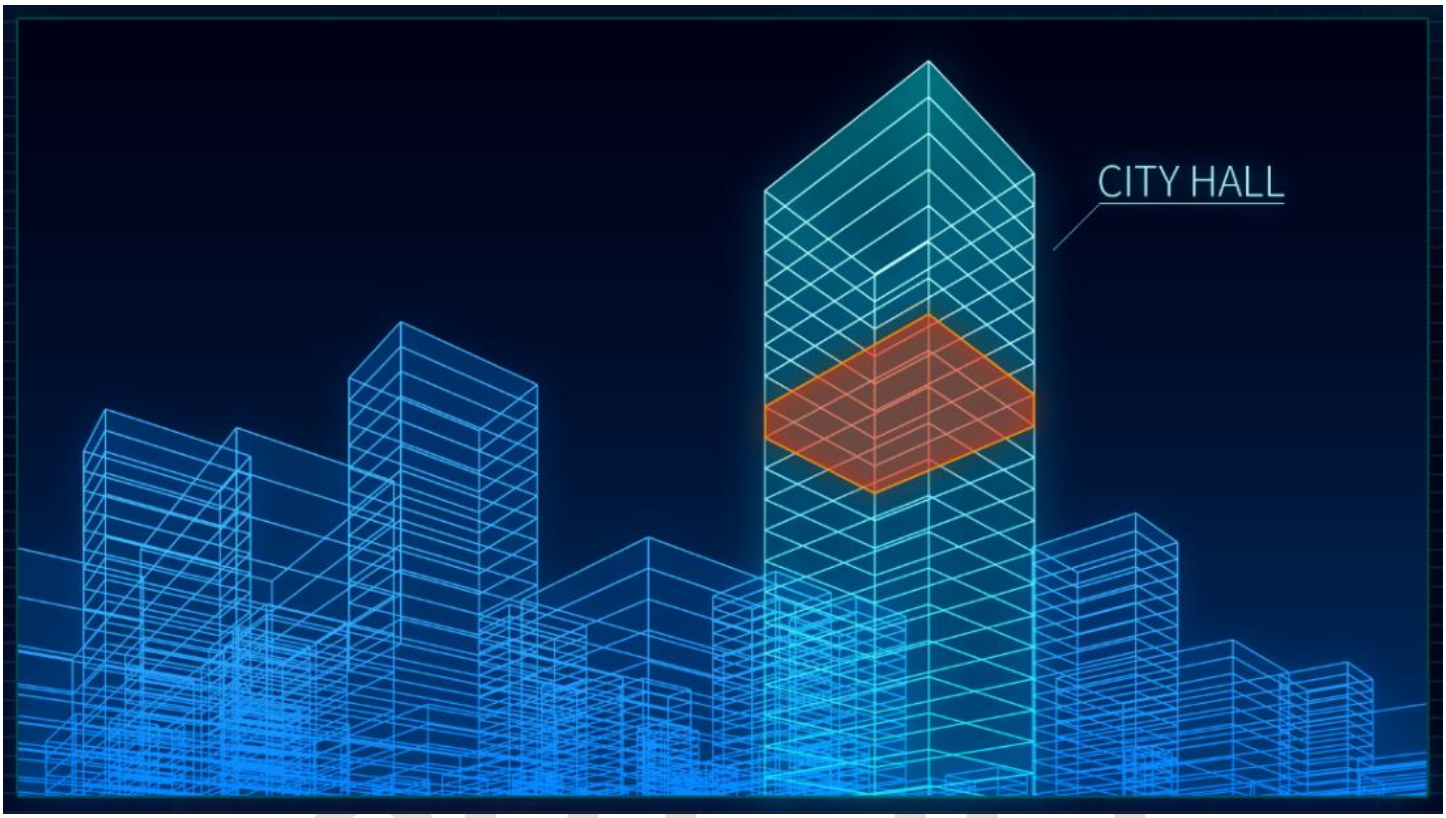
[헤이그 도시 사수 작전]



빌더버그 조직이 시청을 시작으로 헤이그 도시를 파괴하려고 계획하고 있다는 정보를 알아낸
이선과 파이는 우선 시청에 있는 사람들을 안전한 곳으로 대피시킨다.
시청 건물 어딘가에 설치된 폭발물을 찾아 헤이그 도시를 사수할 수 있을지….

오늘의 목표	파괴 위기에 처한 헤이그 도시 사수하기
주요 함수	print, range, append

➤ 학습 목표

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

WizLive

→ 반복문의 for 사용법에 대해 알고 반복적인 일을 효율적으로 처리할 수 있습니다.
→ 조건문의 if 사용법에 대해 알고 참과 거짓을 판단하는 프로그램을 만들 수 있습니다.
→ 항목을 서로 비교하는 연산자와 참, 거짓을 판단하는 논리연산자에 대해 알고 적절하게 사용할 수 있습니다.
→ 리스트에 요소를 추가할 수 있습니다.
→ 주요 함수를 배우고 활용하며 데이터를 분석하여 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 스토리 이해]

● 이전 차시 스토리에 관해 이야기를 나눕니다.

<이전 스토리>

이선은 남은 파일 1 개에서 장소에 대한 힌트가 있을 것이라 예상했다. 그리고 데이터분석 전문가들에게 파일을 보내 단서를 확보할 수 있었다. 파이는 재빠르게 단서를 확인하고 프로그래밍을 해서 빌더버그 조직이 헤이그 도시 파괴 계획 장소의 첫 번째 타깃인 city hall 을 알아냈다.

→ 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. 시청 건물의 용도 확인하여 타깃 층 찾기.

● 미션의 목표를 설명합니다

[미션 목표]

1) 스토리 속에서 시청 건물의 용도 종류를 파악할 수 있습니다.
2) 반복 제어문을 사용하여 같은 일을 반복하는 프로그램을 만들 수 있습니다.
3) 조건 제어문을 사용하여 참과 거짓 조건에 따라 어떤 일을 수행하는 프로그램을 만들 수 있습니다.

[스토리 이해]

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

< 단서 >

(빌더버그 조직의 타깃은 시청으로 좁혀졌다.)
(우선 시청에 있는 사람들을 안전한 곳으로 대피시켰다.)

"이선", "빌더버그 조직은 왜 시청을 타깃으로 잡았을까?"
"파이", "헤이그 도시에 관한 중요한 정보를 먼저 확보하려고 하는 것 같아."
"이선", "시청에 무슨 자료가 있는데?"
"파이", "도시의 시스템을 제어하는 장치에 관한 자료가 있을 거야."
"이선", "시청은 40 층으로 이루어져 있어. 각 층은 다른 용도로 사용되고 있다던데."
"파이", "응, 맞아. 시청 건물은 층마다 일반자료실, 기밀자료실, 회의실, 사무실로 사용하고 있어."
"파이", "내가 시청 건물의 각 층 용도가 적힌 리스트를 확인해볼게."
"이선", "그래 파이, 빌더버그 조직은 기밀자료실을 타깃으로 생각할 거야."

<<< 단서 확인하기 >>>

<단서>
기밀자료실이 있는 층을 모두 찾아라.

→ 파이와 이선의 현재 상황에 관해 이야기를 나누어 봅니다.
(예시) 빌더버그 조직의 헤이그 도시 파괴 계획의 첫 번째 타깃인 시청에 설치된 폭발물을 찾으려고 합니다. 시청 건물은 총 40 층으로 각 층마다

용도가 다르게 일반자료실, 기밀자료실, 회의실, 사무실 4 가지로 구분하여 사용하고 있습니다. 빌더버그 조직은 전체 층 중에 기밀자료실이 있는 층을 타깃으로 생각할 것이라 예상하고 있습니다.

→ 기밀자료실로 사용하는 층을 어떻게 찾을 수 있을지 질문합니다.

(예시) 시청 건물의 각 층 용도가 적힌 목록을 확인하여 프로그래밍한 후 기밀자료실이 보관되어 있는 층을 분류합니다.

[학습 자료]

● 학습하게 될 개념에 대해 해당 코드를 구현하면서 적절하게 설명합니다.

 # 반복문이란?

- 같은 일을 반복할 때 사용하는 제어문
- 순서가 있는 자료형에서만 사용
- 구조 : for __ in

```
for 반복 변수 in 반복할 조건:
    수행할 문장

ex) numbers = [1, 2, 3]
    for v in numbers:
        print(v)
```

- range(값) : 연속된 정수를 만드는 함수
  - range(stop)
  - range(start, stop)
  - range(start, stop, step)

```
ex) for v in range(0, 3):
        print(v)
```

 # 조건문이란?

- 조건의 참과 거짓을 판단하는 제어문
- 구조 : if

```
if 조건:
  수행할 문장

ex) rain = True
    if rain:
        print("it's raining")
```

 # 비교 연산자

- 숫자, 문자열 등 항목을 서로 비교
  - == : 두 항목이 같은지 비교
  - != : 두 항목이 다른지 비교

x == y	x 와 y 가 같은가
x != y	x 와 y 는 같지 않은가

- 결과는 True 또는 False 반환

ex) 2 == 2
    1 == 'one'
    'abc' != 'bc'

 # 리스트

- 순서가 있는 요소들의 집합
- 대괄호([ ]) 안에 요소들을 ',' 로 구분하여 저장
- 선언 : 변수 = ['요소 1', '요소 2', …, '요소 n']
- 인덱스
  - 리스트 내 요소의 순서로 0 부터 시작

    ex) list_test = [1, 2, 3, 4, 5]
        list_test[2]

list_test	1	2	3	4	5
인덱스	0	1	2	3	4

● 요소 추가 가능

- 변수.append('요소') : 리스트 마지막에 요소 추가

ex) list_new = ['a', 'b', 'c']
list_new.append('d')

→ 미션 1 에 있는 학습 개념들은 미션 2 에서 동일하게 배우기 때문에 학생의 수준에 맞게 적절하게 분배하여 설명하도록 합니다.

● 사용할 API 에 대해 설명합니다.

[코드 이해]

< API >

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

→ range 함수는 () 안에 값을 넣으면 그 값보다 1 작은 연속된 정수를 만들어 줍니다.

→ append 함수는 () 안에 넣은 요소를 해당 리스트 마지막에 추가해 줍니다.

● 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
purpose_list = ['general', 'office', 'office', 'general', 'office', 'general',
```

'office', 'office', 'office', 'secret', 'conference', 'conference', 'office', 'general', 'office', 'general', 'secret', 'office', 'general', 'secret', 'general', 'office', 'office', 'general', 'office', 'conference', 'office', 'office', 'office', 'conference', 'conference', 'conference', 'office', 'general', 'secret', 'general', 'office', 'secret', 'general', 'office']

#기밀자료실(secret)이 있는 층 찾기.

```
secret_list=[]
for i in range(40):
    if purpose_list[i] == 'secret':
        secret_list.append(i+1)
print(secret_list)
```

#정답 확인:

[10, 17, 20, 35, 38]

→ 시청 건물의 각각 용도가 적힌 목록에서 무엇을 해야할지 확인하도록 합니다.

(예시) 시청 건물은 4 가지의 용도로 사용한다고 했고 4 가지 중 빌더버그 조직이 폭발물을 설치했을 것 같은 층을 기밀자료실이라고 예상했으니 전체 층의 용도를 확인하여 그 중 기밀자료실을 저장하고 있는 층을 확인합니다. 이 때 각 층을 모두 확인해야 하기 때문에 40 번을 반복하게 됨으로 range 괄호 안에 40 을 넣어줍니다. range 함수는 괄호 안에 넣은 값의 1 작은 수인 39 까지의 연속된 정수 횟수 만큼 반복하는 것이나 인덱스는 0 부터 시작함으로 인덱스 0 부터 39 까지의 purpose_list 의 모든 요소를 확인할 수 있는 것입니다. 그리고 요소를 하나씩 확인하면서 그 층이 기밀자료실 즉 'secret' 이라는 조건을 만족하면 새로운 리스트(secret_list)에 해당 층을 추가해줍니다.

→ 실행결과 창에 출력된 값을 정답창에 입력하여 확인하도록 합니다.

(예시) secret_list 변수에 저장하여 출력한 후 그대로 입력하여 정답을 확인하도록 합니다.

→ 스스로 코딩하도록 유도합니다.

→ 미션 2 템플릿으로 이동합니다.

Mission2. 시청 건물의 센서 확인하여 타깃 층 찾기.

● 미션의 목표를 설명합니다

[미션 목표]

1) 스토리 속에서 단서를 확인하여 타깃 층을 예상할 수 있습니다.
2) 반복 제어문을 사용하여 같은 일을 반복하는 프로그램을 만들 수 있습니다.
3) 조건 제어문을 사용하여 참과 거짓 조건에 따라 어떤 일을 수행하는 프로그램을 만들 수 있습니다.

[스토리 이해]

● 스토리의 내용을 이해합니다.

- 이전 미션에서 나온 프로그래밍 결과를 생각해보며 스토리를 확인합니다.

< 단서 >

"파이", "시청 건물 40 층 중 5 개의 층만 기밀자료실로 사용하는 중이었어."
"이선", "근데 여기 시청 건물에는 위험물을 감지하는 센서가 있다고 하는데."
"이선", "왜 감지를 못했을까?"
"파이", "그래? 센서가 있었다고?"
"이선", "응. 시청 건물의 짝수 층마다 위험물 감지 센서가 있어."
"파이", "그럼 홀수 층에는 센서가 없어?"
"이선", "응! 그렇지만 짝수 층에 있는 센서가 2 층씩 위험물을 감지할 수 있대."
"파이", "그럼 그 당시에 센서에 오류가 있어서 폭발물을 감지하지 못했군."
"이선", "오류가 있거나 빌더버그 조직이 일부러 센서를 망가뜨렸을 수도 있을 것 같아. "
"파이", "시청 감지 센서 제어 시스템을 확인해봐야겠어."
"이선", "알았어. 센서 에러가 있는 층이 어딘지 찾아줘. "

<<< 단서 확인하기 >>>

<단서>

센서가 작동되지 않은 층을 모두 찾아라.

[학습 자료]

→ 센서가 작동되지 않은 층을 찾기 위한 방법에 관해 질문합니다.
(예시) 시청 건물의 센서는 짝수 층마다 존재합니다. 그리고 센서는 2 개의 층에 대해서 위험물을 감지해주는데, 예를 들어 2 층의 감지 센서가 고장이 났다면 1, 2 층의 위험물에 대해 감지하지 못했다는 것입니다. 시청 감지 센서 제어 시스템에서 확보한 데이터를 통해 에러(error)가 있는 층을 찾습니다.

● 학습하게 될 개념에 대해 해당 코드를 구현하면서 적절하게 설명합니다.

 # 미션 1 과 학습하는 개념은 동일합니다.

● 사용할 API 에 대해 설명합니다.

< API >

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

[코드 이해]

→ range 함수는 () 안에 값을 넣으면 그 값보다 1 작은 연속된 정수를 만들어 줍니다.
→ append 함수는 () 안에 넣은 요소를 해당 리스트 마지막에 추가해 줍니다.

● 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
sensor_list = ['error', 'error', '054057', '054324', '054326', '054327',
'error', 'error', '054345', '054352', '054353', '054359', '054404',
'054406', '054411', '054412', '054413', '054414', 'error', 'error',
```

'054415', '054416', 'error', 'error', 'error', 'error', '054421', '054422', 'error', 'error', '054425', '054426', '054427', '054428' , '054429', '054430', '054431', '054432', '054433', '054434']

#위험물 감지 센서가 고장난 층 찾기

```
error_list=[]
for i in range(40):
    if sensor_list[i] == 'error':
        error_list.append(i+1)
print(error_list)
```

#정답 확인 :

[1, 2, 7, 8, 19, 20, 23, 24, 25, 26, 29, 30]

→ 시청 건물에 있는 위험물 감지 센서의 작동 유무를 알아내기 위해 어떻게 확인하도록 합니다.

(예시) 시청 건물의 위험물 감지 센서 제어 시스템에 기록된 데이터에서 정상적인 숫자가 아닌 'error' 라고 기록된 요소를 찾아 내도록 합니다. 이 때 각 층을 모두 확인해야 하기 때문에 40 번을 반복하게 됨으로 range 괄호 안에 40 을 넣어줍니다. range 함수는 미션 1 에서와 같이 error_list 의 모든 요소를 하나씩 확인하게 합니다. 확인하는 층이 센서가 고장난 층 즉 'error' 라고 되어 있는 요소인 조건을 만족하면 새로운 리스트(error_list)에 해당 층을 추가해줍니다.

→ 실행결과 창에 출력된 값을 정답창에 입력하여 확인하도록 합니다.

(예시) error_list 변수에 저장하여 출력한 후 그대로 입력하여 정답을 확인하도록 합니다.

→ 스스로 코딩하도록 유도합니다.

→ 미션 3 템플릿으로 이동합니다.

Mission3. 폭발물이 설치된 층 찾기.

[미션 목표]

● 미션의 목표를 설명합니다

1) 대화 속에서 사건을 해결할 중요한 힌트를 파악할 수 있습니다.
2) 미션을 해결하기 위해 만족하는 조건을 설정할 수 있습니다.
3) 조건의 참과 거짓을 판단하는 논리연산자를 사용할 수 있습니다.

[스토리 이해]

● 스토리의 내용을 이해합니다.

- 이전 미션에서 나온 프로그래밍 결과를 생각해보며 스토리를 확인합니다.

< 단서 >

"이선", "어느 층의 센서가 고장이야?"
"파이", "2, 8, 20, 24, 26, 30 층에 있는 6 개의 센서가 에러야. "
"이선", "아. 그래서 폭발물을 감지하지 못했구나."
"파이", "그렇지, 빌더버그 조직은 아무래도 센서가 작동하지 않는 기밀자료실 층을 노렸을 거야. "
"이선", "맞아. 그게 쉬웠겠지. 그럼 이제 그 층으로 가서 폭발물만 찾으면 되겠어."
"이선", "시간이 없으니 어서 찾아줘. 파이!"
"파이", "응. 아까 찾은 데이터들을 비교해볼게."

<<< 단서 확인하기 >>>

〈단서〉

폭발물이 설치된 층을 찾아라.

→ 빌더버그 조직이 폭발물을 어느 층에 설치했을 것이라 예상했는지 이야기를 나누어 봅니다.
(예시) 빌더버그 조직이 헤이그 도시를 파괴하기 위해 첫 번째 타깃으로 시청을 정한 이유가 헤이그 도시 시스템을 제어할 수 있는 중요한 자료가

[학습 자료]

시청에 있을 것이라고 했습니다. 그래서 그 자료를 얻기 위해 기밀자료실이 있는 층을 타깃으로 정했을 것 같고 위험물을 감지 하는 센서가 오작동 하거나, 그들이 일부러 센서를 망가뜨린 층에 폭발물을 설치했을 것이라고 예상하고 있습니다.

● 학습하게 될 개념에 대해 해당 코드를 구현하면서 적절하게 설명합니다.

 # boolean

- 참 또는 거짓을 나타내는 자료형
- 'True', 'False' 키워드로 표시
  - True : 참을 의미
  - False : 거짓을 의미

 # 논리 연산자

- 조건의 참과 거짓에 따라 사용하는 연산자
- and, or, not

x **and** y	x 와 y 모두 참이어야 참
x **or** y	x 와 y 둘 중 하나만 참이어도 참
not x	x 가 거짓이면 참

ex) money = False
card = True
if money == True or card == True:
print("can buy something")

if money == False and card == False:
print("can nothing")

if not money:

```
print(“no money”)
```

[코드 이해]

● 사용할 API 에 대해 설명합니다.

< API >

1) range(값) : 연속된 정수를 만드는 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 >

```
#기밀자료실 이면서 위험물 감지 센서가 고장난 층 찾기
for i in range(40):
    if purpose_list[i] == 'secret' and sensor_list[i] == 'error':
        print(i+1)
        #print("폭발물이 설치된 층은 ",i+1, " 이다.")

#정답 확인 :
폭발물이 설치된 층은  20  이다.
```

→ 조건 제어문의 if 문에 어떤 조건을 넣어야 하는지 질문을 합니다.
(예시) 빌더버그 조직은 기밀자료실이 있는 층 중에서 위험물 감지 센서가 고장 난 층을 타깃으로 생각했기 때문에 그 두 조건을 만족시킬 수 있도록 ‘and’ 논리 연산자를 사용합니다.

→ range 함수 안의 연속된 정수와 해당 층과는 1 씩 차이가 나기 때문에 마지막에 ‘i+1’ 을 출력하도록 합니다.

→ 결과 창에 출력된 값을 입력 버튼을 누르고 정답을 확인하도록 합니다.
(예시) 시청 건물 전체 층 중에서 빌더버그 조직이 폭발물을 설치한 장소는 20 층 입니다. 정답을 확인합니다.

→ 스스로 코딩하도록 유도합니다.

→ 마무리 템플릿으로 이동합니다.

Summary.

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

 # 반복문이란?

- 같은 일을 반복할 때 사용하는 제어문
- 순서가 있는 자료형에서만 사용
- 구조 : for __ in

```
for 반복 변수 in 반복할 조건:
    수행할 문장


ex) numbers = [1, 2, 3]
```

```
for v in numbers:
    print(v)
```

- range(값) : 연속된 정수를 만드는 함수
  - range(stop)
  - range(start, stop)
  - range(start, stop, step)

ex) for v in range(0, 3):
    print(v)

 # 조건문이란?

- 조건의 참과 거짓을 판단하는 제어문
- 구조 : if

```
if 조건:
  수행할 문장

ex) rain = True
    if rain:
      print("it's raining")
```


 # 비교 연산자

- 숫자, 문자열 등 항목을 서로 비교
  - == : 두 항목이 같은지 비교
  - != : 두 항목이 다른지 비교

x == y	x 와 y 가 같은가
x != y	x 와 y 는 같지 않은가

- 결과는 True 또는 False 반환

  ex) 2 == 2

  1 == 'one'

  'abc' != 'bc'

 # 리스트

- 순서가 있는 요소들의 집합
- 대괄호([ ]) 안에 요소들을 ',' 로 구분하여 저장
- 선언 : 변수 = ['요소 1', '요소 2', …, '요소 n']
- 인덱스
  - 리스트 내 요소의 순서로 0 부터 시작

    ex) list_test = [1, 2, 3, 4, 5]

    list_test[2]

list_test	1	2	3	4	5
인덱스	0	1	2	3	4

- 요소 추가 가능
  - 변수.append('요소') : 리스트 마지막에 요소 추가

    ex) list_new = ['a', 'b', 'c']

    list_new.append('d')

 # boolean

- 참 또는 거짓을 나타내는 자료형
- 'True', 'False' 키워드로 표시
  - True : 참을 의미
  - False : 거짓을 의미

 # 논리 연산자

- 조건의 참과 거짓에 따라 사용하는 연산자
- and, or, not

x **and** y	x 와 y 모두 참이어야 참
x **or** y	x 와 y 둘 중 하나만 참이어도 참
not x	x 가 거짓이면 참

ex) money = False
card = True
if money == True or card == True:
print("can buy something")

if money == False and card == False:
print("can nothing")

if not money:
print("no money")

 # API

- range(값) : 연속된 정수를 만드는 함수
- append('요소') : 리스트 마지막에 요소 추가
- print(값) : 값을 출력하는 함수

더 나아가기

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

〈 Mission 1 〉

→ 시청 건물의 층별 용도가 적힌 리스트에서 보안을 위해 기밀자료실은 'secret'으로 표기되지 않고 어떤 단어로 표기되었는지 알려지지 않았다면, 리스트에서 기밀 자료실이 위치한 층을 어떻게 알아낼 수 있을까요?
(예시) 비교 연산자 != 를 이용하여 용도가 'general', 'office', 'conference'가 아닌 층을 찾습니다.

〈 Mission 2 〉

→ 위험물 감지 센서는 시청 건물 짝수 층마다 설치되어 있고, 각 센서는 두 층씩 감지할 수 있다고 하였습니다. sensor_list 중 센서가 작동되지 않은 모든 층이 아닌 센서가 위치한 층 만을 출력하려면 어떻게 할 수 있을까요?
(예시) range() 함수의 step 을 이용하면 됩니다. range(0,40,2)와 같이 건너뛸 요소의 개수를 입력합니다.

〈 Mission 3 〉

→ 센서 데이터 값에 작동되지 않음을 의미하는 데이터로 'error' 외에 '....'가 추가되었다면 작동되지 않는 모든 층의 번호를 어떻게 알아낼 수 있을까요?
(예시) 논리 연산자 or 를 활용하여 sensor_list 중 값이 'error' 이거나 '....'인 요소를 찾습니다.

평가 기준

평가 내용	1~5	강사 메모
학습		
사건 해결에 필요한 데이터를 스토리 속에서 파악할 수 있다.		
같은 일을 반복할 때 사용하는 반복 제어문에 대해 알고 사용할 수 있다.		
조건의 참과 거짓을 판단하는 조건 제어문에 대해 알고 사용할 수 있다.		
비교 및 논리연산자에 대해 알고 참과 거짓을 판단하는 프로그램을 만들 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

	purpose_list = ['general', 'office', 'office', 'general', 'office', 'general', 'office', 'office', 'office', 'secret', 'conference', 'conference', 'office', 'general', 'office', 'general', 'secret', 'office', 'general', 'secret', 'general', 'office', 'office', 'general', 'office', 'conference', 'office', 'office', 'office', 'conference', 'conference', 'conference', 'office', 'general', 'secret', 'general', 'office', 'secret', 'general', 'office']
	
	#기밀자료실(secret)이 있는 층 찾기.
	secret_list=[]
	for i in range(40):
	  if purpose_list[i] == 'secret':
	  secret_list.append(i+1)
	print(secret_list)
	
	#정답 확인:
	[10, 17, 20, 35, 38]

Mission2	**#data**
	sensor_list = ['error', 'error', '054057', '054324', '054326', '054327', 'error', 'error', '054345', '054352', '054353', '054359', '054404', '054406', '054411', '054412', '054413', '054414', 'error', 'error', '054415', '054416', 'error', 'error', 'error', 'error', '054421', '054422', 'error', 'error', '054425', '054426', '054427', '054428' , '054429', '054430', '054431', '054432', '054433', '054434']
	
	#위험물 감지 센서가 고장난 층 찾기
	error_list=[]
	for i in range(40):
	  if sensor_list[i] == 'error':
	  error_list.append(i+1)
	print(error_list)
	
	#정답 확인 :
	[1, 2, 7, 8, 19, 20, 23, 24, 25, 26, 29, 30]

Mission3	**#기밀자료실이면서 위험물 감지 센서가 고장난 층 찾기**

	for i in range(40):
	  if purpose_list[i] == 'secret' and sensor_list[i] == 'error':
	  print(i+1)
	  #print("폭발물이 설치된 층은 ",i+1, " 이다.")
	
	#정답 확인 :
	폭발물이 설치된 층은 20 이다.

더 나아가기
Mission1
	secret_list=[]
	for i in range(40):
	  if purpose_list[i] != 'general' and purpose_list[i] != 'office' and purpose_list[i] != 'conference' :
	  secret_list.append(i+1)
	print(secret_list)

더 나아가기
Mission2
	error_list=[]
	for i in range(0,40,2):
	  if sensor_list[i] == 'error':
	  error_list.append(i+2)
	print(error_list)

더 나아가기
Mission3
	for i in range(40):
	  if sensor_list[i] == 'error' or sensor_list[i] == '....':
	  print(i+1)