

파이선으로 배우는 데이터구조

Data Structures Learning with Python

김영훈

한양대학교 ERICA 인공지능학과


기본 반복문 프로그래밍 패턴

2

- ▶ Pattern 1. 새 List / Dictionary 만들기
- ▶ Pattern 2. 세기 / 합산하기
- ▶ Pattern 3. 최대 / 최소찾기
- ▶ Pattern 4. 어떤 / 모든 조건 확인하기
- ▶ Pattern 5. 사전 카운팅

Pattern 1. 새 List / Dictionary 만들기

- ▶ 단어들의 목록에서 길이가 n보다 긴 단어들로 구성된 리스트를 반환하는 함수를 작성하십시오.
- ▶ 입력: n=3, ["The", "quick", "brown", "fox"], 출력: ["quick", "brown"]

```
def long_words(n, ls):  
    word_len = []  
      
    return word_len  
print(long_words(3, ["The", "quick", "brown", "fox"]))
```

Pattern 1. 새 List / Dictionary 만들기

- ▶ 주어진 dictionary에서 중복된 정보를 제거하는 코드를 작성하시오.
- ▶ 입력 : {'id1': {'name': ['Sara'], 'score': [90]},
'id2': {'name': ['David'], 'score': [70]},
'id3': {'name': ['Sara'], 'score': [90]}}

```
student_data = {  
    'id1': {'name': ['Sara'], 'score': [90]},  
    'id2': {'name': ['David'], 'score': [70]},  
    'id3': {'name': ['Sara'], 'score': [90]}  
}  
result = {}
```

?

```
print(result)
```

Pattern 2. 세기 / 합산하기

- ▶ 서브목록들의 목록에서, 각 서브목록의 첫 번째 원소들의 합을 출력하는 코드를 작성하십시오.
- ▶ 입력: `[[1, 2], [2, 3, 4], [8, 4], [10, 11, 1, 2], [3, 10]]`
- ▶ 출력: 24

```
lst = [[1, 2], [2, 3, 4], [8, 4], [10, 11, 1, 2], [3, 10]]  
first_elems = 0
```

?

```
print(sum(first_elems))
```

Pattern 2. 세기 / 합산하기

- ▶ 서브목록들의 목록에서, 각 서브목록의 마지막 원소들 중, 값이 3보다 큰 원소들의 개수를 을 출력하는 코드를 작성하십시오.
- ▶ 입력: `[[1, 2], [2, 3, 4], [8, 4], [10, 11, 1, 2], [3, 10]]`
- ▶ 출력: `[0, 1, 2, 2, 1]`

```
lst = [[1, 2], [2, 3, 4], [8, 4], [10, 11, 1, 2], [3, 10]]  
res = []
```

?

```
print(res)
```

Pattern 2. 세기 / 합산하기

- ▶ 주어진 dictionary의 모든 값의 곱을 구하는 코드를 작성하시오.
- ▶ 입력 : {'data1':100,'data2':-54,'data3':247}
- ▶ 출력 : -1333800

```
my_dict = {'data1':100,'data2':-54,'data3':247}
```

```
result=1
```

```
?
```

```
print(result)
```

Pattern 3. 최대 / 최소찾기

- ▶ 주어진 dictionary에서 최대, 최소 값을 출력하는 코드를 작성하시오.
- ▶ 입력 : {"m1":78 , "m2":89 , "m3":64 , "m4":35 , "m5":71}
- ▶ 출력 : "Max" = 89, "Min" = 35

```
marks={"m1":78 , "m2":89 , "m3":64 , "m4":35 , "m5":71}  
maxi = marks.values()[0]  
mini = marks.values()[0]
```

?

```
print("Max =", maxi, "Min =", mini)
```


Pattern 4. 어떤(any) / 모든(all) 조건 확인하기

- ▶ 두 목록이 주어졌을 때, 공통인자가 있으면 True, 없으면 False를 반환하는 함수를 작성하십시오.

```
def common_data(list1, list2):
```

```
    ?
```

```
    return result
```

```
print(common_data([1,2,3,4,5], [5,6,7,8,9]))
```

```
print(common_data([1,2,3,4,5], [6,7,8,9]))
```

Pattern 4. 어떤(any) / 모든(all) 조건 확인하기

- ▶ 목록들의 목록에서, 일정 범위의 값만 포함하는 목록들의 목록을 계산하는 함수를 작성하십시오.
- ▶ 출력: [[13, 14, 15, 17]]

```
list1 = [ [2], [10, 20], [3, 4, 5, 6] ]
```

```
def remove_list_range(lst, left_range, right_range):  
    res = []
```

?

```
    return res
```

```
print(remove_list_range(list1, 1, 5))
```

Pattern 5. 사전 카운팅

- ▶ 주어진 문자열에 각 문자의 출현 횟수를 세는 프로그램을 작성하시오. (힌트: for와 dictionary를 사용)
- ▶ 샘플 문자열: str1 = 'The quick brown fox jumps over the lazy dog.'
- ▶ {'T': 1, 'h': 2, 'e': 3, ' ': 8, 'q': 1, 'u': 2, 'i': 1, 'c': 1, 'k': 1, 'b': 1, 'r': 2, 'o': 4, 'w': 1, 'n': 1, 'f': 1, 'x': 1, 'j': 1, 'm': 1, 'p': 1, 's': 1, 'v': 1, 't': 1, 'l': 1, 'a': 1, 'z': 1, 'y': 1, 'd': 1, 'g': 1, '.': 1}

```
str1 = 'The quick brown fox jumps over the lazy dog.'
```

```
cnt = {}
```

?

```
print(cnt)
```

파이선 데이터구조 필수함수

꼭 외워야 할 필수 함수

데이터구조	함수	설명
List	s.append(x)	s의 끝에 요소 x를 추가합니다.
	s.extend(x)	리스트 x를 s에 추가합니다.
	zip(a, b)	리스트 a와 b에서 순서대로 동일 인덱스의 원소의 쌍을 가져옵니다.
Dictionary	s.items()	딕셔너리 s의 각 키-값 쌍에 대한 튜플을 포함하는 리스트를 반환합니다.
	s.keys()	딕셔너리 s의 모든 키의 리스트를 반환합니다.
	s.values()	딕셔너리 s의 모든 값의 리스트를 반환합니다.
String	s.upper() / s.lower()	문자열 s를 대문자 / 소문자로 바꾼 문자열을 반환합니다.
	s.split([separator])	문자열 s를 공백 단위로 잘라줍니다. 이때 separator를 지정해주면 공백대신 이 문자열을 이용해 자릅니다.
	s.join(x)	리스트 x의 문자열 들을 연결해줍니다. 문자열 들 사이에 이때 s를 끼워서 연결 합니다.
	s[a:b:c]	문자열 s의 부분문자열($a \leq$ 인덱스 $< b$)을 반환합니다. c를 지정하면 c-1개씩 건너뛰며 부분문자열을 생성합니다.

zip()

- ▶ 두개의 리스트 중 l1을 key로 l2를 value로 하는 dictionary를 만드는 코드를 작성하시오.
- ▶ 입력 : l1 = ['a', 'b', 'c', 'd', 'e', 'f'], l2 = [1, 2, 3, 4, 5]
- ▶ 출력 : {'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5}

```
def test(keys, values):
```

```
    ?
```

```
l1 = ['a', 'b', 'c', 'd', 'e', 'f']
```

```
l2 = [1, 2, 3, 4, 5]
```

```
print(test(l1, l2))
```

sort()

- ▶ 주어진 dictionary를 key의 알파벳 역순으로 정렬하시오.
- ▶ 입력 : { ' banana': ' yellow', ' kiwi': ' green', ' apple': 'red', 'grape': 'purple'}
- ▶ 출력 : {'kiwi': 4, 'grape': 3, 'banana': 1, 'apple': 2}

```
d = { 'banana': 1, 'apple': 2, 'grape': 3, 'kiwi': 4 }  
res = {}  
keys = list(d.keys())
```

?

```
print(res)
```

split() / join()

- ▶ 주어진 문자열을 단어들 중 길이가 5이상인 단어들만 대문자로 변환한 문자열을 출력하시오.
- ▶ 샘플 문자열 : "The-quick-brown-fox-jumps-over-the-lazy-dog."
- ▶ 예상 출력 : The-QUICK-BROWN-fox-JUMPS-over-the-lazy-dog.

```
str1 = 'The-quick-brown-fox-jumps-over-the-lazy-dog.'
```

```
res = []
```

```
?
```

```
print('-'.join(res))
```