[폭발물 해체 작전]

폭발물 감지 드론은 폭발물이 설치된 정확한 위치를 파악했다.
중요한 기밀 자료가 많이 보관된 시청의 폭파를 막기 위해 NIS 본부에서는
현장에 폭발물 해체 전문 요원들을 투입하는데….

오늘의 목표	시청에 설치된 폭발물 해체하기
주요 함수	print, count

➤ 학습 목표

→ 스토리의 **사건**을 이해하며 미션을 해결하기 위해 필요한 데이터를 수집할 수 있습니다.
→ 리스트 요소의 존재 여부를 판단하는 포함 연산자를 알고 적절하게 사용할 수 있습니다.

→ 조건을 여러 개 설정하여 참과 거짓을 판단하는 프로그램을 만들 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 스토리 이해]

● 이전 차시 스토리에 관해 이야기를 나눕니다.

<이전 스토리>

이선과 파이는 빌더버그 조직이 설치한 폭발물이 시청 건물의 20 층이라는 사실을 알았고 기밀 자료를 안전하게 보존하기 위해 폭발물 탐지 드론을 현장에 보냈다. 하지만 드론은 20 층 전체 면적을 탐지하기에는 비행시간에 제한이 있어 폭발물을 설치할 것이라고 예상되는 호수를 찾아 드론으로 수색하여 20 층 33 호에 설치되어 있던 폭발물을 찾아냈다.

→ 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. 첫 번째 폭발물 해체하기.

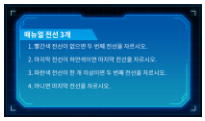
● 미션의 목표를 설명합니다

[미션 목표]

1) 스토리 속에서 단서를 확인하여 폭발물 해체 방법을 찾아낼 수 있습니다.
2) 조건 제어문을 사용하여 상황에 따라 다르게 실행하는 프로그램을 만들 수 있습니다.
3) 리스트 내의 같은 요소의 개수를 구하는 함수를 사용할 수 있습니다.

[스토리 이해]

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



< 단서 >

(이선과 파이는 드론이 보내온 데이터로 폭발물의 정확한 위치를 파악했다.)
(폭발물을 안전하게 제거하기 위해 NIS 는 폭발물 해체 전문 요원을 현장에 투입한다.)

“파이”, “현장에 투입된 전문 요원들에게 연락 왔어?”
“이선”, “응, 폭발물의 케이스를 조심스럽게 열어서 폭발물의 종류를 알아냈대.”
“파이”, “어떤 폭발물이래?”
“이선”, “PNF 라는데.. 어떤 폭발물인지 알아?”
“파이”, “PNF? 다른 폭발물에 비해 엄청 강력한 폭발력이 있는 폭탄이지!”
“이선”, “아! 그래서 전문 요원들도 케이스만 열어놓고 NIS 상황실로 해체 매뉴얼을 찾아달라고 한 거구나.”
“파이”, “그게 안전하겠지. 곧 본부에서 암호 해독기를 만들라고 하겠군….”
“파이”, “폭발물 해체 박스는 몇 개로 되어 있대?”
“이선”, “3 개로 이루어져 있는데 2 개는 크기가 같고 나머지 1 개는 조금 더 크대.”
“이선”, “그리고 같은 크기 박스 2 개 중에서 1 개만 열려 있대.”
“파이”, “그럼 전선 색깔을 알 수 있겠네?”
“이선”, “응! 파란색, 하얀색, 검은색 순서로 연결되어 있대….”
“파이”, “음… 암호 해독기에 돌려보면 되겠어.”
“이선”, “파이! 해체 매뉴얼이 도착했는데 암호 해독기를 빨리 만들어 달래.”
“파이”, “알았어. 매뉴얼 보내줘.”

[학습 자료]

<<< 단서 확인하기 >>>

<단서>

첫 번째 폭탄을 해체하기 위해 절단할 전선의 색깔을 찾아라.

→ 33 호에서 발견된 폭발물은 어떤 것인지 이야기 나누어 봅니다.

(예시) 발견된 폭발물은 다른 폭발물에 비해 강력한 폭발력이 있지만 매뉴얼을 찾아 해독기 프로그램을 만들면 해체할 수 있을 것이다. 폭발물은 총 3 개로 1 개는 크기가 크고 2 개는 동일한 크기로 되어 있으며 전선의 색깔이 다르고 연결된 순서도 다릅니다.

● 학습하게 될 개념에 대해 해당 코드를 구현하면서 적절하게 설명합니다.

 # 리스트

- 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

 # 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.')
```

 # 조건문

- 조건의 참과 거짓을 판단하는 제어문
- 구조 : if - elif - else

```
if 조건 1:
  수행할 문장 1
elif 조건 2:
  수행할 문장 2
else:
  수행할 문장 3

ex)
weather = 'rain'
if weather == 'rain':
   print('it's raining')
elif weather == 'snow':
   print('it's snowing')
elif weather == 'cloud':
   print('it's cloudy')
else:
   print('it's sunny')
```

 # 비교 연산자

- 숫자, 문자열 등 두 항목을 서로 비교하는 연산자

파이썬 연산자 기호	의미
x == y	x 와 y 가 같다
x != y	x 와 y 는 같지 않다

● 결과는 True 또는 False 반환

ex) 1 == 'one'

'Alpha' != 'Avocado'

→ 파이썬에서 유용하게 쓰이는 연산자 중에 포함 연산자가 있습니다. 'in' 은 멤버 연산자로 어떤 리스트가 있을 때 그 리스트에 특정한 값이 있는가를 찾아서 그 값이 있으면 'True' 없으면 'False' 를 반환해 줍니다. 'not in' 은 반대의 의미로 사용합니다. 이 때 정수 3 과 실수 3.0 은 동일한 자료형이므로 같은 값을 반환하지만 정수 3 과 문자열화된 정수 '3' 은 다른 자료형이기 때문에 다른 값을 반환합니다.

[코드 이해]

● 사용할 API 에 대해 설명합니다.

< API >

1) count('문자열') : 리스트 내부 특정 문자열의 개수 반환
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 >

```
wires1 = ['blue','white','black']

if 'red' not in wires1:
    print("cut the",wires1[1],"wire")
elif wires1[-1] == 'white':
    print("cut the", wires1[-1], "wire")
elif wires1.count('blue') >= 1:
    print("cut the", wires1[1], "wire")
else:
```

```
  print("cut the", wires1[-1],"wire")

#정답 확인:
cut the white wire
```

→ 폭발물을 해체 하기 위한 암호 해독기를 만들 방법에 관해 이야기를 나누어 봅니다.

(예시) 전선 3 개 일 때의 해체 매뉴얼에는 네 가지의 조건 제시문이 있습니다. 이 제시문을 그대로 프로그램으로 만들기 위해서는 if 문과 else 문을 사용하게 되는데 이때, if 문이 거짓일 때 else 문의 명령이 조건 없이 실행하게 됩니다. 그리고 else 문은 if 문 없이 단독으로 사용할 수 없습니다. 만약 여러 개의 조건을 판단해야 한다면 if 문과 else 문 사이에 elif 문을 사용합니다. if 문 조건을 만족하지 않는다면 elif 문의 명령이 실행되고 elif 문 조건도 만족하지 않는다면 else 문이 실행됩니다.

매뉴얼의 첫 번째 제시문을 if 문으로, 두 번째 제시문과 세 번째 제시문을 elif 문으로, 마지막 제시문을 else 문으로 구현합니다.

→ 실행결과 창에 출력된 값을 정답창에 입력하여 확인하도록 합니다.

(예시) wires1 리스트의 전선 색깔 요소 순서를 'blue','white','black' 으로 입력하여 나온 결과값인 **cut the white wire** 를 입력한 후 정답을 확인하도록 합니다.

→ 스스로 코딩하도록 유도합니다.

→ 미션 2 템플릿으로 이동합니다.

Mission2. 두 번째 폭발물 해체하기.

● 미션의 목표를 설명합니다

[미션 목표]

1) 스토리 속에서 단서를 확인하여 폭발물 해체 방법을 찾아낼 수 있습니다.
2) 조건 제어문을 사용하여 상황에 따라 다르게 실행하는 프로그램을 만들 수 있습니다.
3) 리스트 내의 같은 요소의 개수를 구하는 함수를 사용할 수 있습니다.

[스토리 이해]

< 단서 >

● 스토리의 내용을 이해합니다.

- 이전 미션에서 나온 프로그래밍 결과를 생각해보며 스토리를 확인합니다.

"이선", "파이! 만들어준 암호 해독기로 첫 번째 박스를 해체하니까 두 번째 박스가 열렸대."

"파이", "응! 두 번째 박스의 전선은 무슨 색깔이래?"

"이선", "전문 요원들이 사진을 보내줬어. 확인해봐!"

<<< 단서 확인하기 >>>

<단서>

두 번째 폭탄을 해체하기 위해 절단할 전선의 색깔을 찾아라.

→ 두 번째 박스의 전선의 색깔은 어떤 순서로 되어 있는지 이야기 나누어 봅니다.

(예시) 전문 요원들이 보낸 사진을 보면 하얀색, 파란색, 빨간색의 순서로 전선이 연결되어 있습니다.

[학습 자료]

● 학습하게 될 개념에 대해 해당 코드를 구현하면서 적절하게 설명합니다.

 # 미션 1 과 동일합니다.

[코드 이해]

● 사용할 API 에 대해 설명합니다.

< API >

1) count('문자열') : 리스트 내부 특정 문자열의 개수 반환
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 >

```
wires2 =['white','blue','red']

if 'red' not in wires2:
    print("cut the",wires2[1],"wire")
elif wires2[-1] == 'white':
    print("cut the", wires2[-1], "wire")
elif wires2.count('blue') >= 1:
    print("cut the", wires2[1], "wire")
else:
    print("cut the", wires2[-1],"wire")

#정답 확인 :
cut the blue wire
```

→ 전선 3 개로 이루어진 두 번째 폭발물을 해체하기 위한 방법에 관해 이야기 나누어 봅니다.

(예시) 두 번째 폭발물 박스도 첫 번째와 동일한 전선의 개수이므로 미션 1 에서 만든 암호 해독기를 이용하여 절단할 전선을 찾습니다. 전선의 색깔 순서에 맞게 리스트의 요소에 추가하여 암호 해독기를 실행해 봅니다.

→ 실행결과 창에 출력된 값을 정답창에 입력하여 확인하도록 합니다.
(예시) wires2 리스트의 전선 색깔 요소 순서를 'white','blue','red' 으로 입력하여 나온 결과값인 **cut the blue wire** 를 입력한 후 정답을 확인하도록 합니다.
→ 스스로 코딩하도록 유도합니다.
→ 미션 3 템플릿으로 이동합니다.

Mission3. 세 번째 폭발물 해체하기.

[미션 목표]

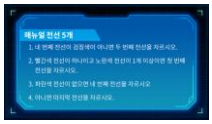
● 미션의 목표를 설명합니다

1) 스토리 속에서 단서를 확인하여 폭발물 해체 방법을 찾아낼 수 있습니다.
2) 조건 제어문을 사용하여 상황에 따라 다르게 실행하는 프로그램을 만들 수 있습니다.
3) 리스트 내의 같은 요소의 개수를 구하는 함수를 사용할 수 있습니다.

[스토리 이해]

● 스토리의 내용을 이해합니다.

- 이전 미션에서 나온 프로그래밍 결과를 생각해보며 스토리를 확인합니다.



< 단서 >

"이선", "이제 마지막 큰 박스만 남았어!"
"파이", "그래, 아까 사진에서 보니까 왠지 큰 박스는 전선이 많을 것 같은데. "
"이선", "응! 맞아! 전선이 5 개래. "
"이선", "노란색, 파란색, 하얀색, 검은색, 빨간색의 순서로 연결되어 있다고 전문 요원이 말해줬어. "
"파이", "전선의 개수가 다르다면 지금 만든 해독기는 사용 못 할 거야. "
"파이", "아까 받은 매뉴얼을 보고 해독기 프로그램을 좀 수정해 볼게. "
"이선", "응! 서둘러줘. 파이. "

<<< 단서 확인하기 >>>

〈단서〉

세 번째 폭탄을 해체하기 위해 절단할 전선의 색깔을 찾아라.

→ 세 번째 폭발물 박스를 해체하기 위한 방법에 관해 이야기 나누어 봅니다.
(예시) 앞서 해체했던 폭발물 박스들과 다르게 세 번째 폭발물 박스는 크기가 크고 전선이 5 개가 있습니다. 만들어 놓은 암호 해독기를 전선 5 개

[학습 자료]

일 때의 매뉴얼에 맞게 수정할 필요가 있으며 전선의 색깔은 노란색, 파란색, 하얀색, 검은색, 빨간색의 순서로 되어 있습니다.

● 학습하게 될 개념에 대해 해당 코드를 구현하면서 적절하게 설명합니다.

 # 미션 1 과 동일합니다.

● 사용할 API 에 대해 설명합니다.

[코드 이해]

< API >

1) count('문자열') : 리스트 내부 특정 문자열의 개수 반환
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 >

```
wires3=['yellow','blue','white','black','red']

if  wires3[3] != 'black':
    print("cut the", wires3[1], "wire")
elif wires3.count('red') == 1 and wires3.count('yellow') >= 1:
    print("cut the", wires3[0], "wire")
elif 'blue' not in wires3:
    print("cut the", wires3[3] ,"wire")
else:
    print("cut the",wires3[-1],"wire")

 #정답 확인 :
 cut the yellow wire
```

→ 세 번째 폭발물 박스를 해체하기 위한 방법에 관해 이야기 나누어 봅니다.
(예시) 전선 5 개 일 때의 매뉴얼을 확인하여 만들어진 암호 해독기를 수정합니다.
→ 결과 창에 출력된 값을 입력 버튼을 누르고 정답을 확인하도록 합니다.
(예시) wires3 리스트의 전선 색깔 요소 순서를 'yellow', 'blue', 'white', 'black', 'red' 으로 입력하여 나온 결과값인 **cut the yellow wire** 를 입력한 후 정답을 확인하도록 합니다.
→ 스스로 코딩하도록 유도합니다.
→ 마무리 템플릿으로 이동합니다.

Summary.

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

 # 리스트

- 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

 # 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.’)

 # 조건문

- 조건의 참과 거짓을 판단하는 제어문
- 구조 : if - elif - else

```
if 조건 1:
  수행할 문장 1
elif 조건 2:
  수행할 문장 2
else:
  수행할 문장 3


ex)
weather = 'rain'
if weather == 'rain':
   print('it's raining')
elif weather == 'snow':
   print('it's snowing')
elif weather == 'cloud':
   print('it's cloudy')
```

```
else:
    print('it's sunny')
```

 # 비교 연산자

- 숫자, 문자열 등 두 항목을 서로 비교하는 연산자

파이썬 연산자 기호	의미
x == y	x 와 y 가 같다
x != y	x 와 y 는 같지 않다

- 결과는 True 또는 False 반환

  ex) 1 == 'one'

  'Alpha' != 'Avocado'

 # API

- count('문자열') : 리스트 내부 특정 문자열의 개수 반환
- print(값) : 값을 출력하는 함수

더 나아가기

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

〈 Mission 1 〉

→ 폭탄 해체 매뉴얼 중 '노란색 전선이 있다면 마지막 전선을 자르시오'라는 조건이 추가된다면 암호 해독기 코드를 어떻게 고칠 수 있을까요?
(예시) 포함 연산자 in 을 사용하는 elif 문을 추가하여 조건 제어문을 완성합니다..

〈 Mission 2 〉

→ 폭탄 해체 매뉴얼 중 세 번째 다음 조건으로 '첫 번째 전선이 파란색이 아니라면 첫 번째 전선을 자르시오'라는 내용이 추가되어 있다면 암호 해독기 코드를 어떻게 고칠 수 있을까요?
(예시) 비교 연산자 != 를 사용하는 elif 문을 추가하여 조건 제어문을 완성합니다.

〈 Mission 3 〉

→ 6 개의 전선을 해체하는 매뉴얼로 6 개의 전선 색을 검사할 수 있도록 암호 해독기를 준비하라는 지시를 받았다면 결과는 어떻게 달라질까요?
매뉴얼
전선 6 개

1. 첫 번째 전선이 검정색이 아니면 두 번째 전선을 자르시오.
2. 파란색 전선이 2 개이면 마지막 전선을 자르시오.
3. 검정색 전선이 없으면 네 번째 전선을 자르시오.
4. 빨간색 전선이 있으면 첫 번째 전선을 자르시오.
5. 아니면 마지막 전선을 자르시오.

(예시) 6 번째 전선 색을 저장한 리스트를 만듭니다. 잘라야 할 전선의 색을 리스트의 인덱스를 이용하여 조건문의 구조에 매뉴얼의 내용을 추가하여 암호 해독기를 완성합니다. 리스트의 요소와 순서에 따라 출력되는 전선 색의 정보가 달라집니다.

평가 기준

평가 내용	1~5	강사 메모
학습		
사건 해결에 필요한 데이터를 스토리 속에서 파악할 수 있다.		
조건 중첩 제어문을 사용하여 참과 거짓을 구별하고 상황에 따라 다르게 실행되는 프로그램을 만들 수 있습니다.		
리스트 요소의 존재 여부를 판단하는 포함 연산자를 알고 적절하게 사용할 수 있습니다.		
리스트 요소 내의 같은 문자열의 개수를 확인하는 프로그램을 만들 수 있습니다.		
주요 함수를 사용하여 프로그래밍하고 결과 값을 찾아 미션을 해결할 수 있다.		
태도		
어려운 점이 있어도 포기하지 않고 끝까지 해결하려고 노력하였다.		

모범 답안

코드	

Mission1	wires1 = ['blue','white','black']
	
	if 'red' not in wires1:
	  print("cut the",wires1[1],"wire")
	elif wires1[-1] == 'white':
	  print("cut the", wires1[-1], "wire")

	elif wires1.count('blue') >= 1:
	  print("cut the", wires1[1], "wire")
	else:
	  print("cut the", wires1[-1],"wire")
	
	#정답 확인:
	cut the white wire

Mission2	wires2 =['white','blue','red']
	
	if 'red' not in wires2:
	  print("cut the",wires2[1],"wire")
	elif wires2[-1] == 'white':
	  print("cut the", wires2[-1], "wire")
	elif wires2.count('blue') >= 1:
	  print("cut the", wires2[1], "wire")
	else:
	  print("cut the", wires2[-1],"wire")
	
	#정답 확인 :
	cut the blue wire

Mission3	wires3=['yellow','blue','white','black','red']
	
	if wires3[3] != 'black':
	  print("cut the", wires3[1], "wire")
	elif wires3.count('red') == 1 and wires3.count('yellow') >= 1:
	  print("cut the", wires3[0], "wire")
	elif 'blue' not in wires3:
	  print("cut the", wires3[3] ,"wire")
	else:
	  print("cut the",wires3[-1],"wire")
	
	#정답 확인 :
	cut the yellow wire

더 나아가기
Mission1
	wires1 = ['blue','white','black']
	
	if 'red' not in wires1:
	  print("cut the",wires1[1],"wire")
	elif 'yellow' in wires1:
	  print("cut the last wire.")
	elif wires1[-1] == 'white':
	  print("cut the", wires1[-1], "wire")
	elif wires1.count('blue') >= 1:
	  print("cut the", wires1[1], "wire")
	else:
	  print("cut the", wires1[-1],"wire")
	
	#정답 확인 :
	cut the white wire

더 나아가기
Mission2
	wires2 =['white','blue','red']
	
	if 'red' not in wires1:
	  print("cut the",wires1[1],"wire")
	elif wires1[-1] == 'white':
	  print("cut the", wires1[-1], "wire")
	elif wires1.count('blue') >= 1:
	  print("cut the", wires1[1], "wire")
	elif wires1[0] != 'blue':
	  print("cut the", wires1[0], "wire")
	else:
	  print("cut the", wires1[-1],"wire")
	
	#정답 확인 :
	cut the blue wire

더 나아가기 Mission3	매뉴얼 전선 6 개 6. 첫 번째 전선이 검정색이 아니면 두 번째 전선을 자르시오. 7. 파란색 전선이 2 개이면 마지막 전선을 자르시오.

	8. 검정색 전선이 없으면 네 번째 전선을 자르시오.
	9. 빨간색 전선이 있으면 첫 번째 전선을 자르시오.
	10. 아니면 마지막 전선을 자르시오.
	
	wires3 = ['black', 'purple', 'red', 'white', 'blue', 'yellow']
	
	if wires3[0] != 'black':
	print("cut the",wires3[1],"wire")
	elif wires3.count('blue') == 2:
	print("cut the", wires3[-1], "wire")
	elif 'black' not in wires3:
	print("cut the", wires3[3], "wire")
	elif 'red' in wires3:
	print("cut the", wires3[0], "wire")
	else:
	print("cut the", wires3[-1],"wire")
	
	#정답 확인:
	cut the black wire