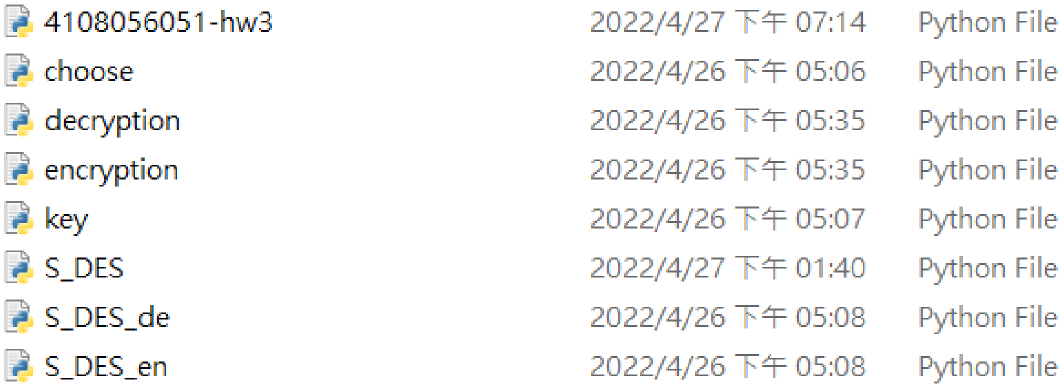
S-DES 實作作業

4108040040 魯姵妤：Qt 介面設計
4108056048 王宥嘉：S-DES 加解密
4108056051 鄭穎：PyQt5 主程式

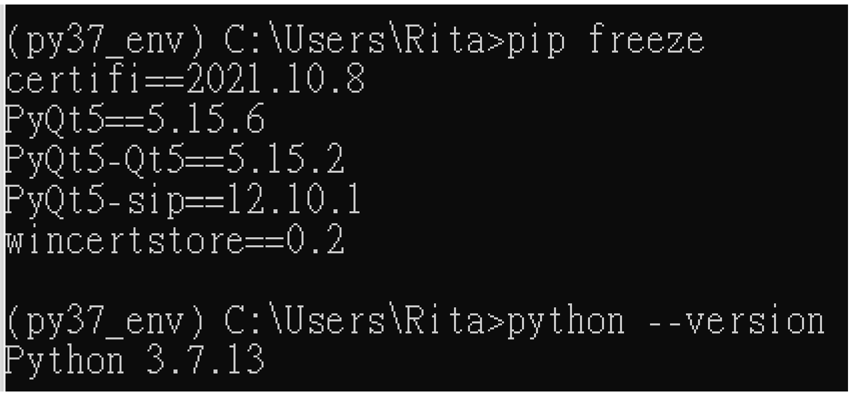
1. 程式執行方式
   (1) 下載附件中的八個.py 檔並放在同一資料夾內



4108056051-hw3	2022/4/27 下午 07:14	Python File
choose	2022/4/26 下午 05:06	Python File
decryption	2022/4/26 下午 05:35	Python File
encryption	2022/4/26 下午 05:35	Python File
key	2022/4/26 下午 05:07	Python File
S_DES	2022/4/27 下午 01:40	Python File
S_DES_de	2022/4/26 下午 05:08	Python File
S_DES_en	2022/4/26 下午 05:08	Python File

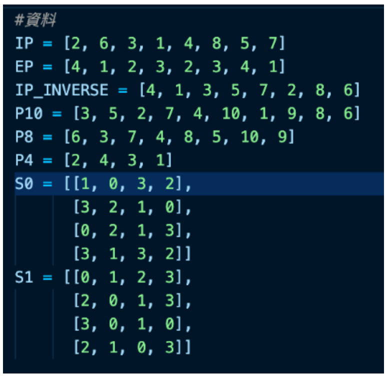
   (2) 終端機輸入：python 4108056051-hw3.py

2. Requirement:



```
(py37_env) C:\Users\Rita>pip freeze
certifi==2021.10.8
PyQt5==5.15.6
PyQt5-Qt5==5.15.2
PyQt5-sip==12.10.1
wincertstore==0.2

(py37_env) C:\Users\Rita>python --version
Python 3.7.13
```

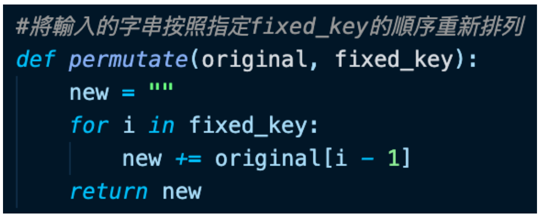
3. S-DES 程式碼說明
   (1) 資料：最上方為 S-DES 需要用到的資料



```
#資料
IP = [2, 6, 3, 1, 4, 8, 5, 7]
EP = [4, 1, 2, 3, 2, 3, 4, 1]
IP_INVERSE = [4, 1, 3, 5, 7, 2, 8, 6]
P10 = [3, 5, 2, 7, 4, 10, 1, 9, 8, 6]
P8 = [6, 3, 7, 4, 8, 5, 10, 9]
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]]
```

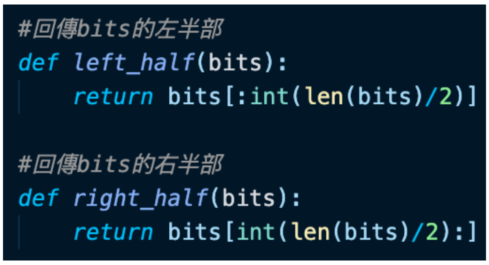
(2) 函式

permutate 會將傳入字串根據指定 fixed_key 的順序做重新排列



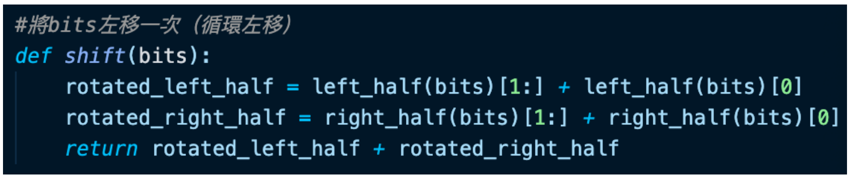
```
#將輸入的字串按照指定fixed_key的順序重新排列
def permutate(original, fixed_key):
    new = ""
    for i in fixed_key:
        new += original[i - 1]
    return new
```

left_half, right_half 分別回傳傳入字串的左半部和右半部



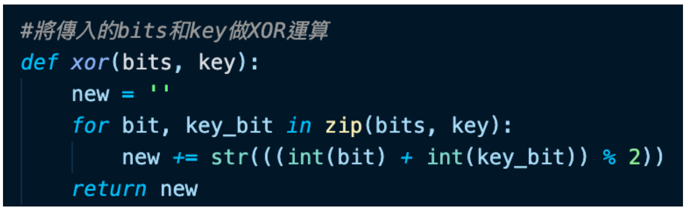
```
#回傳bits的左半部
def left_half(bits):
    return bits[:int(len(bits)/2)]

#回傳bits的右半部
def right_half(bits):
    return bits[int(len(bits)/2):]
```

將字串循環左移一次



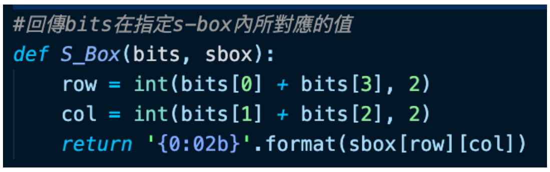
```
#將bits左移一次（循環左移）
def shift(bits):
    rotated_left_half = left_half(bits)[1:] + left_half(bits)[0]
    rotated_right_half = right_half(bits)[1:] + right_half(bits)[0]
    return rotated_left_half + rotated_right_half
```

將傳入字串和 key 做 XOR 運算



```
#將傳入的bits和key做XOR運算
def xor(bits, key):
    new = ''
    for bit, key_bit in zip(bits, key):
        new += str(((int(bit) + int(key_bit)) % 2))
    return new
```

根據字串回傳在指定 s-box 內對應的值

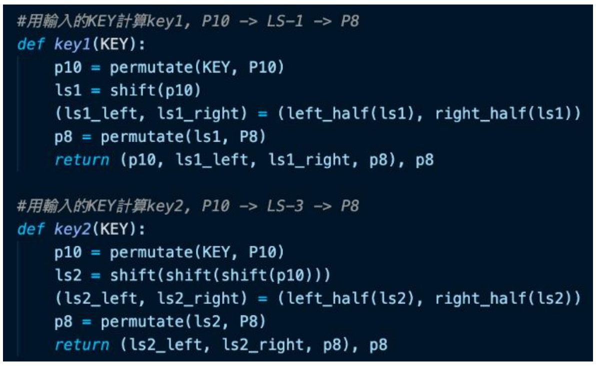


```python
#回傳bits在指定s-box內所對應的值
def S_Box(bits, sbox):
    row = int(bits[0] + bits[3], 2)
    col = int(bits[1] + bits[2], 2)
    return '{0:02b}'.format(sbox[row][col])
```

利用 permutate 和 shift 計算 key1 和 key2，並回傳各步驟計算結果

key 1 : P10 -> LS-1 -> P8

key 2 : P10 -> LS-3 -> P8



```python
#用輸入的KEY計算key1, P10 -> LS-1 -> P8
def key1(KEY):
    p10 = permutate(KEY, P10)
    ls1 = shift(p10)
    (ls1_left, ls1_right) = (left_half(ls1), right_half(ls1))
    p8 = permutate(ls1, P8)
    return (p10, ls1_left, ls1_right, p8), p8

#用輸入的KEY計算key2, P10 -> LS-3 -> P8
def key2(KEY):
    p10 = permutate(KEY, P10)
    ls2 = shift(shift(shift(p10)))
    (ls2_left, ls2_right) = (left_half(ls2), right_half(ls2))
    p8 = permutate(ls2, P8)
    return (ls2_left, ls2_right, p8), p8
```

加解密使用的 round function，並回傳各步驟計算結果

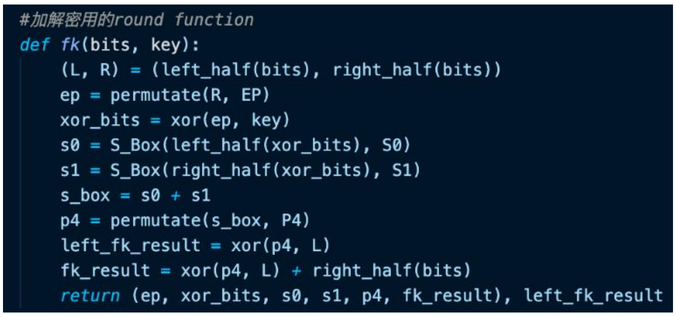
i. 將字串右半部依照 EP 做 permutate

ii. 和指定的 key 做 XOR

iii. 取得 s-box 內相對應值

iv. 根據 P4 做 permutate，得到 fk 的左半部結果

v. 和最初字串的左半部做 XOR，並加上最初字串的右半部



```python
#加解密用的round function
def fk(bits, key):
    (L, R) = (left_half(bits), right_half(bits))
    ep = permutate(R, EP)
    xor_bits = xor(ep, key)
    s0 = S_Box(left_half(xor_bits), S0)
    s1 = S_Box(right_half(xor_bits), S1)
    s_box = s0 + s1
    p4 = permutate(s_box, P4)
    left_fk_result = xor(p4, L)
    fk_result = xor(p4, L) + right_half(bits)
    return (ep, xor_bits, s0, s1, p4, fk_result), left_fk_result
```

根據 plaintext 和 key 執行加密

i. 將 plaintext 依照 IP 進行 permutate

ii. 取得 key1 和計算 key1 間的各步驟結果

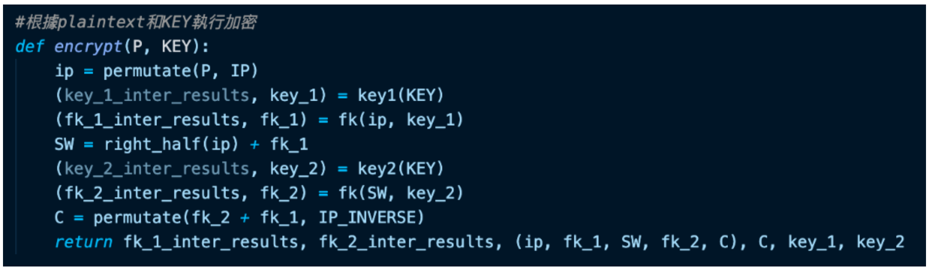
iii. 用 key1 執行 fk，取得各步驟計算結果

iv. 執行 SW，將右半部 ip 結果接上左半部的 fk_1 結果

v. 取得 key2 和計算 key2 間的各步驟結果

vi. 用 key2 執行 fk，取得各步驟計算結果

vii. 將左半部的 fk_2 結果接上左半部的 fk_1 結果，並依照 IP_INVERSE 進行 permutate，得到最終 ciphertext



```
#根據plaintext和KEY執行加密
def encrypt(P, KEY):
    ip = permutate(P, IP)
    (key_1_inter_results, key_1) = key1(KEY)
    (fk_1_inter_results, fk_1) = fk(ip, key_1)
    SW = right_half(ip) + fk_1
    (key_2_inter_results, key_2) = key2(KEY)
    (fk_2_inter_results, fk_2) = fk(SW, key_2)
    C = permutate(fk_2 + fk_1, IP_INVERSE)
    return fk_1_inter_results, fk_2_inter_results, (ip, fk_1, SW, fk_2, C), C, key_1, key_2
```

根據 ciphertext 和 key 執行解密

i. 將 ciphertext 依照 IP 進行 permutate

ii. 取得 key2 和計算 key2 間的各步驟結果

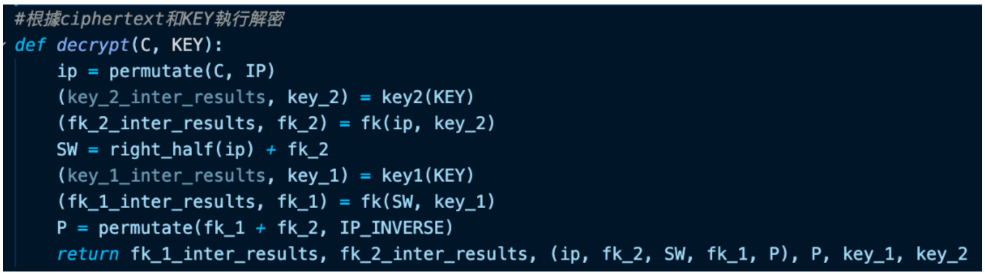
iii. 用 key2 執行 fk，取得各步驟計算結果

iv. 執行 SW，將右半部 ip 結果接上左半部的 fk_2 結果

v. 取得 key1 和計算 key1 間的各步驟結果

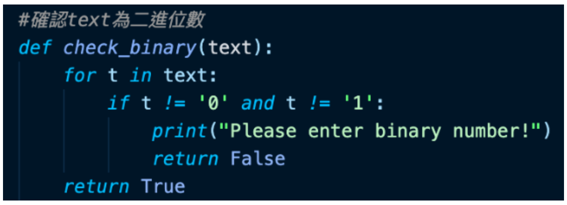
vi. 用 key1 執行 fk，取得各步驟計算結果

vii. 將左半部的 fk_1 結果接上左半部的 fk_2 結果，並依照 IP_INVERSE 進行 permutate，得到最終 plaintext



```
#根據ciphertext和KEY執行解密
def decrypt(C, KEY):
    ip = permutate(C, IP)
    (key_2_inter_results, key_2) = key2(KEY)
    (fk_2_inter_results, fk_2) = fk(ip, key_2)
    SW = right_half(ip) + fk_2
    (key_1_inter_results, key_1) = key1(KEY)
    (fk_1_inter_results, fk_1) = fk(SW, key_1)
    P = permutate(fk_1 + fk_2, IP_INVERSE)
    return fk_1_inter_results, fk_2_inter_results, (ip, fk_2, SW, fk_1, P), P, key_1, key_2
```

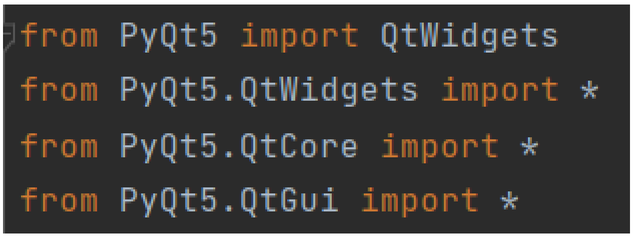
check_binary 確認 text 皆由 0 和 1 組成



```
#確認text為二進位數
def check_binary(text):
    for t in text:
        if t != '0' and t != '1':
            print("Please enter binary number!")
            return False
    return True
```

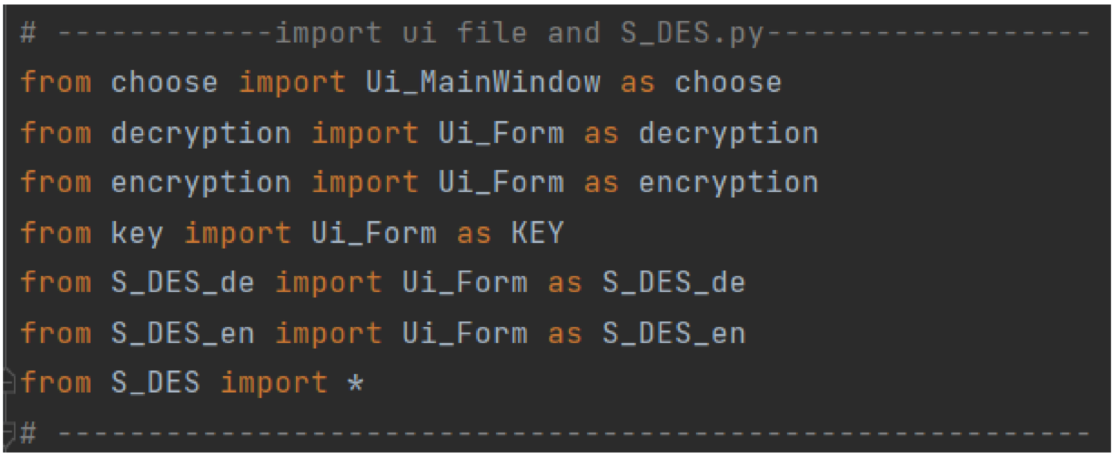
4. PyQt5 主程式

(1) Import 套件



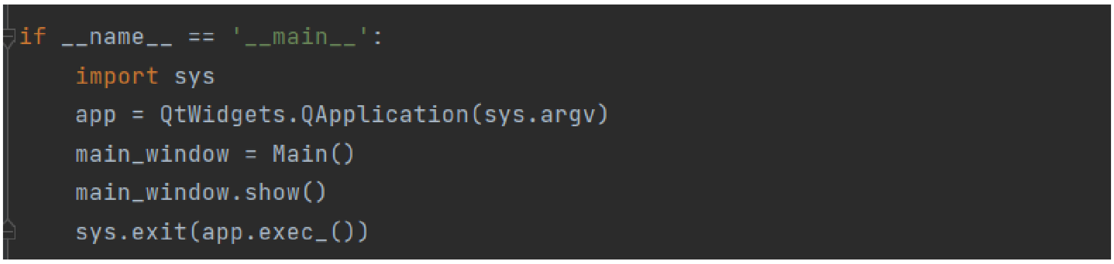
```
from PyQt5 import QtWidgets
from PyQt5.QtWidgets import *
from PyQt5.QtCore import *
from PyQt5.QtGui import *
```

(2) Import S_DES 加解密函式和介面程式



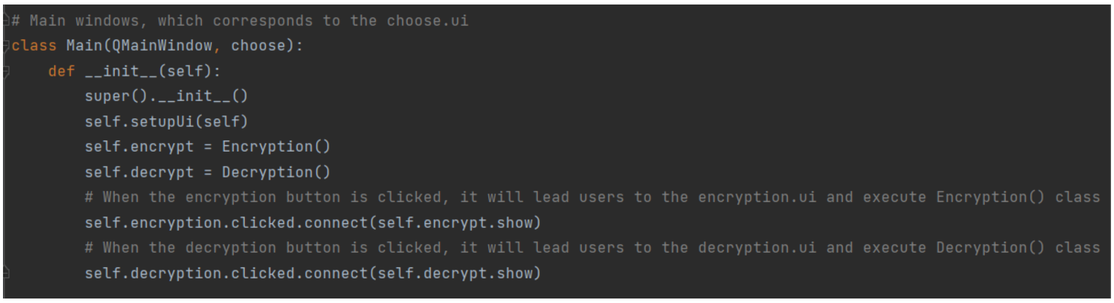
```
# ------------import ui file and S_DES.py------------------
from choose import Ui_MainWindow as choose
from decryption import Ui_Form as decryption
from encryption import Ui_Form as encryption
from key import Ui_Form as KEY
from S_DES_de import Ui_Form as S_DES_de
from S_DES_en import Ui_Form as S_DES_en
from S_DES import *
# ---------------------------------------------------------
```

(3) Main function



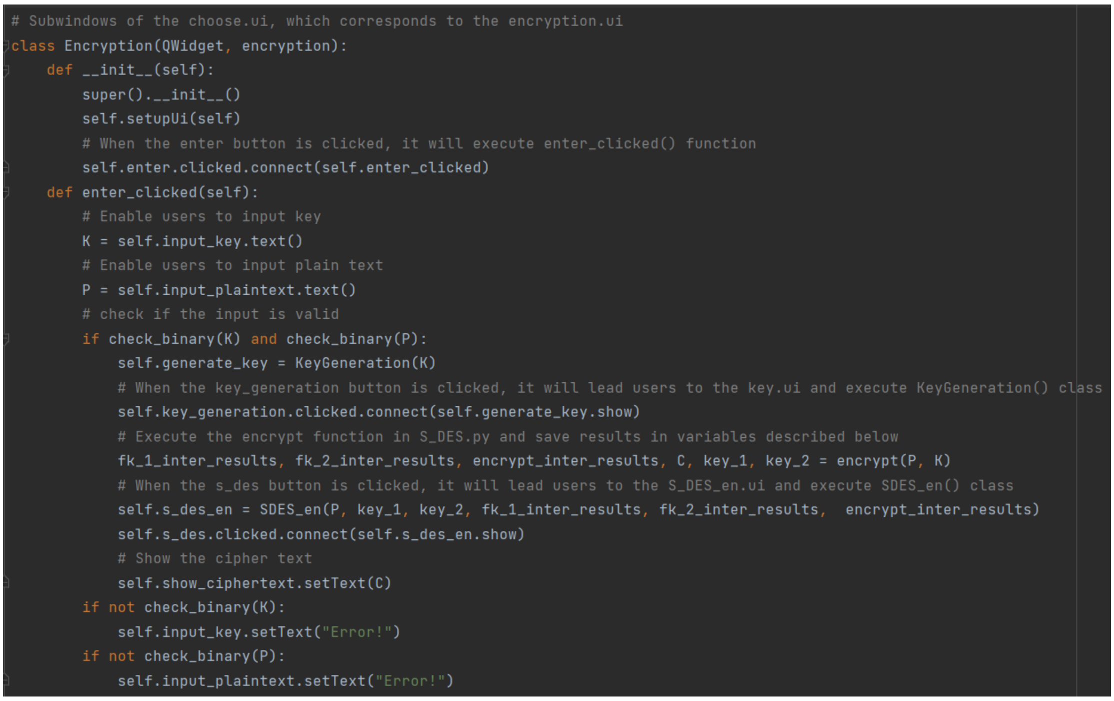
```
if __name__ == '__main__':
    import sys
    app = QtWidgets.QApplication(sys.argv)
    main_window = Main()
    main_window.show()
    sys.exit(app.exec_())
```

(4) 主畫面類別（由此可導入到加密畫面和解密畫面）



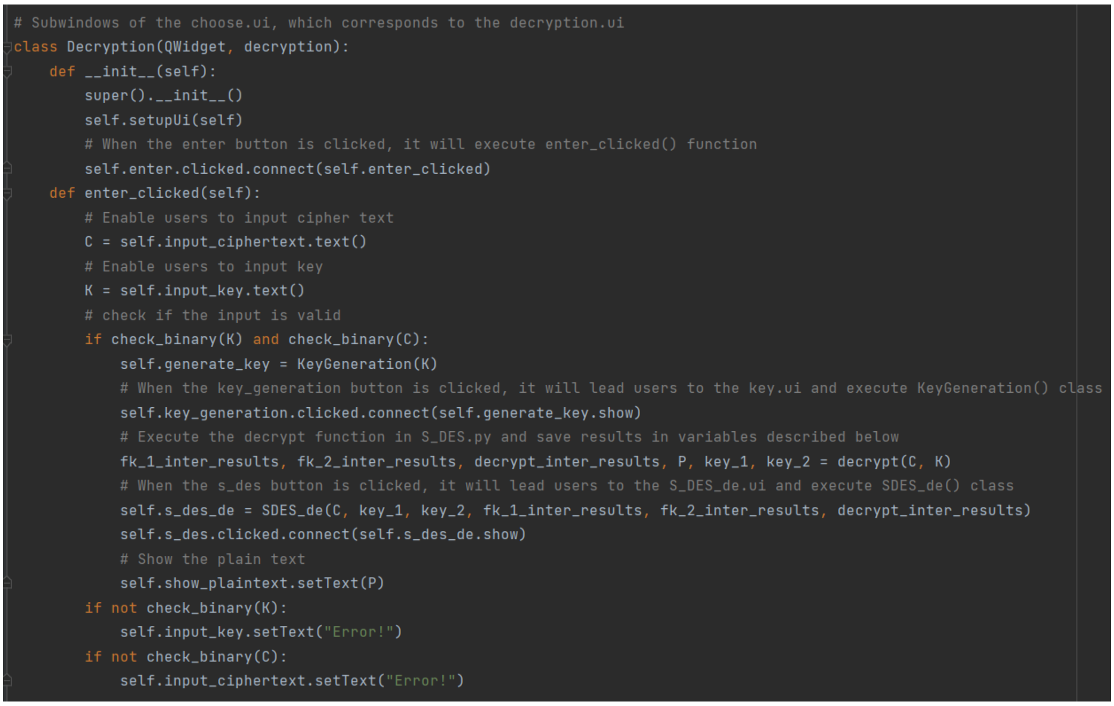
```
# Main windows, which corresponds to the choose.ui
class Main(QMainWindow, choose):
    def __init__(self):
        super().__init__()
        self.setupUi(self)
        self.encrypt = Encryption()
        self.decrypt = Decryption()
        # When the encryption button is clicked, it will lead users to the encryption.ui and execute Encryption() class
        self.encryption.clicked.connect(self.encrypt.show)
        # When the decryption button is clicked, it will lead users to the decryption.ui and execute Decryption() class
        self.decryption.clicked.connect(self.decrypt.show)
```

(5) 加密畫面類別（由此可導入到產生子金鑰的畫面和 S-DES 加密流程的畫面）



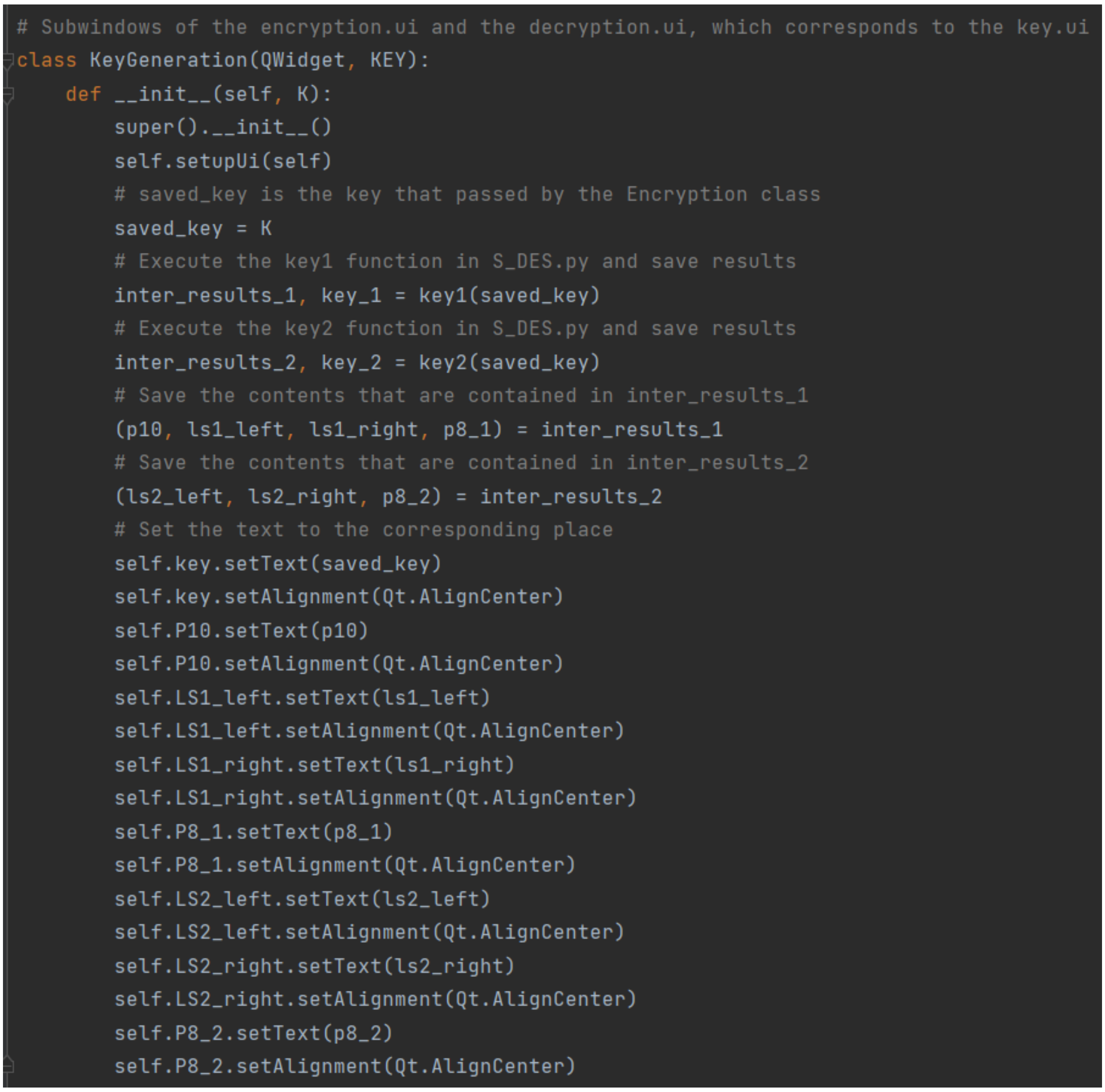
```
# Subwindows of the choose.ui, which corresponds to the encryption.ui
class Encryption(QWidget, encryption):
    def __init__(self):
        super().__init__()
        self.setupUi(self)
        # When the enter button is clicked, it will execute enter_clicked() function
        self.enter.clicked.connect(self.enter_clicked)
    def enter_clicked(self):
        # Enable users to input key
        K = self.input_key.text()
        # Enable users to input plain text
        P = self.input_plaintext.text()
        # check if the input is valid
        if check_binary(K) and check_binary(P):
            self.generate_key = KeyGeneration(K)
            # When the key_generation button is clicked, it will lead users to the key.ui and execute KeyGeneration() class
            self.key_generation.clicked.connect(self.generate_key.show)
            # Execute the encrypt function in S_DES.py and save results in variables described below
            fk_1_inter_results, fk_2_inter_results, encrypt_inter_results, C, key_1, key_2 = encrypt(P, K)
            # When the s_des button is clicked, it will lead users to the S_DES_en.ui and execute SDES_en() class
            self.s_des_en = SDES_en(P, key_1, key_2, fk_1_inter_results, fk_2_inter_results,  encrypt_inter_results)
            self.s_des.clicked.connect(self.s_des_en.show)
            # Show the cipher text
            self.show_ciphertext.setText(C)
        if not check_binary(K):
            self.input_key.setText("Error!")
        if not check_binary(P):
            self.input_plaintext.setText("Error!")
```

(6) 解密畫面類別（由此可導入到產生子金鑰的畫面和 S-DES 解密流程的畫面）



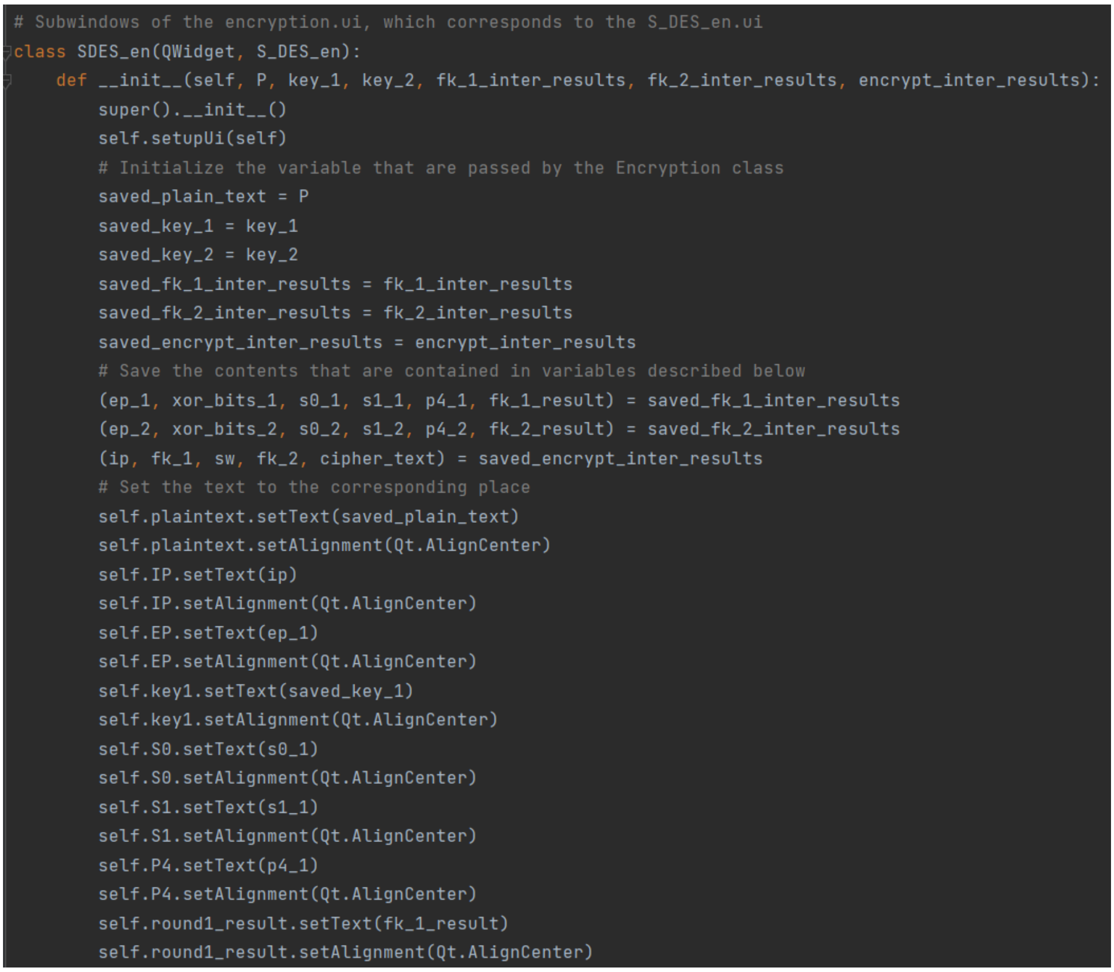
```
# Subwindows of the choose.ui, which corresponds to the decryption.ui
class Decryption(QWidget, decryption):
    def __init__(self):
        super().__init__()
        self.setupUi(self)
        # When the enter button is clicked, it will execute enter_clicked() function
        self.enter.clicked.connect(self.enter_clicked)
    def enter_clicked(self):
        # Enable users to input cipher text
        C = self.input_ciphertext.text()
        # Enable users to input key
        K = self.input_key.text()
        # check if the input is valid
        if check_binary(K) and check_binary(C):
            self.generate_key = KeyGeneration(K)
            # When the key_generation button is clicked, it will lead users to the key.ui and execute KeyGeneration() class
            self.key_generation.clicked.connect(self.generate_key.show)
            # Execute the decrypt function in S_DES.py and save results in variables described below
            fk_1_inter_results, fk_2_inter_results, decrypt_inter_results, P, key_1, key_2 = decrypt(C, K)
            # When the s_des button is clicked, it will lead users to the S_DES_de.ui and execute SDES_de() class
            self.s_des_de = SDES_de(C, key_1, key_2, fk_1_inter_results, fk_2_inter_results, decrypt_inter_results)
            self.s_des.clicked.connect(self.s_des_de.show)
            # Show the plain text
            self.show_plaintext.setText(P)
        if not check_binary(K):
            self.input_key.setText("Error!")
        if not check_binary(C):
            self.input_ciphertext.setText("Error!")
```

(7) 產生子金鑰類別

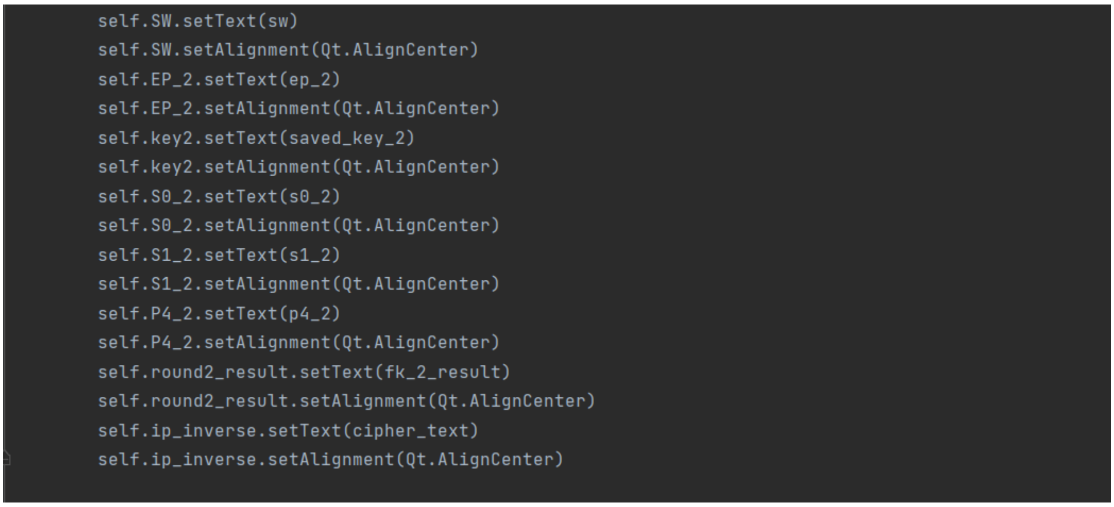


```
# Subwindows of the encryption.ui and the decryption.ui, which corresponds to the key.ui
class KeyGeneration(QWidget, KEY):
    def __init__(self, K):
        super().__init__()
        self.setupUi(self)
        # saved_key is the key that passed by the Encryption class
        saved_key = K
        # Execute the key1 function in S_DES.py and save results
        inter_results_1, key_1 = key1(saved_key)
        # Execute the key2 function in S_DES.py and save results
        inter_results_2, key_2 = key2(saved_key)
        # Save the contents that are contained in inter_results_1
        (p10, ls1_left, ls1_right, p8_1) = inter_results_1
        # Save the contents that are contained in inter_results_2
        (ls2_left, ls2_right, p8_2) = inter_results_2
        # Set the text to the corresponding place
        self.key.setText(saved_key)
        self.key.setAlignment(Qt.AlignCenter)
        self.P10.setText(p10)
        self.P10.setAlignment(Qt.AlignCenter)
        self.LS1_left.setText(ls1_left)
        self.LS1_left.setAlignment(Qt.AlignCenter)
        self.LS1_right.setText(ls1_right)
        self.LS1_right.setAlignment(Qt.AlignCenter)
        self.P8_1.setText(p8_1)
        self.P8_1.setAlignment(Qt.AlignCenter)
        self.LS2_left.setText(ls2_left)
        self.LS2_left.setAlignment(Qt.AlignCenter)
        self.LS2_right.setText(ls2_right)
        self.LS2_right.setAlignment(Qt.AlignCenter)
        self.P8_2.setText(p8_2)
        self.P8_2.setAlignment(Qt.AlignCenter)
```

(8) S-DES 加密流程類別

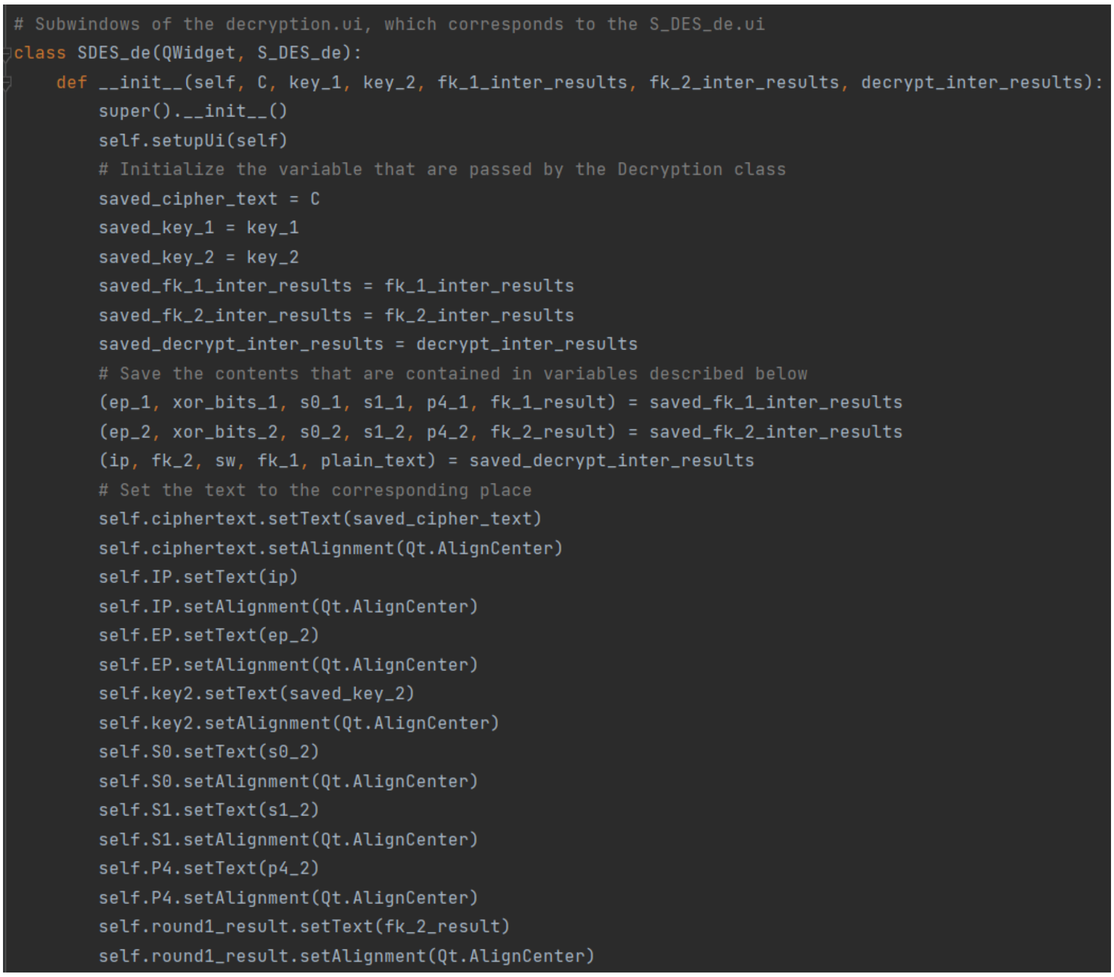


```python
# Subwindows of the encryption.ui, which corresponds to the S_DES_en.ui
class SDES_en(QWidget, S_DES_en):
    def __init__(self, P, key_1, key_2, fk_1_inter_results, fk_2_inter_results, encrypt_inter_results):
        super().__init__()
        self.setupUi(self)
        # Initialize the variable that are passed by the Encryption class
        saved_plain_text = P
        saved_key_1 = key_1
        saved_key_2 = key_2
        saved_fk_1_inter_results = fk_1_inter_results
        saved_fk_2_inter_results = fk_2_inter_results
        saved_encrypt_inter_results = encrypt_inter_results
        # Save the contents that are contained in variables described below
        (ep_1, xor_bits_1, s0_1, s1_1, p4_1, fk_1_result) = saved_fk_1_inter_results
        (ep_2, xor_bits_2, s0_2, s1_2, p4_2, fk_2_result) = saved_fk_2_inter_results
        (ip, fk_1, sw, fk_2, cipher_text) = saved_encrypt_inter_results
        # Set the text to the corresponding place
        self.plaintext.setText(saved_plain_text)
        self.plaintext.setAlignment(Qt.AlignCenter)
        self.IP.setText(ip)
        self.IP.setAlignment(Qt.AlignCenter)
        self.EP.setText(ep_1)
        self.EP.setAlignment(Qt.AlignCenter)
        self.key1.setText(saved_key_1)
        self.key1.setAlignment(Qt.AlignCenter)
        self.S0.setText(s0_1)
        self.S0.setAlignment(Qt.AlignCenter)
        self.S1.setText(s1_1)
        self.S1.setAlignment(Qt.AlignCenter)
        self.P4.setText(p4_1)
        self.P4.setAlignment(Qt.AlignCenter)
        self.round1_result.setText(fk_1_result)
        self.round1_result.setAlignment(Qt.AlignCenter)
```

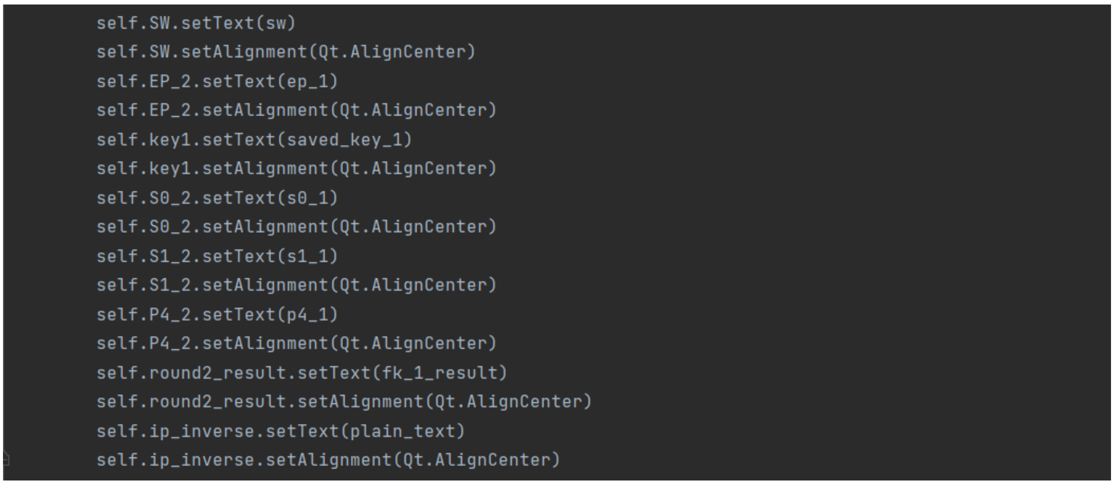


```python
        self.SW.setText(sw)
        self.SW.setAlignment(Qt.AlignCenter)
        self.EP_2.setText(ep_2)
        self.EP_2.setAlignment(Qt.AlignCenter)
        self.key2.setText(saved_key_2)
        self.key2.setAlignment(Qt.AlignCenter)
        self.S0_2.setText(s0_2)
        self.S0_2.setAlignment(Qt.AlignCenter)
        self.S1_2.setText(s1_2)
        self.S1_2.setAlignment(Qt.AlignCenter)
        self.P4_2.setText(p4_2)
        self.P4_2.setAlignment(Qt.AlignCenter)
        self.round2_result.setText(fk_2_result)
        self.round2_result.setAlignment(Qt.AlignCenter)
        self.ip_inverse.setText(cipher_text)
        self.ip_inverse.setAlignment(Qt.AlignCenter)
```

(9) S-DES 解密流程類別



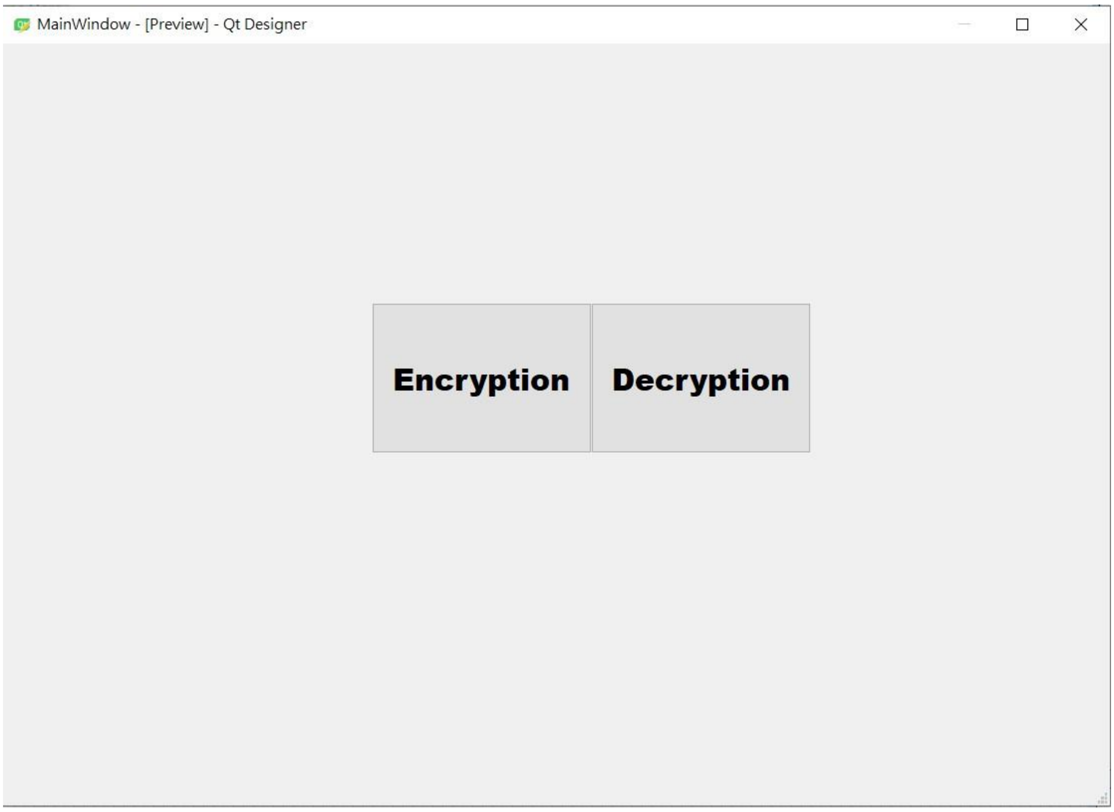
```
# Subwindows of the decryption.ui, which corresponds to the S_DES_de.ui
class SDES_de(QWidget, S_DES_de):
    def __init__(self, C, key_1, key_2, fk_1_inter_results, fk_2_inter_results, decrypt_inter_results):
        super().__init__()
        self.setupUi(self)
        # Initialize the variable that are passed by the Decryption class
        saved_cipher_text = C
        saved_key_1 = key_1
        saved_key_2 = key_2
        saved_fk_1_inter_results = fk_1_inter_results
        saved_fk_2_inter_results = fk_2_inter_results
        saved_decrypt_inter_results = decrypt_inter_results
        # Save the contents that are contained in variables described below
        (ep_1, xor_bits_1, s0_1, s1_1, p4_1, fk_1_result) = saved_fk_1_inter_results
        (ep_2, xor_bits_2, s0_2, s1_2, p4_2, fk_2_result) = saved_fk_2_inter_results
        (ip, fk_2, sw, fk_1, plain_text) = saved_decrypt_inter_results
        # Set the text to the corresponding place
        self.ciphertext.setText(saved_cipher_text)
        self.ciphertext.setAlignment(Qt.AlignCenter)
        self.IP.setText(ip)
        self.IP.setAlignment(Qt.AlignCenter)
        self.EP.setText(ep_2)
        self.EP.setAlignment(Qt.AlignCenter)
        self.key2.setText(saved_key_2)
        self.key2.setAlignment(Qt.AlignCenter)
        self.S0.setText(s0_2)
        self.S0.setAlignment(Qt.AlignCenter)
        self.S1.setText(s1_2)
        self.S1.setAlignment(Qt.AlignCenter)
        self.P4.setText(p4_2)
        self.P4.setAlignment(Qt.AlignCenter)
        self.round1_result.setText(fk_2_result)
        self.round1_result.setAlignment(Qt.AlignCenter)
```



```
        self.SW.setText(sw)
        self.SW.setAlignment(Qt.AlignCenter)
        self.EP_2.setText(ep_1)
        self.EP_2.setAlignment(Qt.AlignCenter)
        self.key1.setText(saved_key_1)
        self.key1.setAlignment(Qt.AlignCenter)
        self.S0_2.setText(s0_1)
        self.S0_2.setAlignment(Qt.AlignCenter)
        self.S1_2.setText(s1_1)
        self.S1_2.setAlignment(Qt.AlignCenter)
        self.P4_2.setText(p4_1)
        self.P4_2.setAlignment(Qt.AlignCenter)
        self.round2_result.setText(fk_1_result)
        self.round2_result.setAlignment(Qt.AlignCenter)
        self.ip_inverse.setText(plain_text)
        self.ip_inverse.setAlignment(Qt.AlignCenter)
```

5. 使用說明

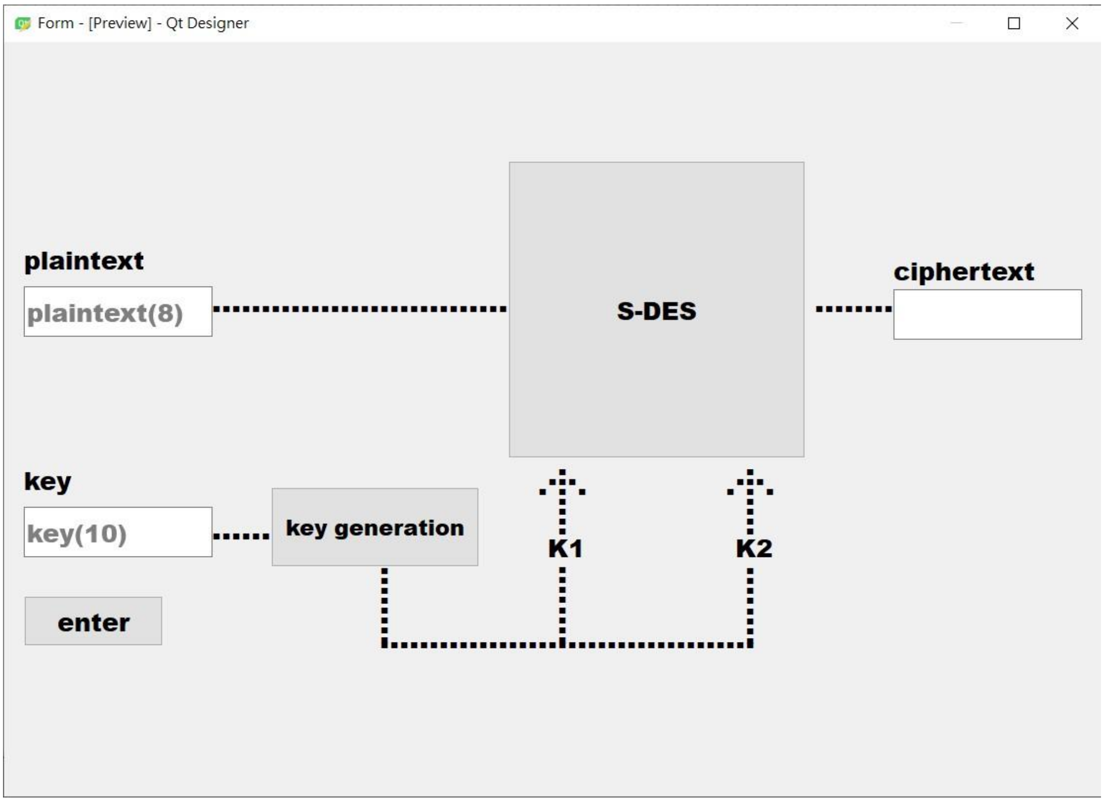
(1) 主畫面



按下(1)：執行加密

按下(2)：執行解密

(2) Encryption 加密



(1)：輸入要加密的字串

(2)：輸入 main key

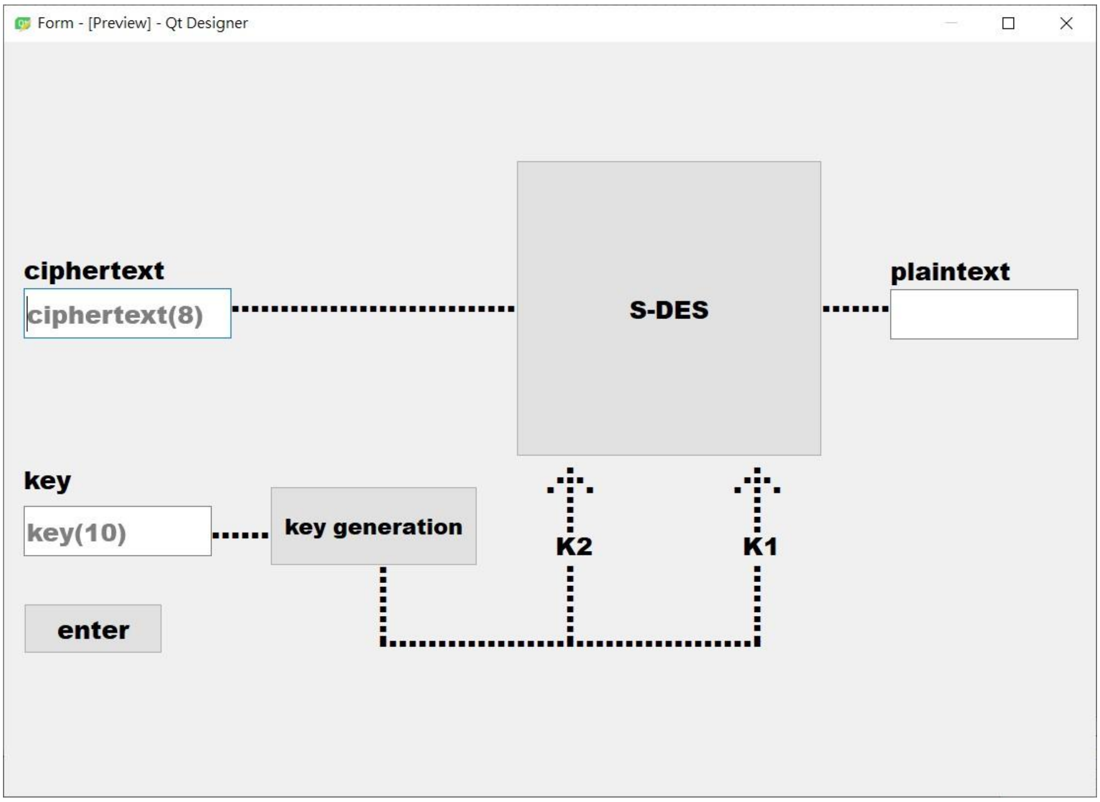
(3)：送出輸入的 plaintext 和 main key

(4)：查看 subkey 產生過程

(5)：查看 S-DES 加密過程

(6)：顯示加密過後的 ciphertext

(3) Decryption 解密



(1)：輸入要解密的字串

(2)：輸入 main key

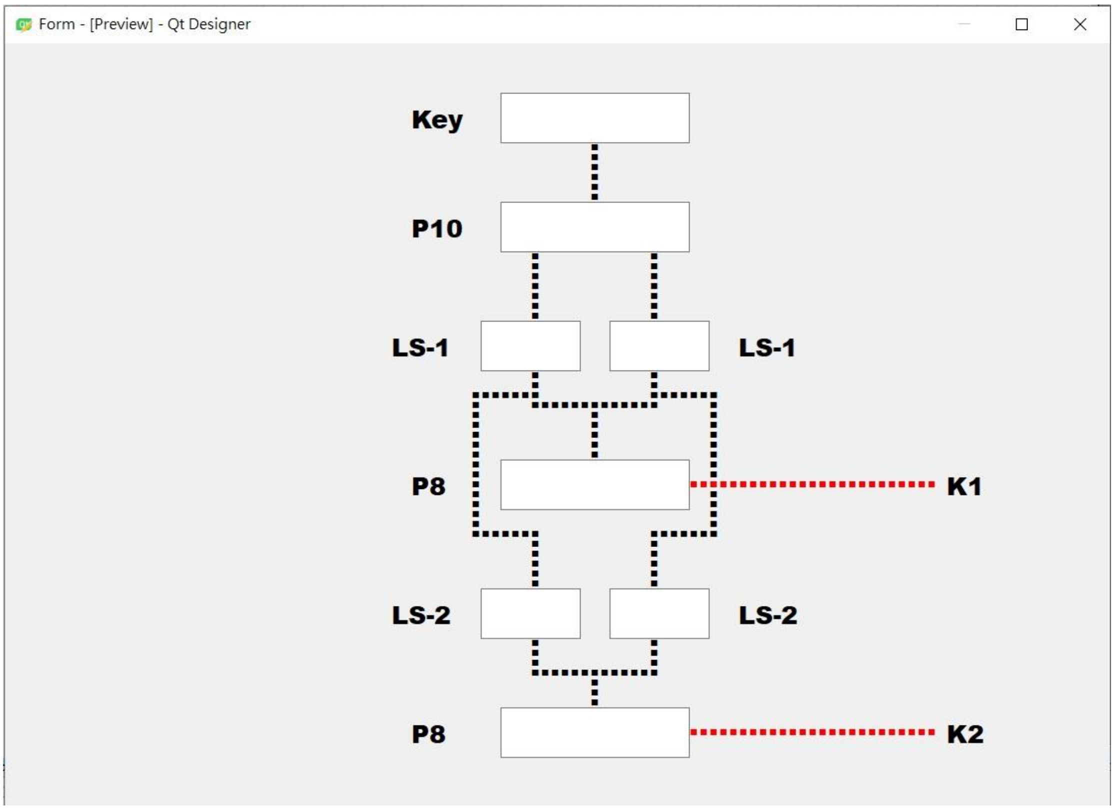
(3)：送出輸入的 ciphertext 和 main key

(4)：查看 subkey 產生過程

(5)：查看 S-DES 加密過程

(6)：顯示加密過後的 plaintext

(4) 產生 subkey



(1)：上個頁面所輸入的 main key

(2)：P10 的結果

(3)：P10 後左半邊 shift left1 格的結果

(4)：P10 後右半邊 shift left1 格的結果

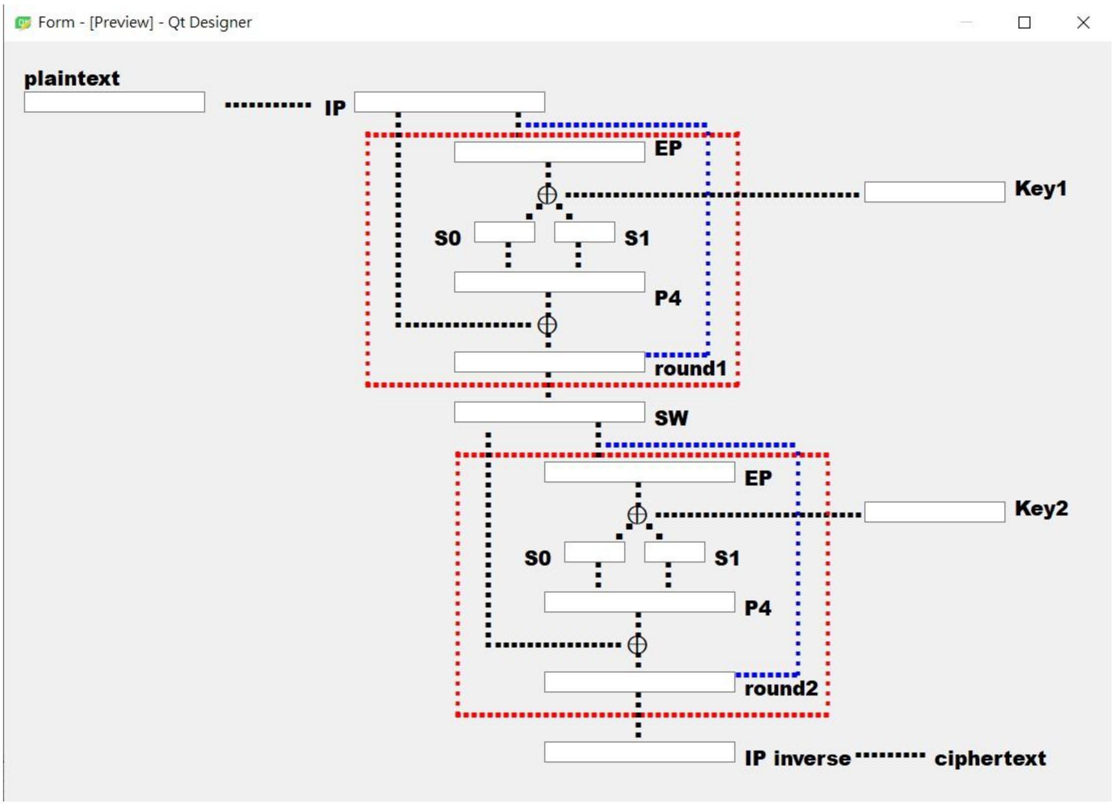
(5)：(3)(4)合併再 P8 所產生 key1(第一把 subkey)

(6)：左半邊 shift left 1 格後再 shift left 2 格的結果

(7)：右半邊 shift left 1 格後再 shift left 2 格的結果

(8)：(6)(7)合併再 P8 所產生 key2(第二把 subkey)

(5) 顯示加密過程



(1)：上個頁面所輸入的 plaintext

(2)：IP 的結果

(3)：EP 的結果

(4)：顯示之前所算出的 key1

(5)：對照 S0 的結果

(6)：對照 S1 的結果

(7)：P4 的結果

(8)：第一個 round 做完的結果

(9)：(8)做左半右半交換的結果

(10)：EP 的結果

(11)：顯示之前所算出的 key2

(12)：對照 S0 的結果

(13)：對照 S1 的結果

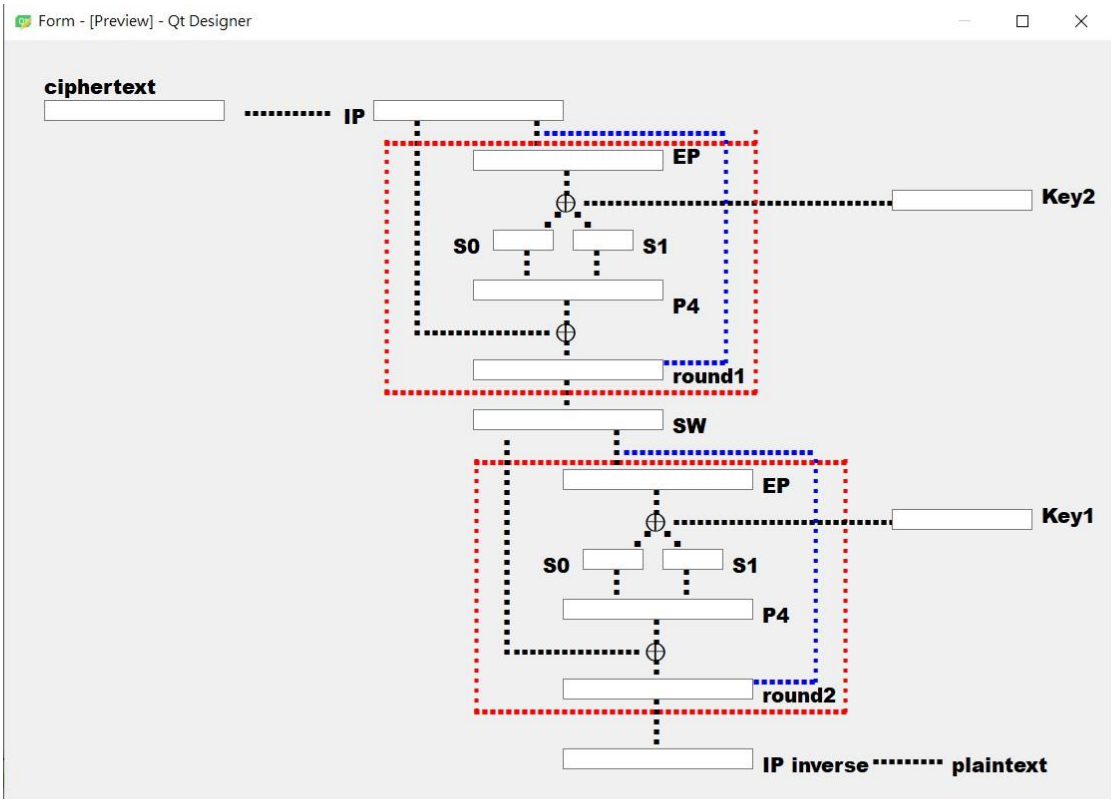
(14)：P4 的結果

(15)：第二個 round 做完的結果

(16)：IP inverse 的結果也就是 ciphertext

紅色框框內為 fk function

(6) 顯示解密過程



(1)：上個頁面所輸入的 ciphertext

(2)：IP 的結果

(3)：EP 的結果

(4)：顯示之前所算出的 key2

(5)：對照 S0 的結果

(6)：對照 S1 的結果

(7)：P4 的結果

(8)：第一個 round 做完的結果

(9)：(8)做左半右半交換的結果

(10)：EP 的結果

(11)：顯示之前所算出的 key1

(12)：對照 S0 的結果

(13)：對照 S1 的結果

(14)：P4 的結果

(15)：第二個 round 做完的結果

(16)：IP inverse 的結果也就是 plaintext

紅色框框內為 fk function