# 

23차시

내장 함수 zip()과 enumerate(), 시퀀스 간의 변환



## ⚠ 학습개요

- ··· 내장 함수 zip()
- ··· 내장 함수 enumerate()
- … 리스트와 튜플 간의 변환
- … 리스트와 집합 간의 변환
- … 딕셔너리를 리스트, 튜플, 집합으로 변환
- ··· 프로젝트 lab1, lab2

## ⚠ 학습목표

- ··· 내장 함수 zip()을 사용해 각각의 항목으로 구성된 튜플 항목을 생성할 수 있다.
- ··· 내장 함수 enumerate()을 사용해 수의 나열과 항목을 생성할 수 있다.
- … 리스트와 튜플 간의 변환과 리스트와 집합 간의 변환을 수행할 수 있다.
- … 가위바위보 게임을 구현할 수 있다.
- ··· K-pop 차트를 딕셔너리로 구현할 수 있다.

Chapter 1.

## 내장 함수 zip()

#### ■ 파이썬 프로그래밍 내장 함수 zip()과 enumerate(), 시퀀스 간의 변환

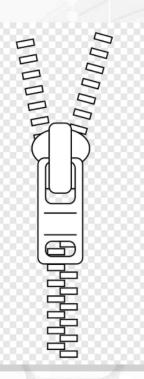


## ① 내장 함수 zip()

#### + 몇 개의 리스트나 튜플의 항목으로 조합된 튜플을 생성

- 동일한 수로 이뤄진 여러 개의 튜플 항목 시퀀스를 각각의 리스트로 묶어 주는 역할을 하는 함수
- 함수 zip()의 결과는 자료형 zip
- 자료형 zip은 간단히 리스트나 튜플로 변환

```
>>> a = ['FTP', 'telnet', 'SMTP', 'DNS']
>>> b = (20, 23, 25, 53)
>>> z = zip(a, b)
>>> type(z)
<clss 'zip'>
>>> list(zip(a, b))
[('FTP', 20), ('telnet', 23), ('SMTP', 25), ('DNS', 53)]
```





## ⚠ 내장 함수 zip()

+ 몇 개의 리스트나 튜플의 항목으로 조합된 튜플을 생성

```
>>> list(zip(a, b))
[('FTP', 20), ('telnet', 23), ('SMTP', 25), ('DNS', 53)]
>>> tuple(zip(a, b))
(('FTP', 20), ('telnet', 23), ('SMTP', 25), ('DNS', 53))
>>> list(zip('ABCD', a, b))
[('A','FTP', 20), ('B', 'telnet', 23), ('C', 'SMTP', 25), ('D', 'DNS', 53)]
>>> tuple(zip('abcd', 'XY'))
(('a', 'X'), ('b', 'Y'))
```



## ⚠ 2개의 리스트나 튜플로 키-값 항목인 딕셔너리를 생성

```
>>> a = ['FTP', 'telnet', 'SMTP', 'DNS']
>>> b = (20, 23, 25, 53)
>>> dict(zip(a, b))
{'FTP': 20, 'telnet': 23, 'SMTP': 25, 'DNS': 53}
>>> dict(zip('ABCD', a))
{'A': 'FTP', 'B': 'telnet', 'C': 'SMTP', 'D': 'DNS'}
>>> dict(zip('abcd', 'XY'))
{'a': 'X', 'b': 'Y'}
>>> dict(zip(a, b, b))
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: dictionary update sequence element # 0 has length 3; 2 is required
```

Chapter 2.

## 내장 함수 enumerate()



### ⚠ 내장 함수 enumerate()

+ 0부터 시작하는 첨자와 항목 값의 튜플 리스트를 생성

```
>>> lst = [10, 20, 30]
>>> list(enumerate(lst))
[(0, 10), (1, 20), (2, 30)]
>>> list(enumerate([10, 20, 30], start = 1))
[(1, 10), (2, 20), (3, 30)]
>>> lst = 'python'
>>> list(enumerate(lst))
[(0, 'p'), (1, 'y'), (2, 't'), (3, 'h'), (4, 'o'), (5, 'n')]
```



### ⚠ 내장 함수 enumerate()

+ 0부터 시작하는 첨자와 항목 값의 튜플 리스트를 생성

```
>>> subj = ['국어', '영어', '수학']
>>> for tp in enumerate(subj):
... print('lst[{}]: {}'.format(tp[0], tp[1]))
... print('lst[{}]: {}'.format(*tp))
...
lst[0]: 국어
lst[0]: 국어
lst[1]: 영어
lst[1]: 영어
lst[2]: 수학
lst[2]: 수학
```

```
>>> subj = ['국어', '영어', '수학']
>>> for i, name in enumerate(subj):
... print('lst[{}]: {}'.format(i,name))
...
lst[0]: 국어
lst[1]: 영어
lst[2]: 수학
```

Chapter 3.

## 리스트와 튜플 간의 변환



## ⚠ 튜플과 시퀀스 간의 변환

```
>>> space = '밤', '낮', '해', '달'
>>> print(space)
('밤', '낮', '해', '달')
>>> print(list(space))
['밤', '낮', '해', '달']

>>> singer = ['BTS', '볼빨간사춘기', 'BTS', '블랙핑크']
>>> print(singer)
['BTS', '볼빨간사춘기', 'BTS', '블랙핑크']
>>> print(tuple(singer))
('BTS', '볼빨간사춘기', 'BTS', '블랙핑크')
```

Chapter 4.

## 리스트와 집합 간의 변환



## ① 리스트와 집합 간의 변환

```
>>> singer = ['BTS', '볼빨간사춘기', 'BTS', '블랙핑크']
>>> print(singer)
['BTS', '볼빨간사춘기', 'BTS', '블랙핑크']
>>> print(set(singer))
{'블랙핑크', 'BTS', '볼빨간사춘기'}
```

Chapter 5.

## 디셔너리를 리스트, 튜플, 집합으로 변환



## ⚠ 딕셔너리를 다른 시퀀스로 변환하면 항목이 키로만 구성

```
>>> game = dict(일월='소나무', 이월='매화', 삼월='벚꽃', 사월='등나무')
>>> lgame = list(game)
>>> print(lgame)
['일월', '이월', '삼월', '사월']
>>> print(tuple(game))
('일월', '이월', '삼월', '사월')
>>> print(set(game))
{'삼월', '이월', '사월', '일월'}
```



### ⚠ 일상 코딩: 구기 종목의 팀원 수를 딕셔너리로 만들기

#### [코딩실습] 구기 종목과 팀원 수의 리스트에서 딕셔너리 구성

난이도응용

```
1 # 구기 종목 리스트
2 sports = ['축구', '야구', '농구', '배구']
3 # 위 종목에 대응하는 팀원 수를 항목으로 구성
4 num = [11, 9, 5, 6]
5 print(sports)
6 print(num)
7 print()
8
```



### ② 일상 코딩: 구기 종목의 팀원 수를 딕셔너리로 만들기

#### [코딩실습] 구기 종목과 팀원 수의 리스트에서 딕셔너리 구성

난이도 응용

Chapter 6.

## 프로젝트 lab1, lab2





#### 철수와 영희의 가위바위보 게임

난이도 실전

딕셔너리와 리스트 튜플 등을 활용해 철수와 영희의 가위바위보 게임을 구현해 보자. 철수와 영희에게 난수를 사용해 가위바위보 중 하나를 선정해 승부를 판정한다.



게임은 10회 연속하며 각각의 승부 내용을 출력한다.

구현하려는 가위바위보 게임에서 매우 유용한 구조가 딕셔너리다.

다음 딕셔너리 dcs는 가위바위보 게임의 승부를 결정(decision)하는

주요 구조로, 키: 값의 쌍을 '가위': '보오'처럼 키(key)로 이기는 패와

값(value)으로 지는 패를 선택한다. 그러므로 항목은 3개가 나온다.

다만 한글 형식화 출력에 정렬 문제가 있어 보를 두 글자인 '보오'로 표현하자.

dcs = {'가위':'보오', '바위':'가위', '보오':'바위'}





#### 철수와 영희의 가위바위보 게임

난이도 실전

두 게임 참여자의 승부를 결정하는 데 있어 이전 딕셔너리는 매우 효과적이다. 철수와 영희가서로 다른 것을 내고 철수가 낸 것을 키로 검색해 'dcs[철수 낸 것] == 영희 낸 것'이라면 철수가승자가 된다. 이와 반대로 조건이 아니면 영희가 승리한 것으로 판정하면 된다. 매우 간단하지만 처음에는 익숙하지 않을 수 있다.

또한 모듈 random의 함수 choice()는 임의로 여러 가지 중 하나를 선택하는 데 활용되는 함수다. 다음과 같이 튜플이나 리스트인 rsp를 인자로 choice(rsp)를 호출하면 임의로 '가위', '바위', '보오' 중의 하나를 선택해 반환한다.

```
rsp = ('가위', '바위', '보오')
# 철수 결정
cs = choice(rsp)
# 영희 결정
yh = choice(rsp)
```





#### 철수와 영희의 가위바위보 게임

난이도 실전

문제 이해 가위바위보게임은 누구나 아는 게임이다. 이 프로그램의 관건은 두 참여자의 선택이고 이 선택에 의한 승부 판정이다. 위에서 설명한 딕셔너리 dcs와 모듈 random의 choice()를 잘 이해하고 구현 하도록 한다.





#### 딕셔너리로 만드는 k-pop차트

난이도 실전

딕셔너리로 가상의 K-pop 차트를 만들어 출력하는 프로그램을 작성해 보자. 우선 가수의 모음인 리스트와 노래의 모음인 리스트를 만든 후, 함수 zip()과 enumerate()를 사용한다. 함수 zip()으로 가수와 노래를 조합하고, 다시 이 조합된 리스트를 함수 enumerate()로 순위를 키로 하는 딕셔너리를 만든다.

```
      {1: ('BTS', '작은 것들을 위한 시'),

      2: ('볼빨간사춘기', '나만 봄'),

      3: ('BTS', '소우주'),

      4: ('블랙핑크', 'Kill This Love'),

      5: ('태연', '사계')}
```





#### 딕셔너리로 만드는 k-pop차트

난이도 실전

```
singer = ['BTS', '볼빨간사춘기', 'BTS', '블랙핑크', '태연']
song = ['작은 것들을 위한 시', '나만 봄', '소우주', 'Kill This Love', '사계']
kpop = list(zip(singer, song))
# dict()와 enumeratezip() 함수로 순위를 키로 가수와 곡을 사전으로 구성
kpchart = dict(enumerate(kpop, start = 1))
```

최종으로 만든 kpchart를 위와 같이 보기 좋게 출력하려면 모듈 pprint의 함수 pprint()를 사용한다.

```
# 모듈 pprint의 pprint() 함수 활용 import pprint pprint(kpchart)
```





#### 딕셔너리로 만드는 k-pop차트

난이도 실전

문제 이해 가수의 리스트 singer와 이에 대응되는 곡의 리스트인 song을 이해하고, 함수 zip()으로 두 리스트의 항목을 결 합한다. 다시 위 결과의 리스트를 값, 순위를 키로 조합하는 딕셔너리 kpchart를 만든다.

- ① 내장 함수 zip()
- ① 내장 함수 enumerate()
- ① 리스트와 튜플 간의 변환
- ① 리스트와 집합 간의 변환
- <u>(1)</u> 딕셔너리를 리스트, 튜플, 집합으로 변환