Task 1: Implement the Shift Cipher

Write a Python function to implement the shift cipher for both encryption and decryption. Function Specifications:

- Input: A string (plaintext or ciphertext) and an integer (shift value).
- Output: The encrypted or decrypted string.

```
 #implementation of shift cipher

def encryption(): #defined function for encryption
    a = input("Enter String for encryption: ")
    ascii_values = [] #created a list for storing the ascii values
    for char in a:
      if char.isalpha() and char.isupper(): #checking whether  character is alphabetic and
        ascii_values.append((ord(char)+3-65)%26+65) #shift the character to 3 letter and ap
      elif char.isalpha() and char.islower():#checking whether  character is alphabetic and
        ascii_values.append((ord(char)+3-97)%26+97) #shift the character to 3 letter and ap
      else:
        ascii_values.append(ord(char)) #append other charcter excluding alphabets
    print("The shift cipher : ", end = " ")
    for val in ascii_values:
        print(chr(val) ,end="") #printing the encrypted cipher
    print()
encryption()

def decryption(): #defined function for decryption
  b= input("Enter cipher text for decryption: ")
  ascii_values = [] #created a list for storing the ascii values
  for char in b:
    if char.isalpha() and char.isupper(): #checking whether  character is alphabetic and up
      ascii_values.append((ord(char)-3-65)%26+65) #shift the character to 3 letter and appe
    elif char.isalpha() and char.islower(): #checking whether  character is alphabetic and
      ascii_values.append((ord(char)-3-97)%26+97) #shift the character to 3 letter and appe
    else:
      ascii_values.append(ord(char)) #append other charcter excluding alphabets
  print("The decrypted shift cipher : ", end = " ")
  for val in ascii_values:
      print(chr(val) ,end="") #printing the encrypted cipher
decryption()
```

```
Enter String for encryption: gOVIND
The shift cipher :  jRYLQG
Enter cipher text for decryption: jRYLQG
The decrypted shift cipher :  gOVIND
```

Task 2: Implement the Vigenère Cipher

Write a Python function to implement the Vigenère cipher for both encryption and decryption.
Function Specifications:

- Input: A string (plaintext or ciphertext) and a keyword (string).
- Output: The encrypted or decrypted string.

```
def encrypt_vigenere(text, keyword):
    result = ""  # Initialize result string
    key_length = len(keyword)

    for i, char in enumerate(text):
        if char.isalpha():  # Only process alphabetic characters
            shift = ord(keyword[i % key_length].upper()) - ord('A')  # Get shift value from
            if char.isupper():
                new_char = chr(((ord(char) - ord('A') + shift) % 26) + ord('A'))  # Encrypt
            else:
                new_char = chr(((ord(char) - ord('a') + shift) % 26) + ord('a'))  # Encrypt
            result += new_char
        else:
            result += char  # Keep non-alphabetic characters unchanged

    return result

def decrypt_vigenere(text, keyword):
    result = ""  # Initialize result string
    key_length = len(keyword)

    for i, char in enumerate(text):
        if char.isalpha():  # Only process alphabetic characters
            shift = ord(keyword[i % key_length].upper()) - ord('A')  # Get shift value from
            if char.isupper():
                new_char = chr(((ord(char) - ord('A') - shift) % 26) + ord('A'))  # Decrypt
            else:
                new_char = chr(((ord(char) - ord('a') - shift) % 26) + ord('a'))  # Decrypt
            result += new_char
        else:
            result += char  # Keep non-alphabetic characters unchanged

    return result

# Collect input from user
text = input("Enter text: ")
keyword = input("Enter keyword: ")
mode = input("Encrypt or Decrypt (E/D): ").strip().upper()

if mode == "E":
    result_text = encrypt_vigenere(text, keyword)
elif mode == "D":
    result_text = decrypt_vigenere(text, keyword)
else:
    result_text = "Invalid mode selected!"

print("Result:", result_text)
```

```
Enter text: Govind
Enter keyword: Duk
Encrypt or Decrypt (E/D): e
Result: Jiflhn
```

Task 3: Implement the One-Time Pad Cipher

Write a Python function to implement the One-Time Pad cipher for both encryption and decryption.

Function Specifications:

- Input: A string (plaintext or ciphertext) and a randomly generated key of the same length.
- Output: The encrypted or decrypted string.
- Note: The key must be truly random and as long as the plaintext. Deliverables

1. A Python file (assignment.py) containing the implementations of all three tasks.
2. Each function should include comments explaining the logic and flow of the code.
3. Submit your Python file via the submission portal.

```
def encrypt_vigenere(text, keyword):
    # Encrypts text using Vigenère cipher
    result = ""
    key_length = len(keyword)

    for i, char in enumerate(text):
        if char.isalpha():  # Check if character is a letter
            shift = ord(keyword[i % key_length].upper()) - ord('A')  # Get shift value
            if char.isupper():  # Encrypt uppercase letters
                result += chr((ord(char) - ord('A') + shift) % 26 + ord('A'))
            else:  # Encrypt lowercase letters
                result += chr((ord(char) - ord('a') + shift) % 26 + ord('a'))
        else:
            result += char  # Keep non-alphabetic characters unchanged

    return result

def decrypt_vigenere(text, keyword):
    # Decrypts text using Vigenère cipher
    result = ""
    key_length = len(keyword)

    for i, char in enumerate(text):
        if char.isalpha():  # Check if character is a letter
            shift = ord(keyword[i % key_length].upper()) - ord('A')  # Get shift value
            if char.isupper():  # Decrypt uppercase letters
                result += chr((ord(char) - ord('A') - shift) % 26 + ord('A'))
            else:  # Decrypt lowercase letters
                result += chr((ord(char) - ord('a') - shift) % 26 + ord('a'))
        else:
            result += char  # Keep non-alphabetic characters unchanged

    return result

def encrypt_otp(text, key):
    # Encrypts text using One-Time Pad cipher
    return "".join(chr(ord(t) ^ ord(k)) for t, k in zip(text, key))

def decrypt_otp(text, key):
    # Decrypts text using One-Time Pad cipher
    return "".join(chr(ord(t) ^ ord(k)) for t, k in zip(text, key))

# Collecting input from the user
print("Vigenère Cipher Example:")
plain_text = input("Enter plaintext: ")
key = input("Enter key: ")
encrypted = encrypt_vigenere(plain_text, key)
decrypted = decrypt_vigenere(encrypted, key)
print("Encrypted:", encrypted)
print("Decrypted:", decrypted)

print("\nOne-Time Pad Example:")
otp_text = input("Enter plaintext: ")
otp_key = input("Enter key (same length as plaintext): ")
if len(otp_text) != len(otp_key):
    print("Error: Key must be the same length as the plaintext!")
else:
    otp_encrypted = encrypt_otp(otp_text, otp_key)
    otp_decrypted = decrypt_otp(otp_encrypted, otp_key)
    print("Encrypted:", otp_encrypted.encode('utf-8'))  # Display encrypted text in byte fo
    print("Decrypted:", otp_decrypted)
```

```
Vigenère Cipher Example:
Enter plaintext: Govind
Enter key: Duk
Encrypted: Jiflhn
Decrypted: Govind

One-Time Pad Example:
Enter plaintext: GovindDuk
Enter key (same length as plaintext): abcdefxyz
```

```
Encrypted: b'&\r\x15\r\x0b\x02<\x0c\x11'
Decrypted: GovindDuk
```