

파이썬의 데이터 다루기

이번 장에서는 파이썬에서 데이터 다루는 방법을 알아봅니다.



1. 숫자 다루기
2. 문자 다루기
3. 리스트
4. 튜플
5. 딕셔너리





1. 숫자 다루기

□ 파이썬의 숫자

- 정수, 실수, 복소수

□ 예

```
>>> a = 2020 + 100
```

☞ 정수

```
>>> a
```

```
2120
```

```
>>> b = 22 / 7
```

☞ 실수

```
>>> b
```

```
3.142857142857143
```

```
>>> c = 2 + 3j
```

☞ 복소수 : i 대신 j를 사용

```
>>> d = 1 + 5j
```

```
>>> c+d
```

```
(3+8j)
```



1. 숫자 다루기 : 연산자

□ 사칙 연산자

연산	연산자
더하기	+
빼기	-
곱하기	*
제곱 구하기	**
나눗셈의 몫 구하기	//
나눗셈의 나머지 구하기	%
나누기	/





1. 숫자 다루기 : 연산자

□ 논리 연산자

연산자	설명
and	피연산자 값이 모두 참일때만 참
or	피연산자 값이 모두 거짓일때만 거짓
not	피연산자 값의 참/거짓을 바꿈





1. 숫자 다루기 : 연산자

□ 비트 연산자

연산자	설명
<<	왼쪽 시프트 연산자
>>	오른쪽 시프트 연산자
&	논리곱(AND) 연산자
	논리합(OR) 연산자
^	배타적 논리합(XOR) 연산자
~	보수(NOT) 연산자



supervisor
Entry level
Assistant manager
supervisor
Assistant manager
Assistant manager
Assistant manager
supervisor
supervisor



1. 숫자 다루기 : math 모듈을 이용한 연산

□ 모듈 : 함수나 변수 또는 클래스를 모아둔 파이썬 파일.
파이썬 프로그램에서 불러와서 사용 가능

□ math 모듈에서 제공하는 함수를 호출하여 연산

□ 예

```
>>> import math
```

☞ import : 사용할 모듈 선언 (나중에 다룸)

```
>>> math.factorial(5)
```

```
120
```

```
>>> math.sqrt(4)
```

```
2.0
```

```
>>> math.pow(3, 3)
```

```
27.0
```





2. 문자 다루기

□ 파이썬의 문자, 문자열

- 작은 따옴표(') 또는 큰 따옴표(")의 쌍으로 감싸서 표현
- 문자열 안에 따옴표를 넣고 싶다면 \ 사용
- 문자열이 여러 줄에 걸쳐 있는 경우는 따옴표 3개의 쌍으로 감쌘

□ 예

```
>>> e = 'a'
>>> f = 'abc'
>>> g = 'abc\ndef'
>>> h = '''welcome to
python'''
>>> h
'welcome to\npython'
```

☞ 문자열 입력 중에 엔터 키 입력





2. 문자 다루기 (계속)

□ 인덱스

- 0부터 시작하며, 음수 인덱스도 가능
- 음수 인덱스는 '끝에서 몇번째..' 라는 의미임

□ 예

str = 'hello'

str	h	e	l	l	o
인덱스	0	1	2	3	4
	-5	-4	-3	-2	-1





2. 문자 다루기 (계속)

□ 문자열 결합 : + 연산자

□ 예

```
>>> h = 'hello'
>>> w = 'world'
>>> h_w = h + ',' + w
>>> h_w
'hello, world'
```

☞ 문자열 결합





2. 문자 다루기 (계속)

□ 문자열 분리(slicing) : [] 연산자

[시작 인덱스 : 끝 인덱스 : 스텝]

- 시작 인덱스: 범위의 시작
- 끝 인덱스: 범위의 끝
- 스텝: 간격 (생략 시 기본값은 1)

□ 예

```
>>> h='hello, world'
```

```
>>> h[0:5]
```

```
'hello'
```

```
>>> h[-3:]
```

```
'rld'
```

```
>>> h[0:10:2]
```

```
'hlo o'
```

☞ 문자열의 (0~4)

☞ 끝에서 3번째부터

☞ 문자열의 (0~9), 스텝 값 2개씩 띄어서



2. 문자 다루기 (계속)

□ in 연산자

- 문자열에 원하는 내용이 있는 지 확인

□ not in 연산자

- 문자열에 원하는 내용이 없는 지 확인

□ 예

```
>>> a = 'hello world'
```

```
>>> 'hello' in a
```

```
True
```

```
>>> 'W' in a
```

```
False
```

```
>>> 'W' not in a
```

```
True
```



2. 문자 다루기 (계속)

□ len()

- 문자열 길이를 출력

□ 예

```
>>> a = 'hello world'
>>> len(a)
11
```

과제: python의 common sequence operation 중에서 10개를 조사하시오.

<https://docs.python.org/3/library/stdtypes.html>



2. 문자 다루기 (계속) : 문자열 메소드

메소드	설명
startswith()	<p>원본 문자열이 매개변수로 입력한 문자열로 시작되는지를 판단. 결과는 True 또는 False</p> <pre>>>> a = 'Hello' >>> a.startswith('He') True >>> a.startswith('lo') False</pre>
endswith()	<p>원본 문자열이 매개변수로 입력한 문자열로 끝나는지를 판단. 결과는 True 또는 False</p> <pre>>>> a = 'Hello' >>> a.endswith('He') False >>> a.endswith('lo') True</pre>
find()	<p>원본 문자열 안에 매개변수로 입력한 문자열이 존재하는 위치를 앞에서부터 찾음. 존재하지 않으면 -1</p> <pre>>>> a = 'Hello' >>> a.find('l') 2 >>> a.find('H') 0 >>> a.find('K') -1</pre>



2. 문자 다루기 (계속) : 문자열 메소드

메소드	설명
rfind()	<p>원본 문자열 안에 매개변수로 입력한 문자열이 존재하는 위치를 뒤에서부터 찾음. 존재하지 않으면 -1</p> <pre> >>> a = 'Hello' >>> a.rfind('H') 0 >>> a.rfind('lo') 3 >>> a.rfind('M') -1 </pre>
count()	<p>원본 문자열 안에 매개변수로 입력한 문자열이 몇 개 있는 지를 계산</p> <pre> >>> a = 'Hello' >>> a.count('l') 2 </pre>
lstrip()	<p>원본 문자열 왼쪽에 있는 공백을 제거</p> <pre> >>> ' Left Strip'.lstrip() 'Left Strip' </pre>
rstrip()	<p>원본 문자열 오른쪽에 있는 공백을 제거</p> <pre> >>> 'Right Strip '.rstrip() 'Right Strip' </pre>
strip()	<p>원본 문자열 양쪽에 있는 공백을 제거</p> <pre> >>> ' Strip '.strip() 'Strip' </pre>



2. 문자 다루기 (계속) : 문자열 메소드

메소드	설명
isalpha()	<p>원본 문자열이 숫자와 기호를 제외한 알파벳(영문, 한글 등)으로만 이루어져 있는지를 평가.</p> <pre>>>> 'ABCDefgh'.isalpha() True >>> '123ABC'.isalpha() False</pre>
isnumeric()	<p>원본 문자열이 수로만 이루어져 있는지를 평가.</p> <pre>>>> '1234'.isnumeric() True >>> '123ABC'.isnumeric() False</pre>
isalnum()	<p>원본 문자열이 알파벳과 수로만 이루어져 있는지를 평가.</p> <pre>>>> '1234ABC'.isalnum() True >>> '1234'.isalnum() True >>> 'ABC'.isalnum() True >>> '1234 ABC'.isalnum() False</pre>



2. 문자 다루기 (계속) : 문자열 메소드

메소드	설명
replace()	<p>원본 문자열에서 특정 문자열을 다른 문자열로 변경.</p> <pre>>>> a = 'Hello, World' >>> b = a.replace('World', 'Korea') >>> a 'Hello, World' >>> b 'Hello, Korea'</pre>
split()	<p>매개변수로 입력한 문자열을 기준으로 원본 문자열을 나누어 리스트로 생성. (리스트는 목록을 다루는 자료형이며, 나중에 다룸)</p> <pre>>>> a = 'Apple, Orange, Kiwi' >>> b = a.split(',') >>> b ['Apple', ' Orange', ' Kiwi'] >>> type(b) <class 'list'></pre>
upper()	<p>원본 문자열을 모두 대문자로 바꿈.</p> <pre>>>> a = 'lower case' >>> b = a.upper() >>> a 'lower case' >>> b 'LOWER CASE'</pre>



2. 문자 다루기 (계속) : 문자열 메소드

메소드	설명
lower()	<p>원본 문자열을 모두 소문자로 바꿈</p> <pre>>>> a = 'UPPER CASE' >>> b = a.lower() >>> a 'UPPER CASE' >>> b 'upper case'</pre>
format()	<p>형식을 갖춘 문자열을 만들 때 사용. 문자열 안에 중괄호 {와 }로 다른 데이터가 들어갈 자리를 만들어 두고 format() 함수를 호출할 때 이 자리에 들어갈 데이터를 순서대로 넣어주면 원하는 형식의 문자열을 만들어 낼 수 있음.</p> <pre>>>> a = 'My name is {0}. I am {1} years old.'.format('Mario', 40) >>> a 'My name is Mario. I am 40 years old.'</pre> <pre>>>> b = 'My name is {name}. I am {age} years old.'.format(name='Luigi', age=35) >>> b 'My name is Luigi. I am 35 years old.'</pre>



2. 문자 다루기 (계속)

□ 실습 : 문자열 메소드

- 다음과 같은 문자열 a:b:c:d가 있다. 문자열의 replace 함수를 사용하여 a#b#c#d로 바꿔서 출력해보시오. (replace)
- [1, 3, 5, 4, 2] 리스트를 [5, 4, 3, 2, 1]로 만들어 보시오. (리스트 내장함수)
- ['Life', 'is', 'too', 'short'] 리스트를 Life is too short 문자열로 만들어 출력해 보시오. (join)





2. 문자 다루기 : 문자열 포매팅

□ 문자열에 값 대입하기

```
>>> "I have %d apples." %3
'I have 3 apples.'
```

```
>>> "I have %s apples." %"three"
'I have three apples.'
```

```
>>> number = 3
>>> "I have %d apples." %number
'I have 3 apples.'
```

```
>>> days = 3
>>> "I have %d apples for % days." %(number, day)
'I have 3 apples for 3 days.'
```

코드	의미
%d	정수
%c	문자 1개
%s	문자열
%f	실수
%o	8진수
%x	16진수
%%	% 자체



2. 문자 다루기 : 숫자와 문자 변환

❑ 문자열을 숫자로 변환 : `int()`, `float()`, `complex()`

❑ 숫자를 문자열로 변환 : `str()`

❑ 예

```
>>> int('12345')
```

```
12345
```

```
>>> float('34.1')
```

```
34.1
```

```
>>> complex('1+2j')
```

```
(1+2j)
```

```
>>> text = "I have " + str(4) + " apples"
```

```
>>> text
```

```
'I have 4 apples'
```



- input() : 키보드 입력을 받아 문자열로 저장

```
print("수를 입력하시오: ")
a = input()
print("수를 입력하시오: ")
b = input()
result = int(a) + int(b)
print("{0} + {1} = {2}".format(a, b, result))
```





3. 리스트

□ 리스트 : 데이터의 목록으로 [와]를 사용

□ 예

```
>>> a = ['apple', 'banana', 'orange']
```

```
>>> a
```

```
['apple', 'banana', 'orange']
```

```
>>> a[0]
```

```
'apple'
```

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

```
'banana'
```

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

```
>>> b
```

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



3. 리스트 (계속)

□ 리스트의 분할(slicing) : 문자열의 분할과 같은 방식

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

```
>>> a[0:2]
```

```
[1, 2]
```

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

```
[3, 4]
```

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

```
[1, 2]
```

□ 리스트의 결합

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

```
>>> b = [5, 6, 7]
```

```
>>> a + b
```

```
[1, 2, 3, 4, 5, 6, 7]
```




3. 리스트 (계속)

□ 리스트 내의 특정 값 변경

```
>>> a = [1, 2, 3, 4]
>>> a[2] = 30
>>> a
[1, 2, 30, 4]
```

□ 리스트의 길이 값 구하기

```
>>> a = [1, 2, 3, 4]
>>> len(a)
4
```





3. 리스트 (계속)

□ 리스트 함축 (list comprehension)

< 예 1 >

```
>>> a = [ x ** 2 for x in range(10) ]
```

```
>>> a
```

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

range(10)에 속하는 모든 정수에 대해 x^2 을 계산하여 리스트를 생성하였음

<예 2>

```
>>> a = [ x for x in range(10) if x % 2 == 0 ]
```

```
>>> a
```

```
[0, 2, 4, 6, 8]
```

range(10)에 속하는 모든 정수 중에서 짝수 리스트를 생성하였음



3. 리스트 (계속)

메소드	설명
append()	<p>리스트의 끝에 새 요소를 추가</p> <pre>>>> a = [1, 2, 3] >>> a.append(4) >>> a [1, 2, 3, 4]</pre>
extend()	<p>기존 리스트에 다른 리스트를 이어 붙이기 (+ 연산자와 같은 기능)</p> <pre>>>> a = [1, 2, 3] >>> a.extend([4, 5, 6]) >>> a [1, 2, 3, 4, 5, 6]</pre>
insert()	<p>첨자로 명시한 리스트 내의 위치에 새 요소를 삽입. insert(첨자, 데이터)의 형식으로 사용합니다.</p> <pre>>>> a = [2, 4, 5] >>> a.insert(0, 1) >>> a [1, 2, 4, 5] >>> a.insert(2, 3) >>> a [1, 2, 3, 4, 5]</pre> <p># 0 위치(첫 번째)에 데이터 1을 삽입합니다.</p> <p># 2 위치(세 번째)에 데이터 3을 삽입합니다.</p>



1. 리스트 (계속)

메소드	설명
remove()	<p>매개 변수로 입력한 데이터를 리스트에서 찾아 발견한 첫 번째 요소를 제거</p> <pre>>>> a = ['BMW', 'BENZ', 'VOLKSWAGEN', 'AUDI'] >>> a.remove('BMW') >>> a ['BENZ', 'VOLKSWAGEN', 'AUDI']</pre>
pop()	<p>리스트의 마지막 요소를 뽑아내어 리스트에서 제거</p> <pre>>>> a = [1, 2, 3, 4, 5] >>> a.pop() 5 >>> a [1, 2, 3, 4] >>> a.pop() 4 >>> a [1, 2, 3]</pre> <p>마지막이 아닌 특정 요소를 제거할 때에는 pop() 메소드에 제거하고자 하는 요소의 인덱스를 입력.</p> <pre>>>> a = [1, 2, 3, 4, 5] >>> a.pop(2) # 3번째 요소 제거 3 >>> a [1, 2, 4, 5]</pre>



3. 리스트 (계속)

메소드	설명
index()	<p>입력한 데이터와 일치하는 첫번째 요소의 첨자 구하기. 찾고자 하는 데이터가 없으면 오류 발생</p> <pre>>>> a = ['abc', 'def', 'ghi'] >>> a.index('def') 1 >>> a.index('jkl') Traceback (most recent call last): File "<pyshell#2>", line 1, in <module> a.index('jkl') ValueError: 'jkl' is not in list</pre>
count()	<p>매개변수로 입력한 데이터와 일치하는 요소의 개수 구하기</p> <pre>>>> a = [1, 100, 2, 100, 3, 100] >>> a.count(100) 3 >>> a.count(200) 0</pre>



3. 리스트 (계속)

메소드	설명
sort()	<p>리스트 내의 요소를 정렬. (매개변수로 <code>reverse = True</code>를 입력하면 내림차순, 아무 것도 입력하지 않으면 오름차순)</p> <pre>>>> a = [3, 4, 5, 1, 2] >>> a.sort() >>> a [1, 2, 3, 4, 5] >>> a.sort(reverse = True) >>> a [5, 4, 3, 2, 1]</pre>
reverse()	<p>리스트 내 요소의 순서를 반대로 만들기.</p> <pre>>>> a = [3, 4, 5, 1, 2] >>> a.reverse() >>> a [2, 1, 5, 4, 3] >>> b = ['안', '녕', '하', '세', '요'] >>> b.reverse() >>> b ['요', '세', '하', '녕', '안']</pre>



4. 튜플

- 튜플은 데이터의 목록으로 데이터 변경이 불가함
 - 리스트는 데이터의 목록으로 데이터 변경 가능

□ 예

```
>>> a = ('apple', 'banana', 'orange')
```

```
>>> a
```

```
('apple', 'banana', 'orange')
```

```
>>> type(a)
```

```
<class 'tuple'>
```

```
>>> b = 1, 2, 3, 4
```

```
>>> b
```

```
(1, 2, 3, 4)
```



4. 튜플 (계속)

□ 요소가 1개인 튜플을 생성할 때

```
>>> a = (1,)
>>> a
(1,)
>>> type(a)
<class 'tuple'>
```

```
>>> a = 1,
>>> a
(1,)
```

```
cf) >>> a = (1)
>>> a
1
>>> type(a)
<class 'int'>
```




4. 튜플 (계속)

□ 튜플의 슬라이싱

```
>>> a = (1, 2, 3, 4)
>>> a[:3]
(1, 2, 3)
>>> a[3:4]
(4,)
```

□ 튜플의 결합

```
>>> a = (1, 2, 3)
>>> b = (4, 5, 6)
>>> c = a + b
>>> c
(1, 2, 3, 4, 5, 6)
```



4. 튜플 (계속)

□ 튜플의 패킹(packaging)

```
>>> a = 1, 2, 3, 4  
>>> a  
(1, 2, 3, 4)
```

☞ 여러 데이터를 하나로 묶는 것

□ 튜플 언패킹(unpacking)

```
>>> a = (4, 5, 6)  
>>> apple, banana, orange = a  
  
>>> apple  
4  
>>> banana  
5
```

☞ 튜플의 각 요소를 변수에 할당.
튜플의 요소와 변수 개수가 일치해야 함



4. 튜플 (계속)

메소드	설명
index()	<p>매개변수로 입력한 데이터와 일치하는 튜플 내 요소의 첨자 구하기. 찾고자 하는 데이터와 일치하는 요소가 없으면 에러 발생</p> <pre>>>> a = ('abc', 'def', 'ghi') >>> a.index('def') 1 >>> a.index('jkl') Traceback (most recent call last): File "<pyshell#4>", line 1, in <module> a.index('jkl') ValueError: tuple.index(x): x not in tuple</pre>
count()	<p>매개변수로 입력한 데이터와 일치하는 요소의 개수 구하기</p> <pre>>>> a = (1, 100, 2, 100, 3, 100) >>> a.count(100) 3 >>> a.count(200) 0</pre>



4. 튜플

□ 리스트 VS 튜플

리스트 - 수정 **가능**

튜플 - 수정 **불가능**

```
Python 3.7.4 Shell
File Edit Shell Debug Options Window Help
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul 8 2019, 20:34:20) [MSC v.1916 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> my_list = [1, 2, 3]
>>> type(my_list)
<class 'list'>
>>> my_tuple = (1, 2, 3)
>>> type(my_tuple)
<class 'tuple'>
>>> my_list[1] = "two"
>>> my_list
[1, 'two', 3]
>>> my_tuple[1] = "two"
Traceback (most recent call last):
  File "<pyshell#6>", line 1, in <module>
    my_tuple[1] = "two"
TypeError: 'tuple' object does not support item assignment
>>> |
```

Ln: 17 Col: 4



5. 딕셔너리(dictionary)

□ 딕셔너리

- 리스트와 비슷하지만, 첨자를 숫자 대신 임의의 값을 사용할 수 있으며 첨자를 키(key)라고 부름
 - { } 를 사용해서 생성

□ 예

```
>>> a = {}  
>>> a['name'] = 'Ryu Yeonseung'  
>>> a['sex'] = 'male'  
>>> a['position'] = 'professor'  
>>> type(a)  
<class 'dict'>
```

```
>>> a  
{'name': 'Ryu Yeonseung', 'sex': 'male', 'position': 'professor'}
```



5. 딕셔너리 (계속)

□ 딕셔너리 메소드

- 딕셔너리 메소드는 객체를 돌려줌
- 딕셔너리 객체는 리스트로 변환하지 않더라도 기본 반복 구문(예: for문)을 실행할 수 있음
- 하지만 리스트의 append, insert, pop, remove, sort 함수는 수행할 수 없음.
- 리스트 메소드를 사용하려면 캐스팅 필요함 (`list(a.keys())`)





5. 딕셔너리 (계속)

메소드	설명
keys()	<p>Key만을 모아서 dict_keys 객체를 돌려줌</p> <pre>>>> a {'name': 'Ryu Yeonseung', 'sex': 'male', 'position': 'professor'} >>> a.keys() dict_keys(['name', 'sex', 'position'])</pre>
values()	<p>values 값만 모아서 dict_values 객체를 돌려준다.</p> <pre>>>> a {'name': 'Ryu Yeonseung', 'sex': 'male', 'position': 'professor'} >>> a.values() Dict_values(['Ryu Yeonseung', 'male', 'professor'])</pre>



5. 딕셔너리 (계속)

메소드	설명
items()	<p>Key,value 쌍을 dict_items 객체를 돌려줌.</p> <pre>>>> a {'name': 'Ryu Yeonseung', 'sex': 'male', 'position': 'professor'} >>> a.items() dict_items([('name', 'Ryu Yeonseung'), ('sex', 'male'), ('position', 'professor')])</pre>
get('key')	<p>Key에 대응되는 Value를 돌려줌. 존재하지 않는 키(nokey)로 값을 가져오려고 할 경우 a['nokey']는 Key 오류를 발생시키고 a.get('nokey')는 None을 돌려줌.</p> <pre>>>> a = {'name': 'Ryu Yeonseung', 'sex': 'male', 'position': 'professor'} >>> a.get('name') 'Ryu Yeonseung'</pre>



5. 딕셔너리 (계속)

□ 딕셔너리에서 요소 삭제

```
>>> a.pop('position')  
'professor'  
>>> a
```

👉 키가 'position'인 요소를 삭제

```
>>> del a['position']
```

□ 딕셔너리 삭제

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





5. 딕셔너리

□ 딕셔너리에서 키 값 존재 확인

```
>>> a = {'name': 'Ryu Yeonseung', 'sex': 'male', 'position': 'professor'}
```

```
>>> 'name' in a
```

True

```
>>> 'email' in a
```

False

