# Data Encryption and Decryption using Python

**Encryption:**

Encryption is the process of encoding the data. i.e converting plain text into ciphertext. This conversion is done with a key called an encryption key.

**Decryption:**

Decryption is the process of decoding the encoded data. Converting the ciphertext into plain text. This process requires a key that we used for encryption.

We require a key for encryption. There are two main types of keys used for encryption and decryption. They are Symmetric-key and Asymmetric-key.

**Symmetric-key Encryption:**

In symmetric-key encryption, the data is encoded and decoded with the same key. This is the easiest way of encryption, but also less secure. The receiver needs the key for decryption, so a safe way need for transferring keys. Anyone with the key can read the data in the middle.

**Example:**

Install the python cryptography library with the following command.

pip install cryptography

**Steps:**

* Import Fernet
* Then generate an encryption key, that can be used for encryption and decryption.
* Convert the string to a byte string, so that it can be encrypted.
* Instance the Fernet class with the encryption key.
* Then encrypt the string with the Fernet instance.
* Then it can be decrypted with Fernet class instance and it should be instanced with the same key used for encryption.

## Python3

|  |
| --- |
| **from** cryptography.fernet **import** Fernet  message **=** "hello tiaa"  key **=** Fernet.generate\_key()  fernet **=** Fernet(key)   encMessage **=** fernet.encrypt(message.encode())   print("original string: ", message)  print("encrypted string: ", encMessage)  decMessage **=** fernet.decrypt(encMessage).decode()  **print**("decrypted string: ", decMessage) |

**Asymmetric-key Encryption:**

In Asymmetric-key Encryption, we use two keys a public key and a private key. The public key is used to encrypt the data and the private key is used to decrypt the data. By the name, the public key can be public (can be sent to anyone who needs to send data). No one has your private key, so no one in the middle can read your data.

**Example:**

Install the python rsa library with the following command.

pip install rsa

**Steps:**

* Import rsa library
* Generate public and private keys with rsa.newkeys() method.
* Encode the string to byte string.
* Then encrypt the byte string with the public key.
* Then the encrypted string can be decrypted with the private key.
* The public key can only be used for encryption and the private can only be used for decryption.

## Python3

|  |
| --- |
| **import** rsa  publicKey, privateKey **=** rsa.newkeys(512)  message **=** "hello tiaa"  encMessage **=** rsa.encrypt(message.encode(),  publicKey)  **print**("original string: ", message)  **print**("encrypted string: ", encMessage)  decMessage **=** rsa.decrypt(encMessage, privateKey).decode()  **print**("decrypted string: ", decMessage) |