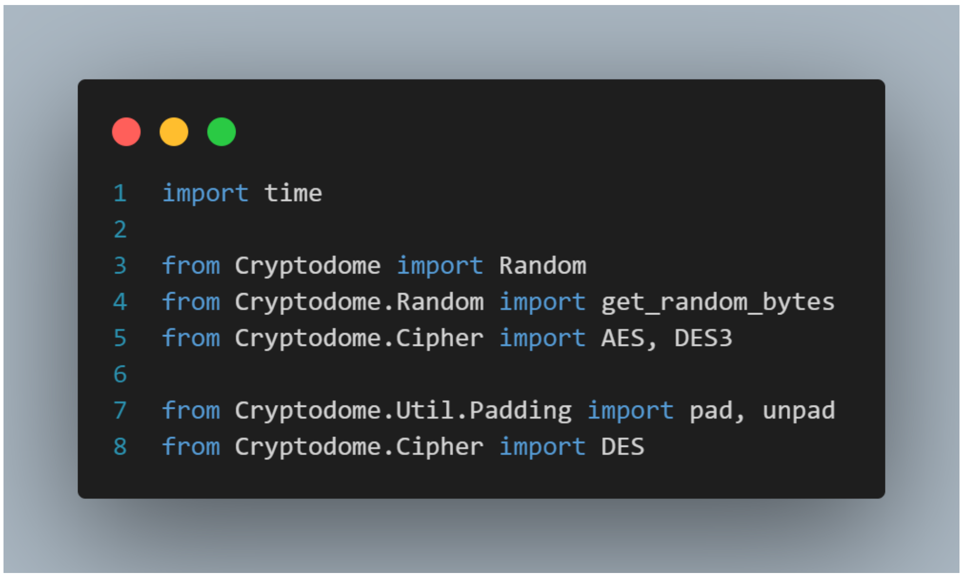
Q. Measure and compare the time taken for encryption and decryption using DES, 3DES, and AES, with different input sizes. Analyze the three algorithms and identify which components made an algorithm particularly fast/slow, weak/strong. You can implement your own algorithms or use others' implementation.

Before Starting will import time and PyCryptodomex library to get started



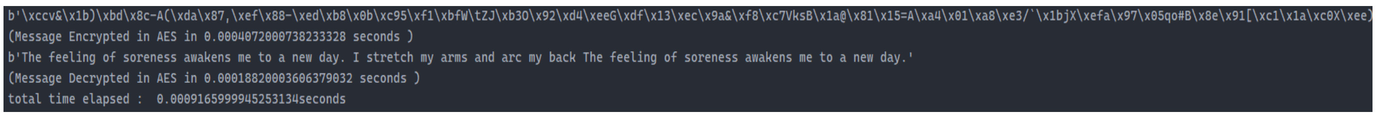
```python
import time

from Cryptodome import Random
from Cryptodome.Random import get_random_bytes
from Cryptodome.Cipher import AES, DES3

from Cryptodome.Util.Padding import pad, unpad
from Cryptodome.Cipher import DES
```

1. AES:

Advanced Encryption Standard (AES): AES is an acronym for Advanced Encryption Standard, a symmetric block cypher with a fixed data block size of 16 bytes. A 256-bit key can be used. I'm working with 16 bytes for my AES encryption algorithm

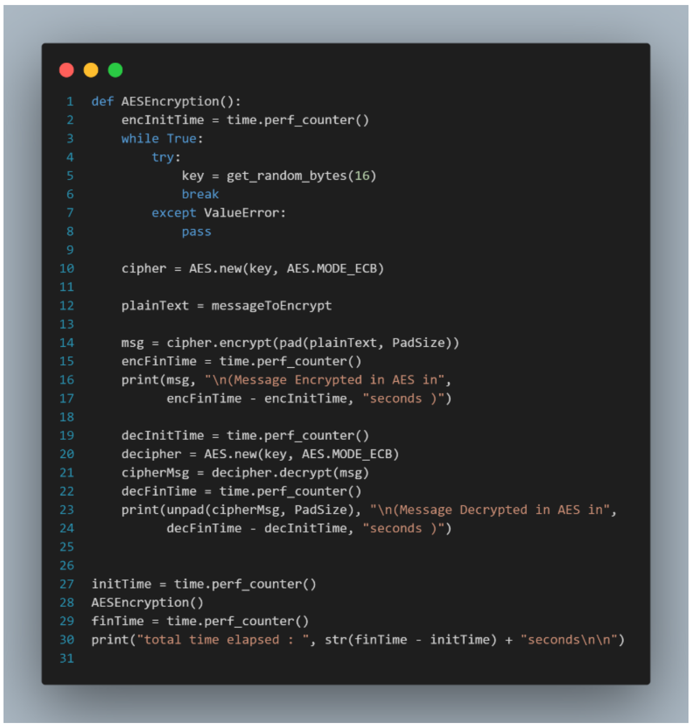
For Encryption, Firstly I'm generating a 16-bytes key randomly, and then I'm constructing an AES cipher in AES.MODE_ECB Mode and passing it the key that was randomly generated in the first step, then I'm using my Cipher which I constructed before to encrypt our Message(Padded)

Secondly for decryption, I'm Constructing a decipher with AES.MODE_ECB and passing it the key, and then I'm using my decipher to decrypt our AES encrypted message and then Unpadding our message to retrieve original message



```
b'\xccv&\x1b)\xbd\x8c-A(\xda\x87,\xef\x88-\xed\xb8\x0b\xc95\xf1\xbfW\tZJ\xb3O\x92\xd4\xeeG\xdf\x13\xec\x9a&\xf8\xc7VksB\x1a@\x81\x15=A\xa4\x01\xa8\xe3/`\x1bjX\xefa\x97\x05qo#B\x8e\x91[\xc1\x1a\xc0X\xee)
(Message Encrypted in AES in 0.0004072000738233328 seconds )
b'The feeling of soreness awakens me to a new day. I stretch my arms and arc my back The feeling of soreness awakens me to a new day.'
(Message Decrypted in AES in 0.00018820003606379032 seconds )
total time elapsed :  0.0009165999945253134seconds
```

Code:



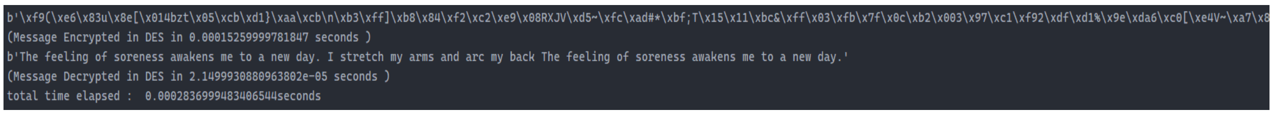
```python
def AESEncryption():
    encInitTime = time.perf_counter()
    while True:
        try:
            key = get_random_bytes(16)
            break
        except ValueError:
            pass

    cipher = AES.new(key, AES.MODE_ECB)

    plainText = messageToEncrypt

    msg = cipher.encrypt(pad(plainText, PadSize))
    encFinTime = time.perf_counter()
    print(msg, "\n(Message Encrypted in AES in",
          encFinTime - encInitTime, "seconds )")

    decInitTime = time.perf_counter()
    decipher = AES.new(key, AES.MODE_ECB)
    cipherMsg = decipher.decrypt(msg)
    decFinTime = time.perf_counter()
    print(unpad(cipherMsg, PadSize), "\n(Message Decrypted in AES in",
          decFinTime - decInitTime, "seconds )")


initTime = time.perf_counter()
AESEncryption()
finTime = time.perf_counter()
print("total time elapsed : ", str(finTime - initTime) + "seconds\n\n")
```

2. 3DES:

A fixed data block size of 16 bytes will be used for the symmetric block cypher **Triple DES**. It consists of three single DES cyphers cascaded together. The 16-byte key will be generated randomly. Here, we'll use a length of 16 bytes and a random number of bytes, and if it succeeds, we'll break; if not, we'll only do value error.

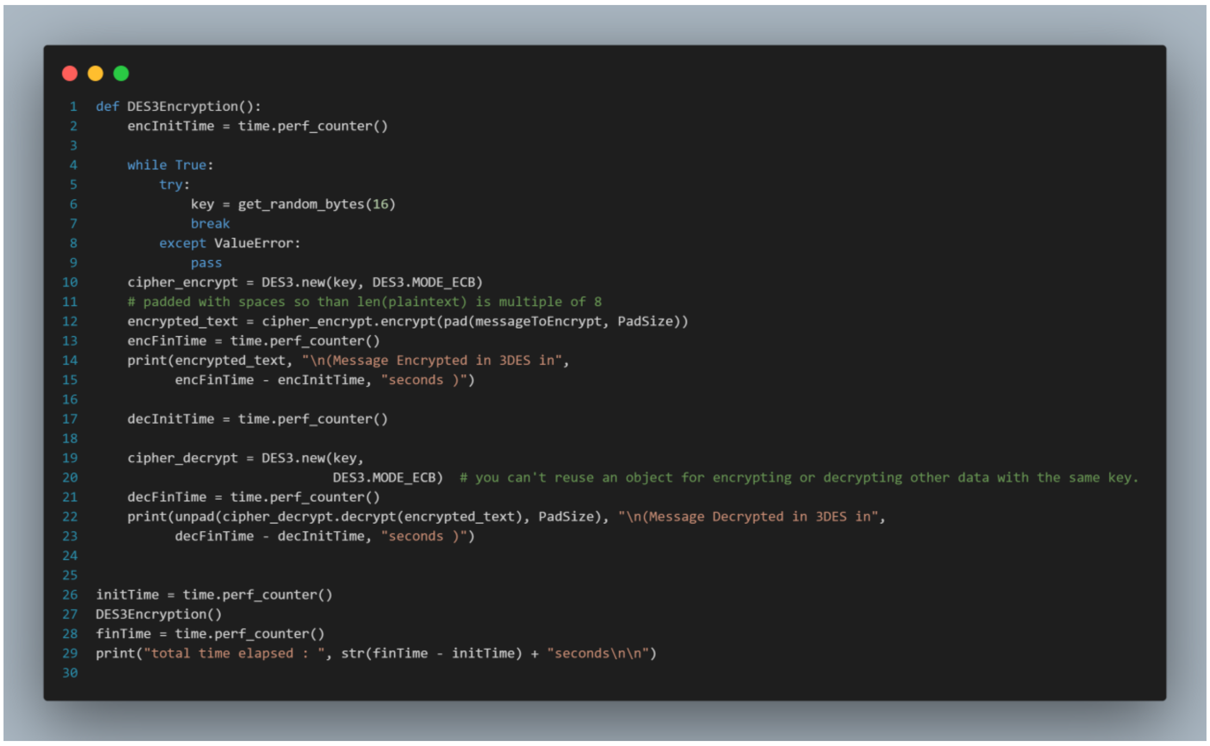
For Encryption, Firstly I'm generating a 16-bytes key randomly, and then I'm constructing an DES3 cipher in DES3.MODE_ECB Mode and passing it the key that was randomly generated in the first step, then I'm using my Cipher which I constructed before to encrypt our Message(Padded)

Secondly for decryption, I'm Constructing a decipher with DES3.MODE_ECB and passing it the key, and then I'm using my decipher to decrypt our DES3 encrypted message and then Unpadding our message to retrieve original message



```
b'\xf9(\xe6\x83u\x8e[\x014bzt\x05\xcb\xd1}\xaa\xcb\n\xb3\xff]\xb8\x84\xf2\xc2\xe9\x08RXJV\xd5~\xfc\xad#*\xbf;T\x15\x11\xbc&\xff\x03\xfb\x7f\x0c\xb2\x003\x97\xc1\xf92\xdf\xd1%\x9e\xda6\xc0[\xe4V~\xa7\x8
(Message Encrypted in DES in 0.00015259999781847 seconds )
b'The feeling of soreness awakens me to a new day. I stretch my arms and arc my back The feeling of soreness awakens me to a new day.'
(Message Decrypted in DES in 2.1499930880963802e-05 seconds )
total time elapsed :  0.0002836999483406544seconds
```

Code:



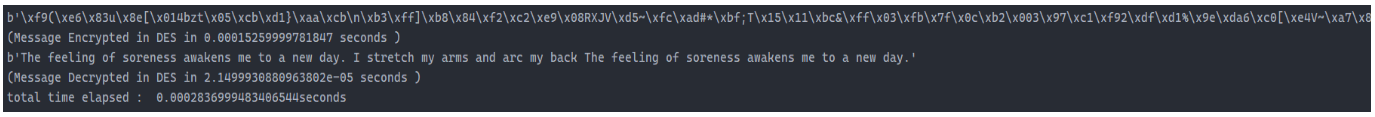
```python
def DES3Encryption():
    encInitTime = time.perf_counter()

    while True:
        try:
            key = get_random_bytes(16)
            break
        except ValueError:
            pass
    cipher_encrypt = DES3.new(key, DES3.MODE_ECB)
    # padded with spaces so than len(plaintext) is multiple of 8
    encrypted_text = cipher_encrypt.encrypt(pad(messageToEncrypt, PadSize))
    encFinTime = time.perf_counter()
    print(encrypted_text, "\n(Message Encrypted in 3DES in",
          encFinTime - encInitTime, "seconds )")

    decInitTime = time.perf_counter()

    cipher_decrypt = DES3.new(key,
                              DES3.MODE_ECB)  # you can't reuse an object for encrypting or decrypting other data with the same key.
    decFinTime = time.perf_counter()
    print(unpad(cipher_decrypt.decrypt(encrypted_text), PadSize), "\n(Message Decrypted in 3DES in",
          decFinTime - decInitTime, "seconds )")


initTime = time.perf_counter()
DES3Encryption()
finTime = time.perf_counter()
print("total time elapsed : ", str(finTime - initTime) + "seconds\n\n")

```

3. DES:

Data Encryption Standards (DES) is an acronym for a symmetric block cypher with a fixed data block size of just 8 bytes.

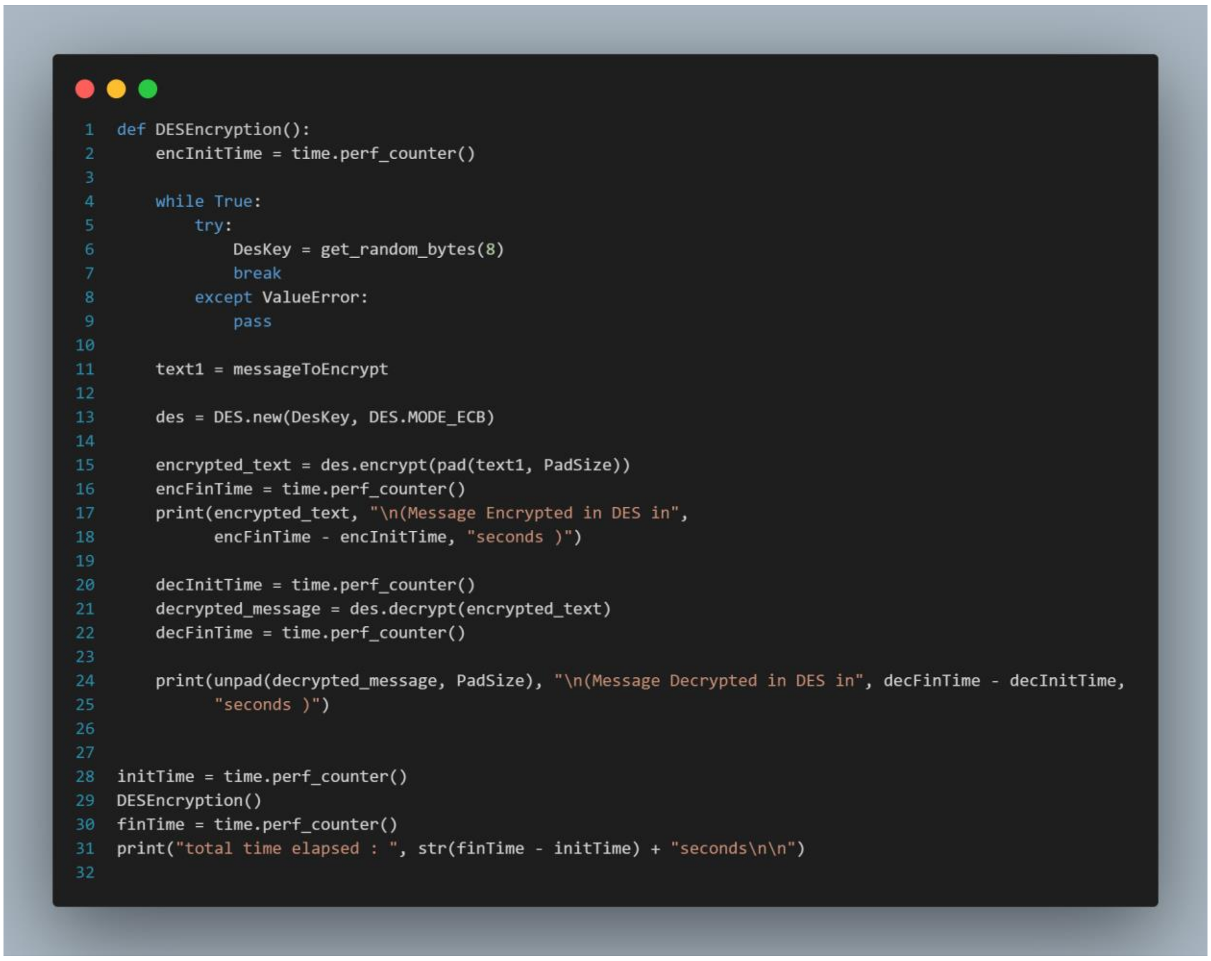
For Encryption, Firstly I'm generating a 8-bytes key randomly, and then I'm constructing an DES in DES.MODE_ECB Mode and passing it the key that was randomly generated in the first step, then I'm using my Cipher which I constructed before to encrypt our Message(Padded)

Secondly for decryption, I'm Constructing a decipher with DES.MODE_ECB and passing it the key, and then I'm using my decipher to decrypt our DES encrypted message and then Unpadding our message to retrieve original message



```
b'\xf9(\xe6\x83u\x8e[\x014bzt\x05\xcb\xd1}\xaa\xcb\n\xb3\xff]\xb8\x84\xf2\xc2\xe9\x08RXJV\xd5~\xfc\xad#*\xbf;T\x15\x11\xbc&\xff\x03\xfb\x7f\x0c\xb2\x003\x97\xc1\xf92\xdf\xd1%\x9e\xda6\xc0[\xe4V~\xa7\x8
(Message Encrypted in DES in 0.00015259999781847 seconds )
b'The feeling of soreness awakens me to a new day. I stretch my arms and arc my back The feeling of soreness awakens me to a new day.'
(Message Decrypted in DES in 2.1499930880963802e-05 seconds )
total time elapsed :  0.0002836999483406544seconds
```

Code:



```python
def DESEncryption():
    encInitTime = time.perf_counter()

    while True:
        try:
            DesKey = get_random_bytes(8)
            break
        except ValueError:
            pass

    text1 = messageToEncrypt

    des = DES.new(DesKey, DES.MODE_ECB)

    encrypted_text = des.encrypt(pad(text1, PadSize))
    encFinTime = time.perf_counter()
    print(encrypted_text, "\n(Message Encrypted in DES in",
          encFinTime - encInitTime, "seconds )")

    decInitTime = time.perf_counter()
    decrypted_message = des.decrypt(encrypted_text)
    decFinTime = time.perf_counter()

    print(unpad(decrypted_message, PadSize), "\n(Message Decrypted in DES in", decFinTime - decInitTime,
          "seconds )")


initTime = time.perf_counter()
DESEncryption()
finTime = time.perf_counter()
print("total time elapsed : ", str(finTime - initTime) + "seconds\n\n")

```