4주차: 리스트, 딕셔너리와 관련된 기본 함수

함수에 대해

함수

- 함수의 매개변수로 적절한 값을 넣어주면 적절한 값을 반환함
- 어떤 값인지는 함수에 따라 다름

함수의 사용

- `함수이름(매개변수1, 매개변수2, ...)`
- 매개변수의 개수와 형태는 함수마다 다름

매개변수

- 함수를 사용할 때 소괄호 안에 넣어 전달하는 변수 혹은 상수

반환

- 함수에 따라 반환되는 값이 다르므로 유의

4-1. 리스트에 적용할 수 있는 기본 함수

`min()`

- 리스트 내부에서 최솟값을 탐색

`max()`

- 리스트 내부에서 최댓값을 탐색

`sum()`

- 리스트 내부의 값을 모두 더함

```
numbers = [103, 41, 273, 32, 77]
print(min(numbers))
```

```
print(max(numebrs))
print(sum(numbers))
```

reversed()

- 리스트 요소의 순서를 뒤집기

```
list_a = [1, 2, 3, 4, 5]
list_reversed = reversed(list_a)

print(list_a)
print(list_reversed)

# 반복하여 출력해 보기
for element in list_reversed:
    print(element)
```

```
# 다음과 같은 코드는?
list_reversed = reversed([1, 2, 3, 4, 5])

# 반복하여 출력해 보기
for element in list_reversed:
    print(element)
# 두 번째 반복문은 실행이 안 됨
for element in list_reversed:
    print(element)
```

- `reversed()` 의 결과는 리스트가 아니다.
- `reversed()` 는 iterator를 리턴하는 generator

enumerate()

- 리스트의 요소를 반복할 때, 현재 인덱스가 몇 번째인지 확인해야 할 경우 편리하게 사용

```
# 변수 선언
example_list = ["요소A", "요소B", "요소C"]

# 출력
```

```
print("# 단순 출력")
print(example_list)
print()

# enumerate()를 적용하여 출력
print("# enumerate() 함수 적용 출력")
print(enumerate(example_list))
print()

# list() 함수로 리스트로 변환하여 출력
print("# list() 함수로 변환 출력")
print(list(enumerate(example_list)))
print()

# 반복문과 조합하기
print("# 반복문과 조합하기")
for i, value in enumerate(example_list):
    print(i, "번째 요소는", value, "입니다.")
```

4-2. 딕셔너리

items() 함수와 반복문 조합하기

```
# 변수 선언
example_dictionary = {
    "키A": "값A",
    "키B": "값B",
    "키C": "값C"
}

# 딕셔너리의 items() 함수 결과 출력하기
print("# 딕셔너리의 items() 함수와 반복문 조합하기")

for key, element in example_dictionary.items():
    print("dictionary[", key, "] = ", element, sep="")
```

4-3. 리스트 내포

반복문을 사용해 리스트를 생성할 때

```
# range(0, 20, 2)의 리스트를 만들어 나가는 과정

# 리스트 변수를 선언
array = []

# 반복문을 적용
for i in range(0, 20, 2):
    array.append(i)

print(array)
```

- `리스트 이름 = [표현식 for 반복자 in 반복할 수 있는 것]`

```
# 같은 코드, 리스트 내포

# 리스트 변수를 선언
array = [i for i in range(0, 20, 2)]

print(array)
```

- `리스트 이름 = [표현식 for 반복자 in 반복할 수 있는 것 if 조건문]`

```
# 리스트 선언
array = ["사과", "자두", "초콜릿", "바나나", "체리"]
output = [fruit for fruit in array if fruit != "초콜릿"]

# 출력
print(output)
```

리스트 내포를 활용한 2차원 리스트 선언

```
# 10×5(세로x가로) 크기의 2차원 리스트 선언
array = [[0 for j in range(5)] for i in range(10)]
```

이터레이터(iterator, 반복자)

iterable

- `for {반복자} in {반복할 수 있는 것}`
- 여기서 '반복할 수 있는 것'을 이터러블(iterable)이라고 함
- 즉, 내부에 있는 요소들을 차례차례 꺼낼 수 있는 객체

사용하는 이유

- 공간 및 시간의 효율성
    - `reversed()` 를 예로, 리스트를 복제하고 뒤집어 새로운 리스트를 만드는 것 보다 기존의 리스트를 반대로 탐색하는 것이 더 빠름