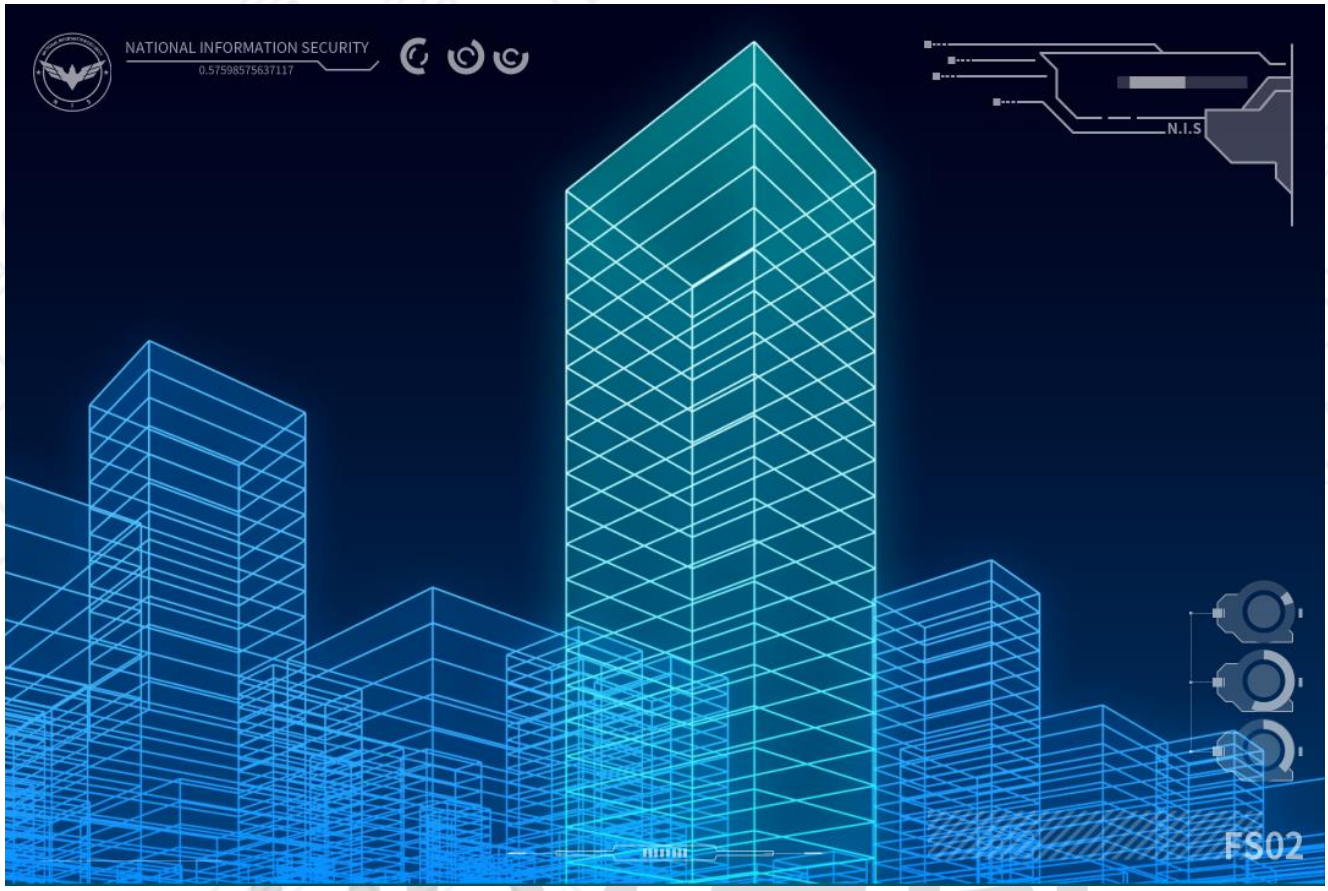
[폭발물 설치 범인 추적 작전]



폭발물 해체 매뉴얼을 이용하여 암호해독기를 완성한 파이

폭발물을 성공적으로 해체하여 시청은 안전해졌다.

시청 안에 남아있는 단서를 이용해 폭발물을 설치한 범인을 찾아야 하는데….

오늘의 목표	폭발물을 설치한 범인 추적하기
주요 함수	print, range, append, decode

➤ 학습 목표

→ 스토리의 **사건**을 이해하며 미션을 해결하기 위해 필요한 데이터를 수집할 수 있습니다.

→ while 반복 제어문을 사용하여 조건에 따라 반복을 중지하는 코드를 작성할 수 있습니다.

→ 2 차원 리스트의 형태를 알고 각 리스트의 값에 접근하여 요소를 추출할 수 있습니다.

→ 항목을 서로 비교하는 비교 연산자에 대해 알고 적절하게 사용할 수 있습니다.

→ 요소의 포함 여부를 판단하는 포함 연산자를 활용하여 값의 존재 여부를 알아낼 수 있습니다.

→ 큰 수와 문자를 간결하게 나타내는 16 진수 표기법에 대해 알고, hex code 수로 표현된 값을 디코딩하여 문자열로 변환할 수 있습니다.

→ 주요 함수를 배우고 활용해 데이터를 분석함으로써 사건을 해결할 수 있습니다.

✔ 수업 전 체크하기

→ '수업 환경 테스트' 를 눌러 수업 환경을 체크합니다.

→ 학생과 반갑게 인사를 나누고 전 시간 스토리에 대해 이야기를 나눕니다.

✔ 수업 중 체크하기

→ 스토리의 내용을 잘 이해했는지 적절한 질문을 통해 지속적으로 확인합니다.

→ 코드를 입력할 때, 대소문자를 정확히 입력하였는지, 따옴표와 괄호의 짝이 잘 맞는지 확인하도록 합니다.

→ 띄어쓰기가 잘 되어 있는지 확인합니다.

✔ 수업 후 체크하기

→ 스토리를 잘 이해했는지 확인합니다.

→ 파이썬 프로그래밍 개념을 잘 이해했는지 확인합니다.

→ 화면의 '수업 종료' 버튼을 누른 뒤 피드백을 작성합니다. (하단 피드백 예시 참고)

Intro.

● 이전 차시 스토리에 관해 이야기를 나눕니다.

[이전 스토리 이해]

<이전 스토리>

파이와 이선은 시청 건물 20 층 33 호에서 발견된 폭발물의 종류를 파악했다. 곧바로 현장에 투입된 전문요원이 해체 작업을 할 수 있도록 암호 해독기를 만들었다. 매뉴얼에 따라 전선으로 이루어진 기폭장치를 해체할 수 있는 해독기 프로그램을 작성하여 전선 색과 수가 다른 박스 총 3 개를 성공적으로 해체했다.

→ NIS 시스템에 접속하기 위한 ID 와 PW 를 학생에게 전달합니다.

- NIS 정보 요원이 되어 사건을 해결할 ID 를 부여합니다.
- ID : agent@nis.com / PW : python

Mission1. cctv 파일 확인하기.

● 미션의 목표를 설명합니다

[미션 목표]

1) 스토리 속에서 단서를 확인하여 유실된 cctv 파일의 정확한 구간을 찾아낼 수 있습니다.
2) while 반복 제어문을 사용하여 조건에 따라 반복을 중지하는 코드를 작성할 수 있습니다.
3) 2 차원 리스트의 형태를 알고 각 리스트의 값에 접근하여 요소를 추출할 수 있습니다.

[스토리 이해]

< 단서 >

● 스토리의 내용을 이해합니다. 학생 스스로 클릭하면서 스토리를 읽도록 합니다.

(폭발물을 성공적으로 해체하여 시청은 안전해졌다.)
(이선과 파이는 폭발물을 설치한 범인을 찾기 위해 단서를 확보하고 있다.)

"파이", "시청에 요청했던 33 호 내부에 설치된 CCTV 파일이 왔어."
"이선", "응. 한번 볼게."
"파이", "파이! 시청에서 보내준 파일은 총 7 일 분량이고 파일 1 개당 1 시간의 영상 기록물이 있대."
"이선", "그래? 다 확인해 보려면 시간이 좀 걸리겠는걸."
"파이", "오래 걸릴 것 같아?"
"이선", "… 잠깐만! 이선! 7 일 분량의 모든 파일을 보내온 게 맞아?"
"파이", "응! 내가 시청 담당자한테 한 번 더 확인했었는데… 왜?"
"이선", "파일 개수가 안 맞는거 같은데…."
"파이", "그럴 리 없어! "
"이선", "누군가 조작한 흔적이 있는데 어느 구간이 이상한지 확인해 봐야겠어."
"파이", "응 알겠어!"

<<< 단서 확인하기 >>>

<단서>
사라진 파일의 날짜를 찾아라.

[학습 자료]

→ 파이는 cctv 파일 리스트를 보고 어떻게 7 일 분량의 파일 개수에 문제가 있는지 알아냈는지 이야기 나누어 봅니다.

(예시) cctv 기록 파일은 1 시간 간격으로 저장되고, 하루에는 24 개의 파일이 쌓일 것이므로 7 일 동안 누적된 파일의 개수는 총 7*24 = 168 개여야 합니다. 데이터가 저장된 파일 목록을 보고 이를 확인하던 중 개수가 맞지 않는 점을 먼저 알아차렸을 것입니다.

● 학습하게 될 개념에 대해 해당 코드를 구현하면서 적절하게 설명합니다.

 # 2 차원 리스트

- 리스트 안에 리스트가 들어 있는 형태
- 선언 : 변수 =[['요소 1',...'요소 n']['요소 1',...'요소 n']...]

  ex) double_list = [['a', 'b'], ['c', 'd'], ['e', 'f']]
- 인덱스
  - 리스트 요소의 순서로 0 부터 시작

    ex) list_test = [[10, 20], [30, 40], [50, 60]]

내부 인덱스	[0, 1]	[0, 1]	[0, 1]
list_test 인덱스	0	1	2

- 요소 접근 방법 : 리스트[리스트 인덱스][내부 리스트 인덱스]

  ex) print(list_test[1])

  print(list_test[0][2])

 # 반복문

- 같은 일을 반복할 때 사용하는 제어문
- 구조 : while_break
  - while 조건이 참인 동안 반복 수행
  - 반복을 중지할 때 break 사용

```
while 조건:
  수행할 문장
  if 반복을 중지할 조건:
    break

ex)
count = 5
while count > 0:
  print(count)
  count = count - 1
  if count == 1:
    break
```

 # 비교 연산자

- 숫자, 문자열 등 두 항목을 서로 비교하는 연산자

x == y	x 와 y 가 같다
x != y	x 와 y 는 같지 않다

- 결과는 True 또는 False 반환

  ex) 1 != 'one'

  'Sun' == 'Sun'

→ while 반복문은 for 반복문과 달리 반복할 횟수를 정확히 알지 못할 때 사용합니다. 반복 제어문의 기본 구조와 같이 반복할 조건을 지정해 주고, 들여쓰기를 지켜 수행할 문장을 작성한 다음 반복을 중지하고자 하는 시점이나 조건 아래에 break 를 써줍니다. while 반복문 내를 순회하다 반복 조건이 종료되거나 break 를 만나면 중지됩니다.

[코드 이해]

- 사용할 API 에 대해 설명합니다.

< API >

1) len(리스트) : 리스트 요소의 개수 반환 함수

2) print(값) : 값을 출력하는 함수

● 학생이 스스로 코드를 구현하여 정답을 찾아 확인하도록 합니다.

- 스토리 속에서 해결해야 할 미션에 대해 질문하며 해당 코드를 학생 스스로 구현하도록 설명해줍니다. 코드는 제공하지 않습니다.

< CODE >

```
#data : 모범답안에서 확인
#일주일치의 cctv 파일,하루에 24 개의 파일(7*24=168 개)
print(len(file_list))

#정답 확인:
144

#없어진 파일 구간 확인하기
i=0
while i < len(file_list):
    if file_list[i+1][0] - file_list[i][0] != 1:
        print(file_list[i][0],file_list[i+1][0])
        break
    i=i+1

#결과 확인:
72 97

#시간대 확인하기
i=0
while i < len(file_list):
    if file_list[i+1][0] - file_list[i][0] != 1:
        print(file_list[i][1], file_list[i+1][1])
        break
```

```
    i=i+1

#결과 확인:
012723 012900

#정답 확인:
0128
```

→ 주어진 data 에 대해 이야기 나누어 봅니다.

(예시) 일주일 분량의 cctv 파일을 나타낸 2 차원 리스트 입니다. 1 시간마다 1 개의 파일을 저장한다고 했으므로 하루에는 총 24 개의 파일이, 일주일 동안에는 24*7 일 = 168 개의 파일이 존재해야 합니다.

→ 리스트에 저장된 파일의 총 개수를 구하기 위해 어떻게 해야 할지 이야기 해 봅니다.

(예시) 리스트 요소의 개수를 반환하는 len 함수를 이용하여 총 파일의 개수를 알아낼 수 있습니다.

→ cctv 기록 파일 중 사라진 시간대를 알아내기 위한 방법이 무엇인지 이야기 나누어 봅니다.

(예시) 없어진 파일 구간을 확인하기 위해서는 리스트의 요소로 '3, 4, 5…'와 같이 저장된 정수값 중 간격이 일정하지 않은, 즉 그 수의 차가 1 이 아닌 구간을 찾아야 합니다. while 반복문을 사용하여 i 가 총 파일의 개수보다 작을 때 코드를 반복합니다. 2 차원 리스트중 내부 리스트 요소 값의 차를 알아냅니다. 선행하는 인덱스와 후행하는 인덱스를 각각 비교하여 그 값의 차가 1 이 아니라면 출력하도록 합니다.

→ 사라진 파일 구간의 구체적인 날짜와 시간대를 알아내려면 어떻게 해야 하는지 이야기 나누어 봅니다.

(예시) 앞에서 찾아낸, 정수값의 간격이 일정하지 않은 구간을 검색하는 코드를 수정하여 해당 리스트 인덱스의 두 번째 값을 출력하도록 합니다.

→ 실행결과 창에 출력된 값을 보고 유추한 정답을 입력하여 확인하도록 합니다.

(예시) 리스트에서 사라진 파일 구간 인덱스에 해당하는 날짜 및 시간 값을 출력하면 **012723 012900** 이 출력됩니다. 01 월 27 일 23 시부터 01 월 29 일 00 시를 의미하는 날짜 사이에 01 월 28 일의 데이터가 사라진 것임을 추리합니다. 따라서 cctv 기록 파일이 사라진 일자는 0128 이 됩니다. 입력창에 **0128** 을 입력한 후 정답을 확인합니다.

→ 스스로 코딩하도록 유도합니다.

→ 미션 2 템플릿으로 이동합니다.

Mission2. 용의자 특정하기.

[미션 목표]

● 미션의 목표를 설명합니다.

1) 스토리 속에서 단서를 확인하여 용의자를 선별할 수 있는 조건을 알아낼 수 있습니다.
2) 2 차원 리스트의 형태를 알고 요소에 접근할 수 있습니다.
3) 포함연산자를 이용하여 데이터 내 특정 값의 포함 여부를 알아낼 수 있습니다.
4) 조건에 따라 원하는 값을 추출하여 리스트의 마지막에 요소를 추가할 수 있습니다.

[스토리 이해]

● 스토리의 내용을 이해합니다.

- 이전 미션에서 나온 프로그래밍 결과를 생각해보며 스토리를 확인합니다.

< 단서 >

(시청에서 보내온 CCTV 파일은 조작이 되었음이 증명되었다.)

"파이", "파이, 네 말대로 시청에서 보내온 파일이 조작되었어. "
"이선", "그래 맞아, 내가 확인해 보니 28 일에 기록된 파일이 삭제되었더라고."
"파이", " 아무래도 찜찜해서 다른 시청 담당자에게 시청 로비의 출입 명단을 달라고 요청했어."
"이선", "잘했어! 사라진 시간대에 해당하는 출입 명단을 우선 확인해 보는게 좋겠어."
"파이", "응! 마침 도착했어!"
"이선", "바로 보내줘!"

<<< 단서 확인하기 >>>

<단서>
시청에 출입한 총 인원을 찾아라.

→ cctv 기록 파일이 사라진 시간대에 시청 로비에 출입한 명단을 확인하려는 이유를 생각해 봅니다.

(예시) cctv 기록 파일의 특정 구간이 지워진 것은 그 시간동안 폭발물을 설치하는 범행이 이루어졌고, 이를 숨기기 위해 cctv 파일을 삭제했을 것으로 추리가 가능합니다. 시청 로비의 입구를 통과하지 않으면 20 층 기밀 자료실까지 접근할 수 없었을 것이므로 미션 1 에서 알아낸 사라진 특정 시간대 동안 시청 로비에 출입한 명단 리스트를 조회하는 것입니다.

[학습 자료]

- 학습하게 될 개념에 대해 해당 코드를 구현하면서 적절하게 설명합니다.

 # 2 차원 리스트

- 리스트 안에 리스트가 들어 있는 형태
- 선언 : 변수 =[[‘요소 1’,…‘요소 n’][‘요소 1’,…‘요소 n’]…]

  ex) double_list = [[‘a’, ‘b’], [‘c’, ‘d’], [‘e’, ‘f’]]
- 인덱스
  - 리스트 요소의 순서로 0 부터 시작

  ex) list_test = [[10, 20], [30, 40], [50, 60]]

내부 인덱스	[0, 1]	[0, 1]	[0, 1]
list_test 인덱스	0	1	2

- 요소 접근 방법 : 리스트[리스트 인덱스][내부 리스트 인덱스]

  ex) print(list_test[1])

  print(list_test[0][2])

 # 포함 연산자

- 요소의 존재 여부를 판단하는 연산자

x in y	x 가 y 내에 존재한다
x not in y	x 가 y 내에 존재하지 않는다

- if ‘a’ in [‘a’, ‘b,’ ‘c’]:

  print(‘a exists.’)

[코드 이해]

● 사용할 API 에 대해 설명합니다.

〈 API 〉

1) range(값) : 연속된 정수를 만드는 함수
2) append('요소') : 리스트 마지막에 요소 추가
3) len(리스트) : 리스트 요소의 개수 반환 함수
4) print(값) : 값을 출력하는 함수

● 학생이 스스로 코드를 구현하여 정답을 찾아 확인하도록 합니다.

- 스토리 속에서 해결해야 할 미션에 대해 질문하며 해당 코드를 학생 스스로 구현하도록 설명해줍니다. 코드는 제공하지 않습니다.

〈 CODE 〉

```
#data : 모범답안에서 확인
#해당 시간대에 시청에 출입한 인원 확인하기
print(len(entrance_list))
suspects_list=[]

for i in range(len(entrance_list)):
    if entrance_list[i] not in suspects_list:
        suspects_list.append(entrance_list[i])
print(len(suspects_list))

#정답 확인 :
27
```

→ 미션 1 에서 알아낸 특정 시간동안 시청에 출입한 사람들의 수를 알아내기 위해서는 어떻게 해야 하는지 이야기 나누어 봅니다.

(예시) for 반복문을 통해 1 월 28 일에 시청에 출입한 entrance_list 의 요소를 비교하여 검색합니다. 중복된 명단을 제거하여 suspects_list 에 용의자 목록을 저장합니다.

→ 실행결과 창에 출력된 값을 정답창에 입력하여 확인하도록 합니다.

(예시) 조건에 맞는 값을 suspects_list 에 저장하여 출력한 후 리스트의 길이를 출력합니다. 결과값을 그대로 입력하여 정답을 확인하도록 합니다.

→ 스스로 코딩하도록 유도합니다.

→ 미션 3 템플릿으로 이동합니다.

Mission3. id 번호를 해독하여 범인 찾기.

- 미션의 목표를 설명합니다.

[미션 목표]

1) 2 차원 리스트의 형태를 알고 각 리스트의 값에 접근하여 요소를 추출할 수 있습니다.
2) 큰 수와 문자를 간결하게 나타내는 16 진수 표기법에 대해 알고, hex code 수로 표현된 값을 디코딩하여 문자열로 변환할 수 있습니다.

[스토리 이해]

● 스토리의 내용을 이해합니다.

- 이전 미션에서 나온 프로그래밍 결과를 생각해보며 스토리를 확인합니다.

< 단서 >

"파이", "28 일에 출입한 명단이 총 27 명이야. "
"이선", "이 명단 안에 폭탄을 설치한 범인이 있다는 거네. "
"파이", "그렇지. 흠, 시청 직원들의 ID 번호가 좀 특이한걸. "
"이선", "뭐가 특이한데?"
"파이", "16 진법 방식이야. 컴퓨터 분야에서 문자를 숫자로 표현할 때 사용해."
"파이", "뭘 나타낸 거지?"
"이선", "잠깐만, 시청 담당자에게 물어볼게. "

(이선은 바로 시청 담당자와 통화한다.)

"이선", "확인해 봤어. 시청 직원의 ID 번호는 부서 이름을 기준으로 만들어진대."
"이선", "잘됐다. 일일이 소속 부서를 확인할 필요가 없겠어."
"파이", "거기에 하나 더."
"파이", "범인이 위조 ID 를 사용했다면 내부 구조를 잘 모를 가능성이 커."
"파이", "ID 번호가 부서 이름을 나타내는 것도 모르고, 아무 숫자나 썼을 거야."
"이선", "그러면, 범인의 ID 번호는 문자로 바꾸어도 부서 이름이 나오지 않겠네…!"
"파이", "ID 번호를 복호화해볼게. 단서가 나올지도 몰라."
"이선", "응, 부탁해!"

<<< 단서 확인하기 >>>

〈단서〉

출입 명단에서 수상한 사람을 찾아라.

→ 용의자를 특정하기 위해 조회할 시청 직원 id 를 해독하려면 어떻게 해야 할지 이야기 나누어 봅니다.

(예시) 시청 직원 id 리스트는 hex code 번호로 되어 있어 원래의 문자로 해독이 필요합니다. 이러한 과정을 복호화(decode)한다 합니다.

[학습 자료]

- 학습하게 될 개념에 대해 해당 코드를 구현하면서 적절하게 설명합니다.

#16 진수란?

- 큰 수와 문자를 간결하게 나타내는 16 진수 표기법
- 10 진수 정수 10 개 + 추가 기호 6 개를 이용하여 문자 표현

문자	16 진수
A	41
B	42
⋮	
Y	59
Z	5A
a	61
b	62

문자열	r	a	i	n	b	o	w
16 진수	72	61	69	6e	62	6f	77

- 16 진수 → 문자열 변환

[코드 이해]

```
bytes.fromhex(변환할값).decode()


ex)
word = '48 69 20 74 68 65 72 65'
print(bytes.fromhex(word).decode())
```

● 사용할 API 에 대해 설명합니다.

〈 API 〉

1) bytes.fromhex(값).decode() : 16 진수 바이트 코드를 문자열로 변환
2) len(리스트) : 리스트 요소의 개수 반환 함수
3) print(값) : 값을 출력하는 함수

● 학생이 스스로 코드를 구현하여 정답을 찾아 확인하도록 합니다.

- 스토리 속에서 해결해야 할 미션에 대해 질문하며 해당 코드를 학생 스스로 구현하도록 설명해줍니다. 코드는 제공하지 않습니다.

〈 CODE 〉

#시청 직원 ID 확인하기

```
for i in range(len(suspects_list)):
    print(bytes.fromhex(suspects_list[i][1]).decode())

 #결과 확인 :

management support team
technical support team
section of personnel
management department
management support team
section of personnel
administration department
administration department
finance accounting team
administration department
administration department
administration department
administration department
administration department
administration department
technical support team
administration department
administration department
administration department
administration department
administration department
administration department
```

administration department
administration department
administration department
administration department
wkkps://builderbug.dem/main/1_40yADtib8

#범인 확인하기
print(suspects_list[-1][0])

#정답 확인 : jayden

→ hex code 수로 표현된 리스트의 시청 직원 id 를 어떻게 해독할지 이야기 나누어 봅니다.

(예시) hex code 수로 표현된 값을 원래의 문자로 되돌리는 decode 과정이 필요합니다. 파이썬은 바이트 코드를 문자열로 변환하는 bytes.decode() 함수를 지원합니다. 16 진수 hex code 로 표현된 바이트 값을 변환할 것이므로 fromhex()를 명시하여 변환할 값의 변수명을 지정해줍니다. 이를 print 함수로 출력하면 원래의 문자열 내용이 결과로 출력됩니다.

→ 출력된 ID 결과를 보고 누가 범인일지 수상한 점에 대해 이야기 나누어 봅니다.

(예시) 마지막 인물이 수상하므로 범인입니다. 다른 ID 의 경우 해독한 결과가 team, department 등 부서명으로 되어 있지만 마지막 사람의 경우만 알 수 없는 URL 이 출력됩니다.

→ ID 를 복호화하여 출력한 결과가 수상한 인물을 범인으로 지목하고, 그 사람의 이름을 출력하려면 어떻게 할 수 있을지 이야기 나누어 봅니다.

(예시) 수상한 인물은 리스트의 마지막 순서에 위치하고 있으므로 suspects_list 의 마지막 인덱스에 접근하여 이름에 해당하는 요소 값을 출력하는 코드를 작성하여 확인합니다.

→ 결과 창에 출력된 값을 입력 버튼을 누르고 정답을 확인하도록 합니다.

(예시) 해당 시간에 출입한 수상한 시청 직원의 이름은 jayden 입니다. 정답을 확인합니다.

→ 스스로 코딩하도록 유도합니다.

→ 마무리 템플릿으로 이동합니다.

Summary.

● 이번 차시의 미션을 해결하면서 배웠던 프로그래밍 개념과 파이썬 API 를 복습합니다.

 # 2 차원 리스트

- 리스트 안에 리스트가 들어 있는 형태
- 선언 : 변수 =[[‘요소 1’,…‘요소 n’][‘요소 1’,…‘요소 n’]…]

  ex) double_list = [[‘a’, ‘b’], [‘c’, ‘d’], [‘e’, ‘f’]]
- 인덱스
  - 리스트 요소의 순서로 0 부터 시작

    ex) list_test = [[10, 20], [30, 40], [50, 60]]

내부 인덱스	[0, 1]	[0, 1]	[0, 1]
list_test 인덱스	0	1	2

- 요소 접근 방법 : 리스트[리스트 인덱스][내부 리스트 인덱스]

ex) print(list_test[1])
print(list_test[0][2])

 # 반복문

- 같은 일을 반복할 때 사용하는 제어문
- 구조 : while_break
  - while 조건이 참인 동안 반복 수행
  - 반복을 중지할 때 break 사용

```
while 조건:
  수행할 문장
  if 반복을 중지할 조건:
    break

ex)
count = 5
while count > 0:
  print(count)
  count = count - 1
  if count == 1:
    break
```

 # 비교 연산자

- 숫자, 문자열 등 두 항목을 서로 비교하는 연산자

x == y	x 와 y 가 같다
x != y	x 와 y 는 같지 않다

- 결과는 True 또는 False 반환

  ex) 1 != 'one'

  'Sun' == 'Sun'

 # 포함 연산자

- 요소의 존재 여부를 판단하는 연산자

x in y	x 가 y 내에 존재한다
x not in y	x 가 y 내에 존재하지 않는다

- if 'a' in ['a', 'b,' 'c']:

  print('a exists.')

- 결과는 True 또는 False 반환

  ex) 1 != 'one'

  'Sun' == 'Sun'

 # 16 진수란?

- Hexadecimal number, hex number
- 큰 수와 문자를 간결하게 나타내는 16 진수 표기법
- 10 진수 정수 10 개 + 추가 기호 6 개를 이용하여 문자 표현

문자	16 진수
A	41
B	42
⋮	
Y	59
Z	5A

a	61
b	62

문자열	r	a	i	n	b	o	w
16 진수	72	61	69	6e	62	6f	77

- 16 진수 → 문자열 변환

```
bytes.fromhex(변환할값).decode()


ex)
word = '48 69 20 74 68 65 72 65'
print(bytes.fromhex(word).decode())
```

더 나아가기

- 학습한 코드를 응용하도록 추가 질문을 합니다. 기존 코드에서 수정 또는 처음부터 다시 구현해 보도록 합니다.

〈 Mission 1 〉

→ 일주일 치의 cctv 기록이 저장되어있는 리스트에서 각 요소 값의 간격을 비교하여 일정하지 않은 인덱스의 일자와 시간을 나타내는 요소 값을 출력하였습니다. 만약 cctv 기록 조작이 의심되는 일자가 1 월 27 일이고, 해당 날짜의 cctv 기록 파일 번호를 알아내야 했다면 어떻게 할 수 있을까요?
(예시) 리스트 인덱스로 접근하여 두 번째 요소값으로 '0127'을 포함하는 요소가 있을 경우 해당 인덱스의 첫 번째 요소값을 출력하여 알아냅니다.

〈 Mission 2 〉

→ 리스트 entrance_list 의 내부 요소 첫 번째 항목에 접근하여 나타내고 있는 시간대를 기준으로 몇 명의 직원 이름을 출력했습니다. 제보에 따라 'sophia' 이름을 가진 직원의 ID 내용을 모두 추출해야 한다면 어떻게 할 수 있을까요?
(예시) 내부 리스트의 두 번째 요소가 이름이므로 해당 인덱스에 접근하여 원하는 값을 추출할 수 있습니다. entrance_list 의 이름에 해당하는 요소 인덱스에 접근하여 값이 'sophia'와 일치하는 인덱스를 검색하여 해당 요소값을 출력합니다.

〈 Mission 3 〉

→ 16 진수 hex code 로 표현된 시청 직원 ID 값을 복호화하여 부서명을 알아낸 다음, technical support team 에 소속된 직원의 행적이 의심되어 해당 팀에 소속된 직원들의 이름을 알아보려면 어떻게 할 수 있을까요?
(예시) suspects_list 의 16 진수 hex code 로 암호화된 시청 직원 ID 정보를 검사하여 technical support team 인 인덱스의 이름에 해당하는 요소를 추출합니다.

평가 기준

평가 내용	1~5	강사 메모
학습		
사건 해결에 필요한 데이터를 스토리 속에서 파악할 수 있다.		
while 반복 제어문을 사용하여 조건에 따라 반복을 중지하는 코드를 작성할 수 있다.		
비교 연산자에 대해 알고 적절하게 사용할 수 있다.		
2 차원 리스트의 형태를 알고 각 리스트의 값에 접근할 수 있다.		
16 진수 표기법을 이해하고 문자열로 변환하는 코드를 작성할 수 있다.		

주요 함수를 사용하여 프로그래밍하고 결과 값을 찾아 미션을 해결할 수 있다.		
태도		
어려운 점이 있어도 포기하지 않고 끝까지 해결하려고 노력하였다.		

모범 답안

코드	
Mission1	**#data** file_list = [[1,'012500'], [2,'012501'], [3,'012502'], [4,'012503'], [5,'012504'], [6,'012505'], [7,'012506'], [8,'012507'], [9,'012508'], [10,'012509'], [11,'012510'], [12,'012511'], [13,'012512'], [14,'012513'], [15,'012514'], [16,'012515'], [17,'012516'], [18,'012517'], [19,'012518'], [20,'012519'], [21,'012520'], [22,'012521'], [23,'012522'], [24,'012523'], [25,'012600'], [26,'012601'], [27,'012602'], [28,'012603'], [29,'012604'], [30,'012605'], [31,'012606'], [32,'012607'], [33,'012608'], [34,'012609'], [35,'012610'], [36,'012611'], [37,'012612'], [38,'012613'], [39,'012614'], [40,'012615'], [41,'012616'], [42,'012617'], [43,'012618'], [44,'012619'], [45,'012620'], [46,'012621'], [47,'012622'], [48,'012623'], [49,'012700'], [50,'012701'], [51,'012702'], [52,'012703'], [53,'012704'], [54,'012705'], [55,'012706'], [56,'012707'], [57,'012708'], [58,'012709'], [59,'012710'], [60,'012711'], [61,'012712'], [62,'012713'], [63,'012714'], [64,'012715'], [65,'012716'], [66,'012717'], [67,'012718'], [68,'012719'], [69,'012720'], [70,'012721'], [71,'012722'], [72,'012723'], [97,'012900'], [98,'012901'],[99,'012902'],[100,'012903'],[101,'012904'], [102,'012905'], [103,'012906'], [104,'012907'], [105,'012908'], [106,'012909'], [107,'012910'], [108,'012911'], [109,'012912'], [110,'012913'], [111,'012914'], [112,'012915'], [113,'012916'], [114,'012917'], [115,'012918'], [116,'012919'], [117,'012920'], [118,'012921'], [119,'012922'], [120,'012923'], [121,'013000'], [122,'013001'], [123,'013002'], [124,'013003'], [125,'013004'], [126,'013005'], [127,'013006'],

[128,'013007'], [129,'013008'], [130,'013009'], [131,'013010'], [132,'013011'], [133,'013012'], [134,'013013'], [135,'013014'], [136,'013015'], [137,'013016'], [138,'013017'], [139,'013018'], [140,'013019'], [141,'013020'], [142,'013021'], [143,'013022'], [144,'013023'], [145,'013100'], [146,'013101'], [147,'013102'], [148,'013103'], [149,'013104'], [150,'013105'], [151,'013106'], [152,'013107'], [153,'013108'], [154,'013109'], [155,'013110'], [156,'013111'], [157,'013112'], [158,'013113'], [159,'013114'], [160,'013115'], [161,'013116'], [162,'013117'], [163,'013118'], [164,'013119'], [165,'013120'], [166,'013121'], [167,'013122'], [168,'013123']]

#일주일치의 cctv 파일,하루에 24 개의 파일(7*24=168 개)

```
print(len(file_list))
```

#정답 확인:

144

#없어진 파일 구간 확인하기

```
i=0
while i<len(file_list):
    if file_list[i+1][0] - file_list[i][0] != 1:
        print(file_list[i][0],file_list[i+1][0])
        break
    i=i+1
```

#결과 확인:

72 97

#시간대 확인하기

```
i=0
while i<len(file_list):
    if file_list[i+1][0] - file_list[i][0] != 1:
        print(file_list[i][1], file_list[i+1][1])
        break
```

	i=i+1 #결과 확인: 012723 012900 #정답 확인: 0128
Mission2	**#data** **#이름, id 명단의 형태** entrance_list = [['sophia', "6d 61 6e 61 67 65 6d 65 6e 74 20 73 75 70 70 6f 72 74 20 74 65 61 6d"],['noah', "74 65 63 68 6e 69 63 61 6c 20 73 75 70 70 6f 72 74 20 74 65 61 6d"],['jacob', "73 65 63 74 69 6f 6e 20 6f 66 20 70 65 72 73 6f 6e 6e 65 6c"],['oliva', 'T 6d 65 6e 74 20 64 65 70 61 72 74 6d 65 6e 74'],['emily', "6d 61 6e 61 67 65 6d 65 6e 74 20 73 75 70 70 6f 72 74 20 74 65 61 6d"],['isabella', "73 65 63 74 69 6f 6e 20 6f 66 20 70 65 72 73 6f 6e 6e 65 6c"],['ethan', "61 64 6d 69 6e 69 73 74 72 61 74 69 6f 6e 20 64 65 70 61 72 74 6d 65 6e 74"],['owen', "61 64 6d 69 6e 69 73 74 72 61 74 69 6f 6e 20 64 65 70 61 72 74 6d 6t5 6e 74"],['jack', "66 69 6e 61 6e 63 65 20 61 63 63 6f 75 6e 74 69 6e 67 20 74 65 61 6d"],['connor', "61 64 6d 69 6e 69 73 74 72 61 74 69 6f 6e 20 64 65 70 61 72 74 6d 65 6e 74"], ['ethan', "61 64 6d 69 6e 69 73 74 72 61 74 69 6f 6e 20 64 65 70 61 72 74 6d 65 6e 74"],['david', "61 64 6d 69 6e 69 73 74 72 61 74 69 6f 6e 20 64 65 70 61 72 74 6d 65 6e 74"],['levi', "61 64 6d 69 6e 69 73 74 72 61 74 69 6f 6e 20 64 65 70 61 72 74 6d 65 6e 74"], ['julian', "61 64 6d 69 6e 69 73 74 72 61 74 69 6f 6e 20 64 65 70 61 72 74 6d 65 6e 74"],['daniel', "61 64 6d 69 6e 69 73 74 72 61 74 69 6f 6e 20 64 65 70 61 72 74 6d 65 6e 74"], ['samuel', "61 64 6d 69 6e 69 73 74 72 61 74 69 6f 6e 20 64 65 70 61 72 74 6d 65 6e 74"], ['henry', "74 65 63 68 6e 69 63 61 6c 20 73 75 70 70 6f 72 74 20 74 65 61 6d"],['jack', "66 69 6e 61 6e 63 65 20 61 63 63 6f 75 6e 74 69 6e 67 20 74 65 61 6d"], ['lsaac', "61 64 6d 69 6e 69 73 74 72 61 74 69 6f 6e 20 64 65 70 61 72 74 6d 65 6e 74"], ['jacob', "73 65 63 74 69 6f 6e 20 6f 66 20 70 65 72 73 6f 6e 6e 65 6c"],['aiden', "61 64 6d 69 6e 69 73 74 72 61 74 69 6f 6e 20 64 65 70 61 72 74 6d 65 6e 74"],['daniel', "61 64 6d 69 6e 69 73 74 72 61 74 69 6f 6e 20 64 65 70 61 72 74 6d 65 6e 74"], ['james', "61 64 6d 69 6e 69 73 74 72 61 74 69 6f 6e 20 64 65 70 61 72 74 6d 65 6e 74"], ['mason', "61 64 6d 69 6e 69 73 74 72 61 74 69 6f 6e 20 64 65 70 61 72 74 6d 65 6e 74"],['joseph', "61 64 6d 69 6e 69 73 74 72 61 74 69 6f 6e 20 64 65 70 61 72 74 6d 65 6e 74"], ['julian', "61 64 6d 69 6e 69 73 74 72 61 74 69 6f 6e 20 64 65 70 61 72 74 6d 65 6e 74"],['jack', "66 69 6e 61 6e 63 65 20 61 63 63 6f 75 6e 74 69 6e 67 20 74 65 61 6d"], ['adam', "61 64 6d 69 6e 69 73 74 72 61 74 69 6f 6e 20 64 65 70 61 72 74 6d 65 6e 74"], ['ezra', "61 64 6d 69 6e 69 73 74 72 61 74 69 6f 6e 20 64 65 70 61 72 74 6d 65 6e 74"], ['sophia', "6d 61 6e 61 67 65 6d 65 6e 74 20 73 75 70 70 6f 72 74 20 74 65 61 6d"], ['owen', "61 64 6d 69 6e 69 73 74 72 61 74 69 6f 6e 20 64 65 70 61 72 74 6d 65 6e 74"], ['aiden', "61 64 6d 69 6e 69 73 74 72 61 74 69 6f 6e 20 64 65 70 61 72 74 6d 65 6e 74"],['hunter', "61 64 6d 69 6e 69 73 74 72 61 74 69 6f 6e 20 64 65 70 61 72 74 6d 65 6e 74"],['luke', "61 64 6d 69 6e 69 73 74 72 61 74 69 6f 6e 20 64 65 70 61 72 74 6d 65 6e 74"], ['ezra', "61 64 6d 69 6e 69 73 74 72 61 74 69 6f 6e 20 64 65 70 61 72 74 6d 65 6e 74"],['connor', "61 64 6d 69 6e 69 73 74 72 61 74 69 6f 6e 20 64 65 70 61 72 74 6d 65 6e 74"], ['amy', "61 64 6d 69 6e 69 73 74 72 61 74 69 6f 6e

	20 64 65 70 61 72 74 6d 65 6e 74"], ['hunter', "61 64 6d 69 6e 69 73 74 72 61 74 69 6f 6e 20 64 65 70 61 72 74 6d 65 6e 74"], ['noah', "74 65 63 68 6e 69 63 61 6c 20 73 75 70 70 6f 72 74 20 74 65 61 6d"], ['emily', "6d 61 6e 61 67 65 6d 65 6e 74 20 73 75 70 70 6f 72 74 20 74 65 61 6d"], ['jayden', "77 6b 6b 70 73 3a 2f 2f 62 75 69 6c 64 65 72 62 75 67 2e 64 65 6d 2f 6d 61 69 6e 2f 31 5f 34 30 79 41 44 74 69 62 38"]]
	
	#해당 시간대에 시청에 출입한 명단 확인하기
	print(len(entrance_list))
	suspects_list=[]
	
	for i in range(len(entrance_list)):
	  if entrance_list[i] not in suspects_list:
	  suspects_list.append(entrance_list[i])
	print(len(suspects_list))
	
	#정답 확인 :
	27

Mission3	**#시청 직원 ID 확인하기** for i in range(len(suspects_list)): print(bytes.fromhex(suspects_list[i][1]).decode()) #결과 확인 : management support team technical support team section of personnel management department management support team section of personnel administration department administration department finance accounting team administration department administration department administration department administration department administration department administration department technical support team administration department administration department administration department administration department administration department

	administration department administration department administration department administration department administration department wkkps://builderbug.dem/main/1_40yADtib8 **#범인 확인하기** print(suspects_list[-1][0]) #정답 확인 : jayden

더 나아가기
Mission1
	i = 0
	while i< len(file_list):
	  i = i + 1
	  if '0127' in file_list[i][1]:
	  print(file_list[i][0])
	  break
	
	#정답 확인 : 51

더 나아가기
Mission2
	people_list = []
	for i in range(len(entrance_list)):
	  if entrance_list[i][0] == 'sophia':
	  print(entrance_list[i])
	
	#정답 확인 :
	['sophia', '6d 61 6e 61 67 65 6d 65 6e 74 20 73 75 70 70 6f 72 74 20 74 65 61 6d']
	['sophia', '6d 61 6e 61 67 65 6d 65 6e 74 20 73 75 70 70 6f 72 74 20 74 65 61 6d']

더 나아가기
Mission3
	for i in range(len(suspects_list)):
	  if bytes.fromhex(suspects_list[i][1]).decode()== 'technical support team':
	  print(suspects_list[i][0])
	
	#정답 확인 :
	noah

	henry