

2주

파이썬 기초



염 희 균

1. 기본 자료형

- 문자열, 숫자(정수, 실수), 불

2. 컨테이너 자료형

- 리스트, 튜플, 딕셔너리

3. 연산자

4. 확인문제

시작하기 전에

[핵심 키워드] 자료형, 문자열, 이스케이프 문자, 문자열 연산자

[핵심 포인트] 프로그램이 처리할 수 있는 모든 것을 자료라 부른다. 자료란 무엇인지, 이를 처리하는 방법은 무엇인지, 그리고 가장 일반적으로 쓰이는 문자열 자료형은 어떤 것이 있는지 알아본다.

2주

1. 기본 자료형

기본 자료형

- 자료 (data)
 - 프로그램이 처리할 수 있는 모든 것
 - 프로그램은 자료를 처리하기 위한 모든 행위
- 자료형 (data type)
 - 자료를 기능과 역할에 따라 구분한 것
- 기본 자료형 종류
 - 문자열 (string) : 메일 제목, 메시지 내용 등 ->예) "안녕하세요", "Hello"
 - 숫자 (number) : 물건의 가격, 학생의 성적 등->예) 52,273 , 103.32
 - 불 (boolean) : 친구의 로그인 상태 등->예) True, False

기본 자료형

- 자료형 확인 (data type)
 - 자료의 형식 확인
 - `type()` 함수로 확인

```
>>> print(type("안녕하세요"))  
<class 'str'>  
>>> print(type(273))  
<class 'int'>
```

- `str` : 문자열
- `int` : 정수

숫자 : 정수(Integer, int)

- 메모리가 허용하는 범위 안에서 무한 정수 사용 가능
- 정수 종류
 - 음수, 0, 양수
 - 예)

```
>>> a = 123
```

```
>>> a = -178
```

```
>>> a = 0
```

숫자 : 실수(Float, float)

- 8바이트(byte)에서만 허용
- 8바이트 이상의 데이터 손실 발생
- 예제:

```
norFloat = 1.2345678901234567  
print("norFloat = " , norFloat)  
bigFloat = 1.2345678901234567890123  
print("bigFloat=" , bigFloat)
```



```
norFloat = 1.2345678901234567  
bigFloat = 1.2345678901234567
```

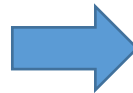

불 (Boolean, bool)

- 참(True), 거짓(False)을 나타낸다.
- 예제:

```
myBool = True  
print(myBool)
```

```
myBool = False  
print(myBool)
```

```
print(type(myBool))
```



```
True  
False  
<class 'bool'>
```

- True와 False를 소문자(true, false)로 하면 에러 발생한다.

```
myBool = true
```



```
Traceback (most recent call last):  
File "C:/pythonLec/04/ex_4_4.py", line 1, in <module>  
    myBool = true  
NameError: name 'true' is not defined
```

불 (Boolean, bool)

- 자료형의 참과 거짓

| 값 | 참 or 거짓 |
|----------|---------|
| "Python" | 참 |
| "" | 거짓 |
| [1,2,3] | 참 |
| [] | 거짓 |
| () | 거짓 |
| {} | 거짓 |
| 1 | 참 |
| 0 | 거짓 |
| None | 거짓 |

불 (Boolean, bool) 예제

- 자료형의 참과 거짓은 어떻게 사용되나?

```
a = [1,2,3,4]
```

```
while a:
```

```
    a.pop()
```

```
    print(a)
```



```
[1,2,3]
```

```
[1,2]
```

```
[1]
```

리스트 자료가 없으면 False로 보고 반복을 종료함!

문자열 (String, str)

- 문자열 (string)

- 작은 따옴표, 큰 따옴표로 둘러싸서 글자가 나열된 것

```
"Hello"    'String'    '안녕하세요'    "Hello Python Programming"
```

- 큰 따옴표 3개를 연속("''")으로 써서 양쪽 둘러싸기

```
"""Life is too short, You need python"""
```

- 작은 따옴표 3개를 연속(''')으로 써서 양쪽 둘러싸기

```
'''Life is too short, You need python'''
```

문자열 만들기

- 문자열 내부에 따옴표 넣기

"안녕하세요"라고 말했습니다

>>> print("안녕하세요"라고 말했습니다)

출력할 큰따옴표

문자열을 만들기 위해 사용한 큰따옴표

위 경우 오류 발생

- 파이썬 프로그래밍 언어는 자료와 자료를 단순 나열할 수 없음
- 구문 오류 (syntax error)

오류

SyntaxError: invalid syntax

문자열 만들기

- 작은따옴표로 문자열 만들어 큰따옴표 포함 문제 해결
 - 반대로도 가능

```
>>> print("'안녕하세요'라고 말했습니다')  
"안녕하세요"라고 말했습니다
```

```
>>> print("'배가 고플니다'라고 생각했습니다")  
'배가 고플니다'라고 생각했습니다
```

문자열 만들기

- 이스케이프 문자 (escape character)
 - 역슬래시 기호와 함께 조합해서 사용하는 특수한 문자
 - \“ : 큰따옴표를 의미
 - \‘ : 작은따옴표를 의미

```
>>> print("\안녕하세요\라고 말했습니다")
"안녕하세요"라고 말했습니다
>>> print('\배가 고픈다\'라고 생각했습니다')
'배가 고픈다'라고 생각했습니다
```

- \n : 줄바꿈 의미
- \t : 탭 의미

문자열 만들기

```
>>> print("안녕하세요\n안녕하세요")
안녕하세요
안녕하세요
>>> print("안녕하세요\t안녕하세요")
안녕하세요      안녕하세요
```

```
01 print("이름\n나이\n지역")
02 print("윤인성\n25\n강서구")
03 print("윤아린\n24\n강서구")
04 print("구름\n3\n강서구")
```

| 실행결과 | | |
|------|----|-----|
| 이름 | 나이 | 지역 |
| 윤인성 | 25 | 강서구 |
| 윤아린 | 24 | 강서구 |
| 구름 | 3 | 강서구 |

문자열 만들기

- \\: 역슬래시를 의미

```
>>> print("\\ \\ \\ \\")  
\\ \\ \\ \\
```

- 여러 줄 문자열 만들기

- \n 사용

```
>>> print("동해물과 백두산이 마르고 닳도록\n하느님이 보우하사 우리나라 만세\n무궁화 삼천리 화려강산 대한사람\n대한으로 길이 보전하세")  
동해물과 백두산이 마르고 닳도록  
하느님이 보우하사 우리나라 만세  
무궁화 삼천리 화려강산 대한사람  
대한으로 길이 보전하세
```

문자열 만들기

- 여러 줄 문자열 기능 활용: 큰따옴표 혹은 작은따옴표를 세 번 반복

```
>>> print("""동해물과 백두산이 마르고 닳도록  
하느님이 보우하사 우리나라 만세  
무궁화 삼천리 화려강산 대한사람  
대한으로 길이 보전하세""")  
동해물과 백두산이 마르고 닳도록  
하느님이 보우하사 우리나라 만세  
무궁화 삼천리 화려강산 대한사람  
대한으로 길이 보전하세
```

숫자형을 활용하기 위한 연산자

- 산술 연산은 일상생활에서 사용하는 사칙연산과 나머지 또는 몫만 구하기를 포함한다.

| 연산기호 | 연산내용 | 연산 예제 |
|------|-----------|--------------------|
| + | 덧셈 연산 | 20+30 =50 |
| - | 뺄셈 연산 | 20-30=-10 |
| * | 곱셈 연산 | 20*30 = 600 |
| / | 나누기 연산 | 3.6666666666666666 |
| % | 나머지 연산 | 20 |
| // | 몫만 구하는 연산 | 0 |

- 제공 연산자 : **
 - 숫자를 제곱함

```
>>> print("2 ** 1 =", 2 ** 1)
2 ** 1 = 2
>>> print("2 ** 2 =", 2 ** 2)
2 ** 2 = 4
>>> print("2 ** 3 =", 2 ** 3)
2 ** 3 = 8
>>> print("2 ** 4 =", 2 ** 4)
2 ** 4 = 16
```

문자열 연산자1

- 문자열 연결 연산자 : +

"문자열" + "문자열"

문자열 연결 연산자

- 더하기와 같은 기호이나 다른 수행임에 주의


문자열 연산자1

- 두 문자열 연결하여 새로운 문자열 만들어냄

```
>>> print("안녕" + "하세요")
안녕하세요
>>> print("안녕하세요" + "!")
안녕하세요!
```

- 문자열과 숫자 사이에는 사용할 수 없음

```
>>> print("안녕하세요" + 1)
```

 오류

TypeError: can only concatenate str (not "int") to str

- 문자열은 문자끼리, 숫자는 숫자끼리 연결
- 문자열과 숫자 연결하여 연산하려면 큰따옴표 붙여
문자열로 인식하게 함

문자열 연산자2

- 문자열 반복 연산자 : *

- 문자열을 숫자와 * 연산자로 연결

```
>>> print("안녕하세요" * 3)  
안녕하세요안녕하세요안녕하세요
```

```
>>> print(3 * "안녕하세요")  
안녕하세요안녕하세요안녕하세요
```

- 파이참으로 실행해보기(multistring.py)

```
multistring.py  
print("=" * 50)  
print("My Program")  
print("=" * 50)
```

```
=====  
My Program  
=====
```

Process finished with exit code 0

문자열 연산자3

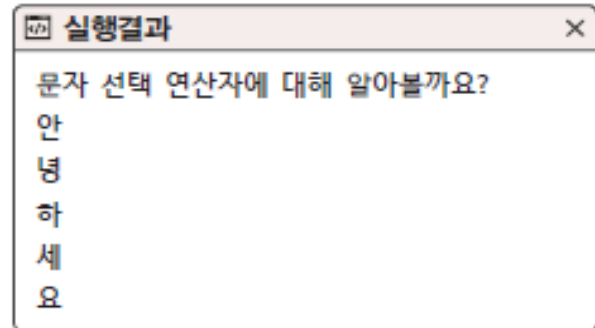
- 문자 선택 연산자 (인덱싱) : []
 - 문자열 내부의 문자 하나를 선택
 - 대괄호 안에 선택할 문자의 위치를 지정
 - 인덱스 (index)
 - 제로 인덱스 (zero index) : 숫자를 0부터 셈
 - 원 인덱스 (one index) : 숫자를 1부터 셈
 - 파이썬은 제로 인덱스 유형 사용

| | | | | |
|-----|-----|-----|-----|-----|
| 안 | 녕 | 하 | 세 | 요 |
| [0] | [1] | [2] | [3] | [4] |

문자열 연산자3

- 예시

```
01 print("문자 선택 연산자에 대해 알아보까요?")
02 print("안녕하세요"[0])
03 print("안녕하세요"[1])
04 print("안녕하세요"[2])
05 print("안녕하세요"[3])
06 print("안녕하세요"[4])
```



문자열 연산자4

- 예시

```
>>> print("안녕하세요"[0:2])
안녕
>>> print("안녕하세요"[1:3])
녕하
>>> print("안녕하세요"[2:4])
하세
```

- 대괄호 안에 넣는 숫자 둘 중 하나를 생략하는 경우

- 뒤의 값 생략 : n번째부터 끝의 문자까지
- 앞의 값 생략 : 0번째부터 뒤의 숫자 n번째 앞의 문자까지

```
[1:]
[:3]
```

```
>>> print("안녕하세요"[1:])
녕하세요
>>> print("안녕하세요"[:3])
안녕하
```

문자열 연산자

- **인덱싱** (indexing): 가리킨다
 - [] 기호 이용해 문자열의 특정 위치에 있는 문자 참조하는 것
- **슬라이싱** (slicing): 잘라낸다
 - [:] 기호 이용해 문자열 일부를 추출하는 것
 - 문자열 선택 연산자로 슬라이스해도 원본은 변하지 않음에 주의

```
>>> hello = "안녕하세요" → ❶  
>>> print(hello[0:2]) → ❷  
안녕  
>>> hello → ❸  
'안녕하세요'
```

- 예제: Pithon이라는 문자열을 Python으로 바꾸려면?

```
>>> a="Pithon"
```

```
>>> a[:1]
```

```
'P'
```

```
>>>a[2:]
```

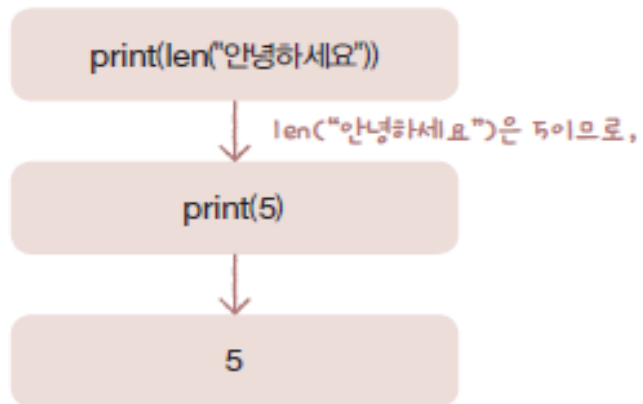
```
'thon'
```

```
>>>a[:1] + 'y' +a[2:]
```

문자열 길이

- len() 함수
 - 문자열 길이 구할 때 사용
 - 괄호 내부에 문자열 넣으면 문자열의 문자 개수 세어 줌
 - 중첩된 구조의 함수는 괄호 안쪽부터 먼저 실행

```
>>> print(len("안녕하세요"))  
5
```



키워드로 정리하는 핵심 포인트

- **자료형** : 자료의 형식
- **문자열** : 문자의 나열. 큰따옴표 혹은 작은따옴표로 입력
- **이스케이프 문자** : 문자열 내부에서 특수한 기능 수행하는 문자열
- **문자열 연산자** : 문자열 연결 연산자 (+), 문자열 반복 연산자 (*), 문자열 선택 연산자 ([]), 문자열 범위 선택 연산자 ([:])
- **type()** : 자료형 확인하는 함수
- **len()** : 문자열 길이 구하는 함수

확인문제

1. 문자열을 만드는 파이썬 구문의 빈칸에 알맞은 기호를 넣어보세요.

| 구문 | 의미 |
|---|----------------|
| <input type="text"/> 글자 <input type="text"/> | 큰따옴표로 문자열 만들기 |
| <input type="text"/> 글자 <input type="text"/> | 작은따옴표로 문자열 만들기 |
| <input type="text"/> 문자열 문자열 문자열 <input type="text"/> | 여러 문자열 만들기 |

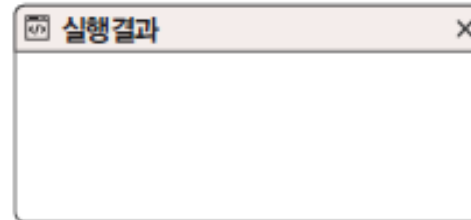
2. 이스케이프 문자의 의미를 보고 알맞은 기호 혹은 문자를 넣어보세요.

| 이스케이프 문자 | 의미 |
|----------------------|---------------|
| <input type="text"/> | 큰따옴표를 의미합니다. |
| <input type="text"/> | 작은따옴표를 의미합니다. |
| <input type="text"/> | 줄바꿈을 의미합니다. |
| <input type="text"/> | 탭을 의미합니다. |
| <input type="text"/> | \을 의미합니다. |

확인문제

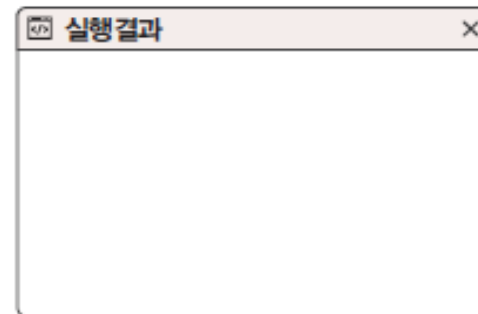
3. 다음 프로그램의 실행결과를 예측해보세요.

```
print("# 연습 문제")  
print("\\\\\\\\\\\\\\\\")  
print("-" * 8)
```



4. 다음 프로그램의 실행결과를 예측해보세요. 오류가 발생하는 것은 어느 행인가요? 그리고 그 이유는 무엇인가요?

```
print("안녕하세요"[1])  
print("안녕하세요"[2])  
print("안녕하세요"[3])  
print("안녕하세요"[4])  
print("안녕하세요"[5])
```



2주

2. 컨테이너 자료형

컨테이너 자료형 - 리스트 (List, [])

- **리스트 (list)**
 - 여러 가지 자료를 저장할 수 있는 자료
 - 자료들을 모아서 사용할 수 있게 해 줌
 - **대괄호 내부에 자료들 넣어 선언**

```
>>> array = [273, 32, 103, "문자열", True, False]
>>> print(array)
[273, 32, 103, '문자열', True, False]
```

1. 리스트 (List, [])

- 요소 (element)
 - 리스트의 대괄호 내부에 넣는 자료

```
[요소, 요소, 요소...]
```

```
>>> [1, 2, 3, 4]                                # 숫자만으로 구성된 리스트
[1, 2, 3, 4]
>>> ["안", "녕", "하", "세", "요"]              # 문자열만으로 구성된 리스트
['안', '녕', '하', '세', '요']
>>> [273, 32, 103, "문자열", True, False]        # 여러 자료형으로 구성된 리스트
[273, 32, 103, '문자열', True, False]
```

1. 리스트 (List, [])

- 리스트 내부의 요소 각각 사용하려면 리스트 이름 바로 뒤에 대괄호 입력 후 자료의 위치 나타내는 숫자 입력

```
list_a = [273, 32, 103, "문자열", True, False]
```

| | | | | | | |
|--------|-----|-----|-----|-----|------|-------|
| list_a | 273 | 32 | 103 | 문자열 | True | False |
| | [0] | [1] | [2] | [3] | [4] | [5] |

- **인덱스** (index)
 - 대괄호 안에 들어간 숫자

1. 리스트 (List, [])

```
>>> list_a = [273, 32, 103, "문자열", True, False]
>>> list_a[0]
273
>>> list_a[1]
32
>>> list_a[2]
103
>>> list_a[1:3]
[32, 103]
```

리스트의 슬라이싱

- 결과로 [32, 103] 출력

리스트 슬라이싱 예

```
>>> a=[1,2,3,4,5]
>>> b=a[:2]
>>> c=a[2:]
>>> b
[1, 2]
>>> c
[3, 4, 5]
```

처음부터 a[1]까지
즉 0이상 2미만

a[2]부터 마지막까지

리스트 연산하기

- 리스트 역시 + 기호를 사용해서 더할 수 있고 *기호를 사용해서 반복할 수 있다.

1. 리스트 더하기(+)

```
>>> a=[1,2,3]
>>> b=[4,5,6]
>>> a+b
[1, 2, 3, 4, 5, 6]
```

2. 리스트 반복하기(*)

```
>>> a=[1,2,3]
>>> a*3
[1, 2, 3, 1, 2, 3, 1, 2, 3]
```

3. 리스트 길이 구하기

```
>>> a=[1,2,3]
>>> len(a)
3
```

```
students = ['홍길동', '김길자', '이길순', '장영웅', '박석기', '송민규']

# 전체학생명단
print(students)

# 학생개인의 이름
print("인덱스 0 :", students[0])
print("인덱스 1 :", students[1])
print("인덱스 2 :", students[2])
print("인덱스 3 :", students[3])

# 전체 학생수
print("전체 학생수 :", len(students))
```

```
['홍길동', '김길자', '이길순', '장영웅', '박석기', '송민규']
인덱스 0 : 홍길동
인덱스 1 : 김길자
인덱스 2 : 이길순
인덱스 3 : 장영웅
전체 학생수 : 6
```

리스트의 수정과 삭제

```
>>> a=[1,2,3]
>>> a[2]=4
>>> a
[1, 2, 4]
```

- del함수 사용해 리스트 요소 삭제하기

```
>>> a=[1,2,3]
>>> del a[1]
>>> a
[1, 3]
```

```
>>> a=[1,2,3,4,5]
>>> del a[2:]
>>> a
[1, 2]
```

리스트 관련 함수와 메소드 예제

```
# 리스트 길이 : len()
students = ['홍길동', '김길자', '이길순', '장영웅', '박석기', '송민규']
print('전체 학생수:', len(students))

# 리스트에 요소 추가 : append()
students.append('유재석')
print(students)

# 리스트 요소 삭제 pop() 함수 사용1
students.pop() # 리스트 마지막 요소 삭제
print(students)

# 리스트 요소 삭제 pop() 함수 사용2
students.pop(3) # 인덱스3 요소 삭제
print(students)

# 리스트에 요소 삽입 : insert()
students.insert(2, '박명수')
print(students)
```

결과



```
전체 학생수: 6
['홍길동', '김길자', '이길순', '장영웅', '박석기', '송민규', '유재석']
['홍길동', '김길자', '이길순', '장영웅', '박석기', '송민규']
['홍길동', '김길자', '이길순', '박석기', '송민규']
['홍길동', '김길자', '박명수', '이길순', '박석기', '송민규']
```


리스트 관련 함수와 메소드 예제

```
# 리스트에 요소 삭제 : remove()
students.remove('박명수')
print(students)
```

```
# 리스트 정렬 : sort()
num_list = [5, 2, 4, 3, 1]
num_list.sort()
print(num_list)
```

```
# 리스트 확장 : extend()
mix_list = ['조세호', 20, True]
num_list.extend(mix_list)
print(num_list)
```

```
# 리스트에 포함된 요소 x의 개수 세기 : count()
print(students.count('홍길동'))
```

결과



```
['홍길동', '김길자', '이길순', '박석기', '송민규']
[1, 2, 3, 4, 5]
[1, 2, 3, 4, 5, '조세호', 20, True]
1
```

2. 컨테이너 자료형 - 튜플(Tuple, ())

- 리스트와 같지만 **요소(데이터) 수정 불가**

리스트



a = [1,2,3]

튜플



b = (1,2,3)

2. 컨테이너 자료형 - 튜플(Tuple, ())

- 리스트와 같지만 **요소(데이터) 수정 불가**
- 튜플 기본 구조

employee = ("홍길동", "김길자", "이길순") ← 소괄호

↑
변수명

↑
소괄호

```
employee = ("홍길동", "김길자", "이길순", "장영웅", "박석기")  
# 전체학생명단  
print(employee)
```



```
('홍길동', '김길자', '이길순', '장영웅', '박석기')
```

2. 컨테이너 자료형 - 튜플(Tuple, ())

- 튜플 요소 값 삭제 시 오류

```
t1 = (1,2, 'a','b')  
del t1[0]
```

Traceback (most recent call last):

File "<pyshell#1>", line 1, in <module>

del t1[0]

TypeError: 'tuple' object doesn't support item deletion

2. 컨테이너 자료형 - 튜플(Tuple, ())

- 튜플 요소 값 변경 시 오류

```
t1 = (1, 2, 'a', 'b')  
t1[0] = 'c'
```

Traceback (most recent call last):

File "<pyshell#1>", line 1, in <module>

t1[0]='c'

TypeError: 'tuple' object does not support item assignment

2. 컨테이너 자료형 - 튜플(Tuple, ())

- 인덱싱

```
t1 = (1, 2, 'a', 'b')  
print(t1[0])  
print(t1[3])
```

- 슬라이싱

```
t1 = (1, 2, 'a', 'b')  
print(t1[1:])  
  
(2, 'a', 'b')
```

2. 컨테이너 자료형 - 튜플(Tuple, ())

- 더하기

```
t1 = (1, 2, 'a', 'b')  
t2 = (3,4)  
print(t1 + t2)  
  
(1,2, 'a','b',3,4)
```

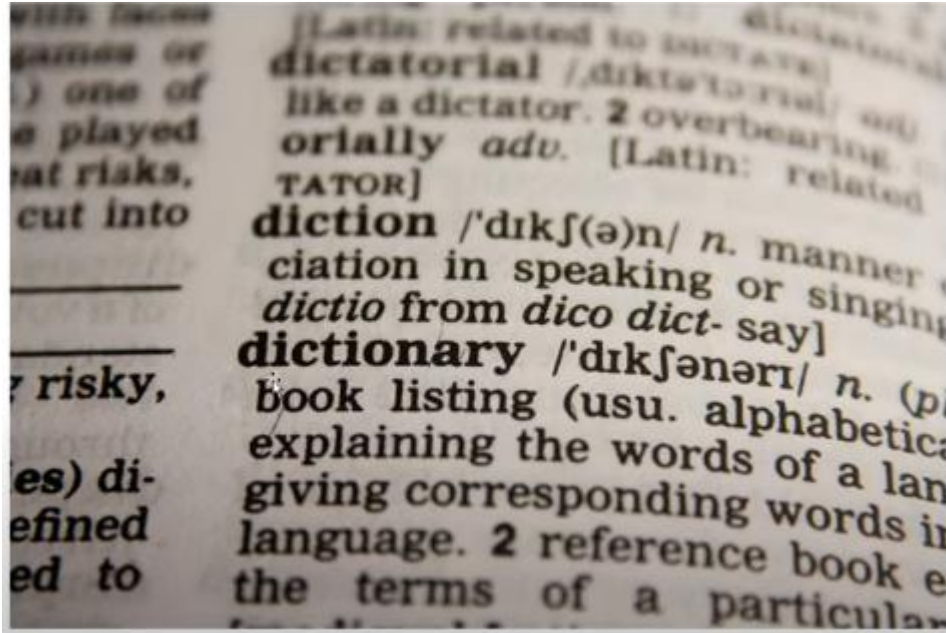
- 곱하기

```
print(t2 *3)  
  
(3, 4, 3, 4, 3,4)
```

튜플 관련 함수와 메소드

- 튜플 길이: `len()`
 - `len(employee)`
- 튜플 결합: `+`
 - `employee + ('a', 'b', 'c')`
- 데이터 슬라이싱: `[n : m]`
 - `employee[1:3]`
- 인덱스 검색 : `index()`
 - `employee.index('장영웅')`
- 데이터 개수 찾기 : `count()`
 - `employee.count('장영웅')`

3. 컨테이너 자료형 - 딕셔너리(Dictionary, {})



- 연관 배열(Associative array) 또는 해시(Hash)
- Key를 통해 Value를 얻는다.

컨테이너 자료형 - 딕셔너리(Dictionary, {})

- 키(key)와 밸류(value)를 이용한 데이터 관리
- 딕셔너리 기본 구조

dic = { 10:'홍길동', 20:'김길자', 30:'이으뜸' } ← 중괄호

↑
변수명

↑
중괄호

```
dic = {10:'홍길동', 20:'김길자', 30:'이으뜸'}  
  
# 전체 데이터  
print(dic)
```



```
{10: '홍길동', 20: '김길자', 30: '이으뜸'}
```

- 딕셔너리 쌍 추가하기

```
>>> a = { 1: 'a' }  
>>> a[2] = 'b'  
>>> a  
{2: 'b', 1: 'a'}
```

- 딕셔너리 요소 삭제하기

```
>>> del a[1]  
>>> a  
{2: 'b'}
```

컨테이너 자료형 - 딕셔너리(Dictionary, {})

- 딕셔너리에서 Key 사용해 Value 얻기

```
>>> grade = { 'pey' : 10 , 'julliet' : 99 }  
>>> grade['pey']  
10  
>>> grade['julliet']  
99
```

- 딕셔너리 만들 때 주의할 사항(key를 중복하면 안됨, 만약 중복되면 마지막 것만 남음)

```
>>> a = {1:'a' , 1:'b'}  
>>> a  
{1: 'b'}
```

컨테이너 자료형 - 딕셔너리(Dictionary, {})

- Key 리스트 만들기(keys)

```
a = { 'name' : 'pey' , 'phone' : '01099993333' , 'birth' : '1225' }  
print(a.keys())  
dict_keys(['name' , 'phone' , 'birth'])
```

- Value 리스트 만들기 (values)

```
print(a.values())  
  
dict_values(['pey' , '01099993333' , '1225'])
```

컨테이너 자료형 - 딕셔너리(Dictionary, {})

- Key, Value 쌍 얻기(items)

```
a = { 'name' : 'pey' , 'phone' : '01099993333' , 'birth' : '1225' }  
print(a.items())  
dict_keys([('name' , 'pey') , ('phone' , '01099993333'), ('birth', '1225' )])
```

- Key : Value 쌍 모두 지우기(clear)

```
a.clear()  
print(a)  
{}
```

컨테이너 자료형 - 딕셔너리(Dictionary, {})

- Key로 Value 얻기(get)

```
a = { 'name' : 'pey' , 'phone' : '01099993333' , 'birth' : '1225' }
```

```
print(a.get('name'))
```

```
print(a.get('phone'))
```

```
'pey'
```

```
01099993333'
```

```
print(a["name"])
```

```
print(a["birth"])
```

```
'pey'
```

```
01099993333'
```

그러나, 만약에 `print(a["ame"])` 이런 경우 'Key Error'를 발생시키고,
`print(a.get("ame"))`일 경우는 'None'을 출력한다.

컨테이너 자료형 - 딕셔너리(Dictionary, {})

- 해당 Key가 딕셔너리 안에 있는지 조사하기(in)

```
a = { 'name' : 'pey' , 'phone' : '01099993333' , 'birth' : '1225' }  
print('name' in a)  
True  
  
print('email' in a)  
False
```


딕셔너리 관련 함수와 메소드 예제

```
cabinet = {3:'유재석', 100:'김태호'}  
print(cabinet[3])  
print(cabinet[100])  
  
print(cabinet.get(3))  
  
print(3 in cabinet) # True  
  
print(5 in cabinet) # False  
cabinet = {'A-3':'유재석', 'B-100':'김태호'}  
print(cabinet)
```

결과



```
유재석  
김태호  
유재석  
True  
False  
{'A-3': '유재석', 'B-100': '김태호'}
```

확인 문제

- 다음 표를 딕셔너리로 만드시오.

| 항목 | 값 |
|-------|------|
| Name | 홍길동 |
| birth | 1128 |
| age | 30 |