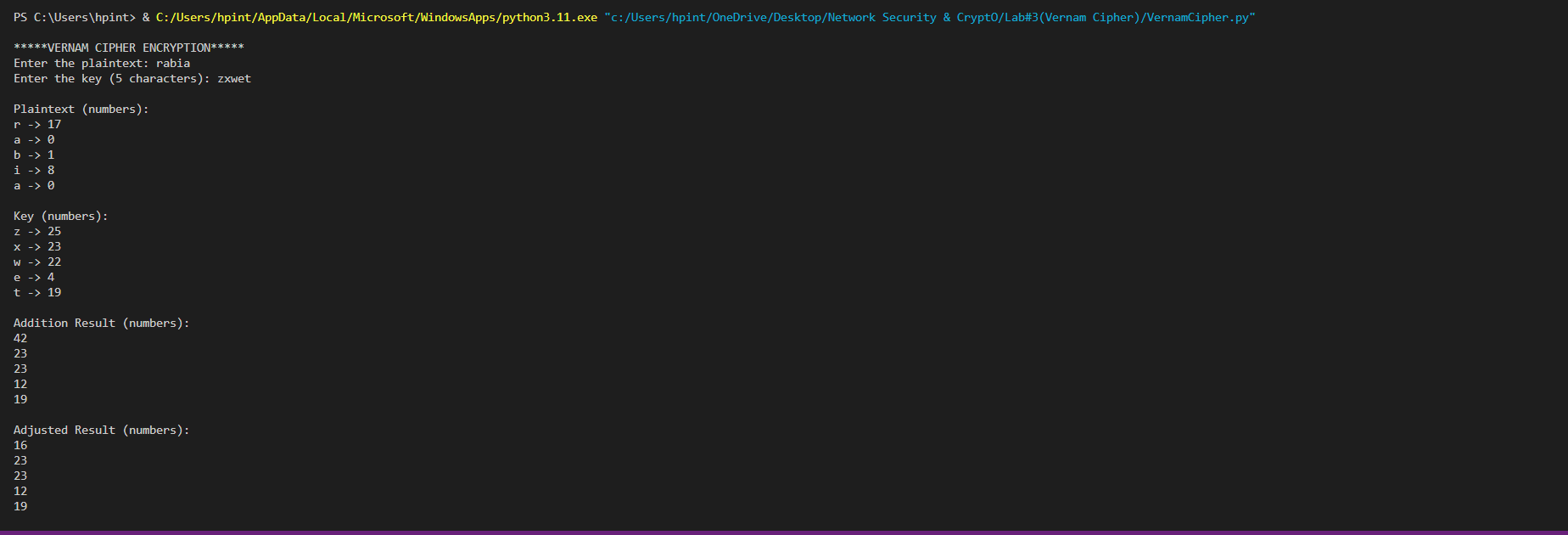
decrypted\_text = decrypt(result\_chars, key)

print("\nDecrypted text:", decrypted\_text)

if \_\_name\_\_ == "\_\_main\_\_":

main()

***OUTPUT***

******

***A black rectangle with white text

Description automatically generated***