Cryptography Fundamentals

Aim: To develop the Simple-DES using Python.

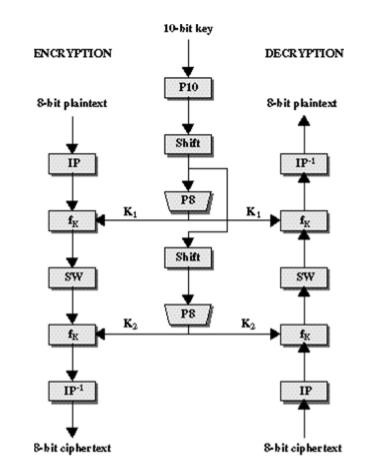
Short-Procedure:

Step1: The overall structure of the simplified DES is given below.

Step2: The S-DES encryption algorithm takes an 8- bit block of plaintext (example: 10111101) and a 10-bit key as input and produces an 8-bit block of cipher-text as output.

Step3: The S-DES decryption algorithm takes an 8-bit block of cipher-text and the same 10-bit key used to produce that cipher-text as input and produces the original 8bit block of plaintext

*STRUCTURE OF ENCRYPTION AND DECRYPTION*



========================================================== Coding:

|  |
| --- |
| FIXED\_IP = [2, 6, 3, 1, 4, 8, 5, 7]  FIXED\_IP = [2, 6, 3, 1, 4, 8, 5, 7]  FIXED\_EP = [4, 1, 2, 3, 2, 3, 4, 1]  FIXED\_IP\_INVERSE = [4, 1, 3, 5, 7, 2, 8, 6]  FIXED\_P10 = [3, 5, 2, 7, 4, 10, 1, 9, 8, 6]  FIXED\_P8 = [6, 3, 7, 4, 8, 5, 10, 9]  FIXED\_P4 = [2, 4, 3, 1]    S0 = [[1, 0, 3, 2],  [3, 2, 1, 0],  [0, 2, 1, 3],  [3, 1, 3, 2]]    S1 = [[0, 1, 2, 3],  [2, 0, 1, 3],  [3, 0, 1, 0],  [2, 1, 0, 3]]    KEY = '0111111101'  def permutate(original, fixed\_key): new = '' for i in fixed\_key: new += original[i - 1] return new  def left\_half(bits): return bits[:(len(bits)) // 2]  def right\_half(bits):  return bits[len(bits) // 2:]  def shift(bits):  rotated\_left\_half = left\_half(bits)[1:] + left\_half(bits)[0] rotated\_right\_half = right\_half(bits)[1:] + right\_half(bits)[0] return rotated\_left\_half + rotated\_right\_half  def key1(): print(' Generating Key1')  return permutate(shift(permutate(KEY, FIXED\_P10)), FIXED\_P8)  def key2():  print(' Generating Key2')  return permutate(shift(shift(permutate(KEY, FIXED\_P10))), FIXED\_P8) |
| def xor(bits, key): new = '' for bit, key\_bit in zip(bits, key): new += str(((int(bit) + int(key\_bit)) % 2)) return new    def lookup\_in\_sbox(bits, sbox):  row = int(bits[0] + bits[3], 2) col = int(bits[1] + bits[2], 2)  return '{0:02b}'.format(sbox[row][col])  def f\_k(bits, key): L = left\_half(bits) R = right\_half(bits) bits = permutate(R, FIXED\_EP) print(" E/P output = " + bits) print(" Key = " + key) bits = xor(bits, key)  bits = lookup\_in\_sbox(left\_half(bits), S0) + lookup\_in\_sbox(right\_half(bits), S1) bits = permutate(bits, FIXED\_P4) print(" P4 = " + bits) return xor(bits, L)  def encrypt(plain\_text): print("Plain Text: " + plain\_text) bits = permutate(plain\_text, FIXED\_IP) print("Initial Permutation = " + bits) print("Round function 1 ") temp = f\_k(bits, key1()) bits = right\_half(bits) + temp print("Round function 2 ") bits = f\_k(bits, key2())  print("Cipher Text = " + permutate(bits + temp, FIXED\_IP\_INVERSE))  def decrypt(cipher\_text):  print("Cipher Text: " + cipher\_text) bits = permutate(cipher\_text, FIXED\_IP) print("Initial Permutation = " + bits) print("Round function 1 ") temp = f\_k(bits, key2()) bits = right\_half(bits) + temp print("Round function 2 ") bits = f\_k(bits, key1())  print("Plain Text = " + permutate(bits + temp, FIXED\_IP\_INVERSE))  if \_\_name\_\_== "\_\_main\_\_": encrypt('11101010') decrypt('00000110') |

==========================================================

Sample Input and Output:

Plain Text: 11101010

Initial Permutation = 10110011

Round function 1

Generating Key1

E/P output = 10010110

Key = 01011111

P4 = 1010

Round function 2

Generating Key2

E/P output = 10000010

Key = 01111101

P4 = 0111

Cipher Text = 00000110

Cipher Text: 00000110

Initial Permutation = 01000001

Round function 1

Generating Key2

E/P output = 10000010

Key = 01111101

P4 = 0111

Round function 2

Generating Key1

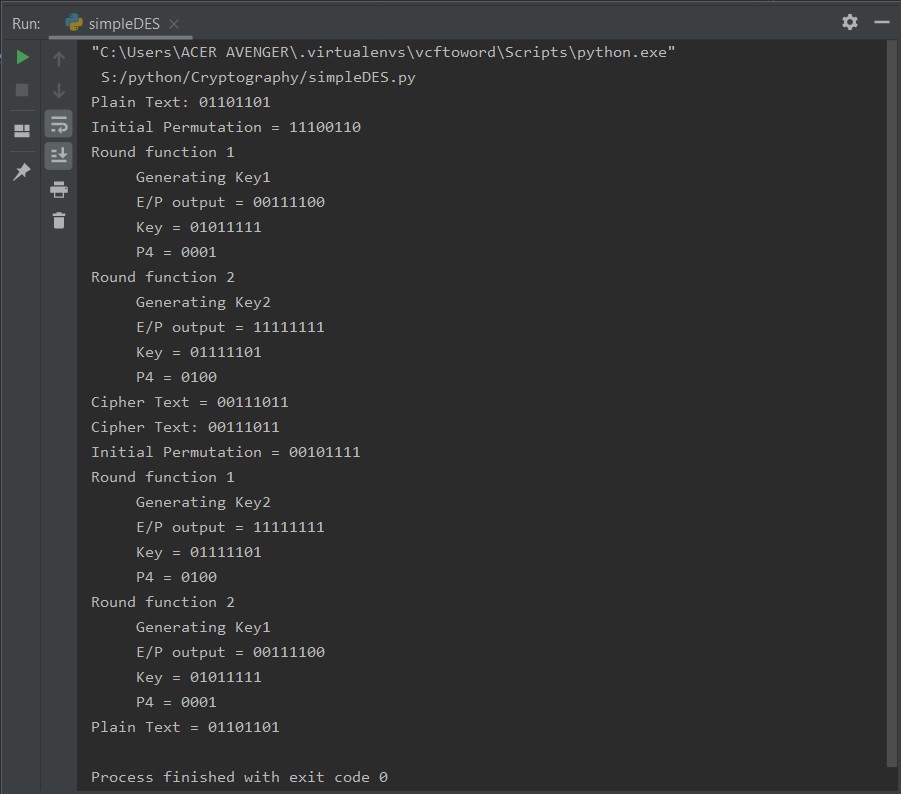
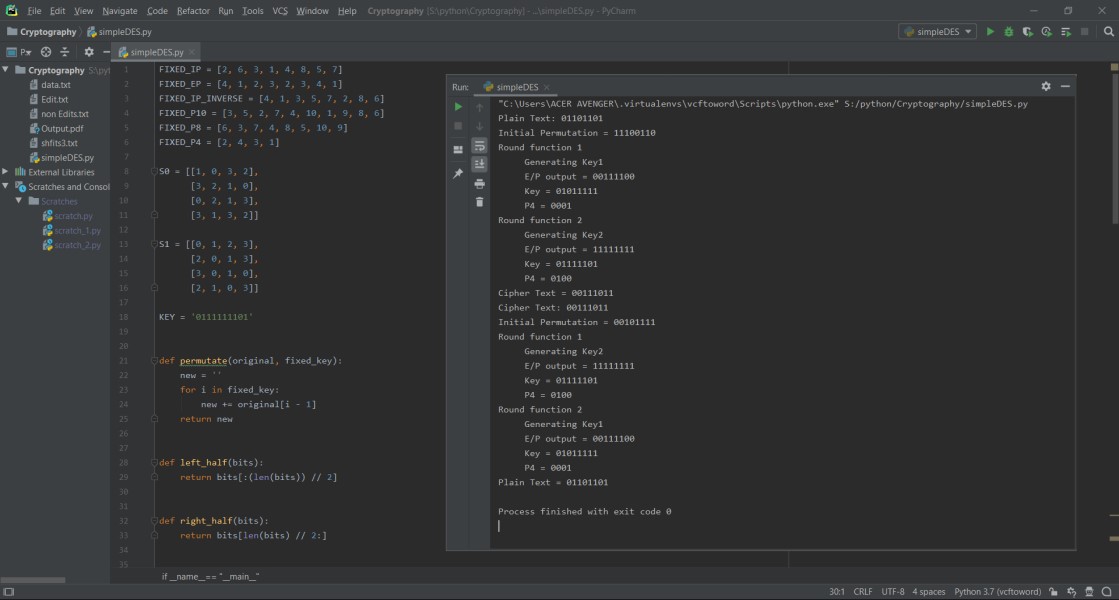
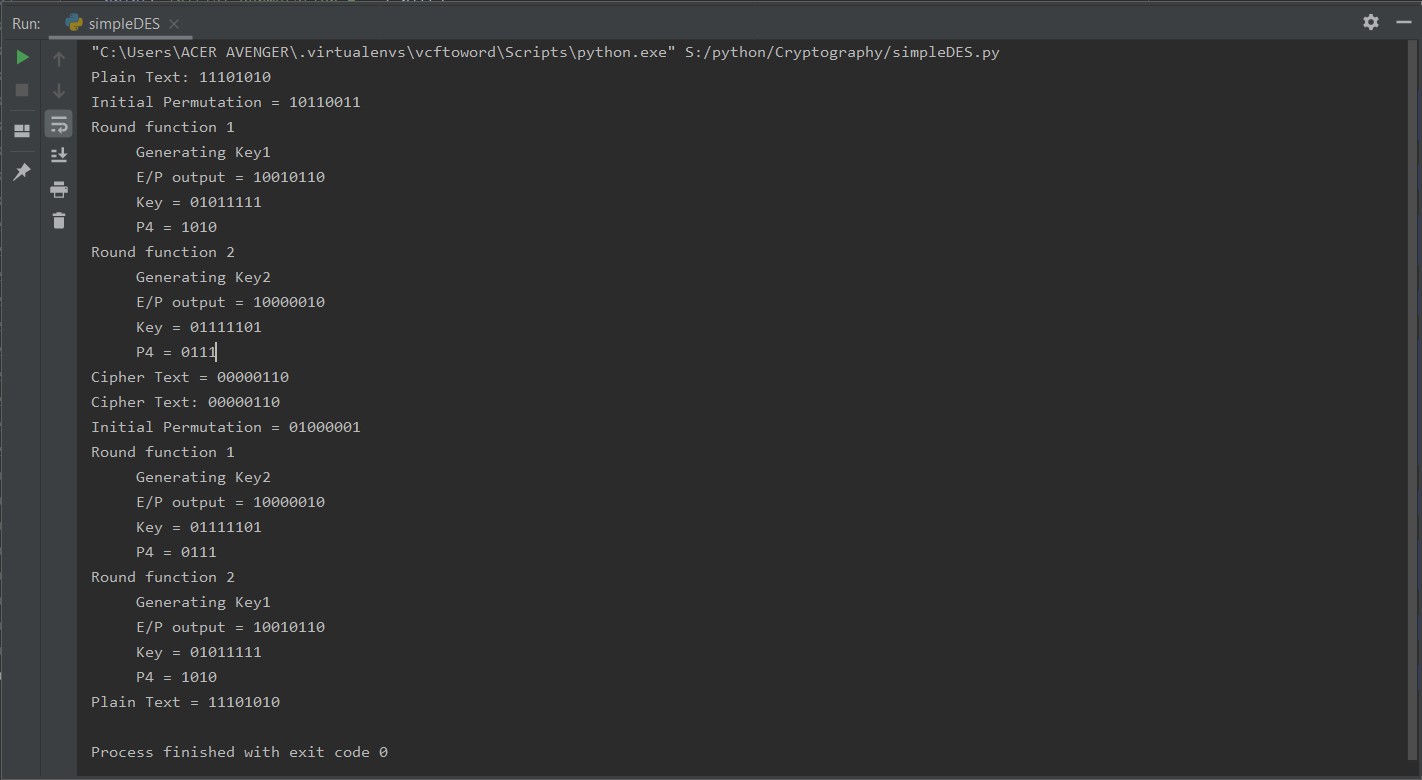
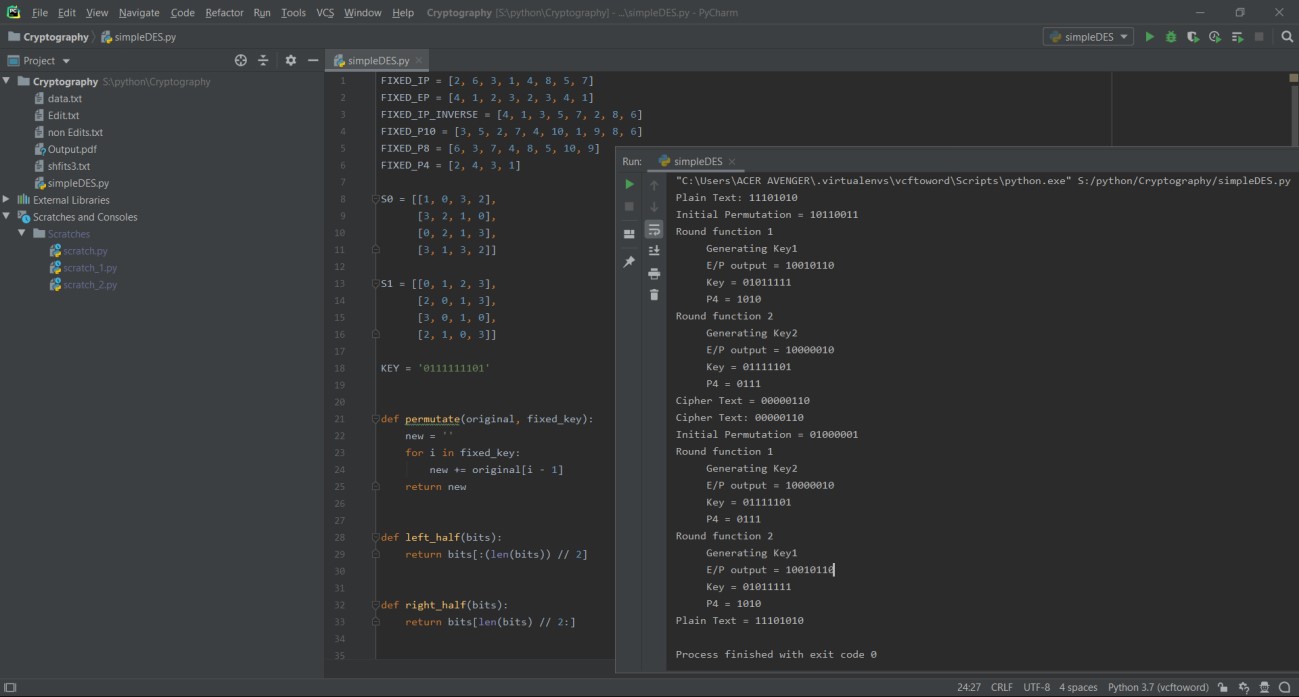
E/P output = 10010110

Key = 01011111

P4 = 1010

Plain Text = 11101010

Code Output:



Result:

Thus the Simple-DES Encryption and decryption algorithm is implemented and verified