BIND SOFT

문자열, 리스트, 딕셔너리와 관련된 기본 함수

BIND SOFT ㈜바인드소프트

BIND SOFT

목차

시작하기 전에

[핵심 키워드] : enumerate(), items(), 리스트 내포

[핵심 포인트]

반복문과 관련된 파이썬만의 기능들에 대해 알아본다.

시작하기 전에

- 파이썬만의 고유한 기능들

  리스트에 적용할 수 있는 기본 함수 : min(), max(), sum()

  리스트 뒤집기 : reversed()

  현재 인덱스가 몇 번째인지 확인하기 : enumerate()

  딕셔너리로 쉽게 반복문 작성하기 : items()

  리스트 안에 for문 사용하기 : 리스트 내포

리스트에 적용할 수 있는 기본 함수

- min(), max(), sum() 함수
  - 리스트를 매개변수로 넣어 사용하는 기본 함수들

함수	설명
min( )	리스트 내부에서 최솟값을 찾습니다.
max( )	리스트 내부에서 최댓값을 찾습니다.
sum( )	리스트 내부에서 값을 모두 더합니다.

```
>>> numbers = [103, 52, 273, 32, 77]
>>> min(numbers)          ⟶ 리스트 내부에서 최솟값을 찾습니다.
32
>>> max(numbers)          ⟶ 리스트 내부에서 최댓값을 찾습니다.
273
>>> sum(numbers)          ⟶ 리스트 내부에서 값을 모두 더합니다.
537
```

reversed() 함수로 리스트 뒤집기

- reversed() 함수
  - 리스트에서 요소 순서 뒤집기

```
01  # 리스트를 선언하고 뒤집습니다.
02  list_a = [1, 2, 3, 4, 5]
03  list_reversed = reversed(list_a)
04
05  # 출력합니다.
06  print("# reversed() 함수")
07  print("reversed([1, 2, 3, 4, 5]):", list_reversed)
08  print("list(reversed([1, 2, 3, 4, 5])):", list(list_reversed))
09  print()
10
11  # 반복문을 적용해 봅니다.
12  print("# reversed() 함수와 반복문")
13  print("for i in reversed([1, 2, 3, 4, 5]):")
14  for i in reversed(list_a):
15      print("-", i)
```

실행결과

```
# reversed() 함수
reversed([1, 2, 3, 4, 5]): <list_reverseiterator object at 0x031F21D0>
list(reversed([1, 2, 3, 4, 5])): [5, 4, 3, 2, 1]

# reversed() 함수와 반복문
for i in reversed([1, 2, 3, 4, 5]):
- 5
- 4
- 3
- 2
- 1
```

(좀 더 알아보기: 이터레이터에서 자세히 설명합니다.(204쪽))

reversed() 함수로 리스트 뒤집기

- reversed() 함수와 반복문 조합할 때는 함수 결과 여러 번 활용하지 않고 for 구문 내부에 reversed() 함수 곧바로 넣어서 사용
  - 잘못된 예

```
temp = reversed([1, 2, 3, 4, 5, 6])

for i in temp:
    print("첫 번째 반복문: {}".format(i))

for i in temp:
    print("두 번째 반복문: {}".format(i))
```

```
첫 번째 반복문: 6
첫 번째 반복문: 5
첫 번째 반복문: 4
첫 번째 반복문: 3
첫 번째 반복문: 2
첫 번째 반복문: 1
```

reversed() 함수로 리스트 뒤집기

- 바른 예

```
numbers = [1, 2, 3, 4, 5, 6]

for i in reversed(numbers):
    print("첫 번째 반복문: {}".format(i))

for i in reversed(numbers):
    print("두 번째 반복문: {}".format(i))
```

→ 필요한 시점에 reversed() 함수를 사용합니다.

enumerate() 함수와 반복문 조합하기

- enumerate() 함수
  - 리스트 요소 반복할 때 현재 인덱스가 몇 번째인지 확인
  - 예시

```
example_list = ["요소A", "요소B", "요소C"]
```

```
0번째 요소는 요소A입니다.
1번째 요소는 요소B입니다.
2번째 요소는 요소C입니다.
```

enumerate() 함수와 반복문 조합하기

- 방법 1

```
example_list = ["요소A", "요소B", "요소C"]
i = 0
for item in example_list:
    print("{}번째 요소는 {}입니다.".format(i, item))
    i += 1
```

- 방법 2

```
example_list = ["요소A", "요소B", "요소C"]
for i in range(len(example_list)):
    print("{}번째 요소는 {}입니다.".format(i, example_list[i]))
```

enumerate() 함수와 반복문 조합하기

- 예시 – enumerate() 함수와 리스트

```
01  # 변수를 선언합니다.
02  example_list = ["요소A", "요소B", "요소C"]
03
04  # 그냥 출력합니다.
05  print("# 단순 출력")
06  print(example_list)
07  print()
08
09  # enumerate() 함수를 적용해 출력합니다.
10  print("# enumberate() 함수 적용 출력")
11  print(enumerate(example_list))
12  print()
13
14  # list() 함수로 강제 변환해 출력합니다.
15  print("# list() 함수로 강제 변환 출력")
16  print(list(enumerate(example_list)))
```

enumerate() 함수와 반복문 조합하기

```
17  print()
18
19  # for 반복문과 enumerate() 함수 조합해서 사용하기
20  print("# 반복문과 조합하기")
21  for i, value in enumerate(example_list):
22      print("{}번째 요소는 {}입니다.".format(i, value))
```

enumerate() 함수를 사용하면 반복 변수를 이런 형태로 넣을 수 있습니다.

실행결과

```
# 단순 출력
['요소A', '요소B', '요소C']

# enumberate() 함수 적용 출력
<enumerate object at 0x02A43CB0>

# list() 함수로 강제 변환 출력
[(0, '요소A'), (1, '요소B'), (2, '요소C')]

# 반복문과 조합하기
0번째 요소는 요소A입니다.
1번째 요소는 요소B입니다.
2번째 요소는 요소C입니다.
```

딕셔너리의 items() 함수와 반복문 조합하기

- 딕셔너리와 items() 함수 함께 사용하면 키와 값을 조합하여 쉽게 반복문 작성할 수 있음

```
01  # 변수를 선언합니다.
02  example_dictionary = {
03      "키A": "값A",
04      "키B": "값B",
05      "키C": "값C",
06  }
07
08  # 딕셔너리의 items() 함수 결과 출력하기
09  print("# 딕셔너리의 items() 함수")
10  print("items():", example_dictionary.items())
11  print()
12
13  # for 반복문과 items() 함수 조합해서 사용하기
14  print("# 딕셔너리의 items() 함수와 반복문 조합하기")
15
16  for key, element  in example_dictionary.items():
17      print("dictionary[{}] = {}".format(key, element))
```

실행결과

```
# 딕셔너리의 items() 함수
items(): dict_items([('키A', '값A'), ('키B', '값B'), ('키C', '값C')])

# 딕셔너리의 items() 함수와 반복문 조합하기
dictionary[키A] = 값A
dictionary[키B] = 값B
dictionary[키C] = 값C
```

리스트 내포

- 반복문 사용하여 리스트 재조합하는 경우
  - 예시 – 반복문을 사용한 리스트 생성

```
01  # 변수를 선언합니다.
02  array = []
03
04  # 반복문을 적용합니다.
05  for i in range(0, 20, 2):
06      array.append(i * i)
07  # 출력합니다.
08  print(array)
```

실행결과

```
[0, 4, 16, 36, 64, 100, 144, 196, 256, 324]
```

리스트 내포

- 예시 – 리스트 안에 for문 사용하기

```
01  # 리스트를 선언합니다.
02  array = [i * i for i in range(0, 20, 2)]
03                  최종 결과를 앞에 작성합니다.
04  # 출력합니다.
05  print(array)
```

- 리스트 내포 (list comprehension)

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

리스트 내포

- 예시 – 조건을 활용한 리스트 내포

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

```
01  # 리스트를 선언합니다.
02  array = ["사과", "자두", "초콜릿", "바나나", "체리"]
03  output = [fruit for fruit in array if fruit != "초콜릿"]
04
05  # 출력합니다.
06  print(output)
```

실행결과

```
['사과', '자두', '바나나', '체리']
```

키워드로 정리하는 핵심 포인트

- **enumerate() 함수** : 리스트를 매개변수로 넣을 경우 인덱스와 값을 쌍으로 사용해 반복문을 돌릴 수 있게 하는 함수

- **items() 함수** : 키와 쌍으로 사용해 반복문을 돌릴 수 있게 하는 딕셔너리 함수

- **리스트 내포** : 반복문과 조건문을 대괄호 안에 넣는 형태를 사용하여 리스트 생성하는 파이썬의 특수 구문. list comprehensions 기억할 것

확인문제

- 다음 중 enumerate() 함수와 items() 함수의 사용법으로 올바른 것은?
  - 리스트.enumerate()
  - enumerate(리스트)
  - 딕셔너리.items()
  - items(딕셔너리)

- 2진수, 8진수, 16진수로 변환하는 코드는 많이 사용됩니다.
  다음과 같은 형태로 10진수를 변환할 수 있습니다.

10진수와 2진수 변환

```
>>> "{:b}".format(10)
'1010' ---> 변환했을 때 따옴표로 둘러싸여 있다면 문자열 자료형입니다.
>>> int("1010", 2)
10
```

확인문제

10진수와 8진수 변환

```
>>> "{:o}".format(10)
'12'
>>> int("12", 8)
10
```

10진수와 16진수 변환

```
>>> "{:x}".format(10)
'a'
>>> int("10", 16)
16
```

```
>>> "안녕안녕하세요".count("안") ⟶ 문자열을 매개변수로 넣어야 합니다.
2
```

확인문제

- 이를 활용해서 1~100 사이에 있는 숫자 중 2진수로 변환했을 때 0이 하나만 포함된 숫자를 찾고, 그 숫자들의 합을 구하는 코드를 만들어보세요.

```
# 리스트 내포를 사용해본 코드입니다.
output =


for i in        :
    print("{} : {}".format(i, "{:b}".format(i)))
print("합계:", sum(output))
```

실행결과

```
2 : 10
5 : 101
6 : 110
11 : 1011
...
62 : 111110
95 : 1011111
합계: 539
```