파이썬프로그래밍

문제 3: 영상 파일 암호화 프로그램

2020년 12월 10일 목

컴퓨터공학과

2019305059

이현수

■ 문제 1: 영상 파일 암호화 프로그램

●src_folder 변수로 지정한 폴더 및 그 하부 폴더의 모든 파일을 검색하여 이중 file_type이 지정한 타입의 파일에 대해 암호화를 행하여 해당 파일을 src 폴더의 하와 같은 이름으로 dst_folder 변수로 지정하는 폴더에 저장하는 프로그램 작성.

●암호화하기: 파일의 각 바이트별로 주어진 변수 key8이 지정하는 8비트 값과 XOR 동작을 행한 결과를 상위 4비트(nibble), 하위 4비트의 값의 서로 교환해 암호화.

●복호화하여 저장된 데이터가 올바로 복원되는지 보이기 dst_folder에 저장된 영상들을 하나씩 읽어서 복호화 작업을해 src_folder에 있는 원본 영상의 데이터 값과 바이트 단위로 비교하여 모든 데이터의 값이 같으면 그 파일은 정 상적으로 저장된 것으로 간주. 이 검증 과정에 대한 결과는 지정된 형식으로 출력.

암호화, 복호화 못해 나머지 코드만이라도 작성함.

file_type리스트에 있는 확장자 파일을 src_folder에서 읽어 dst_folder에 저장하고, src_folder에 있는 원본파일과 dst_folder에 있는 복사한파일을 바이트별로 읽어 비교해 같은지 아닌지를 지정된 형식으로 출력함.

■ 코드

```
1  src_folder = 'mission/'
2  dst_folder = 'dst/'
3  file_type = ['jpg', 'png', 'tif', 'txt', 'py']
4  key8=0b0011_1001
5
6  import os
7  import time
8
9  sz = os.path.getsize(__file__)
10 print(f'Program Size = {sz:#,}\n')#파일사이즈 출력
11
12 s_time = time.time() #시간측정 시작
13
14 if os.path.exists(dst_folder): #dst_folder 폴더가 있다면
15     pass #아무동작도 안함.
16 else: #dst_folder 폴더가 없다면
17     os.mkdir(dst_folder) #dst_folder 폴더 만들기
18
19 for i, j in enumerate(file_type): #처리할 확장자에 .확장자 형태로 저장
20     temp = '.'+j
21     file_type[i] = temp
22
```

파일사이즈 출력(주석문 포함 크기)

암호화된 파일 저장할 폴더 만들기

file_type에 있는 처리할 확장자앞에 . 붙이기

```
lst_src=[] #file_type에 있는 확장자 파일 정보가 담길 리스트
for i, (path, sub_dir, files) in enumerate(os.walk(src_folder)):
    for filename in files:
        ext = os.path.splitext(filename)[-1]
        ext_and_lst = file_type
        for ext_nm in ext_and_lst:
            if ext_nm == ext.lower():  # 소문자로 바꾸어 비교한다.
                a = os.path.join(path, filename) #파일이름 저장
                b = os.path.getsize(path+"/"+ filename) #파일크기 저장
                lst_temp = [a,b]
                lst_src.append(lst_temp)
                break
```

src_folder에서 file_type에 있는 확장자 파일 정보 얻기

```
for i in lst_src:
    dst = dst_folder + i[0][i[0].rfind('\\') + 1:]#저장경로+파일이름+확장자
    f1 = open(i[0], 'rb') # i[0]파일을 'rb'모드로
    f2 = open(dst, 'wb')  # dst파일로 저장. 'wb'모드로
    f = f1.read() #f에 f1내용읽어들이기
    f2.write(f) #f2에 f내용 쓰기
    f1.close() #f1파일닫기
    f2.close() #f2파일닫기

#암호화, 복호화 구현 못함
```

lst_src에 있는 모든 파일 열어서 암호하시키고 dst_folder에 저장하기
(암호화 못시킴. 그냥 저장함)

```
index=1
ch='\\'
for i in lst_src:
    filename_dst = dst_folder+i[0][i[0].rfind('\\')+1:] #암호화후 저장된경로+파일이름+확장자
    f1 = open(filename_dst, 'rb') #암호화후 저장된 파일 'rb'로 열기
    f2 = open(i[0],'rb') #원본파일 i[0]을 'rb'로 열기
    b = True # b가 True로 유지되면 두 파일내용이 같다.
    byte1 = f1.read(1) #byte1변수에 f1파일을 1바이트씩 읽기
    byte2 = f2.read(1) #byte2변수에 f2파일을 1바이트씩 읽기
    while byte1: #파일끝까지 계속해서 읽기
        if byte1!=byte2: #만약 1바이트씩 비교해서 다르면
            b=False #b를 False로
            break #while반복문 빠져나오기
        byte1 = f1.read(1) #f1을 계속해서 1바이트씩 읽기
        byte2 = f2.read(1) #f2를 계속해서 1바이트씩 읽기
    if b==True: #b가 True라면, 만약 두 파일내용이 같다면
        print(f'#{index:03}: {i[0][i[0].rfind(ch)+1:]}({i[1]:#,}), pass!!') #지정
        print(f'location: {i[0][:i[0].rfind(ch)]}')
        index=index+1
        print()
    else: #같지 않으면
        print(f'#{index:03}: {i[0][i[0].rfind(ch) + 1:]}({i[1]:#,}), fail!!')  #지정된형식 + fail 출력
        print(f'location: {i[0][:i[0].rfind(ch)]}')
        index = index + 1
        print()
    f1.close() #f1파일닫기
    f2.close() #f2파일닫기

c_time = time.time() #시간측정 종료
print('\nTotal Execution Time = {0:.4f}[sec]'.format(c_time - s_time)) #소요된 시간 출력
```

dst_folder에 있는 파일을 복호화해 lst_src의 파일을 열어서 복호화된 파일과 비교.
(복호화 못시킴. 그냥 바이트별로 비교함)

■ 코드설명

```
1   src_folder = 'mission/'
2   dst_folder = 'dst/'
3   file_type = ['jpg', 'png', 'tif', 'txt', 'py']
4   key8=0b0011_1001
5
6   import os
7   import time
8
9   sz = os.path.getsize(__file__)
10  print(f'Program Size = {sz:#,}\n')#파일사이즈 출력
11
12  s_time = time.time() #시간측정 시작
13
14  if os.path.exists(dst_folder): #dst_folder 폴더가 있다면
15      pass #아무동작도 안함.
16  else: #dst_folder 폴더가 없다면
17      os.mkdir(dst_folder) #dst_folder 폴더 만들기
18
19  for i, j in enumerate(file_type): #처리할 확장자에 .확장자 형태로 저장
20      temp = '.'+j
21      file_type[i] = temp
22
```

os모듈을 사용해 파일사이즈를 출력한다. 그리고 time모듈을 이용해 시간측정을 시작한다.(s_time = time.time()) (코드14-17) 원하는 파일을 암호화시킨 후 dst_floder 위치에 저장해야 한다. 그러기 위해서는 os.path.exists(dst_folder)를 통해 이미 폴더가 존재하는지를 체크하고 있다면 아무동작도 하지말고, 없다면 os.mkdir 메소드를 사용해 폴더를 만들어 준다.

(코드19-21)문제3은 file_type에 있는 확장자 파일을 암호화 하는 문제이다. file_type리스트 안에는 확장자 이름만 들어있다. 그래서 for반복문에서 enumerate(file_type)을 통해 확장자에 '.'+확장자이름을 temp변수에 만들어 준 후 리스트의 각각의 원소를 temp변수로 변경한다.

```
23  lst_src=[] #file_type에 있는 확장자 파일 정보가 담길 리스트
24  for i, (path, sub_dir, files) in enumerate(os.walk(src_folder)):
25      for filename in files:
26          ext = os.path.splitext(filename)[-1]
27          ext_and_lst = file_type
28          for ext_nm in ext_and_lst:
29              if ext_nm == ext.lower():  # 소문자로 바꾸어 비교한다.
30                  a = os.path.join(path, filename) #파일이름 저장
31                  b = os.path.getsize(path+"/"+ filename) #파일크기 저장
32                  lst_temp = [a,b]
33                  lst_src.append(lst_temp)
34                  break
35
```

확장자별로 따로 저장할필요가 없기 때문에 lst_src리스트 한 개만 만든다. os모듈의 메소드들을 사용해 원하는 확장자 파일을 분석해 a변수에 파일이름, b변수에 파일사이즈를 저장하고 a,b를 리스트로 묶어 lst_src에 append 한다.

```
for i in lst_src:
    dst = dst_folder + i[0][i[0].rfind('\\') + 1:]#저장경로+파일이름+확장자
    f1 = open(i[0], 'rb') # i[0]파일을 'rb'모드로
    f2 = open(dst, 'wb')  # dst파일로 저장. 'wb'모드로
    f = f1.read() #f에 f1내용읽어들이기
    f2.write(f) #f2에 f내용 쓰기
    f1.close() #f1파일닫기
    f2.close() #f2파일닫기

#암호화, 복호화 구현 못함

```

원하는 파일정보가 lst_src에 담겼다면 for반복문을 통해 정보를 한 개씩 가져와 i에 저장한다. i는 리스트로 되어 있다. dst변수는 암호화후 저장할 위치+파일이름으로 만든다.

f1에 원본파일을 'rb'로 열고, f2에 암호화된 파일을 저장하기 위해서 'wb'모드로 dst파일로 저장한다.

그 후 f에 f1.read()를 통해 원본파일을 읽어서 저장한다. 그 후 f2.write(f)로 f2에 f를 쓴다. 그 후 f1, f2를 닫는다.

위 코드에서 파일을 연후 암호화과정을 해야하지만 암호화하지 못함.

```
index=1
ch='\\'
for i in lst_src:
    filename_dst = dst_folder+i[0][i[0].rfind('\\')+1:] #암호화후 저장된경로+파일이름+확장자
    f1 = open(filename_dst, 'rb') #암호화후 저장된 파일 'rb'로 열기
    f2 = open(i[0],'rb') #원본파일 i[0]을 'rb'로 열기
    b = True # b가 True로 유지되면 두 파일내용이 같다.
    byte1 = f1.read(1) #byte1변수에 f1파일을 1바이트씩 읽기
    byte2 = f2.read(1) #byte2변수에 f2파일을 1바이트씩 읽기
    while byte1: #파일끝까지 계속해서 읽기
        if byte1!=byte2: #만약 1바이트씩 비교해서 다르면
            b=False #b를 False로
            break #while반복문 빠져나오기
        byte1 = f1.read(1) #f1을 계속해서 1바이트씩 읽기
        byte2 = f2.read(1) #f2를 계속해서 1바이트씩 읽기
    if b==True: #b가 True라면, 만약 두 파일내용이 같다면
        print(f'#{index:03}: {i[0][i[0].rfind(ch)+1:]}({i[1]:#,}), pass!!') #지정된형식으로 출력
        print(f'location: {i[0][:i[0].rfind(ch)]}')
        index=index+1
        print()
    else: #같지 않으면
        print(f'#{index:03}: {i[0][i[0].rfind(ch) + 1:]}({i[1]:#,}), fail!!')  #지정된형식 + fail 출력
        print(f'location: {i[0][:i[0].rfind(ch)]}')
        index = index + 1
        print()
    f1.close() #f1파일닫기
    f2.close() #f2파일닫기

c_time = time.time() #시간측정 종료
print('\nTotal Execution Time = {0:.4f}[sec]'.format(c_time - s_time)) #소요된 시간 출력
```

위 코드들은 암호화된 파일들을 dst_folder위치에서 읽어서 원본파일과 dst_folder에있는 파일을 바이트별로 비교해 갔을 경우에 지정된형식으로 pass를 출력하고, 아니라면 지정된형식으로 fail를 출력한다.

for반복문으로 lst_src를 한 개씩 읽어서 i변수에 저장한다. 그다음 filename_dst변수는 암호화 위치 + 파일이름과 확장자를 저장해준다. 그 후 f1은 filename_dst 즉 암호화된 파일을 읽고, f2는 i[0] 즉 원본파일을 읽는다.

그다음에 변수 b가 True로 나온다. 만약 b가 False로 변경되면 암호화파일과 복호화파일이 일치하지 않는것이다.

byte1 = f1.read(1), byte2 = f2.read(1)을 통해 f1, f2파일을 각각 1바이트씩 읽어서 저장한다. 그리고 while반복문에 들어가서 f1을 끝까지 읽을때까지 반복한다. while반복문 안에는 byte1!=byte2 조건문이 있다. 만약 두개가 일치하지 않을 경우 b=False로 만들어주고 break를 통해 반복문을 빠져나간다. 만약 일치하면 계속해서 1바이트씩 읽어서 비교한다.

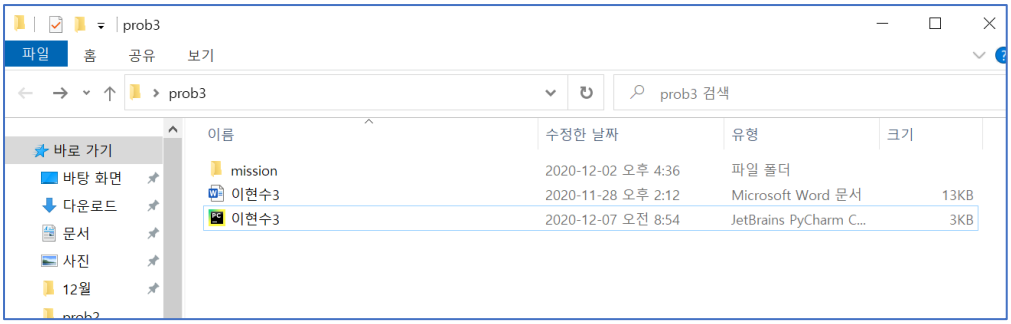
반복문에서 빠져나오면 조건문을 통해 b가 True일 경우와 아닐경우로 나뉜다. True일 경우는 암호화된 파일과 복호화된 파일일 일치하는 경우, 즉 복호화가 잘 처리된경우 지정된형식으로 pass를 출력해주고, 아니라면 지정된 형식과 함께 fail를 출력해준다.

그리고 f1., f2 파일을 닫아준다.

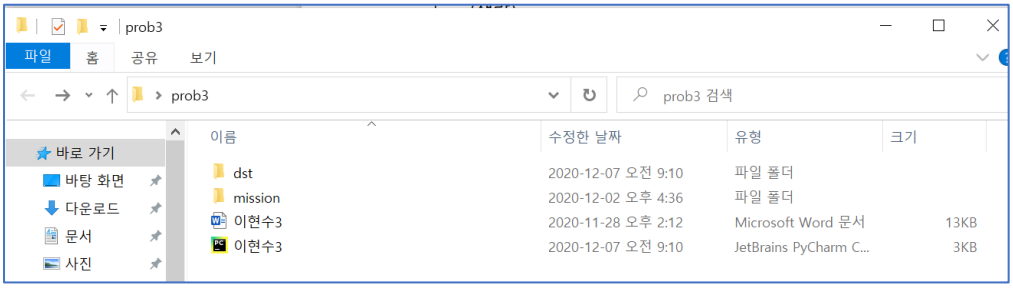
마지막에 시간측정을 종료하고 소요된시간을 출력한다.

위 코드에서 복호화 코드가 있어야 하지만 작성하지 못함.

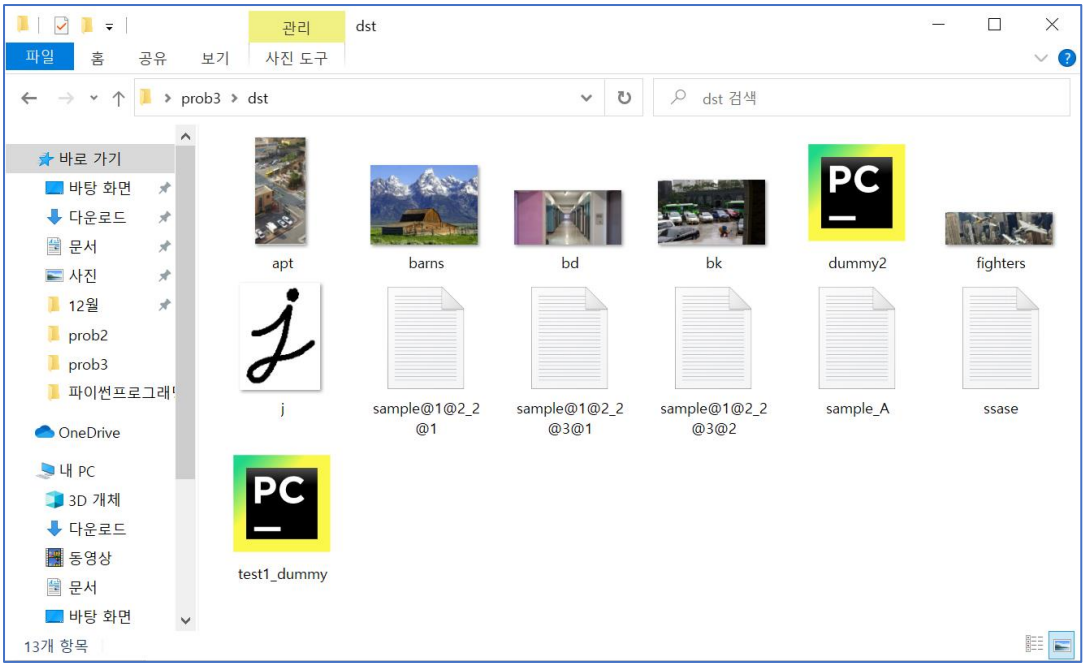
■ 실행



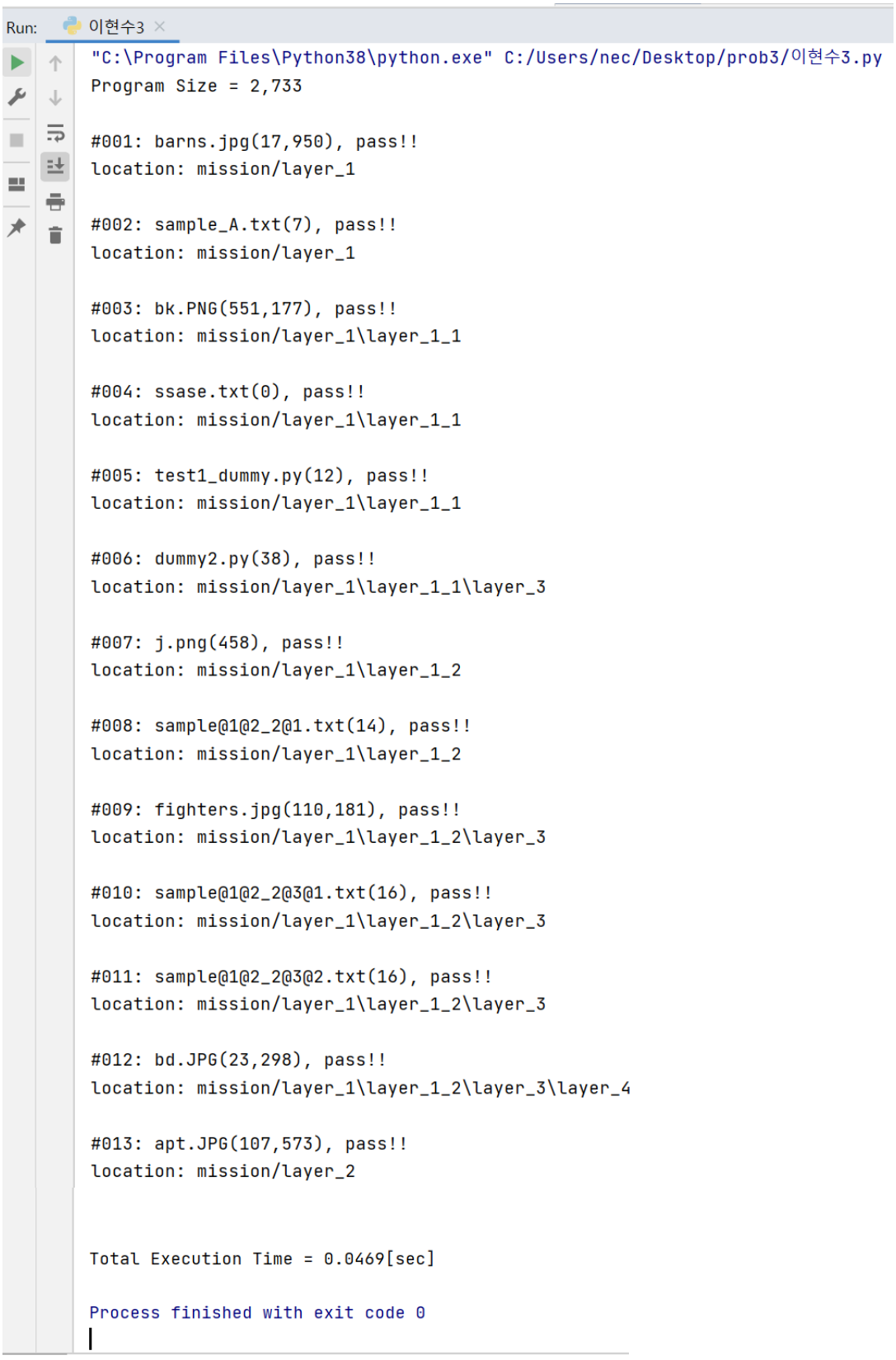
실행전



실행후(dst폴더생김)



dst폴더안에 지정한 jpg, png, tif, txt, py 5개의 확장자파일 13개가 정상적으로 복사됨.



13개 파일에 대해서 번호, 파일이름, 파일크기, 통과여부, 위치가 정확하게 출력된다.

■ 교훈 및 시행착오

xor연산을 시도했지만 성공하지 못해 암호화와 복호화를 하지 못해 나머지 과정이라도 코드를 작성했다.