파이썬 프로그래밍

20차시

키와 값인 쌍의 나열인 딕셔너리

⚠ 학습개요

… 딕셔너리의 이해

… 딕셔너리 생성

⚠ 학습목표

… 키와 값의 목록인 딕셔너리를 이해할 수 있다.

… 딕셔너리를 생성할 수 있다.

… 딕셔너리 키의 제한 조건을 알 수 있다.

Chapter 1.

딕셔너리의 이해

PYTHON PROGRAMMING

⚠ 딕셔너리

✚ 딕셔너리는 말 그대로 '사전'

- 딕셔너리

키와 값의 쌍인 항목을 나열한 시퀀스

⚠ 딕셔너리

+ 콤마로 구분

- 항목(또는 요소, 원소)들의 리스트로 표현

+ 항목은 키: 값, 전체는 중괄호(curly brace) {…} 사용

- 딕셔너리는 중괄호 {…} 사이에 키와 값의 항목을 기술한다.
- 딕셔너리의 항목 순서는 의미가 없으며, 키는 중복될 수 없다.
- 키는 수정될 수 없지만, 값은 수정될 수 있다.
- 값은 키로 참조된다.

⚠ 딕셔너리

✚ 항목은 키: 값, 전체는 중괄호(curly brace) {…} 사용

```
dct = { <key>:<value>, <key>:<value>, ..., <key>:<value> } # 딕셔너리
```

```
groupnumber = {'엑소': 9, '트와이스': 9, '블랙핑크': 4, '방탄소년단': 7}
coffeeprice = {'에스프레소': 2500, '아메리카노': 2800, '카페라떼': 3200}
mycar = {"brand": "현대", "model": "제네시스", "year": 2016}
```

```
>>> mycar = {"brand": "현대", "model": "제네시스", "year": 2016}
>>> print(mycar)
{"brand": "현대", "model": "제네시스", "year": 2016}
>>> print(type(mycar))
<class 'dict'>
```

Chapter 2.

딕셔너리 생성

PYTHON PROGRAMMING

⚠ 빈 딕셔너리

+ 빈 중괄호 {}
+ dict()

```
>>> lect = dict{} #빈 딕셔너리
>>> print(lect)
{}
```

⚠ 일상 코딩: 강좌 정보로 딕셔너리를 생성해 각각의 항목 값 참조

[코딩실습] 강좌 정보로 구성된 딕셔너리 생성과 참조 난이도 기본

```
1  lect = dict() # 빈 딕셔너리
2  lect['강좌명'] = '파이썬 기초';
3  lect['개설년도'] = [2020, 1];
4  lect['학점시수'] = (3, 3);
5  lect['교수'] = '김민국';
6  print(lect)
7  print(len(lect))
8  print()
9  # 딕셔너리의 항목 참조
10 print(lect['개설년도'], lect['학점시수'])
11 print(lect['강좌명'], lect['교수'])
```

주의 3, 4번 줄처럼 딕셔너리의 값은 리스트나 튜플 등 제한이 없다.

⚠ 일상 코딩: 강좌 정보로 딕셔너리를 생성해 각각의 항목 값 참조

[코딩실습] 강좌 정보로 구성된 딕셔너리 생성과 참조 난이도 기본

결과

```
{'강좌명': '파이썬 기초', '개설년도': [2020, 1], '학점시수': (3,
3), '교수': '김민국'}
4

[2020, 1] (3, 3)
파이썬 기초 김민국
```

⚠ 함수 dict()의 사용

+ 함수 dict()

- 일련의 키-값 쌍으로 [키, 값] 리스트 형식과 (키, 값) 튜플 형식을 모두 사용

```
>>> day = dict([['월', 'monday'], ['화', 'tuesday'], ['수', 'wednesday']])
>>> day = dict((['월', 'monday'], ['화', 'tuesday'], ['수', 'wednesday']))
>>> day = dict([('월', 'monday'), ('화', 'tuesday'), ('수', 'wednesday')])
>>> day = dict((('월', 'monday'), ('화', 'tuesday'), ('수', 'wednesday')))
>>> print(day)
{'월': 'monday', '화': 'tuesday', '수': 'wednesday'}
```

⚠ 함수 dict()의 사용

+ 함수 dict()

- 키가 단순 문자열이면 간단히 월 = 'monday'처럼 키 = 값 항목 나열로도 지정 가능

```
>>> day = dict(월 = 'monday', 화 = 'tuesday', 수 = 'wednesday')
>>> print(day)
{'월': 'monday', '화': 'tuesday', '수': 'wednesday'}
```

- 문자열 입력 오류 발생

```
>>> day = dict(월="monday")
>>> day
{'월': 'noday'}
>>> day1 = dict('월'="monday")
SyntaxError: expression cannot contain assignment, perhaps you meant "=="?
>>> day1 = dict(""월"="monday")
SyntaxError: expression cannot contain assignment, perhaps you meant "=="?
>>>
```

⚠ 일상 코딩: 방탄소년단 정보를 저장하는 다양한 딕셔너리 생성 방법

[코딩실습] 방탄소년단 정보를 저장하는 다양한 딕셔너리 생성과 참조 난이도 기본

```
1  bts1 = {'그룹명': '방탄소년단', '인원수': 7, '리더': '김남준'}
2  bts1['소속사'] = '빅히트 엔터테인먼트';
3  print(bts1)
4  bts2 = dict([['그룹명', '방탄소년단'], ['인원수', 7]])
5  print(bts2)
6  bts3 = dict((('리더', '김남준'), ('소속사', '빅히트 엔터테인먼트')))
7  print(bts3)
8
9  bts = dict(그룹명='방탄소년단', 인원수=7, 리더='김남준', 소속사='빅히트 엔터테
10 인먼트')
11 # 구성원 추가
12 bts['구성원'] = ['RM', '진', '슈가', '제이홉', '지민', '뷔', '정국']
13
14 print(bts) # 전체 출력
15 print(bts['구성원']) # 구성원 출력
```

주의 9번 줄에서 키인 그룹명, 인원수 등에는 따옴표를 붙이면 안 된다.

⚠ 일상 코딩: 방탄소년단 정보를 저장하는 다양한 딕셔너리 생성 방법

[코딩실습] 방탄소년단 정보를 저장하는 다양한 딕셔너리 생성과 참조 난이도 기본

결과

```
{'그룹명': '방탄소년단', '인원수': 7, '리더': '김남준', '소속사':
'빅히트 엔터테인먼트'}
{'그룹명': '방탄소년단', '인원수': 7}
{'리더': '김남준', '소속사': '빅히트 엔터테인먼트'}

{'그룹명': '방탄소년단', '인원수': 7, '리더': '김남준', '소속사':
'빅히트 엔터테인먼트', '구성원': ['RM', '진', '슈가', '제이홉', '지민',
'뷔', '정국']}
['RM', '진', '슈가', '제이홉', '지민', '뷔', '정국']
```

방탄소년단의 주요 히트곡을 키 'songs'에 두세 곡 저장해 출력하는 프로그램을 추가해보자.

⚠ 딕셔너리의 키로 정수, 실수 등 사용 가능

✚ 딕셔너리의 키는 수정 불가능한 객체는 모두 가능

- 정수는 물론 실수도 가능

```
1  >>> real = {3.14: '원주율'}
2  >>> real[2.71] = '자연 수'
3  >>> print(real)
4  {3.14: '원주율', 2.71: '자연 수'}
5  >>>
6  >>> real[2.71] = '오일러 수'
7  >>> print(real)
8  {3.14: '원주율', 2.71: '오일러 수'}
9  >>>
10 >>> real[2.72] = '자연 수'
11 >>> print(real)
12 {3.14: '원주율', 2.71: '오일러 수', 2.72: '자연 수'}
13 >>> real[1.61] = '황금비'
14 >>> print(real)
15 {3.14: '원주율', 2.71: '오일러 수', 2.72: '자연 수', 1.61: '황금비'}
```

⚠ 딕셔너리의 키로 정수, 실수 등 사용 가능

+ 새로운 키로 대입

- 새로운 키-값의 항목이 삽입

+ month[0]

- 0은 첨자가 아니라 키를 의미
- month에는 키 0이 없으므로 KeyError가 발생

```
16 >>>
17 >>> print(real[3.14])
18 원주율
19 >>> print(real[2.71])
20 오일러 수
21 21 >>> print(real[1.61])
22 황금비
23 >>> month = {1: 'January', 2: 'February', 3: 'March'}
24 >>>
25 >>> print(month[2])
26 February
27 >>> print(month[0]) # 오류발생
28 Traceback (most recent call last):
29    File "<stdin>", line 1, in <module>
30 KeyError: 0
31 >>>
```

⚠ 일상 코딩: 1월에서 9월의 영어 단어로 구성되는 딕셔너리

[코딩실습] 월 영어 단어 구성과 검색 난이도 응용

```
1  month = {1: 'January', 2: 'February', 3: 'March', 4: 'April'}
2  month[5] = 'May'
3  month[6] = 'June'
4  month[7] = 'July'
5  month[8] = 'August'
6  month[9] = 'September'
7  print(month)
8  print()
9
10 from random import randint
11 # 임의로 5번의 월 단어 출력
12 for i in range(5):
13     r = randint(1, 9)
14     print('%d: %s' % (r, month[r]))
```

주의 2번 줄에서 6번 줄까지 딕셔너리 month에 새로운 5개의 항목을 삽입한다.

⚠ 일상 코딩: 1월에서 9월의 영어 단어로 구성되는 딕셔너리

[코딩실습] 월 영어 단어 구성과 검색 난이도 응용

결과

```
{1: 'January', 2: 'February', 3: 'March', 4: 'April', 5:
'May', 6: 'June', 7: 'July', 8: 'August', 9: 'September'}

4: April
6: June
7: July
9: September
8: August
```

⚠ 딕셔너리의 이해

… 키와 값의 목록

⚠ 딕셔너리 생성

… {}

… 함수 dict()

… 딕셔너리의 키 사용: Immutable 객체만 가능

int, float, str, tuple 등