MAJOR I PRACTICAL (Python)

PRACTICAL NO:-1

STUDY OF CRYPTOGRAPHY

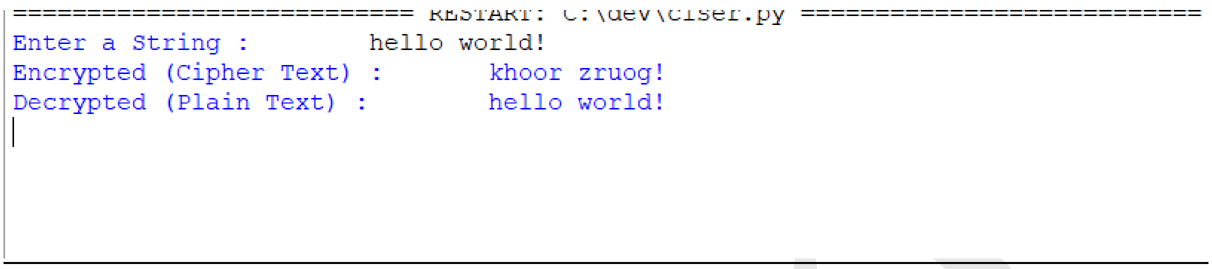
Aim:-To study cryptography substitution technique

Program:-

```
import string

def ceaser_cipher(input_str):
    ENCRYPT_SHIFT = 3
    DECRYPT_SHIFT = -3
    cipher_str = ""
    plain_str = ""
    for char in input_str:
        if char in string.ascii_lowercase:
            i = string.ascii_lowercase.index(char)
            new_i = (i + ENCRYPT_SHIFT) % 26
            cipher_str += string.ascii_lowercase[new_i]
        elif char in string.ascii_uppercase:
            i = string.ascii_uppercase.index(char)
            new_i = (i + ENCRYPT_SHIFT) % 26
            cipher_str += string.ascii_uppercase[new_i]
        else:
            cipher_str += char

    print(f"Encrypted (Cipher Text) :\t{cipher_str}")
    for char in cipher_str:
        if char in string.ascii_lowercase:
            i = string.ascii_lowercase.index(char)
            new_i = (i + DECRYPT_SHIFT + 26) % 26
            plain_str += string.ascii_lowercase[new_i]
        elif char in string.ascii_uppercase:
            i = string.ascii_uppercase.index(char)
            new_i = (i + DECRYPT_SHIFT + 26) % 26
            plain_str += string.ascii_uppercase[new_i]
        else:
            plain_str += char
    print(f"Decrypted (Plain Text) :\t{plain_str}")

print("Enter a String :", end="\t")
user_input = input()
```

ceaser_cipher(user_input)

Output:-



```
========================== RESTART: C:\dev\ciser.py ==========================
Enter a String :        hello world!
Encrypted (Cipher Text) :       khoor zruog!
Decrypted (Plain Text) :        hello world!
```

Conclusion :-The above program is successfully executed

PRACTICAL NO:-2

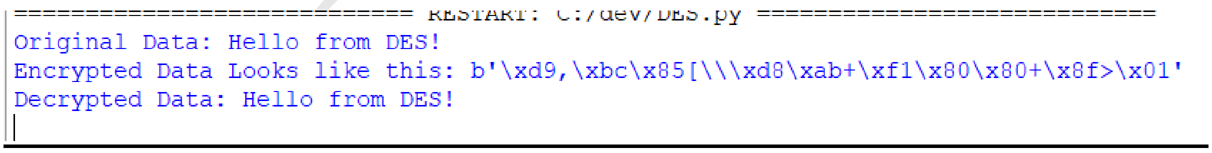
STUDY OF SYMMETRIC KEY - DES & AES

2)a)Aim:-To write python code for DES

Program:-

```
import pyDes

key = b"mysecret"

data_to_encrypt = b"Hello from DES!"

cipher = pyDes.des(key, padmode = pyDes.PAD_PKCS5)

encrypted_data = cipher.encrypt(data_to_encrypt)

print (f"Original Data: {data_to_encrypt.decode()}")

print (f"Encrypted Data Looks like this: {encrypted_data}")

decrypted_data = cipher.decrypt(encrypted_data)

print (f"Decrypted Data: {decrypted_data.decode()}")
```

Output:



```
============================ RESTART: C:/dev/DES.py ============================
Original Data: Hello from DES!
Encrypted Data Looks like this: b'\xd9,\xbc\x85[\\\xd8\xab+\xf1\x80\x80+\x8f>\x01'
Decrypted Data: Hello from DES!
```

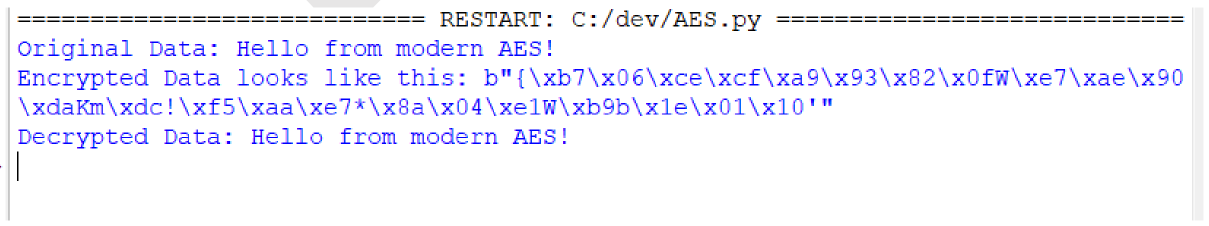
Conclusion :-The above program is successfully executed

2)b)Aim:-To write a code in python for AES

Program:-

```
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad, unpad
key = b'my_secret_key_16' # exactly 16 bytes
data_to_encrypt = b'Hello from modern AES!'
cipher = AES.new(key, AES.MODE_CBC)
iv = cipher.iv

encrypted_data = cipher.encrypt(pad(data_to_encrypt, AES.block_size))

print (f"Original Data: {data_to_encrypt.decode()}")
print (f"Encrypted Data looks like this: {encrypted_data}")
decrypt_cipher = AES.new(key, AES.MODE_CBC, iv)
decrypted_data=unpad(decrypt_cipher.decrypt(encrypted_data), AES.block_size)
print (f"Decrypted Data: {decrypted_data.decode()}")
```

Output:-



```
============================ RESTART: C:/dev/AES.py ============================
Original Data: Hello from modern AES!
Encrypted Data looks like this: b"{\xb7\x06\xce\xcf\xa9\x93\x82\x0fW\xe7\xae\x90
\xdaKm\xdc!\xf5\xaa\xe7*\x8a\x04\xe1W\xb9b\x1e\x01\x10'"
Decrypted Data: Hello from modern AES!
```

Conclusion :-The above program is successfully executed

PRACTICAL NO 3

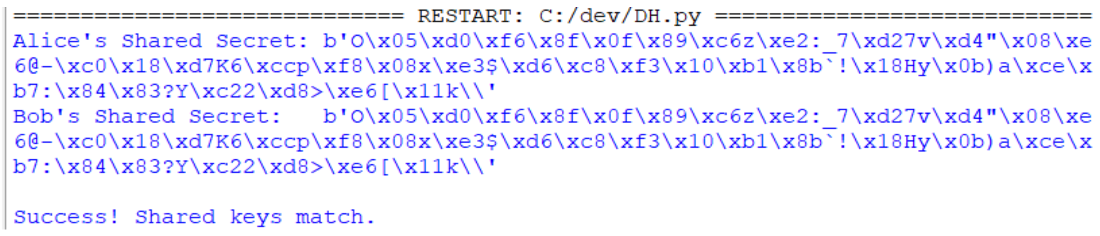
STUDY OF ASYMETRIC KEY (DH & RSA)

3)a)Aim:-To write a python code for DH

Program:-

```
from cryptography.hazmat.primitives.asymmetric import dh

from cryptography.hazmat.primitives import serialization

from cryptography.hazmat.backends import default_backend

parameters = dh.generate_parameters(generator=2, key_size=512,
backend=default_backend())

alice_private_key = parameters.generate_private_key()

alice_public_key = alice_private_key.public_key()

bob_private_key = parameters.generate_private_key()

bob_public_key = bob_private_key.public_key()

alice_shared_key = alice_private_key.exchange(bob_public_key)

bob_shared_key = bob_private_key.exchange(alice_public_key)

print(f"Alice's Shared Secret: {alice_shared_key}")

print(f"Bob's Shared Secret:   {bob_shared_key}")

if alice_shared_key == bob_shared_key:

    print("\nSuccess! Shared keys match.")

else:

    print("\nError! Shared keys do not match.")
```

Output:



```
============================ RESTART: C:/dev/DH.py ============================
Alice's Shared Secret: b'O\x05\xd0\xf6\x8f\x0f\x89\xc6z\xe2:_7\xd27v\xd4"\x08\xe
6@-\xc0\x18\xd7K6\xccp\xf8\x08x\xe3$\xd6\xc8\xf3\x10\xb1\x8b`!\x18Hy\x0b)a\xce\x
b7:\x84\x83?Y\xc22\xd8>\xe6[\x11k\\'
Bob's Shared Secret:   b'O\x05\xd0\xf6\x8f\x0f\x89\xc6z\xe2:_7\xd27v\xd4"\x08\xe
6@-\xc0\x18\xd7K6\xccp\xf8\x08x\xe3$\xd6\xc8\xf3\x10\xb1\x8b`!\x18Hy\x0b)a\xce\x
b7:\x84\x83?Y\xc22\xd8>\xe6[\x11k\\'

Success! Shared keys match.
```

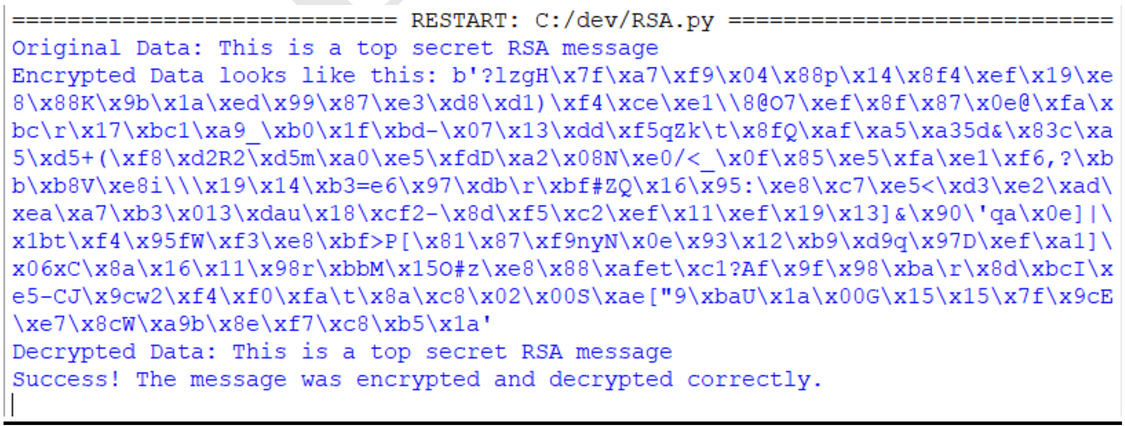
Conclusion :-The above program is successfully executed

3)b)**Aim:-**To write a code in python code for RSA

Program:-

```
from Crypto.PublicKey import RSA

from Crypto.Cipher import PKCS1_OAEP

key = RSA.generate(2048)

private_key = key

public_key = key.public_key()

data_to_encrypt = b'This is a top secret RSA message'

print (f"Original Data: {data_to_encrypt.decode()}")

cipher_rsa = PKCS1_OAEP.new(public_key)

encrypted_data = cipher_rsa.encrypt(data_to_encrypt)

print (f"Encrypted Data looks like this: {encrypted_data}")

decrypt_rsa = PKCS1_OAEP.new(private_key)

decrypted_data = decrypt_rsa.decrypt(encrypted_data)

print (f"Decrypted Data: {decrypted_data.decode()}")

assert data_to_encrypt == decrypted_data

print("Success! The message was encrypted and decrypted correctly.")
```

Output:



```
============================ RESTART: C:/dev/RSA.py ============================
Original Data: This is a top secret RSA message
Encrypted Data looks like this: b'?lzgH\x7f\xa7\xf9\x04\x88p\x14\x8f4\xef\x19\xe
8\x88K\x9b\x1a\xed\x99\x87\xe3\xd8\xd1)\xf4\xce\xe1\\8@O7\xef\x8f\x87\x0e@\xfa\x
bc\r\x17\xbc1\xa9_\xb0\x1f\xbd-\x07\x13\xdd\xf5qZk\t\x8fQ\xaf\xa5\xa35d&\x83c\xa
5\xd5+(\xf8\xd2R2\xd5m\xa0\xe5\xfdD\xa2\x08N\xe0/<_\x0f\x85\xe5\xfa\xe1\xf6,?\xb
b\xb8V\xe8i\\\x19\x14\xb3=e6\x97\xdb\r\xbf#ZQ\x16\x95:\xe8\xc7\xe5<\xd3\xe2\xad\
xea\xa7\xb3\x013\xdau\x18\xcf2-\x8d\xf5\xc2\xef\x11\xef\x19\x13]&\x90\'qa\x0e]|\
x1bt\xf4\x95fW\xf3\xe8\xbf>P[\x81\x87\xf9nyN\x0e\x93\x12\xb9\xd9q\x97D\xef\xa1]\
x06xC\x8a\x16\x11\x98r\xbbM\x15O#z\xe8\x88\xafet\xc1?Af\x9f\x98\xba\r\x8d\xbcI\x
e5-CJ\x9cw2\xf4\xf0\xfa\t\x8a\xc8\x02\x00S\xae["9\xbaU\x1a\x00G\x15\x15\x7f\x9cE
\xe7\x8cW\xa9b\x8e\xf7\xc8\xb5\x1a'
Decrypted Data: This is a top secret RSA message
Success! The message was encrypted and decrypted correctly.
|
```

Conclusion :-The above program is successfully executed

PRACTICAL NO 4

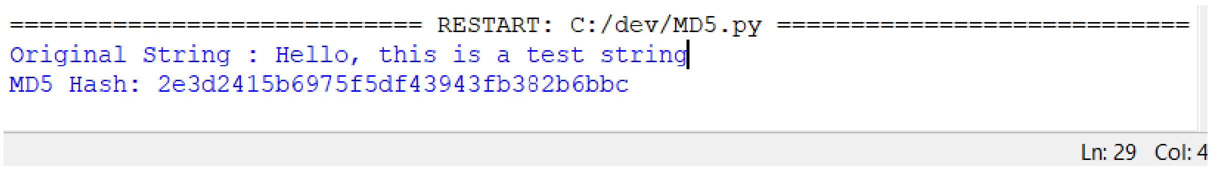
STUDY OF MD5 ALGORITHM

Aim:- To write code in python in MD5

Program:-

```
import hashlib

data = "Hello, this is a test string"

md5_hash_object = hashlib.md5(data.encode())

hex_digest = md5_hash_object.hexdigest()

print (f"Original String : {data}")

print (f"MD5 Hash: {hex_digest}")
```

Output:-



```
============================ RESTART: C:/dev/MD5.py ============================
Original String : Hello, this is a test string
MD5 Hash: 2e3d2415b6975f5df43943fb382b6bbc
```

Conclusion :-The above program is successfully executed

PRACTICAL NO 5

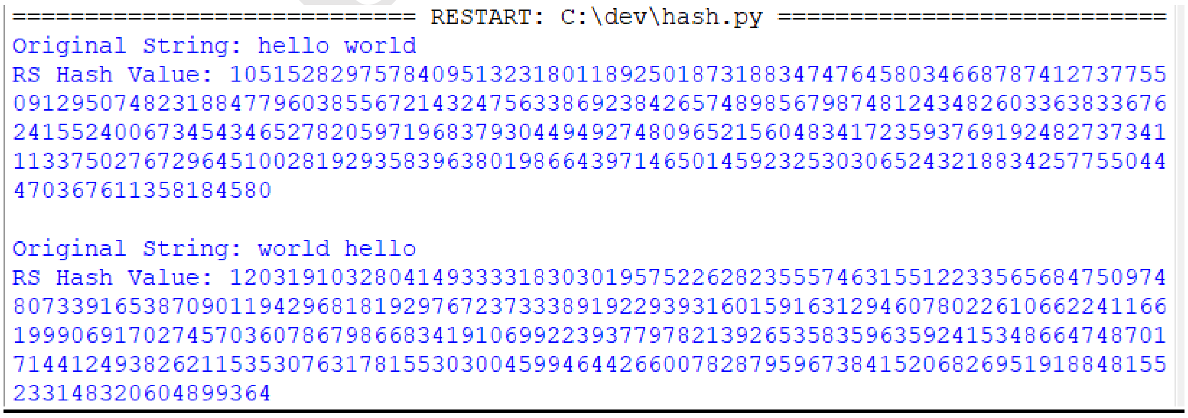
STUDY OF HASH FUNCTION

Aim:-To write code in python for hash fuction

Program:-

```
def RSHash (data_string):
    a = 378551
    b = 63689
    hash_value = 0
    for char in data_string:
        hash_value = hash_value * a + ord(char)
        a = a * b
    return hash_value
data = "hello world"
hash_result = RSHash(data)
print (f"Original String: {data}")
print (f"RS Hash Value: {hash_result}")
data2 = "world hello"
hash_result2 = RSHash(data2)
print (f"\nOriginal String: {data2}")
print (f"RS Hash Value: {hash_result2}")
```

Output:



```
=========================== RESTART: C:\dev\hash.py ===========================
Original String: hello world
RS Hash Value: 10515282975784095132318011892501873188347476458034668787412737755
0912950748231884779603855672143247563386923842657489856798748124348260336383676
24155240067345434652782059719683793044949274809652156048341723593769192482737341
11337502767296451002819293583963801986643971465014592325303065243218834257755044
470367611358184580

Original String: world hello
RS Hash Value: 12031910328041493333183030195752262823555746315512233565684750974
80733916538709011942968181929767237333891922939316015916312946078022610662241166
19990691702745703607867986683419106992239377978213926535835963592415348664748701
71441249382621153530763178155303004599464426600782879596738415206826951918848155
233148320604899364
```

Conclusion :- The above program is successfully executed