

Chapter 7

리스트, 튜플, 딕셔너리

임경태





학습목표

- 리스트의 개념을 이해하고 활용합니다.
- 2차원 리스트 생성법을 익힙니다.
- 튜플을 이해하고 리스트와 차이점을 파악합니다.
- 딕셔너리의 개념과 활용법을 익힙니다.

Section 01

리스트의 기초



리스트의 개념

여러 개 변수의 사용

- num1 ~ num4까지 4개의 정수형 변수 선언하고 합계 출력하기

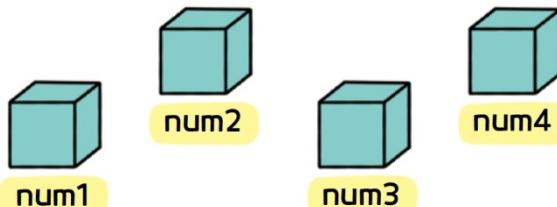


그림 7-1 변수 4개 준비

[코드 7-1]

```
num1, num2, num3, num4 = 0, 0, 0, 0
hap = 0

num1 = int(input("숫자 : "))
num2 = int(input("숫자 : "))
num3 = int(input("숫자 : "))
num4 = int(input("숫자 : "))

hap = num1 + num2 + num3 + num4

print("합계 ==> " ,hap)
```

[실행결과]

숫자 : 10
숫자 : 20
숫자 : 30
숫자 : 40
합계 ==> 100

사용자 입력

리스트의 개념



■ 리스트(List)

- 하나씩 사용하던 변수를 붙여서 한 줄로 붙여놓은 개념
- 리스트는 종이상자를 한 줄로 붙인 후에 박스 전체의 이름(numList)을 지정하여 사용함
- 각각의 데이터에는 번호(첨자)를 붙여서 접근함

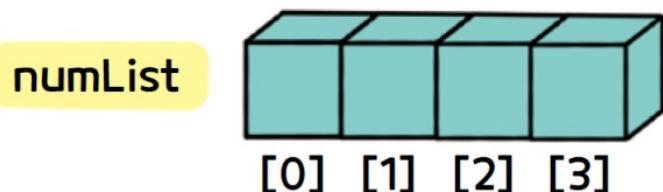


그림 7-2 리스트의 개념



리스트의 개념

■ 리스트 생성

- 대괄호 안에 값을 선언함

리스트 이름 = [값1, 값2, 값3, ...]

- 각 변수를 사용하는 경우 vs 리스트를 사용하는 경우

① 각 변수 사용

num1, num2, num3, num4 = 10, 20, 30, 40

num1 사용

num2 사용

num3 사용

num4 사용

② 리스트 사용

numList = [10, 20, 30, 40]

numList[0] 사용

numList[1] 사용

numList[2] 사용

numList[3] 사용



리스트의 개념

■ 리스트 생성

- [코드 7-1]을 리스트를 사용하도록 수정하기

[코드 7-2]

```
numList = [0, 0, 0, 0]
```

```
hap = 0
```

```
numList[0] = int(input("숫자 : "))
```

```
numList[1] = int(input("숫자 : "))
```

```
numList[2] = int(input("숫자 : "))
```

```
numList[3] = int(input("숫자 : "))
```

```
hap = numList[0] + numList[1] + numList[2] + numList[3]
```

```
print("합계 ==> " ,hap)
```

[실행결과]

숫자 : 10

숫자 : 20

숫자 : 30

숫자 : 40

합계 ==> 100

사용자 입력

리스트의 개념



확인문제

3개 숫자 10, 20, 30을 리스트에 저장하기 위해 빈칸을 채우시오.

numList =

정답

Click!



리스트의 개념

■ 리스트의 다양한 형태

```
❶ numList = []
❷ intList = [10, 20, 30]
❸ strList = ['난생처음', '파이썬', 'Good']
❹ mixList = [10, 20, '난생']
```

❹와 같은 형태는 자주 사용하지 않으므로
참고만 하세요.

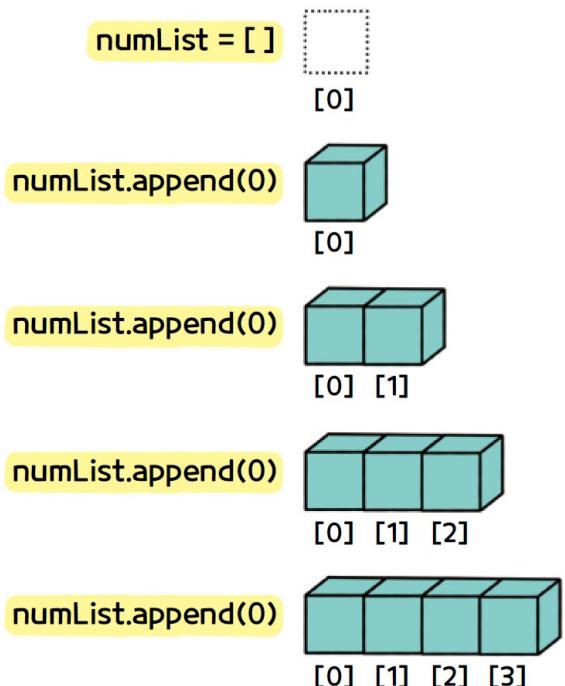
- ① 비어있는 리스트 생성
- ② 정수로만 구성된 리스트
- ③ 문자열로만 구성된 리스트
- ④ 다양한 데이터 형식을 섞은 리스트

for문을 활용한 리스트



■ 리스트를 하나씩 추가하기

- 리스트 이름.append(값)



```
>>> numList = []
>>> numList.append(0)
>>> numList.append(0)
>>> numList.append(0)
>>> numList.append(0)
>>> print(numList)
[0, 0, 0, 0]
```

그림 7-3 빈 리스트의 생성과 추가

for문을 활용한 리스트



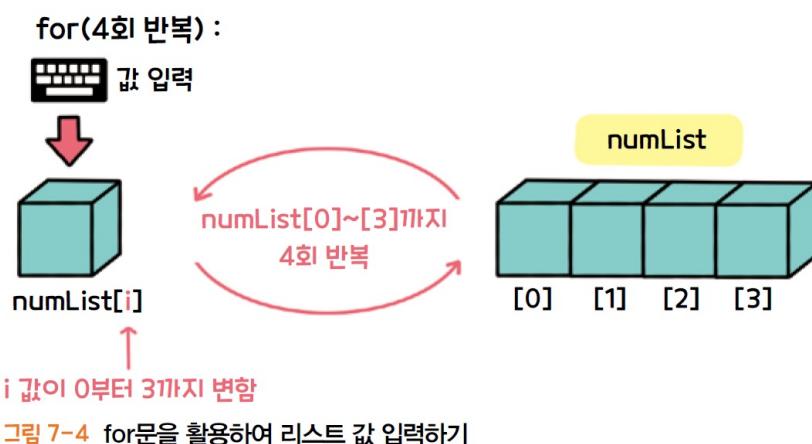
■ 반복문을 활용하여 리스트에 값 추가하기

- for문을 활용하여 리스트에 0을 4번 추가하기

```
>>> numList = []
>>> for i in range(0, 4) :
    numList.append(0)
>>> len(numList)
4
```

■ 리스트에 값 대입

- 리스트의 첨자가 순서대로 변할 수 있도록 반복문과 함께 활용하기



for문을 활용한 리스트



■ 리스트에 값 대입

- [코드 7-2]를 for문을 활용하여 리스트를 생성하고 값을 대입하도록 수정하기

[코드 7-3]

```
numList = []
for i in range(0, 4) :
    numList.append(0)
hap = 0

for i in range(0, 4) :
    numList[i] = int(input("숫자 : "))

hap = numList[0] + numList[1] + numList[2] + numList[3]

print("합계 ==> ", hap)
```

[실행결과]

숫자 : 10
숫자 : 20
숫자 : 30
숫자 : 40
합계 ==> 100

사용자 입력

[문제]

'`hap=numList[0]+numList[1]+numList[2]+numList[3]`' 행을 for문을 활용하도록 수정해봅시다.

for문을 활용한 리스트



확인문제

다음 중 리스트가 아닌 것을 고르시오.

- ① a = (100)
- ② b = [3.14]
- ③ c = ['안녕']
- ④ d = [True]

정답

Click!

하나 더 알기

append()의 사용

append()를 사용할 때도 아무 값이나 append()해도 됩니다. 자주 사용하지는 않지만, 문법상 문제는 없습니다.

```
>>> mixList = []
>>> mixList.append(100)
>>> mixList.append(3.14)
>>> mixList.append(True)
>>> mixList.append("한빛아카데미")
>>> print(mixList)
[100, 3.14, True, '한빛아카데미']
```



리스트 값에 접근하기



■ 첨자를 활용한 리스트 접근 방법

■ 첨자가 음수 값인 경우

- 맨 뒤가 -1로 시작하여 그 바로 앞이 -2가 됨

```
>>> numList = [10, 20, 30, 40]
>>> print("numList[-1]은 ", numList[-1], ", numList[-2]는 ", numList[-2])
numList[-1]은 40, numList[-2]는 30
```



리스트 값에 접근하기

첨자를 활용한 리스트 접근 방법

콜론(:)을 사용하여 범위를 지정하는 경우

- 리스트이름[시작 첨자 : 끝 첨자+1] : 리스트의 시작부터 끝까지 모두 지정함

```
>>> numList = [10, 20, 30, 40]
>>> print(numList[0:3])
[10, 20, 30]
>>> print(numList[2:4])
[30, 40]
```

- 콜론의 앞이나 뒤의 숫자를 생략할 수도 있음

```
>>> numList = [10, 20, 30, 40]
>>> print(numList[2:])
[30, 40]
>>> print(numList[:2])
[10, 20]
```

리스트 값에 접근하기



확인문제

다음 코드의 결과는 무엇인가?

```
numList = [10, 20, 30, 40, 50]  
print(numList[2:4])
```

정답

Click!



리스트의 덧셈과 곱셈

■ 리스트의 덧셈

- 요소들이 합쳐져 하나의 리스트가 됨

■ 리스트의 곱셈

- 곱한 횟수만큼 리스트가 반복됨

```
>>> numList = [10, 20, 30]
>>> myList = [40, 50, 60]
>>> print(numList + myList)
[10, 20, 30, 40, 50, 60]
>>> print(numList * 3)
[10, 20, 30, 10, 20, 30, 10, 20, 30]
```



리스트의 덧셈과 곱셈

하나 더 알기 ✓

인덱스 오류

리스트를 사용할 때 가장 많이 실수하는 것이 리스트 첨자의 범위를 넘어선 것을 접근하는 것입니다. 이럴 때 발생하는 오류를 인덱스 오류(IndexError)라고 부릅니다. 다음 코드를 실행해 봅시다.

```
>>> numList = [10, 20, 30]
>>> print(numList[3])
Traceback (most recent call last):
  File "<pyshell#13>", line 1, in <module>
    print(numList[3])
IndexError: list index out of range
```

오류가 발생했습니다. 오류 메시지의 제일 마지막 행을 보면 `IndexError`와 `out of range`라는 용어를 볼 수 있습니다. 이는 범위를 벗어났다는 것을 의미합니다. `numList`는 현재 3개의 값이 들어있으며 그 첨자는 0~2입니다. 그런데 존재하지 않는 `numList[3]`을 출력하려고 시도했기 때문에 `out of range` 오류가 발생한 것입니다.

자주 실수하는 이유는 `numList`가 3개이므로 `numList[3]`이 있을 것이라 착각하기 쉽기 때문입니다. `numList`를 3개 설정했다면 `numList[0], numList[1], numList[2]`까지만 있다는 것을 주의하여 기억해야 합니다.



오늘의 명언 출력하기

여러 개의 명언을 리스트에 저장해 놓고, 저장된 명언을 랜덤하게 출력하는 코드를 작성해 봅시다.

실행 결과

오늘의 명언 ==> 언제나 현재에 집중할 수 있다면 행복할 것이다



오늘의 명언 출력하기

1. lab07-01.py 파일을 만들고, 컴퓨터가 무작위로 명언을 고르도록 random을 임포트하기

```
import random
```

2. 10개의 명언을 리스트에 준비하기

```
wiseSay = [  
    "삶이 있는 한 희망은 있다",  
    "언제나 현재에 집중할 수 있다면 행복할 것이다",  
    "신은 용기있는 자를 결코 버리지 않는다",  
    "피할 수 없으면 즐겨라",  
    "행복한 삶을 살기위해 필요한 것은 거의 없다",  
    "내일은 내일의 태양이 뜬다",  
    "계단을 밟아야 계단 위에 올라설 수 있다",  
    "행복은 습관이다. 그것을 몸에 지니라",  
    "1퍼센트의 가능성, 그것이 나의 길이다",  
    "작은 기회로 부터 종종 위대한 업적이 시작된다" ]
```

3. 랜덤하게 명언의 위치를 추출하기 위해 randint() 함수를 사용함. 뽑힌 명언을 출력하기

```
today = random.randint(0, len(wiseSay)-1)  
print("오늘의 명언 ==>", wiseSay[today])
```

4. <Ctrl>+<S>를 눌러서 변경된 내용을 저장하고, <F5>를 눌러 실행 결과 확인하기

Section 02

리스트 활용하기

리스트 값 변경하기



■ 리스트 값 변경하기

- ▶ 첨자를 사용하여 값 변경하기

```
>>> numList = [10, 20, 30]  
>>> numList[1] = 200  
>>> print(numList)  
[10, 200, 30]
```

- ▶ 연속된 범위의 값 변경하기

```
>>> numList = [10, 20, 30]  
>>> numList[1:2] = [200, 201]  
>>> print(numList)  
[10, 200, 201, 30]
```



리스트 값 변경하기

■ 리스트 값 변경하기

- 리스트 안에 리스트 추가하기
 - 자주 사용하는 방식은 아니므로 주의가 필요함

```
>>> numList = [10, 20, 30]
>>> numList[1] = [200, 201]
>>> print(numList)
[10, [200, 201], 30]
```

확인문제

다음 코드의 실행 결과는 무엇인가?

```
numList = [10, 20, 30]
numList[2:3] = [200, 201]
print(numList)
```

정답

Click!

리스트 값 삽입/삭제/개수 세기



■ 리스트에 값 삽입하기

- `append(값)` : 맨 뒤에 값 추가하기
- `insert(위치, 값)` : 정해진 위치에 값 삽입하기

```
>>> numList = [10, 20, 30]
>>> numList.insert(1, 123)
>>> print(numList)
[10, 123, 20, 30]
```

리스트 값 삽입/삭제/개수 세기



■ 리스트에 값 삭제하기

- `del()` : 리스트의 항목 삭제

```
>>> numList = [10, 20, 30]
>>> del(numList[1])
>>> print(numList)
[10, 30]
```

- `del()` 함수에 리스트 이름을 넣으면 리스트가 통째로 삭제됨

```
>>> numList = [10, 20, 30]
>>> del(numList)
>>> print(numList)
```

오류 발생

리스트 값 삽입/삭제/개수 세기



■ 리스트에 값 삭제하기

- remove(지울 값) : 리스트에서 특정 값 삭제

```
>>> numList = [10, 20, 30]
>>> numList.remove(10)
>>> print(numList)
[20, 30]
```

- 주의할 점)

- 중복된 값을 지울 때 모든 값을 지우지 않고, 처음 만나는 한 개의 값만 삭제함

```
>>> numList = [10, 20, 30, 10, 10]
>>> numList.remove(10)
>>> print(numList)
[20, 30, 10, 10]
```

리스트 값 삽입/삭제/개수 세기



■ 리스트의 값 추출하기

- `pop()` : 제일 뒤의 값을 뽑아내서 값을 알려준 뒤 삭제함]₩

```
>>> numList = [10, 20, 30]
>>> numList.pop()
30
>>> print(numList)
[10, 20]
```

■ 리스트에서 개수 세기

- `count(찾을 값)` : 찾을 값이 몇 개인지 개수를 세서 알려줌

```
>>> numList = [10, 20, 30, 10, 10]
>>> numList.count(10)
3
```

리스트 정렬/반전/복사하기



■ 리스트 정렬하기

- `sort()` : 리스트의 값을 정렬함

```
>>> numList = [20, 10, 40, 50, 30]
>>> numList.sort()
>>> print(numList)
[10, 20, 30, 40, 50]
```

- `sort(reverse = True)` : 내림차순으로 정렬함

```
>>> numList = [20, 10, 40, 50, 30]
>>> numList.sort(reverse=True)
>>> print(numList)
[50, 40, 30, 20, 10]
```



리스트 정렬/반전/복사하기

■ 리스트의 차례를 반전하기

- reverse() : 리스트의 마지막 인덱스부터 위치가 반대로 됨

```
>>> numList = [20, 10, 40, 50, 30]
>>> numList.reverse()
>>> print(numList)
[30, 50, 40, 10, 20]
```

■ 리스트 복사하기

- copy() : 리스트를 새로운 리스트로 복사함

```
>>> numList = [10, 20, 30]
>>> numList2 = numList.copy()
>>> print(numList2)
[10, 20, 30]
```

리스트 정렬/반전/복사하기



확인문제

1. 다음 빈칸에 들어갈 단어를 채우시오.

리스트에서 특정 값을 지우는 함수는 이며, 마지막 값을 뽑아내는 함수는 이다.

2. 다음 빈칸에 들어갈 단어를 채우시오.

리스트를 정렬하는 함수는 이고, 차례를 반전하는 함수는 이다.

정답

Click!

2차원 리스트



■ 2차원 리스트

- 1차원 리스트를 여러 개 연결한 리스트
- 2개의 첨자를 사용함
 - 리스트이름[행][열]

numList1 = [10, 20, 30]

numList1

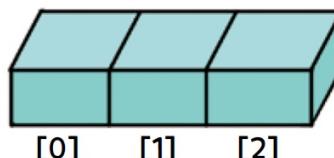


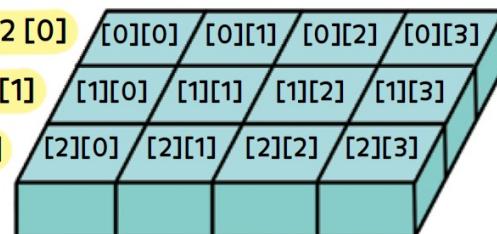
그림 7-5 1차원 리스트의 개념

numList2 = [[1, 2, 3, 4],
[5, 6, 7, 8],
[9, 10, 11, 12]]

numList2 [0]

numList2 [1]

numList2 [2]



전체 이름: numList2

그림 7-6 2차원 리스트의 개념

2차원 리스트



■ for문을 활용한 2차원 리스트 출력

- 중첩 for문 사용

[코드 7-4]

```
numList2 = [[1, 2, 3, 4],  
            [5, 6, 7, 8],  
            [9, 10, 11, 12]]  
  
for i in range(0, 3) :  
    for k in range(0, 4) :  
        print(" ", numList2[i][k], end=' ')  
    print("")
```

[실행결과]

```
1   2   3   4  
5   6   7   8  
9  10  11  12
```

심사위원 점수 결과 구하기

동계 올림픽에서 피겨 스케이팅 경기는, 심사위원 5명이 점수를 각 10점 만점으로 줄 수 있습니다.

홍길동 선수의 경기 후에 5명 심사위원의 점수를 리스트에 저장하고, 점수의 평균을 구하는 프로그램을 구현해 봅시다.

실행 결과

홍길동 선수 경기 끝났습니다~~ 짹쫙쫙

평가 점수 ==>10

평가 점수 ==>9

평가 점수 ==>8

평가 점수 ==>9

평가 점수 ==>10

심사위원 평균 점수 : 9.2

사용자 입력



1. lab07-02.py 파일을 만들고, 흥길동 선수의 경기가 끝났음을 print()로 알리기

```
print("흥길동 선수 경기 끝났습니다~~ 짹쫙쫙")
```

2. 심사위원의 점수를 입력받을 리스트 score를 선언하고, 5번 반복하며 평가 점수를 입력받아 리스트에 추가하기

```
score = []
for i in range(5) :
    jumsu = int(input("평가 점수 ==>"))
    score.append(jumsu)
```

3. 5명의 평가 점수를 더할 합계변수 hap을 선언하고 0으로 초기화하기

- hap에 리스트의 값을 모두 더하기

```
hap = 0
for i in range(5) :
    hap += score[i]
```

4. 심사위원의 평균 점수를 구하여 출력하기

```
avg = hap / 5  
print("심사위원 평균 점수 :", avg)
```

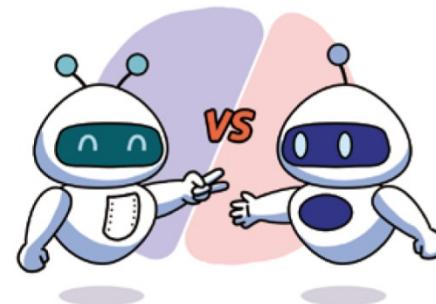
5. <Ctrl>+<S>를 눌러서 변경된 내용을 저장하고, <F5>를 눌러 실행 결과 확인하기

컴퓨터끼리 가위바위보 대결하기

두 대의 컴퓨터끼리 가위바위보 게임을 10000번 시키고 최종적으로 누가 더 많이 이겼는지 결과를 확인해 봅시다.

5장에서도 살펴봤지만, 가위바위보는 다음의 규칙을 따릅니다.

컴퓨터A	컴퓨터B	이긴 컴퓨터
가위	가위 바위 보	없음 B A
바위	가위 바위 보	A 없음 B
보	가위 바위 보	B A 없음



실행 결과

10000번 중 컴퓨터A의 승리 : 3306

10000번 중 컴퓨터B의 승리 : 3267

10000번 중 비긴 경기 : 3427

1. lab07-03.py 파일을 만들고, 컴퓨터가 랜덤하게 가위/바위/보를 내도록 random을 임포트하기
 - 가위바위보에서 이긴 컴퓨터를 저장하기 위한 빈 리스트 toss도 준비하기

```
import random  
  
toss = []
```

2. 10000번을 반복해서 두 컴퓨터가 가위/바위/보 게임을 자동으로 수행하기

```
for i in range(10000) :  
    comA = random.choice(["가위", "바위", "보"])  
    comB = random.choice(["가위", "바위", "보"])
```

3. 각 컴퓨터가 내는 것을 랜덤하게 추출한 후, 표에 따라 어떤 컴퓨터의 승리인지를 리스트에 저장하기
- 비기는 경우는 "없음"으로 리스트에 저장

```
if comA == "가위" :  
    if comB == "가위" :  
        toss.append("없음")  
    elif comB == "바위" :  
        toss.append("B")  
    elif comB == "보" :  
        toss.append("A")  
elif comA == "바위" :  
    if comB == "가위" :  
        toss.append("A")  
    elif comB == "바위" :  
        toss.append("없음")  
    elif comB == "보" :  
        toss.append("B")  
elif comA == "보" :  
    if comB == "가위" :  
        toss.append("B")  
    elif comB == "바위" :  
        toss.append("A")  
    elif comB == "보" :  
        toss.append("없음")
```

4. 컴퓨터A, 컴퓨터B, 비긴 결과의 개수를 세고 출력하기

```
aWin = toss.count("A")
bWin = toss.count("B")
noWin = toss.count("없음")

print("10000번 중 컴퓨터A의 승리 : ", aWin)
print("10000번 중 컴퓨터B의 승리 : ", bWin)
print("10000번 중 비긴 경기 : ", noWin)
```

5. <Ctrl>+<S>를 눌러서 변경된 내용을 저장하고, <F5>를 눌러 실행 결과 확인하기

Section 03

튜플

튜플의 개념



■ 튜플

- 읽기 전용의 리스트
- 소괄호로 생성함
- 리스트와 상당히 비슷하지만 값을 읽을 수만 있고 수정할 수는 없음



그림 7-7 리스트와 튜플의 차이

튜플의 개념



■ 튜플 생성하기

- 튜플은 소괄호로 생성하지만, 괄호가 없어도 무방함

```
>>> numTup1 = (10, 20, 30)  
>>> print(numTup1)  
(10, 20, 30)
```

```
>>> numTup2 = 10, 20, 30  
>>> print(numTup2)  
(10, 20, 30)
```

- 튜플의 항목이 한 개일 때 튜플 뒤에 콤마()를 붙여야 함
 - numTup3, numTup4: 튜플이 아닌 일반 값
 - numTup5, numTup6: 하나의 항목만 가진 튜플

```
>>> numTup3 = (10)  
>>> print(numTup1)  
10
```

```
>>> numTup4 = 10  
>>> print(numTup2)  
10
```

```
>>> numTup5 = (10, )  
>>> print(numTup3)  
(10, )
```

```
>>> numTup6 = 10,  
>>> print(numTup4)  
(10, )
```

튜플의 개념



■ 읽기 전용인 튜플

- 다음은 모두 오류가 발생함

```
>>> numTup1.append(40)
>>> numTup1[0] = 40
>>> del(numTup1[0])
```

- 튜플 자체를 통째로 삭제하려면 **del(튜플이름)** 함수를 사용함

```
>>> del(numTup1)
>>> del(numTup2)
```



튜플은 리스트에서 사용한 `sort()`, `reverse()` 등의 함수도 사용할 수 없습니다.

튜플의 활용



■ 튜플에 접근하기

- 튜플이름[인덱스] : 특정 항목에 접근하기

```
>>> numTup1 = (10, 20, 30, 40)
>>> print(numTup1[0])
10
>>> print(numTup1[0] + numTup1[1] + numTup1[2])
60
```

- 콜론(시작인덱스 : 끝인덱스+1) : 튜플의 범위에 접근하기

```
>>> print(numTup1[1:3])
(20, 30)
>>> print(numTup1[1:])
(20, 30, 40)
>>> print(numTup1[:3])
(10, 20, 30)
```

튜플의 활용



■ 튜플의 더하기 및 곱하기 연산

```
>>> numTup2 = ('A', 'B')
>>> print(numTup1 + numTup2)
(10, 20, 30, 40, 'A', 'B')
>>> print(numTup2 * 3)
('A', 'B', 'A', 'B', 'A', 'B')
```

하나 더 알기

튜플과 리스트의 상호 변환

튜플과 리스트는 서로 변환할 수 있습니다. 예를 들어 튜플의 항목을 변경하려면 튜플을 리스트로 변경해서 항목을 변경한 후, 다시 리스트를 튜플로 변환하는 방법을 사용할 수 있습니다.

`list(튜플)` 함수는 튜플을 리스트로 변환해 주며, `tuple(리스트)` 함수는 리스트를 튜플로 변환해 줍니다. 다음은 튜플 → 리스트 → 튜플로 변환한 예입니다.

```
>>> myTuple = (10, 20, 30) ────────── 튜플 선언
>>> myList = list(myTuple) ────────── 리스트로 변환
>>> myList.append(40)
>>> myTuple = tuple(myList) ────────── 튜플로 변환
>>> print(myTuple)
(10, 20, 30, 40)
```

튜플은 리스트에 비해서 활용도가 다소 떨어지기 때문에 튜플에 대해서는 지금 소개한 정도만 알아도 충분합니다.



튜플의 활용



확인문제

1. 다음 빈칸에 들어갈 단어를 채우시오.

리스트는 대괄호를 사용하며 읽기/쓰기가 되지만, (은)는 (을)를 사용하며 읽기 전용이다.

2. 다음 중 올바른 것을 고르시오.

```
numTup = (10, 20, 30)
```

- ① numTup.append(40)
- ② numTup[0] = 40
- ③ del(numTup[0])
- ④ del(numTup)

정답

Click!

Section 04

딕셔너리



딕셔너리의 개념

■ 딕셔너리

- 단어 의미 그대로 '영어사전'과 같은 구조를 가짐
- 즉 딕셔너리는 2개의 쌍이 하나로 묶이는 자료구조를 의미함
- 딕셔너리는 중괄호({ })로 묶여 있음
- 키(Key)와 값(Value)의 쌍으로 이루어짐
 - 키(Key): 단어, 값(Value): 뜻

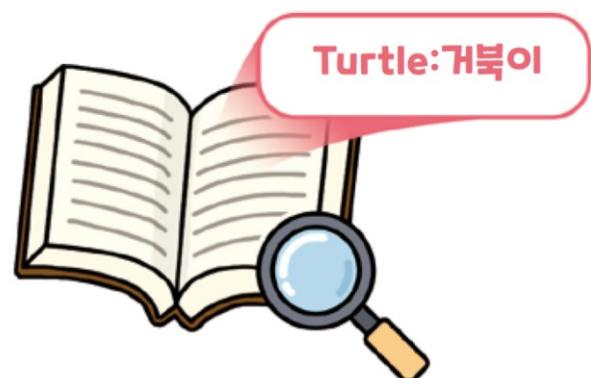


그림 7-8 딕셔너리의 개념

딕셔너리변수 = { 키1:값1 , 키2:값2, 키3:값3, ... }

딕셔너리의 개념



■ 딕셔너리 생성하기

- 키(Key)가 1, 2, 3이, 값(Value)이 'a', 'b', 'c' 인 딕셔너리

```
>>> myDict = {1:'a', 2:'b', 3:'c'}  
>>> print(myDict)  
{1: 'a', 2: 'b', 3: 'c'}
```

- 키와 값을 반대로 생성해도 무방함

```
>>> myDict = {'a':1, 'b':2, 'c':3}  
>>> print(myDict)  
{'a': 1, 'b': 2, 'c': 3}
```

- 즉, 키와 값을 사용자가 지정하는 것이지 어떤 값을 반드시 사용해야 하는 규정은 없으며, 딕셔너리는 순서가 없음



딕셔너리의 생성과 삭제

■ 딕셔너리의 생성 예제

- 딕셔너리는 여러 개의 정보를 하나의 변수로 표현할 때 유용하게 사용됨
- ① 회사원 홍길동의 정보를 딕셔너리로 생성하기

표 7-1 회사원의 정보

키(Key)	값(Value)
사번	1000
이름	홍길동
부서	케이팝

```
>>> empDict = {'사번':1000 , '이름':'홍길동', '부서':'케이팝'}  
>>> print(empDict)  
{'사번': 1000, '이름': '홍길동', '부서': '케이팝'}
```

딕셔너리의 생성과 삭제



■ 딕셔너리의 생성 예제

- ② 딕셔너리에 정보를 더 추가하기
 - 정보를 추가할 때는 키와 값을 쌍으로 추가해야 함
 - '딕셔너리이름[키]' = 값' 형식을 사용함

```
>>> empDict['연락처'] = '010-1111-2222'  
>>> print(empDict)  
{'사번': 1000, '이름': '홍길동', '부서': '케이팝', '연락처':  
'010-1111-2222'}
```

- ③ 정보를 추가할 때 이미 키가 딕셔너리에 존재하는 경우
 - 기존의 키에 대한 값이 변경됨

```
>>> empDict['부서'] = '한빛아카데미'  
>>> print(empDict)  
{'사번': 1000, '이름': '홍길동', '부서': '한빛아카데미',  
'연락처': '010-1111-2222'}
```

딕셔너리의 생성과 삭제



■ 딕셔너리의 생성 예제

- ④ 딕셔너리의 키는 중복되지 않고 유일함
 - 만약 동일한 키를 갖는 딕셔너리를 생성한다면 오류가 발생하지는 않고, 마지막에 있는 키의 값이 적용되어 생성됨

```
>>> empDict2 = { '사번':1000 , '이름':'홍길동', '사번':2000 }  
>>> print(empDict2)  
{'사번': 2000, '이름': '홍길동'}
```



딕셔너리의 값은 중복되어도 상관없습니다.

딕셔너리의 생성과 삭제



■ 딕셔너리의 생성 예제

- ⑤ 딕셔너리의 쌍을 삭제하기
 - del(딕셔너리이름[키]) 함수를 사용

```
>>> del(empDict['부서'])  
>>> print(empDict)  
{'사번': 1000, '이름': '홍길동', '연락처': '010-1111-2222'}
```

하나 더 알기 ✅

딕셔너리의 개수 파악하기

딕셔너리의 개수는 `len(딕셔너리이름)` 함수로 확인할 수 있으며 한 쌍을 한 개로 계산합니다. 현재 `empDict`의 개수는 3이 됩니다.

```
>>> len(empDict)  
3
```



딕셔너리의 사용



■ 딕셔너리에 접근하기

- ▣ ⑥ 키를 이용해 값을 구하기

```
>>> empDict = {'사번':1000, '이름':'홍길동', '부서':'케이팝'}
>>> print(empDict['사번'])
1000
>>> print(empDict['이름'])
'홍길동'
>>> print(empDict['부서'])
'케이팝'
```

딕셔너리의 사용



■ 딕셔너리와 관련된 유용한 함수

- 딕셔너리이름.keys() : 딕셔너리의 모든 키만 뽑아서 반환함

```
>>> empDict.keys()  
dict_keys(['사번', '이름', '부서'])
```

- list(딕셔너리이름.keys()) : 출력 결과에 dict_keys가 붙지 않음

```
>>> list(empDict.keys())  
['사번', '이름', '부서']
```

딕셔너리의 사용



■ 딕셔너리와 관련된 유용한 함수

- 딕셔너리이름.values() : 딕셔너리의 모든 값을 리스트로 만들어서 반환함
 - dict_values가 보기 싫으면 list(딕셔너리이름.values()) 함수를 사용

```
>>> empDict.values()  
dict_values([1000, '홍길동', '케이팝'])
```

- 딕셔너리이름.items() : 튜플 형태로 구하기

```
>>> empDict.values()  
dict_items([('사번', 1000), ('이름', '홍길동'), ('부서', '케이팝')])
```

딕셔너리의 사용



■ 딕셔너리와 관련된 유용한 함수

- in : 딕셔너리 안에 키가 있는지는 확인이 가능함
 - 딕셔너리에 키가 있으면 True, 없으면 False를 반환함
 - in은 if문과 함께 사용되는 경우가 많음

```
>>> '이름' in empDict  
True
```

```
>>> '주소' in empDict  
False
```



딕셔너리의 사용

■ 딕셔너리와 관련된 유용한 함수

- 딕셔너리에 저장된 것을 한 건씩 출력하려면 for문을 사용

```
>>> for key in empDict.keys() :  
    print( key , empDict[key])
```

사번 1000
이름 홍길동
부서 케이팝

- 'for key in 리스트' 구문은 리스트에 있는 것을 하나씩 추출해서 key에 넣고 반복문을 실행하므로 같은 결과가 나옴

```
for key in ['사번', '이름', '부서'] :  
    print( key, empDict[key])
```



딕셔너리의 사용

확인문제

1. 다음 빈칸에 들어갈 단어를 채우시오.

{키:값}의 형태로 구성된 데이터 구조를 (이)라고 부른다.

2. 다음 중 잘못된 것을 고르시오.

- ① 딕셔너리의 개수는 한 쌍을 1개로 계산한다.
- ② 딕셔너리의 쌍을 삭제하려면 del(딕셔너리이름[키]) 함수를 사용한다.
- ③ 딕셔너리의 키는 중복될 수 없다.
- ④ 딕셔너리의 값은 중복될 수 없다.

3. 다음 빈칸에 들어갈 단어를 채우시오.

empDict 딕셔너리의 키를 전부 추출하려면 empDict.() 함수를 사용하고, 값을 전부 추출하려면 empDict.() 함수를 사용한다.

정답

Click!

가수 정보를 딕셔너리에 저장하고 출력하기

딕셔너리 구조를 활용해서 가수그룹의 이름, 구성원수, 데뷔 프로그램, 대표곡을 저장해 보겠습니다. 또, 저장된 딕셔너리를 모두 출력해 봅시다.



실행 결과

이름 ==> 트와이스

구성원수 ==> 9

데뷔 ==> 서바이벌 식스틴

대표곡 ==> CRY FOR ME

가수 정보를 딕셔너리에 저장하고 출력하기

1. lab07-04.py 파일을 만들고, 가수 정보를 저장할 비어있는 딕셔너리를 준비하기

```
singer = {}
```

2. 가수그룹 정보를 딕셔너리에 저장하는 프로그램을 작성하기

```
singer['이름'] = "트와이스"  
singer['구성원수'] = 9  
singer['데뷔'] = "서바이벌 식스틴"  
singer['대표곡'] = "CRY FOR ME"
```

3. 키에 해당하는 값을 차례대로 출력하기

```
for k in singer.keys() :  
    print(k, " ==> ", singer[k])
```

4. <Ctrl>+<S>를 눌러 저장한 후, <F5>를 눌러서 실행하고 결과를 확인합니다.

편의점 재고 관리하기

편의점 물품의 재고를 관리하는 코드를 작성해 봅시다. 사용자가 편의점 물품과 재고량을 입력하면 딕셔너리에 키를 물품으로, 재고량을 값으로 저장시키겠습니다. 그리고 재고량이 있는 물품을 검색하는 프로그램을 작성해 봅시다.



실행 결과

```

    입력 물품 ==>삼각김밥
    재고량 ==>13
    입력 물품 ==>우유
    재고량 ==>7
    입력 물품 ==>도시락
    재고량 ==>5
    입력 물품 ==> z
    *** 물품의 재고량 확인 ***
    찾을 물품 ==>도시락 ●———— 사용자 입력
    5 개 남았어요
    찾을 물품 ==>새우깡 ●———— 사용자 입력
    그 물품은 없어요.
    찾을 물품 ==> Enter
  
```

사용자 입력

편의점 재고 관리하기

1. lab07-05.py 파일을 만들고, 물품 정보를 저장할 비어있는 딕셔너리를 준비하기

```
store = {}
```

2. 물품을 입력하는 작업을 무한 반복하기

- 이때 z를 누르면 무한 반복을 빠져나가도록 조건을 설정함

```
print("*** 물품과 재고량 입력 ***")
while True :
    item = input("입력 물품 ==>")
    if item == "z" :
        break
```

3. 물품에 대한 재고량을 입력받고, 물품명과 재고량의 쌍으로 딕셔너리에 대입합니다.

```
count = int(input("재고량 ==>"))
store[item] = count
```

4. 물품을 검색하는 작업을 무한 반복함.

- 우선 찾을 물품을 입력받음. 이때 <Enter>를 누르면 더 이상 입력할 물품이 없는 것으로 처리하고 무한 반복을 빠져나가기

```
print("*** 물품의 재고량 확인 ***")
while True :
    item = input("찾을 물품 ==>")
    if item == "":
        break
```

5. 딕셔너리에 찾는 물품이 있으면 재고량을 출력하기

- 찾는 물품이 없다면 물품이 없다고 출력하기

```
if item in store :
    print(store[item], "개 남았어요")
else :
    print("그 물품은 없어요.")
```

6. <Ctrl> + <S> 를 눌러 저장한 후, <F5> 를 눌러서 실행하고 결과를 확인하기

[실전 예제] 100마리의 거북이 쇼

실전 예제 100마리의 거북이 쇼

[문제]

거북이 100마리를 만들어서 화면의 중앙에서 각 거북이가 다양한 방향으로 선을 그리면서 나가는 거북이 쇼

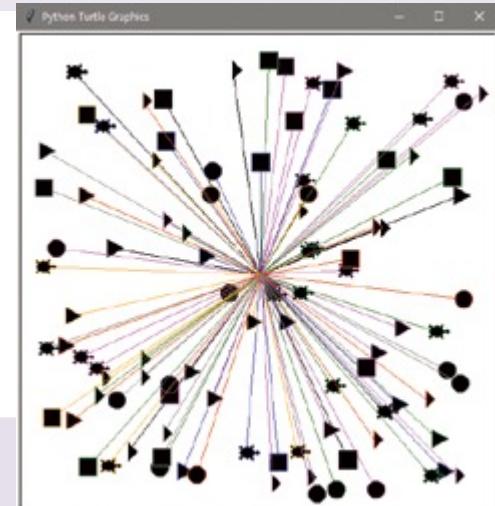
프로그램을 작성하자

- 우선 거북이 한 마리는 다음과 같은 정보를 튜플로 가짐

(거북이, X위치, Y위치, 선 색상)

- '거북이'는 거북이 모양의 종류 중 하나임
 - 거북이 모양, 원 모양, 사각형 모양, 삼각형 모양 등이 있음
- X와 Y위치는 거북이가 화면 중앙에서 선을 그리면서 이동할 위치임
- 선 색상은 거북이가 그릴 선의 색상임
- 거북이 100개를 리스트로 저장함

[(거북이, X위치, Y위치, 선 색상), (거북이, X위치, Y위치, 선 색상), ...]



- 거북이를 모두 생성한 후에는 리스트에 있는 거북이 100마리가 차례대로 선을 그음

실전 예제 100마리의 거북이 쇼

[해결]

```
import turtle
import random

turtleList = [ ]
colorList = ["red", "green", "blue", "black", "magenta", "orange",
"gray"]
shapeList = ["arrow", "circle", "square", "triangle", "turtle"]

turtle.setup(550, 550)
turtle.screensize(500, 500)

for i in range(0, 100) : # 거북이 100마리 생성
    shape = random.choice(shapeList)
    color = random.choice(colorList)
    x = random.randint(-250, 250)
    y = random.randint(-250, 250)
    myTurtle = turtle.Turtle(shape)
    tup = (myTurtle, color, x, y)
    turtleList.append(tup)

for tup in turtleList : # 리스트에 담긴 거북이 100마리 그리기
    myTurtle = tup[0]
    myTurtle.pencolor( tup[1] )
    myTurtle.goto(tup[2], tup[3])

turtle.done( )
```

Preview

감사합니다 :)