Module 2: Sequences and File Operations

Case Study II

To build a secure system for LifeTel Telecom that handles the verification of users through a Reference ID, we need to follow the steps mentioned in the approach. Here's how we can implement this in Python:

Step-by-Step Implementation

1. **Read the input from command line – Reference ID**
2. **Check for validity – it should be 12 digits and allows only numbers and alphabets**
3. **Encrypt the Reference ID and print it for reference**

Enhancements

1. **Allow some special characters in Reference ID**
2. **Give the option for decryption to the user**

Python Code

Below is the Python code implementing the above steps and enhancements:

```
import re
from cryptography.fernet import Fernet

# Generate a key for encryption and decryption
# You must store this key securely. Anyone with this key can decrypt your data.
key = Fernet.generate_key()
cipher_suite = Fernet(key)

def is_valid_reference_id(reference_id):
# Check if the Reference ID is exactly 12 characters long and contains only alphanumeric characters
pattern = re.compile(r'^[a-zA-Z0-9]{12}$')
return bool(pattern.match(reference_id))

def encrypt_reference_id(reference_id):
# Encrypt the Reference ID
```

```
encrypted_id = cipher_suite.encrypt(reference_id.encode())
return encrypted_id

def decrypt_reference_id(encrypted_id):
# Decrypt the Reference ID
decrypted_id = cipher_suite.decrypt(encrypted_id).decode()
return decrypted_id

def main():
# Read input from the command line
reference_id = input("Enter the Reference ID (12 alphanumeric characters): ")
# Validate the Reference ID
if not is_valid_reference_id(reference_id):
print("Invalid Reference ID. It should be exactly 12 alphanumeric characters.")
return
# Encrypt the Reference ID
encrypted_id = encrypt_reference_id(reference_id)
print(f"Encrypted Reference ID: {encrypted_id}")
# Ask user if they want to decrypt it
choice = input("Do you want to decrypt the Reference ID? (yes/no): ")
if choice.lower() == 'yes':
decrypted_id = decrypt_reference_id(encrypted_id)
print(f"Decrypted Reference ID: {decrypted_id}")

if __name__ == "__main__":
main()
```

Explanation

1. **Reading Input**: The `input()` function is used to read the Reference ID from the user.
2. **Validation**: The `is_valid_reference_id()` function uses a regular expression to check if the Reference ID is exactly 12 characters long and contains only alphanumeric characters.
3. **Encryption**: The `encrypt_reference_id()` function uses the `Fernet` class from the `cryptography.fernet` module to encrypt the Reference ID.
4. **Decryption**: The `decrypt_reference_id()` function decrypts the encrypted Reference ID if the user chooses to do so.
5. **Main Function**: The `main()` function orchestrates the process, ensuring that the input is read, validated, encrypted, and optionally decrypted.

Enhancements

1. **Allow Some Special Characters**: To allow special characters, modify the regular expression in the `is_valid_reference_id()` function accordingly.
2. **Decryption Option**: The user is given an option to decrypt the Reference ID after encryption.

Security Considerations

- **Key Management**: The encryption key must be stored securely. In a real-world scenario, you might use a secure vault or environment variable or other storage methods such as Encrypted Databases.
- **Data Protection**: Ensure that the encrypted data and keys are handled securely to prevent unauthorized access.

This implementation provides a basic but effective way to encrypt and decrypt Reference IDs securely, ensuring that LifeTel Telecom can automate and secure their user verification process.