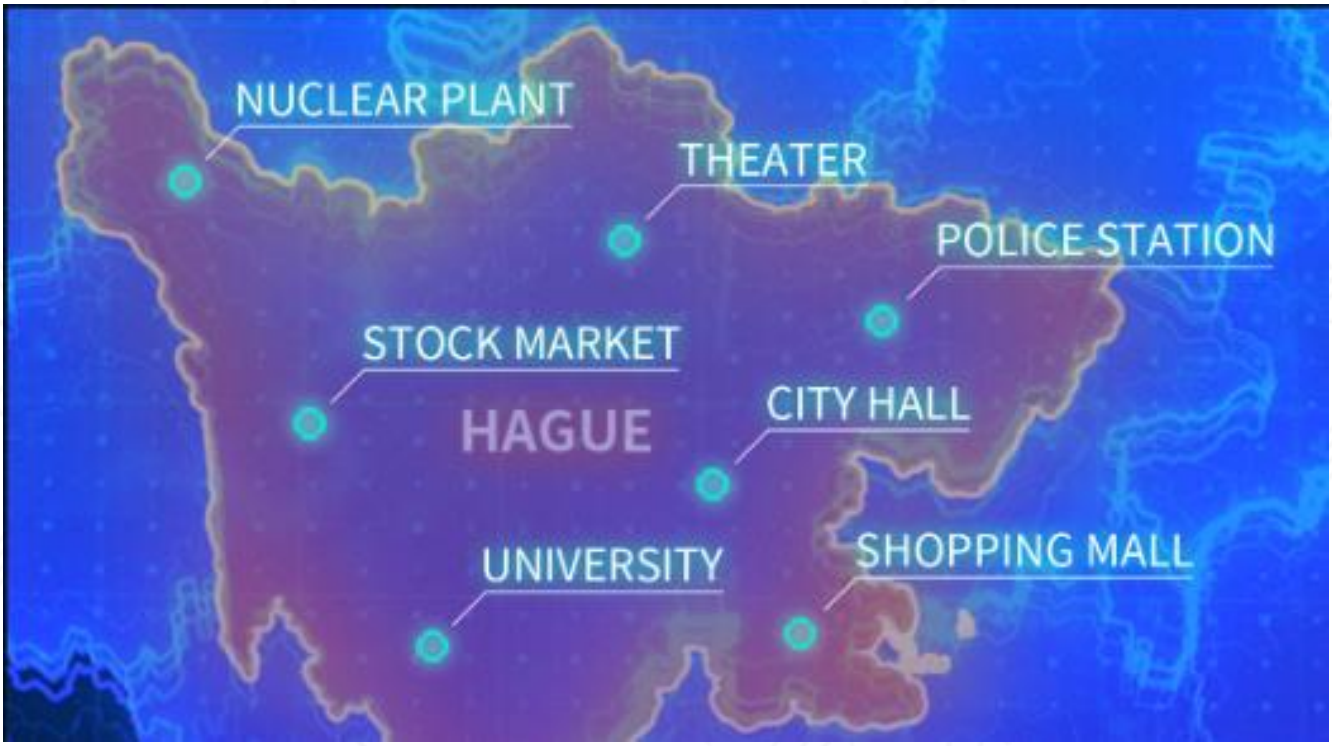
[비밀 작전 계획 장소 추적]



빌더버그 조직의 헤이그 도시 파괴 계획 날짜를 드디어 알아냈다.
이제는 장소를 알아내 그들의 계획을 무산 시켜야 한다.
데이터 분석 전문가에게서 나머지 1 개의 파일에서 장소에 대한 힌트가 있다는 답변이 왔는데...

오늘의 목표	빌더버그 조직의 헤이그 도시 파괴 계획 장소 추적하기
주요 함수	print, len, find, replace, split, sort

➤ 학습 목표

→ 스토리의 **사건**을 이해하며 미션을 해결하기 위해 필요한 데이터를 수집할 수 있습니다.
→ 문자열을 원하는 형태로 **수정**하여 사용할 수 있습니다.
→ 데이터 자료형인 **리스트**에 대해 알고 사용할 수 있습니다.

→ 리스트에 저장된 요소들을 오름차순으로 **정렬**할 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 차시 스토리에 관해 이야기를 나눕니다.

[이전 스토리 이해]

<이전 스토리>

이선과 파이는 빌더버그의 인공위성 **'Intelsat'** 에서 받은 파일 3 개 중 2 개의 파일을 데이터 분석 전문가에게 의뢰하여 단서를 확보한 후에 API 도구를 이용하여 헤이그 도시 파괴 계획 날짜가 1 월 31 일 이라는 정보를 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. 헤이그 도시 파괴 계획 장소 힌트 데이터 찾기.

● 미션의 목표를 설명합니다

[미션 목표]

1) 손상된 데이터를 분석하여 힌트 데이터를 찾을 수 있습니다.
2) 문자열 함수 find 를 사용하여 문자열의 위치를 파악할 수 있습니다.
3) 문자열 인덱싱과 슬라이싱을 이용하여 원하는 문자열을 추출할 수 있습니다.

[스토리 이해]

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


< 단서 >

(빌더버그 조직이 헤이그 도시를 파괴하려는 계획 날짜를 드디어 알아냈다.)
(바로 내일! 하루밖에 남지 않은 상황이다.)
(Intelsat 인공위성도 점점 묘지 궤도를 향해 가고 있어 시간이 많지 않은데....)

"파이", "빌더버그 조직은 이 넓은 헤이그 도시 어디에 폭발물을 설치했을까…. "
"이선", "파이! 잠깐! 데이터 분석 전문가에게 다시 답변이 왔어."
"이선", "내가 혹시 몰라서 나머지 1 개 파일도 해독해달라고 했거든!"
"파이", "뭐라고 답변이 왔어? "
"이선", "역시 내 예상이 맞았어!"
"이선", "이 파일 데이터에는 장소에 대한 힌트가 담겨있는데,"
"이선", "start 와 end 단어 사이에 있는 문자열을 분석해야 한대."
"파이", "그래? 일단 그 문자열만 추출해야겠네."
"이선", "응! 나는 Intelsat 과 통신이 끊기기 전까지 다른 정보를 얻을 수 있는지 시도해 볼게."
"파이", "알았어. 어떤 정보든 모두 수집해 줘. "
"파이", "나는 빨리 데이터를 확인해 볼게. "

<<< 단서 확인하기 >>>

<단서>
파일 데이터 속에서 힌트 문자열을 찾아라.

→ 파이와 이선의 현재 상황에 관해 이야기를 나누어 봅니다.

(예시) 헤이그 도시 파괴 계획 날짜를 알아낸 이선과 파이는 이제 장소를 추적하려고 합니다. 인공위성 Intelsat 에서 확보한 파일 3 개 중 나머지 1 개의 파일 속에서 장소에 대한 힌트 데이터를 찾아내려고 합니다.

→ 장소에 대한 힌트 데이터는 어떻게 찾을 수 있을지 질문합니다.

(예시) 데이터 분석 전문가의 답변에 의하면 장소에 대한 힌트 데이터는 file_b 에서 start 와 end 단어 사이의 문자열을 분석해야 한다고 했으니 그 문자열을 추출합니다.

[학습 자료]

● 학습하게 될 개념에 대해 해당 코드를 구현하면서 적절하게 설명합니다.

 # 문자열

- 작은따옴표(' '), 큰따옴표(" ") 로 묶음
- 문자열 연산 : +, * 연산자 사용 가능
- 인덱스는 0 부터 순서 시작
- 인덱싱
  - 해당 위치의 문자 추출
  - 변수[정수값]
- 슬라이싱
  - 해당 구간의 문자를 추출
  - 변수[시작 : 끝]

→ 지난 차시에 배운 학습 내용을 복습할 수 있도록 합니다.

● 사용할 API 에 대해 설명합니다.

< API >

1) print(값) : 값을 출력하는 함수

2) len('문자열') : 문자열 개수 반환 함수

3) find('문자열') : 문자열의 특정문자 위치 반환 함수

→ print 함수는 () 안에 값을 넣으면 그 값을 실행결과 창에 출력해 줍니다.

→ len 함수는 () 안에 문자열을 넣으면 문자열의 갯수를 정수로 반환해 줍니다.

[코드 이해]

→ find 함수는 () 안에 찾고자 하는 문자열을 넣으면 위치를 정수로 반환해 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
file_b
='djhfaheu___wehiehrhlsfhouhewwehr1238364892hrehwfwhelhewlehrle
whiorhhf3824863___883@hre93734084fdfhieelwhfhieistart#.redro
lacitebahpla yb kcatta trats eW .yti@sr@@evinu,tna@lp rae@@lcun,llam@
gni@@ppoh@s,lla@h@ y@tic,tekr@am kcots,noi@tats
eci@l@op,ret@aeht:secalp gniwollof fo eno si tegrat
ehT#endhfdhsifohifeifhlk368537djs89hds83e____89fwgafg3dbsjhgdiutwf
w823___t93g3%@iu3977e&egd37dheehdgsaioiowi'

#start 와 end 사이의 문자열 찾기
start=file_b.find('start')
end=file_b.find('end')
print(start, end)

#힌트 데이터 찾기
hint_data = file_b[start+len('start'):end]
print(hint_data)

#정답 확인 :
#.redro lacitebahpla yb kcatta trats eW .yti@sr@@evinu,tna@lp
rae@@lcun,llam@ gni@@ppoh@s,lla@h@ y@tic,tekr@am kcots,noi@tats
eci@l@op,ret@aeht:secalp gniwollof fo eno si tegrat ehT#
```

→ b 파일에 있는 문자열에서 start 와 end 문자를 찾는 방법을 질문해 봅니다.
(예시) b 파일에서 해당 문자열의 위치를 찾을 때는 문자열 함수 중에서 find 를 사용하여 위치 값을 찾아 각각 start 의 시작 위치값을 start 변수에 end 의 시작 위치 값을 end 변수에 저장하여 실행결과 창에 출력해 봅니다.

→ start 와 end 사이의 문자열을 추출할 수 있는 방법에 관해 질문합니다.
(예시) 문자열 슬라이싱 방법을 이용하여 시작인덱스에는 start 의 문자가 시작하는 위치를 찾아 start 문자까지 제외시키기 위해 len 함수를 이용하거나, start 문자의 갯수만큼 더한 값을 넣고 끝인덱스에는 end 문자가 시작되는 위치 값을 넣어 해당 힌트 데이터를 hint_data 변수에 저장하여 출력하도록 합니다.

→ 실행결과 창에 출력된 값을 정답창에 입력하여 확인하도록 합니다.
(예시) hint_data 변수에 저장하여 출력한 후 그대로 입력하여 정답을 확인하도록 합니다.

→ 스스로 코딩하도록 유도합니다.

→ 미션 2 템플릿으로 이동합니다.

Mission2. 힌트 데이터 속에서 장소 리스트 찾기.

[미션 목표]

● 미션의 목표를 설명합니다

1) 스토리 속에서 힌트 데이터의 규칙을 찾아낼 수 있습니다.
2) 단서에 따라 문자열을 수정하여 미션을 해결할 수 있습니다.
3) 문자열을 역순으로 변경하거나 제거하여 원하는 데이터를 찾을 수 있습니다.

[스토리 이해]

● 스토리의 내용을 이해합니다.

- 이전 미션에서 나온 프로그래밍 결과를 생각해보며 스토리를 확인합니다.

< 단서 >

"이선", "장소 힌트 데이터 맞아? 전혀 모르겠는데...."
"파이", "내가 보기엔 데이터에 어떤 규칙을 적용해 변형 시켜 놓은 것 같아."
"이선", "규칙? 무슨 규칙인데?"
"파이", "데이터가 손상되어 있긴 하지만, 뭔가 문자열이 뒤집혀 있는 것으로 보여."
"이선", "그래? 조금만 더 서둘러줘. 파이!"
"파이", "알았어."

<<< 단서 확인하기 >>>

<단서>

문자열 속 장소 리스트를 찾아라.

→ 장소 리스트를 찾기 위한 방법에 관해 질문합니다.

(예시) 미션 1 에서 찾은 힌트 데이터에는 규칙이 있다고 했으니 규칙에 따라 문자열을 변경해 봅니다. 데이터 속에 있는 규칙은 문자열이 뒤집혀 있다는 것 입니다.

● 학습하게 될 개념에 대해 해당 코드를 구현하면서 적절하게 설명합니다.

[학습 자료]

 # 문자열 뒤집기

● 문자열의 순서를 거꾸로 변경

- [: : -1] 의 형태로 마지막에 음수(-1)을 넣음

ex) string_test = 'banana'

string_reverse = string_test[: : -1]

● 사용할 API 에 대해 설명합니다.

< API >

1) print(값) : 값을 출력하는 함수
2) find('문자열') : 문자열의 특정문자 위치 반환 함수
3) replace("대상 문자","바꿀 문자") : 문자열의 값을 치환하는 함수

[코드 이해]

● 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 >

```
#힌트 데이터를 뒤집어 출력하기
reverse_data = hint_data[::-1]
print(reverse_data)

#문자열 슬라이싱
place_slice = reverse_data.find(':')
print(place_slice)

place_list = reverse_data[place_slice+1:141]
print(place_list)

#손상된 데이터 복구, 특수문자 제거
real_data = place_list.replace('@', '')
```

```
print(real_data)

#정답 확인 :
theater,police station,stock market,city hall,shopping mall,nuclear plant,university
```

→ 힌트 데이터 속에서 장소 리스트를 찾기 위한 방법에 관해 질문합니다.
(예시) 힌트 데이터에서 발견한 규칙인 문자열 뒤집기를 통해 정상적인 문자열을 확인하고 문자열 함수를 사용하여 변형시키는 과정을 통해 장소 리스트를 찾아내도록 합니다.

→ 장소 리스트 문자열만 추출하기 위해 ':' 을 찾아 문자열의 위치를 찾고 이후 문자열을 슬라이싱 합니다. 이 때 끝 인덱스는 장소와 관련된 단어가 추출되는 과정을 확인하면서 적당한 위치(141)를 슬라이싱 합니다.

→ 결과 창에 출력된 값을 입력 버튼을 누르고 정답을 확인하도록 합니다.
(예시) 힌트데이터 속에서 찾은 장소 리스트를 real_data 변수에 저장하여 실행결과 창에 출력한 후 그대로 복사하여 입력 창에 정답을 확인하도록 합니다.

→ 스스로 코딩하도록 유도합니다.

→ 미션 3 템플릿으로 이동합니다.

Mission3. 헤이그 도시 파괴 계획 장소 추적하기.

[미션 목표]

● 미션의 목표를 설명합니다

1) 대화 속에서 사건을 해결할 중요한 힌트를 파악할 수 있습니다.
2) 데이터 자료형 중 리스트에 대해 알 수 있습니다.
3) 리스트의 요소를 오름차순으로 정렬할 수 있습니다.

[스토리 이해]

● 스토리의 내용을 이해합니다.

- 이전 미션에서 나온 프로그래밍 결과를 생각해보며 스토리를 확인합니다.

< 단서 >

(이선의 컴퓨터 화면은 헤이그 도시의 지도로 가득 차 있다.)

(이선의 눈이 예리하게 빛난다.)

"이선", "파이! 힌트 데이터 속에 있는 장소들은 헤이그 도시 전 지역에 퍼져있어."

"파이", "맞아. 이 많은 곳에 폭발물을 다 설치하진 않았을 것 같은데…. "

"이선", "이 넓은 곳에서 과연 그들은 어디를 타깃으로 두었을까."

"이선", "아! 파이 아까 그 힌트 데이터 속에 단서가 있었어!"

"파이", "맞아! 그들은 알파벳 순서로 파괴한다 했었지!"

<<< 단서 확인하기 >>>

<단서>

첫 번째 폭발물 설치 장소를 찾아라.

[학습 자료]

→ 빌더버그 조직은 헤이그 도시 파괴 계획에서 타깃을 어떻게 선정하는지에 관해 이야기 나누어 봅니다.

(예시) 힌트 데이터를 다시 한번 상기시켜 보았을 때 마지막 문장을 보면 **'We start attack by alphabetical order.'** 이라고 적힌 부분을 확인합니다.

찾아낸 장소 리스트를 알파벳 순서로 정렬하여 첫 번째 나오는 장소가 타깃입니다.

● 학습하게 될 개념에 대해 해당 코드를 구현하면서 적절하게 설명합니다.

 # 문자열 나누기

- 구분자를 기준으로 문자열이 나뉨
- 나누어진 문자열은 리스트에 ‘,’로 구분되어 요소값으로 변환
- 변수.split(‘문자열’)

ex) string_fruits = 'apple$orange$kiwi$banana$mango'
fruits_split = string_fruits.split('$')

 # 리스트(list)란?

- 순서가 있는 요소들의 집합
- 요소들을 수정, 삭제, 추가 가능
- 대괄호([ ]) 안에 요소들을 ‘,’ 로 구분하여 저장

ex) list_test = [0, 2, 4, 6, 8]

 # 리스트 정렬하기

- 리스트의 요소를 오름차순으로 정렬
- 변수.sort()

ex) numbers = [1, 10, 14, 5, 8, 12]
numbers.sort()

● 사용할 API 에 대해 설명합니다.

[코드 이해]

〈 API 〉

1) split(‘문자열’) : 구분자를 기준으로 문자열을 나누는 함수
2) sort() : 리스트의 요소를 정렬하는 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 〉

```
#리스트로 변환하기
places = real_data.split(',')
print(places)

#알파벳 순으로 정렬하기
places.sort()
print(places[0])

#정답 확인 : city hall
```

→ 문자열 함수 중 split 를 사용하여 구분자 ',' 를 기준으로 문자열을 나누어 장소들을 리스트의 형태로 변환하여 places 변수에 저장합니다

→ places 에 저장된 요소들을 알파벳 순서로 정렬하여 첫 번째 요소를 확인합니다.

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

(예시) 빌더버그 조직이 헤이그 도시를 파괴하려는 장소는 **city hall** 입니다. 정답을 확인합니다.

→ 스스로 코딩하도록 유도합니다.

→ 마무리 템플릿으로 이동합니다.

Summary.

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

 # 문자열

- 작은따옴표(‘ ’), 큰따옴표(“ ”) 로 묶음
- 문자열 연산 : +, * 연산자 사용 가능
- 인덱스는 0 부터 순서 시작
- 인덱싱
  - 해당 위치의 문자 추출
  - 변수[정수값]
- 슬라이싱
  - 해당 구간의 문자를 추출
  - 변수[시작 : 끝]

 # 문자열 뒤집기

- 문자열의 순서를 거꾸로 변경
  - [: : -1] 의 형태로 마지막에 음수(-1)을 넣음

ex) string_test = ‘banana’
string_reverse = string_test[: : -1]

 # 문자열 나누기

- 구분자를 기준으로 문자열이 나뉨
- 나누어진 문자열은 리스트에 ‘,’로 구분되어 요소값으로 변환
- 변수.split(‘문자열’)

ex) string_fruits = 'apple$orange$kiwi$banana$mango'
fruits_split = string_fruits.split('$')

 # 리스트(list)란?

- 순서가 있는 요소들의 집합
- 요소들을 수정, 삭제, 추가 가능
- 대괄호([ ]) 안에 요소들을 ‘,’ 로 구분하여 저장

ex) list_test = [0, 2, 4, 6, 8]

 # 리스트 정렬하기

- 리스트의 요소를 오름차순으로 정렬
- 변수.sort()

  ex) numbers = [1, 10, 14, 5, 8, 12]

  numbers.sort()

 # API

- len('문자열') : 문자열의 길이 반환 함수
- find('문자열') : 문자열의 특정문자 위치 반환 함수
- replace("대상 문자","바꿀 문자") : 문자열의 값을 치환하는 함수
- split('문자열') : 구분자를 기준으로 문자열을 나누는 함수
- sort() : 리스트의 요소를 정렬하는 함수
- print(값) : 값을 출력하는 함수

더 나아가기

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

〈 Mission 1 〉

→ 파일 데이터의 중요한 부분은 "start"와 'end'로 감싸져 있었습니다. 만약 파일 데이터의 'start'부터 문자열의 끝까지가 중요한 데이터였다면 문자열을 어떻게 추출할 수 있을까요?
(예시) 문자열 슬라이싱을 할 때 [109+len('start'):]와 같이 끝 인덱스 위치를 비워두면 자동으로 문자열의 끝까지 슬라이싱 됩니다.

〈 Mission 2 〉

→ 암호 문자열을 거꾸로 변경한 결과를 3 번째 인덱스부터 100 번째 인덱스까지만 출력한 결과를 얻으려면 어떻게 해야 할까요?

(예시) 문자열을 거꾸로 변경할 때 썼던 코드 [::-1]의 처음과 끝 인덱스를 수정하면 됩니다. 문자열 슬라이싱 규칙대로 문자열에 [99:3:-1]과 같은 코드를 적용합니다.

〈 Mission 3 〉

→ 리스트로 변환된 장소의 이름 중, 알파벳 역순으로 첫 번째 장소가 타깃이라면 어떻게 해야 할까요?
(예시) 리스트의 요소를 내림차순으로 정렬하고자 할 때는 sort() 함수의 괄호 안에 reverse = True 옵션을 표기해주면 됩니다.

평가 기준

평가 내용	1~5	강사 메모
학습		
사건 해결에 필요한 데이터를 스토리 속에서 파악할 수 있다.		
문자열 함수인 find 를 사용하여 문자열의 위치를 찾을 수 있다.		
문자열 슬라이싱을 사용하여 문자열을 뒤집어 출력할 수 있다.		
데이터 자료형인 리스트에 대해 알고 리스트의 요소들을 기준에 맞게 정렬할 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_b ='djhfaheu___wehiehrhlsfhouhewwehr1238364892hrehwfwhelhewlehrlewhiorhhf3824863___883@hre93734084fdfhieelwhfhieistart#.redro lacitebahpla yb kcatta trats eW .yti@sr@@evinu,tna@lp rae@@lcun,llam@ gni@@ppoh@s,lla@h@ y@tic,tekr@am kcots,noi@tats eci@l@op,ret@aeht:secalp gniwollof fo eno si tegrat ehT#endhfdhsifohifeifhlk368537djs89hds83e____89fwgafg3dbsjhgdiutwfw823___t93g3%@iu3977e&egd37dheehdgsaioiowi' **#start 와 end 사이의 문자열 찾기** start=file_b.find('start') end=file_b.find('end') print(start, end) **#힌트 데이터 찾기** hint_data = file_b[start+len('start'):end] print(hint_data) #정답 확인 : #.redro lacitebahpla yb kcatta trats eW .yti@sr@@evinu,tna@lp rae@@lcun,llam@ gni@@ppoh@s,lla@h@ y@tic,tekr@am kcots,noi@tats eci@l@op,ret@aeht:secalp gniwollof fo eno si tegrat ehT#
Mission2	**#힌트 데이터를 뒤집어 출력하기** reverse_data = hint_data[::-1] print(reverse_data) **#문자열 슬라이싱**

	place_slice = reverse_data.find(':') print(place_slice) place_list = reverse_data[place_slice+1:141] print(place_list) **#손상된 데이터 복구, 특수문자 제거** real_data = place_list.replace('@', '') print(real_data) #정답 확인 : theater,police station,stock market,city hall,shopping mall,nuclear plant,university
Mission3	**#리스트로 변환하기** places = real_data.split(',') print(places) **#알파벳 순으로 정렬하기** places.sort() print(places[0]) #정답 확인 : city hall
더 나아가기 Mission1	start=file_b.find('start') end=file_b.find('end') print(start, end) hint_data = file_b[109+len('start'):] print(hint_data)
더 나아가기 Mission2	reverse_data = hint_data[99:3:-1] print(reverse_data)
더 나아가기 Mission3	places.sort(reverse = True) print(places[0])