Python 데이터타입

데이터 타입(data type)

- 데이터를 식별하는 분류
- 데이터가 가질 수 있는 값을 결정하고 연산을 제어함

타입	설명	예
int	소숫점을 갖지 않는 정수를 갖는 데이타 타입	10
float	소숫점을 갖는 데이타 타입	10.5
bool	True 혹은 False 만을 갖는 타입	True
str	단일인용부호(') 혹은 이중인용부호(") 를 사용하여 표현하는 문자, 단어 등으로 구성된 문자들의 집합	'python'

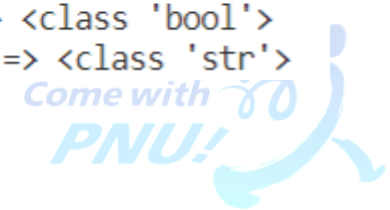
```
x = 10
print(f"{x}의 데이터 타입 => {type(x)}")

x = 10.5
print(f"{x}의 데이터 타입 => {type(x)}")

x = True
print(f"{x}의 데이터 타입 => {type(x)}")

x = 'python'
print(f"{x}의 데이터 타입 => {type(x)}")
```

```
10의 데이터 타입 => <class 'int'>
10.5의 데이터 타입 => <class 'float'>
True의 데이터 타입 => <class 'bool'>
python의 데이터 타입 => <class 'str'>
```



연산자

- 자료를 처리하기 위한 수식들을 결합하여 연산 동작을 수행하도록 하는 기호
- 산술연산자
  - 수학적 계산을 수행할 때 사용하는 연산자
  - 사칙연산자(+, -, *, /), 제곱(**), 나머지(%), 나누기 결과 정수(//)
- 비교연산자(관계연산자)
  - 비교할 때 사용하는 연산자
  - 등호(==), 같지 않음(!=), 부등호(<, >, <=, >=)
- 논리연산자
  - 주어진 논리식을 판단하여, 참(true)과 거짓(false)을 결정하는 연산자
  - and(논리곱), or(논리합), not
- 할당연산자
  - 변수에 값을 할당하기 위하여 사용하는 연산자
  - = (Equal Sign)
  - +=, -=, *=, /=, %=, //=
  - 산술연산자와 함께 사용되어 할당을 보다 간결히 하기 위해 사용

연산자

```
1   #할당 연산자
2   x = 10
3   y = 2.5
4
5   #산술연산자
6   print(f'{x} + {y} = {x+y}')
7
8   #산술연산자와 함께 사용한 할당연산자
9   x = x + 10
10  x += 10
11  print(f'{x}')
12
13  #비교연산자
14  print(f'{x} > {y} = {x > y}')
15
16  #논리연산자
17  print(f'{x} > 5 or {y} > 5 = { x > 5 or y > 5}')
```

```
10 + 2.5 = 12.5
30
30 > 2.5 = True
30 > 5 or 2.5 > 5 = True
```

문자열 연산

- 문자열 더하기(+)
- 문자열 곱하기(*)
- 문자열 인덱싱
  - 문자열의 각 문자는 0부터 시작되는 인덱스를 가짐
  - [인덱스]를 이용하여 인덱스위치의 문자 추출
    - 처음위치 [0], 마지막위치 [-1]
- 문자열 슬라이싱
  - [인덱스1:인덱스2]
    - 인덱스1 위치에서 인덱스2 -1 위치까지 잘라냄
  - [인덱스1:]
    - 인덱스1 위치에서 끝까지 잘라냄
  - [: 인덱스2]
    - 처음부터 인덱스2 -1 위치까지 잘라냄

문자열 함수

- len() : 문자열 길이 구하기
- count() : 문자의 개수 구하기
- find() : 문자가 처음 나오는 위치 반환
  - 없으면 -1
- index() : 문자가 처음 나오는 위치 반환
  - 없으면 오류
- join() : 문자열 사이에 문자 삽입
  - '삽입문자'.join(문자열)
- upper() :소문자를 대문자로 변환
- lower() : 대문자를 소문자로 변환
- replace() : 문자열 바꾸기
  - replace(원본, 변환문자)
- split() : 문자열 나누기
  - 인수가 없으면 공백(스페이스, 탭, 엔터 등)을 기준

문자열 함수

```python
#문자열 함수
x = "Hello Python!"

#len() : 문자열 길이 구하기
print(f'문자열 길이 => {len(x)}')

#count() : 문자의 개수 구하기
print(f'o의 개수  => {x.count("o")}')
print(f'on의 개수  => {x.count("on")}')

#find() : 문자가 처음 나오는 위치 반환
print(f'p 시작위치  => {x.find("p")}')

#index() : 문자가 처음 나오는 위치 반환
print(f'python 시작위치  => {x.index("Python")}')

#join() : 문자열 사이에 문자 삽입
z = '-'.join(x)
print(f'{z}')

#upper() :소문자를 대문자로 변환
print(f' {x} 소문자변환  => {x.upper()}')

#lower() : 대문자를 소문자로 변환
print(f' {x} 대문자변환  => {x.lower()}')

#replace() : 문자열 바꾸기
print(f' {x}  => {x.replace("P", "★")}')

#split() : 문자열 나누기
print(f' {z}  => {z.split("-")}')
```

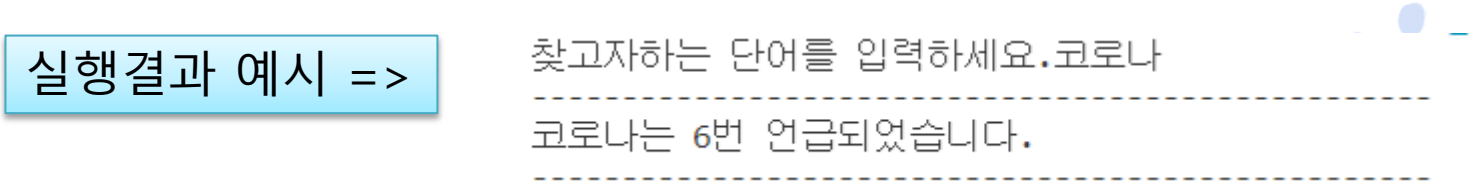
```
문자열 길이 => 13
o의 개수  => 2
on의 개수  => 1
p 시작위치  => -1
python 시작위치  => 6
H-e-l-l-o- -P-y-t-h-o-n-!
Hello Python! 소문자변환  => HELLO PYTHON!
Hello Python! 대문자변환  => hello python!
Hello Python!  => Hello ★ython!
H-e-l-l-o- -P-y-t-h-o-n-!  => ['H', 'e', 'l', 'l', 'o', ' ', 'P', 'y', 't', 'h', 'o', 'n', '!']
```

해결문제

- **다음은 코로나바이러스에 대한 설명이다. 찾고자 하는 단어를 입력 받아서 해당 단어가 몇 번 언급되었는지 확인해 보세요.**

코로나(corona)는 라틴말로 왕관을 뜻하며 통상 태양을 둘러싼 외곽의 빛(광환)을 지칭한다. 코로나바이러스라는 이름은 전자현미경으로 이 바이러스를 관찰했을 때 마치 코로나와 유사한 모양을 띠어 붙여진 이름이다. 이번에 중국의 우환에서 시작된 코로나바이러스는 2019년에 발견된 새로운 코로나바이러스라는 뜻으로 2019-nCoV로 붙여졌다. nCoV는 novel(새로운) CoV(코로나바이러스)라는 뜻이다.

실행결과 예시 =>



해결문제2

- **다음은 코로나바이러스에 대한 설명이다. 몇 개의 문장으로 구성 되어 있는지 예시와 같이 출력하시오.**

코로나(corona)는 라틴말로 왕관을 뜻하며 통상 태양을 둘러싼 외곽의 빛(광환)을 지칭한다. 코로나바이러스라는 이름은 전자현미경으로 이 바이러스를 관찰했을 때 마치 코로나와 유사한 모양을 띠어 붙여진 이름이다. 이번에 중국의 우환에서 시작된 코로나바이러스는 2019년에 발견된 새로운 코로나바이러스라는 뜻으로 2019-nCoV로 붙여졌다. nCoV는 novel(새로운) CoV(코로나바이러스)라는 뜻이다.

실행결과 예시 =>

```
--------------------------------------------------
4개의 문장으로 이루어져있습니다.
['코로나(corona)는 라틴말로 왕관을 뜻하며 통상 태양을 둘러싼 외곽의 빛(광환)을 지칭한다',
' 코로나바이러스라는 이름은 전자현미경으로 이 바이러스를 관찰했을 때 마치 코로나와 유사한
모양을 띠어 붙여진 이름이다', ' 이번에 중국의 우환에서 시작된 코로나바이러스는 2019년에 발
견된 새로운 코로나바이러스라는 뜻으로 2019-nCoV로 붙여졌다', ' nCoV는 novel(새로운) CoV(코
로나바이러스)라는 뜻이다', ' ']
--------------------------------------------------
```