Python 프로그래밍 - I

2023년도 1학기

조상구

Symbols name to learn

	Name		Name
~	Tilde	'	Single quotation mark, Apostrophe
`	Grave	"	Double quotation mark
!	Exclamation mark	/	Slash, Virgule
@	At sign, Commercial at	\	Back slash
#	Number sign, Crosshatch, Pound sign	|	Vertical bar
$	Dollar sign	,	Comma
%	Percent sign	.	Dot, Period, Full stop
^	Hat sign, Circumflex	?	Question mark
&	Ampersand	( )	(Left / Right) parenthesis
*	Asterisk, Star sign	{ }	(Left / Right) braces, curved parenthesis
–	Dash, Hyphen, Minus sign	[ ]	(Left / Right) bracket, squared parenthesis
_	Underscore, Underline	< >	(Left / Right) angle bracket, pointed brackets
+	Plus sign	←→	(Left / Right) [unidirectional] arrows
=	Equal sign	↔	Bidirectional arrow
:	Colon		
;	Semicolon		

Python 배우기

- 사칙연산과 논리식

컴퓨터에게 시킬 수 있는 일은?

- 컴퓨터가 하는 가장 기본적인 기능은?

- 연산의 종류는 무엇이 있나?
  - 사칙 연산만 있나?
  - 논리 연산이란 무엇인가? Boolean?

- 연산간의 우선 순위가 있는가? 있다면 어떤 것이 먼저인가?

숫자 연산

```
# 사칙연산(Elementary arithmetic)

>>> 2+3

>>> 2*3

>>> 2-1

>>> 2-3*4

>>> (2-3)-4

>>> 2*3

>>> 2/3
```

```
>>> 2%3

>>> 5//2

>>> 2//3

>>> 2**3
```

논리 연산(Boolean)

```
>>> # 논리값들의 'and', 'or', 'not' 연산
>>> True and True
>>> not True
>>> not False
```

```
>>> # 수자간 비교
>>> 2 == 3
>>> 2 is 2
>>> 2 !=2
>>> not (2==3)
>>> 2>3
>>> 2<3
>>> 2>=3
>>> 2<=3
```

```
>>> (2 >=3 and 2<=3)
>>> (2>=3 or 2 <=3)
>>> not (2>=3 or 2 <=3)


>>> 2 in [1,2,3] # list type (python built in)
```

```
>>> # True/False는 무슨 의미인가?
>>> True*1
>>> True + 100
>>> False*1
>>> False + 100
```

데이터의 종류- type()

```
>>> # type 함수로 데이터 종류 확인
>>> type(1)

>>> type('a')

>>> type(1.2)

>>> type(1-2j)
```

```
>>> 1.5e10 # e10 → 지수형태 수자
>>> type(1.5e10)

>>> str(23) # 수자를 문자로 변환
>>> type(str(23))

>>> int('23') # 문자를 수자로 변환
>>> type(int('23'))

>>> float(23) # 정수를 실수로 변환
>>> type(float(23))

>>> type(True)
```

문자열 – string

```
>>> print('Hello, there!)

>>> 'Hello' == 'Hello'
>>> 'Hello' == "Hello"
>>> 'Hello' == 'hello'

>>> # ' 가 인쇄되도록 하는 방법
>>> print('₩n'')        # ₩(backslash)와 함께
>>> print("₩'")
>>> print("'")          # ""안에 넣어
```

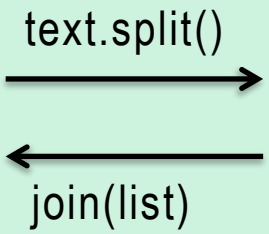
```
>>> # 두 개의 문자열 합치기
>>> print('Hello' + 'there!')

>>> # 다음 줄로 넘기기
>>> print('Hello' + '\' + 'there!')
```

문자열 – strip(), len()

```
# 문자열과 관련 함수 또는 메소드들
# function or method


>>> # 공백제거
>>> '      Hello      '.strip()              # lstrip, rstrip 한쪽만 제거


>>> # 문자열 길이 확인
>>> len('python')
>>> len('    python    ')                    # 공백도 문자길이로 인식
>>> len('    python    '.strip())            # 공백을 먼저 제거하고 문자 길이를 확인
```

문자열 – join(), split()

```
>>> 'Hello' + ' ' + 'World'                    # '+'를 이용해서 합치기
>>> ' '.join(['Hello', 'World'])               # 리스트(list)의 문자열을 합쳐주는 메소드가 join()
>>> ''.join(['Hello', 'World'])
>>> '//'.join(['Hello', 'World'])

>>> 'aBc'.upper()                              # 대문자로
>>> 'aBC'.lower()                              # 소문자로

>>> 'winter is over'.split()                   # 문자열을 공백으로 자르기
>>> 'Wow, winter is over, spring'.split(',')   # 문자열 특정 문자로 자르기
```



: 텍스트 마이닝
(Text mining)

문자열 – 문자열의 순서

```
>>> ‘abd def ghi jkl mno’.find(‘d’)          # 공백을 포함한 문자(열) 찾기
>>> ‘abd def ghi jkl mno’.rfind(‘d’)         # 문자열 내에서 오른쪽 부터 문자(열) 찾기
>>> ‘abd def ghi jkl mno’[0]                 # 문자열중 0번째 문자 가져오기
>>> ‘abd def ghi jkl mno’[4]                 # 문자열중 4번째 문자 가져오기
>>> ‘abd def ghi jkl mno’[2:5]               # 문자열중 2번째부터 4번째 문자 가져오기

>>> ‘0123456789’[0]
>>> int(‘0123456789’[3:7])                   # 문자열을 정수로 변환
>>> list(‘0123456789’[3:7])

>>> ‘0123456789’[3:6]*2
>>> int(‘0123456789’[3:6])*2
>>> list(‘0123456789’[3:6])*2
```

숫자 연산 – 변수(variable)를 사용하여

```
# 사칙연산(Elementary arithmetic)

# Addition
>>> a = 10                                  # a has 10
>>> b = 5                                   # b has 10
>>> c = a + b
>>> print("Addition: ", c)

# Subtraction
>>> c = a - b
>>> print("Subtraction: ", c)

# Multiplication
>>> c = a * b
>>> print("Multiplication: ", c)

# Division
>>> print("Division: ", c)                  # Note that division results in a float by default
```

숫자 연산 – 변수(variable)

```
# 사칙연산(Elementary arithmetic)

# Integer Division (몫)
>>> a = 10 ; b = 3
>>> c = a // b
>>> print("Integer Division: ", c)

# Modulo (나머지)
>>> c = a % b
>>> print("Modulo: ", c)

# Power
>>> c = a**b
>>> print(c)
```

숫자 연산 – 변수(variable)

```
# 사칙연산(Elementary arithmetic)

# Integer Division (몫)
>>> a = 10 ; b = 3
>>> c = a // b
>>> print("Integer Division: ", c)

# Modulo (나머지)
>>> c = a % b
>>> print("Modulo: ", c)

# Power
>>> c = a**b
>>> print(c)
```

```
# 메모리 주소

>>> id(2)
>>> a = 2  # 2,3,4,5를 입력하면서 주소 확인
>>> id(a)

>>> for i in range(11):
        print(id(i))
```

입력과 출력(input, output)

```
# 수자를 입력 받아 여러 형태로 출력

# 수자와 문자를 구분
>>> input('자연수를 입력하세요: ')
>>> x = input('자연수를 입력하세요: ')
>>> y = input('자연수를 하나 더 입력하세요: ')
>>> print(x+y)


# 수자와 문자를 구분
>>> input('자연수를 입력하세요: ')
>>> x = int(input('자연수를 입력하세요: '))
>>> y = int(input('자연수를 하나 더 입력하세요: '))
>>> print(x+y)

# f string사용하기
>>> print(f'{x}과 {y}을 더한 x+y의 결과는 {x+y}')
```

```
for x in range(1,6):
  print(x, x*x, x*x*x)
```

입력과 출력(input, output)

```
# 수자를 입력 받아 여러 형태로 출력

# 수자와 문자를 구분
>>> input('자연수를 입력하세요: ')
>>> x = input('자연수를 입력하세요: ')
>>> y = input('자연수를 하나 더 입력하세요: ')
>>> print(x+y)


# 수자와 문자를 구분
>>> input('자연수를 입력하세요: ')
>>> x = int(input('자연수를 입력하세요: '))
>>> y = int(input('자연수를 하나 더 입력하세요: '))
>>> print(x+y)

# f string사용하기
>>> print(f'{x}과 {y}을 더한 x+y의 결과는 {x+y}')
```

```
1   1   1
2   4   8
3   9   27
4  16   64
5  25  125
```

```
for x in range(1,6):
  print(x, x*x, x*x*x) # x**3
```

입력과 출력(input, output)

```
# 수자를 입력 받아 여러 형태로 출력


>>> r'그냥₩t표시되는₩n₩n문자열'          # 눈에 보이는 대로 값을 가짐
>>> '그냥₩t표시되는₩n₩n문자열'

>>> print('그냥₩t표시되는₩n₩n문자열')     # 특수문자 ₩t(tab)과 ₩n(줄 바꿈) 들이 작동
>>> print(r'그냥₩t표시되는₩n₩n문자열')    # 그냥 보이는 대로 출력
```

변수명 사용하기

- 알파벳, 한글, '_'(underscore), 수자 등을 혼용
  - abc, ABC, _abc, _abc_, abc_0
- 대소문자 구분(case sensitive)
  - abc, Abc, aBc, abC,….
- 숫자로 시작할 수 없음
  - 5abc(X), _3abc(X)
- 문자, 숫자, _(underscore)로 만들 수 있음
  - ['a' for _ in range(5)]

변수명 사용하기

- 형식
  - snake_case(변수명, 파일명에 주로 사용)
  - PascalCase(혹은 UpperCamelCase, class명 등에 사용)
  - Kebab-case

- 예약어(reserved words)는 변수로 사용할 수 없음
  - and, or, not, if, elif, break, while, def, class ….

- 피해야 할 단독 문자들
  - ‘l’ (‘L’의 소문자, 숫자 ‘1’과 대문자 ‘I’와 혼동)
  - ‘O‘(‘o’의 대문자로 숫자 ‘0’과 혼동)
  - ‘I’ (‘I’의 대문자, 숫자 ‘1’, 소문자 ‘l’과 혼동)

Topics to be learned

Task	Single Value	Multiple Values	numpy, pandas
Presentation (value, variable)	int, float, string, boolean	list, tuple, dictionary, set	ndarray, Series, DataFrame
Operation (algebra)	expressions	operations, mutable operations	expressions, get, set
Control flow	if for while		
Use and reuse	Functions Standard libraries Modules and Packages		
Input and output	Standard I/O File I/O		Standard I/O File I/O CSV, Excel, JSON, TXT

Categories of Operators

- **Arithmetic Operators**
  - +, -, *, /, …
- **Relational Operators**
  - >, <, ==, …
- **Logical Operators**
  - not, and, or

Python 배우기

- list
- tuple
- set
- dictionary

자료형
(Python built-in data type)

- if (True or False)
- for ~ in
- while

통제
(Python control syntax)

Topics to be learned

Task	Single Value	Multiple Values	numpy, pandas
Presentation (value, variable)	int, float, string, boolean	list, tuple, dictionary, set	ndarray, Series, DataFrame
Operation (algebra)	expressions	operations, mutable operations	expressions, get, set
Control flow	if for while		
Use and reuse	Functions Standard libraries Modules and Packages		
Input and output	Standard I/O File I/O		Standard I/O File I/O CSV, Excel, JSON, TXT

컴퓨터에게 시킬 수 있는 좀더 복잡한 일은?

- 컴퓨터가 좀더 복잡하게 하는 가장 기본적인 기능은?
- 여러 개의 자료(Multiple values)를 표현할 수 있는 방법은 무엇일까?
- 특정한 조건에 맞는 경우(True)에만 명령을 수행하는 방법은?

자료형 – list

```
animals = [‘dog’, ‘cat’, ‘horse’, ‘dog’]

# list를 지원하는 함수(function)
len(animals)
type(animals)
max(animals)

animals[0]
animals[2]
animals[-1]
animals[1:-1]
animals[1:]
animals[1:0]

# list의 메소드(method)
animals.count(‘dog’)
animals.count(‘cat’)
```

```
# list의 메소드(method)
animals.count(‘dog’
animals.count(‘cat’)
animals.append(‘monkey’)
animals

animals.insert(2, ‘bird’)
animals
# [‘dog’, ‘cat’, ‘bird‘, ‘horse’, ‘dog’, ‘monkey’]
```

자료형 – list

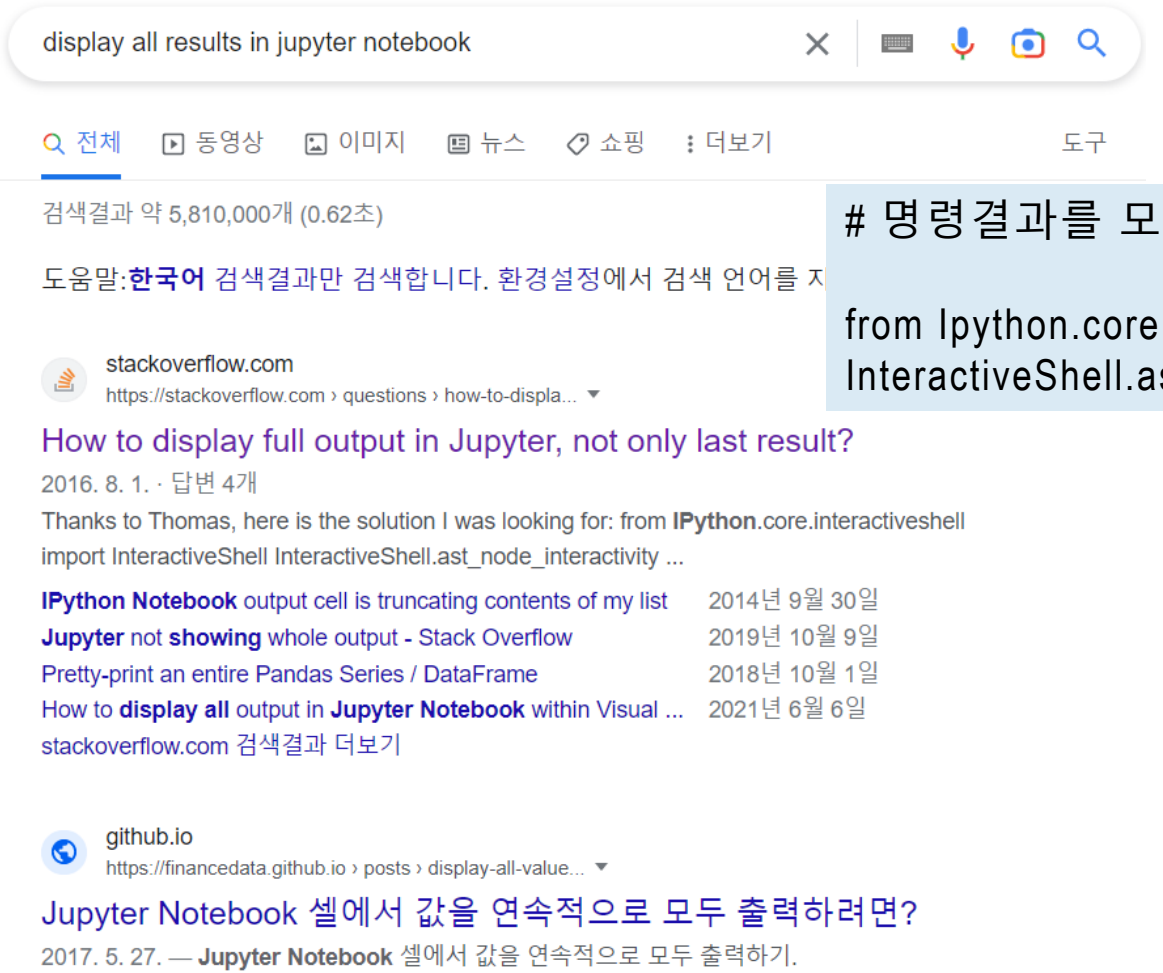
```
# 정렬하려면
sort(animals)
animals                          # 순서 변화 없음
animals.sort()
animals

# 거꾸로 정렬하려면
animals.sort(reverse=True)
animals
animals.pop(len(animals)-1)
animals
```

```
# 중간에서 제거하려면
animals.remove('dog')
animals
del animals[1]
animals
animals.clear()
animals
del animals
animals
```

자료형 – list

```
# 문자열 각각을 list로 변환
letters = list('abcd')
letters

# list 끼리 합치기
animals + letters

# join을 이용해서 문자열로 합치기
''.join(animals)
' ', join(animals)
','. join(animals)
' '.join(animals).split()
```

Tips



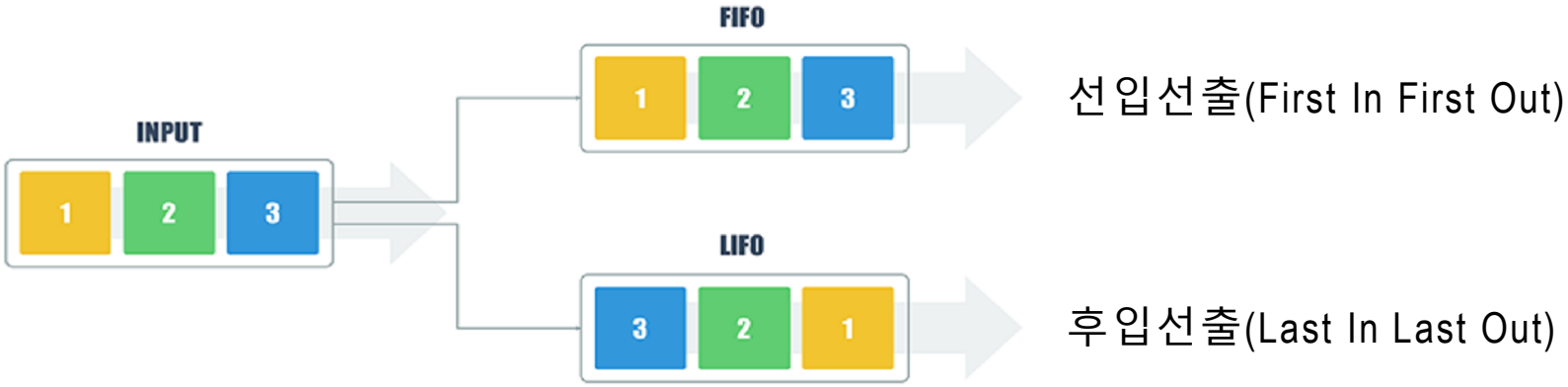
명령결과를 모두 한꺼번에 보고 싶으면 → 구글링

```
from Ipython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity =  " all "
```

stacks & ques (LIFO FIFO)

```
# stack (LIFO)
s = []
s.append(0)
s.append(1)
s.pop()

s.append(2)
s.pop()
```

```
# queue (FIFO)
q = []
q.append(0)
q.append(1)
q.pop(0)

q.append(2)
q.pop(0)
```



자료형 – tuple

```
letters = (‘a’, ‘b’, ‘c’, ‘d’)
# 문자열 각각을 tuple로 변환
letters = tuple(‘abcd’)

# tuple 끼리 합치기
animals = (‘dog’, ‘cat’, ‘horse’, ‘dog’)
animals + letters                          # (‘dog’, ‘cat’, ‘horse’, ‘dog’, ‘a’, ‘b’, ‘c’, ‘d’)

# join을 이용해서 문자열로 합치기
‘’.join(animals)
‘ ‘join(animals)
‘,’.join(animals)

‘ ‘join(animals).split()
```

자료형 – tuple

```
animals = [‘dog’, ‘cat’, ‘horse’, ‘dog’]

# tuple를 지원하는 함수(function)
len(animals)
type(animals)
max(animals)

animals[0]
animals[2]
animals[-1]
animals[1:-1]
animals[1:]
animals[1:0]
```

```
# tuple의 메소드(method)
animals.count(‘dog’)
animals.count(‘cat’)
animals.append(‘monkey’)
animals.insert(2, ‘bird’)
sorted(animals)
```

자료형 – set

```
backpack = {‘book’, ‘paper’, ‘pencil’,
                ‘book’, ‘apple’, ‘tumbler’]

# set는 중복을 허락하지 않음
backpack

# Boolean
‘pencil’ in backpack
‘pen’ in backpack

s1 = set('abracadabra')
s2 = set('aeiou')
s1
s2
```

```
# 합집합
s1 | s2
s1.union(s2)

# 교집합
s1 & s2
s1.intersection(s2)

# 차집합
s1 – s2
s2 – s1

#
s1^s2
s2^s1
set(s1-s2) | set(s2-s1)
```

자료형 – tuple

```
# list로부터 set를 만들때
set_fr_list = set([1, 1, 1, 2, 3, 4, 5, 6])
set_fr_list

# set으로부터 list를 만들때
list_fr_set =  list(set_fr_list)
list_fr_set

s1 = s1 | s2
s1

# 특정항목(원소)를 제거하고 싶을 때
s1 = s1 – {‘e’)
s1
```

자료형 – dictionary

```
# { } 와 : 를 이용하여 상응관계로 짝을 이루어 준다(사전, 단어와 뜻을 짝으로, 전화번호부)
contacts = {‘fire’: 119, ‘police’:112}
contacts

# 다르게 선언하는 방법
[('fire', 119), ('police', 112)]
type([('fire', 119), ('police', 112)])
dict([('fire', 119), ('police', 112)])
dict(fire=119, police=112)

len(contacts)
contacts.keys()
contacts.values()

contacts[‘fire’]
contacts[‘dragon’] = 210
contacts
sorted(contacts)
list(contacts)
```

자료형 – dictionary

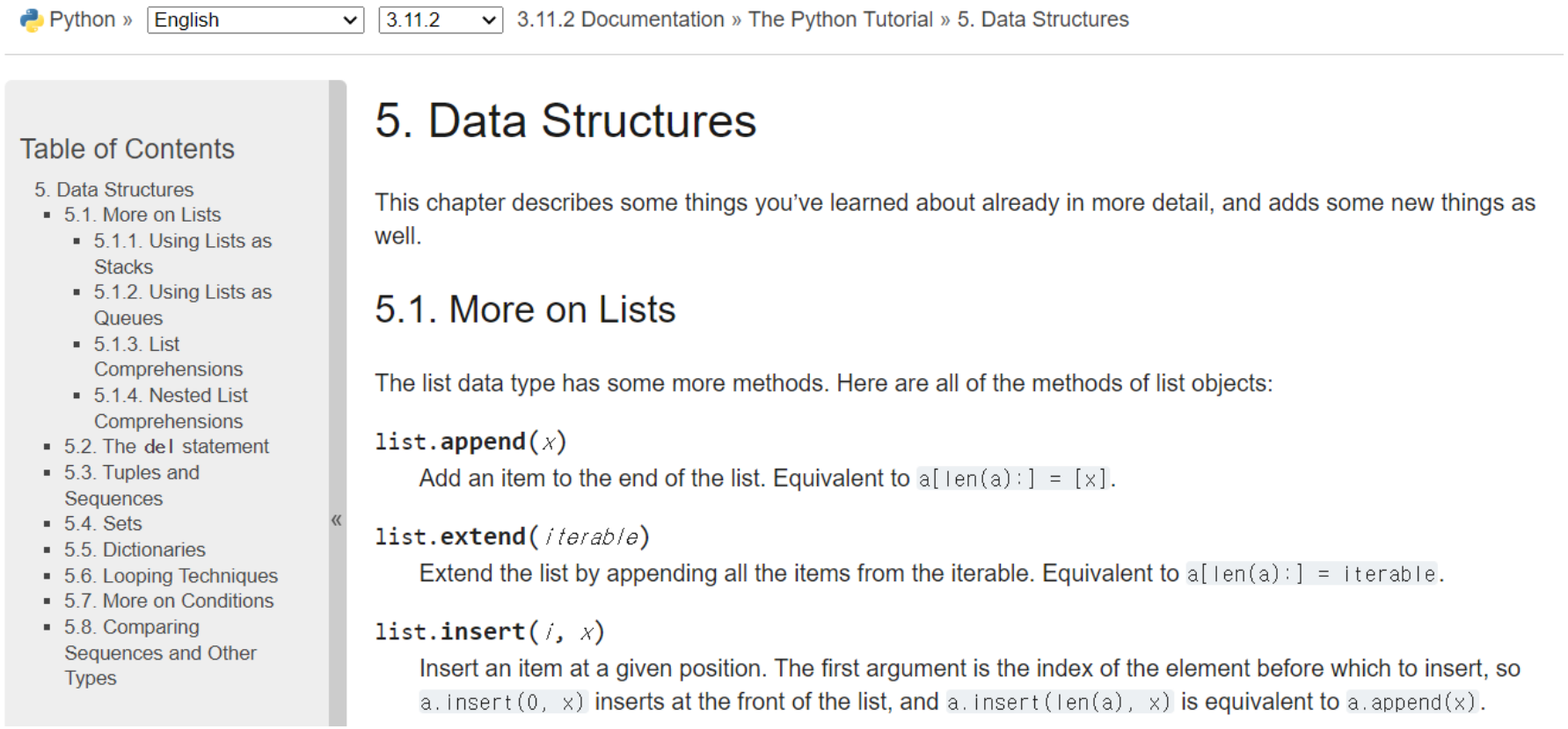
```
‘dragon’ in contacts
del contacts[‘dragon’]
‘dragon’ in contacts
contacts

# 자료형 확인하기 type() 함수로
type([1,2])

type((1,2))

type({1,2})

type({1:2,3:4})
```

자료형 – dictionary

```
# 자료형간 비교하기

'123' < 'ABC' < 'C' < 'abc' < '한글'
'123' < 'DBC' < 'C' < 'abc' < '한글'
'a' < 'A'

(1 ,2, 3)              <              (1, 2, 4)
[1 ,2, 3]              <              [1, 2, 4]
(1 ,2, 3, 4)           <              (1, 2, 4)
(1 ,2)                 <              (1, 2, -1)
(1 ,2, 3)             ==              (1.0, 2.0, 3.0)
1                     ==              1.000
(1, 2, ('aa', 'ab'))   <              (1, 2, ('abc', 'a', 4))
```

자료형 튜토리얼(Tutorial)

https://docs.python.org/3.11/tutorial/datastructures.html



Python » English 3.11.2 3.11.2 Documentation » The Python Tutorial » 5. Data Structures

Table of Contents

5. Data Structures

This chapter describes some things you've learned about already in more detail, and adds some new things as well.

5.1. More on Lists

The list data type has some more methods. Here are all of the methods of list objects:

list.append(*x*)
Add an item to the end of the list. Equivalent to `a[len(a):] = [x]`.

list.extend(*iterable*)
Extend the list by appending all the items from the iterable. Equivalent to `a[len(a):] = iterable`.

list.insert(*i*, *x*)
Insert an item at a given position. The first argument is the index of the element before which to insert, so `a.insert(0, x)` inserts at the front of the list, and `a.insert(len(a), x)` is equivalent to `a.append(x)`.

Control flow – if, 조건문

```
# Boolean(True or False)의 참일 경우만 코드가 실행
a = 1
a == 1                                  # True

if a == 1 :                             # True인 경우(조건)만 다음 문장이 수행
  print('a는 1 이다.')
  print(f'a는 {a} 이다.')
else :
  print('a는 1 이 아니다.')

a = 2                                   # a = -2로 하여 실행해보자
if a == 1:
  print(f'a는 {a}')
elif a > 1:                             # elif(a>1):
  print(f'a는 {a}')
else:
  print(f'a는 {a}')
```

Control flow – for 반복문 활용

```
# 나열형(Iterable type)의 자료를 대상으로 하나씩 호출하면서 명령을 실행
# 리스트 자료의 원소를 하나씩 불러와 프린트하기
a = [1,2,3]
for i in a:
  i
  print('*'*100)
i                                  # for반복문 실행이 끝나면 마지막 원소만 기억함 i=3

for _ in a:
  i
  print('*'*100)

list(range(1,10))
my_list = list(range(1,10))
for i in my_list:
  i
```

Control flow – for 반복문 활용

```
# 빈 리스트(list)와 append 활용 → 가장 많이 사용하게 되는 명령문

# 1 ~ 10까지 리스트에서 짝수와 짝수를 10으로 나눈 실수를 구하기
even_list = []
my_list = []
for i in range(1,11):
  if i % 2 == 0 :
    even_list.append(i)
    my_list.append(i/10)

even_list
my_list

my_list = []
for i in range(1,6):
    my_list.append(i/5)

my_list
```

Control flow – for 반복문 활용

```
# 리스트 축약문(list comprehension)  간결한 문장으로 명령
[i/5 for i in range(1,6)]

[i/10 for i in range(2,11,2) if i%2 ==0]

# 사전 축약문(set comprehension)  간결한 문장으로 명령
[i/5 for i in set(range(1,6))]
{i/10 for i in set(range(2,11,2)) if i%2 ==0}
```

Control flow – for 반복문 활용

```
# 리스트 축약문(list comprehension)  간결한 문장으로 명령
animals = ['dog', 'cat', 'horse', 'dog']
animal_str = 'cumulative_'
for animal in animals :
  animal_str += animal

animal_str

letters = list('abcd')
for i in range(len(letters)):
  animals.append(letters[i])              # animals.append(letters.pop(0))
animals

animals = ('dog', 'cat', 'horse', 'dog')
animal_str = 'cumulative_'
for animal in animals :
  animal_str += animal

animal_str
```

자료형 – dictionary

자료형 – dictionary

자료형 – tuple

```
letters = (‘a’, ‘b’, ‘c’, ‘d’)
# 문자열 각각을 tuple로 변환
letters = tuple(‘abcd’)

# tuple의 각 항목을 하나의 문자열로 합치기
animals = (‘dog’, ‘cat’, ‘horse’, ‘dog’)
animal_str = ‘’
for
```