

Artificial Intelligence

Lecture 2. Python Review I

Spring 2022

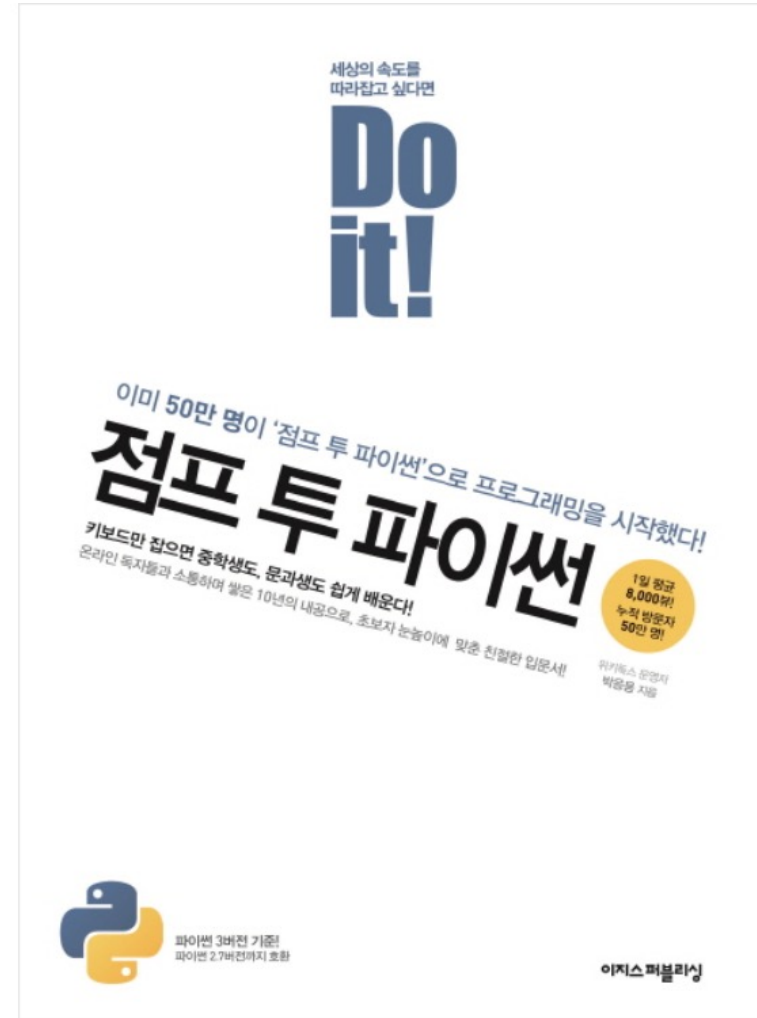
Prof. Jonghoon Chun, Ph.D.

E-mail : jchun@mju.ac.kr

Lecture Note : <http://lms.mju.ac.kr>

점프 투 파이썬

- 위키독스 자료 인용
- <https://wikidocs.net/book/1>



Python 시작하기

- 1990년 암스테르담의 귀도 반 로섬(Guido Van Rossum)이 개발한 인터프리터 언어
- 자신이 좋아하는 코미디 쇼인 "몬티 파이썬의 날아다니는 서커스 (Monty Python's Flying Circus)"에서 따왔음
- 고대 신화에 나오는 파르나소스 산의 동굴에 살던 큰 뱀을 의미



- 구글에서 만들어진 소프트웨어의 많은 비중이 파이썬으로 개발
- 그 외, 인스타그램(Instagram), 드롭박스(dropbox) 등

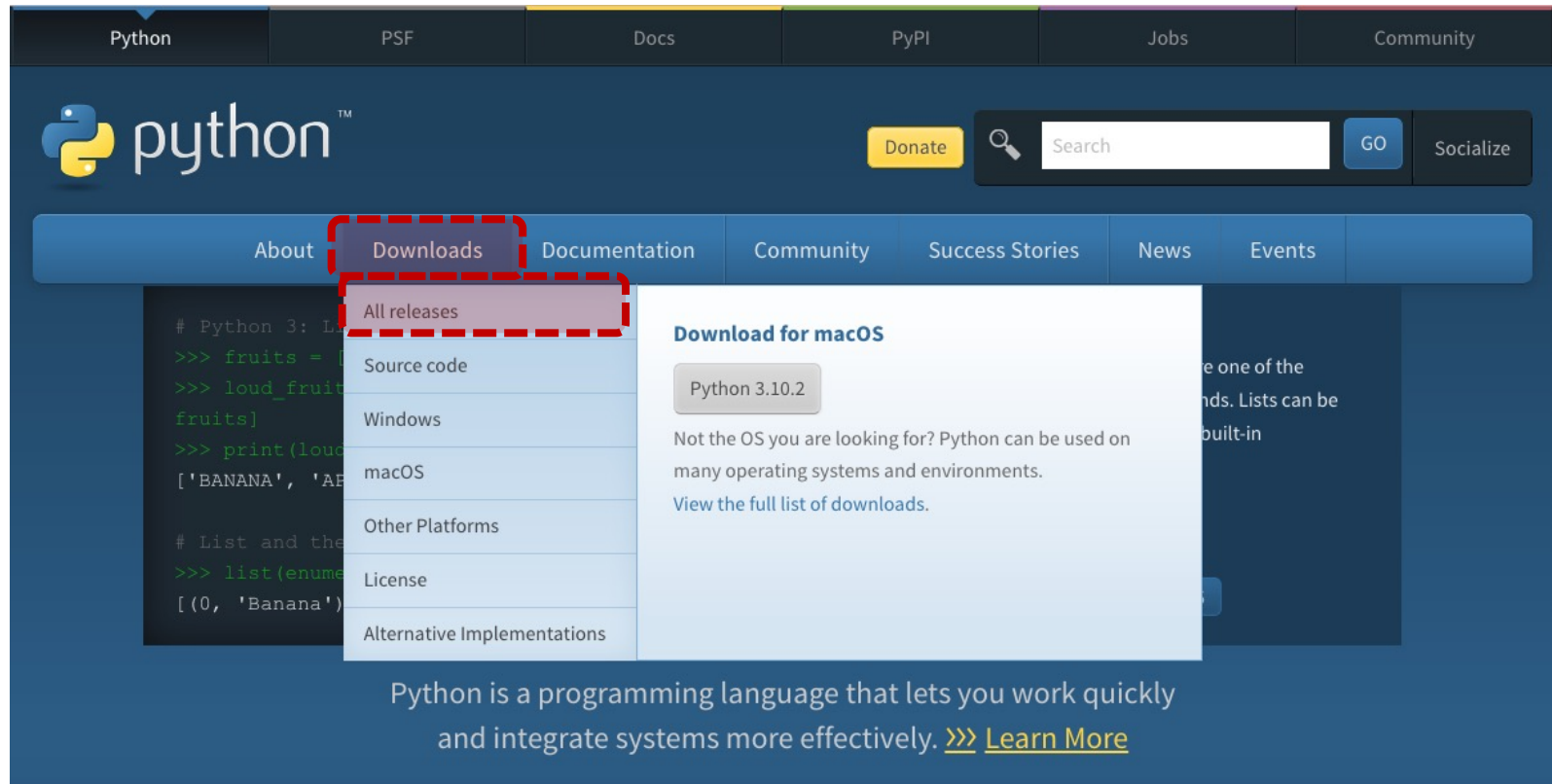
Python 시작하기

- 파이썬으로 할 수 없는 일
 - 시스템 프로그래밍
 - 모바일 프로그래밍
- 파이썬으로 할 수 있는 일
 - 시스템 유틸리티 제작
 - GUI
 - C/C++ 연동
 - 웹프로그래밍
 - 수치연산
 - DB 프로그래밍

Python 시작하기

- Python installation

- 파이썬 홈페이지(<http://www.python.org>)
- Latest version Python 3.10.2 (2022년 2월)
- Tensorflow는 3.4 ~ 3.8에서 동작하므로 주의할 것



Python 시작하기

■ 원하는 version 선택

Looking for a specific release?

Python releases by version number:

Release version	Release date		Click for more
Python 2.7.16	March 4, 2019	Download	Release Notes
Python 3.7.2	Dec. 24, 2018	Download	Release Notes
Python 3.6.8	Dec. 24, 2018	Download	Release Notes
Python 3.7.1	Oct. 20, 2018	Download	Release Notes
Python 3.6.7	Oct. 20, 2018	Download	Release Notes
Python 3.5.6	Aug. 2, 2018	Download	Release Notes
Python 3.4.9	Aug. 2, 2018	Download	Release Notes
Python 3.7.0	June 27, 2018	Download	Release Notes

[View older releases](#)

■ 자신의 환경에 맞는 파일 선택

macOS 64-bit/32-bit installer	Mac OS X	for Mac OS X 10.6 and later	68885dffcd13c5d24699daa0b83315f	28155195	SIG
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later	fee934e3251999a1d353e47ce77be84a	27045163	SIG
Windows help file	Windows		a7caea654e28c8a86ceb017b33b3bf53	8173765	SIG
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64	7617e04b9dafc564f680e37c2f2398b8	7188094	SIG
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64	38cc47776173a45ffec675fc129a46c5	32009096	SIG
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64	6f6b84a5f3c32edd43bffc7c0d65221b	1320008	SIG
Windows x86 embeddable zip file	Windows		a993744c9daa6d159712c8a35374ca9c	6403839	SIG
Windows x86 executable installer	Windows		354023f36de665554bafa21ab10eb27b	30963032	SIG
Windows x86 web-based installer	Windows		da81cf570ee74b59d36f2bb555701cfd	1293456	SIG

■ 기타 02.1_프로그래밍 환경 구성 참고

Python 개발 환경

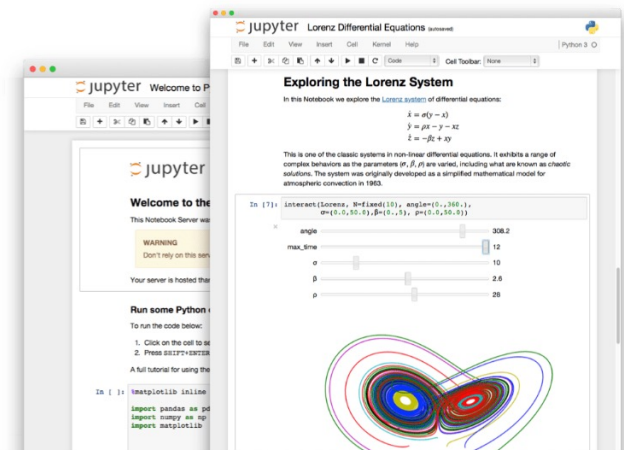
- 파이썬 shell(IDLE)
 - 명령창 또는 편집기 제공
- Eclipse
- PyCharm
- Jupyter Notebook
 - 웹 브라우저에서 파이썬 코드를 작성하고 실행하는 환경
- 일반 source code editor 활용
 - e.g., Sublime text
- Python source code file
 - *.py

Jupyter Notebook jupyter

- <https://jupyter.org/>
- An open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text.
- support for over 40 programming languages, including Python, R, Julia, and Scala.
- Leverage big data tools, such as Apache Spark, from Python, R and Scala.



[Install](#) [About Us](#) [Community](#) [Documentation](#) [NBViewer](#) [JupyterHub](#) [Widgets](#) [Blog](#)



The Jupyter Notebook

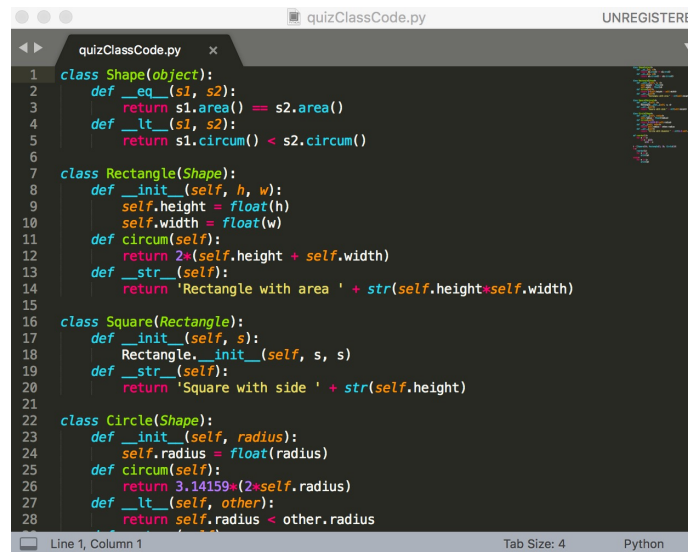
The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

[Try it in your browser](#)

[Install the Notebook](#)

Source Code Editor

- A source code editor is a text editor program
 - designed for editing source code of computer program by programmers



```
quizClassCode.py x UNREGISTERED
1 class Shape(object):
2     def __eq__(s1, s2):
3         return s1.area() == s2.area()
4     def __lt__(s1, s2):
5         return s1.circum() < s2.circum()
6
7 class Rectangle(Shape):
8     def __init__(self, h, w):
9         self.height = float(h)
10        self.width = float(w)
11    def circum(self):
12        return 2*(self.height + self.width)
13    def __str__(self):
14        return 'Rectangle with area ' + str(self.height*self.width)
15
16 class Square(Rectangle):
17     def __init__(self, s):
18         Rectangle.__init__(self, s, s)
19     def __str__(self):
20        return 'Square with side ' + str(self.height)
21
22 class Circle(Shape):
23     def __init__(self, radius):
24         self.radius = float(radius)
25     def circum(self):
26        return 3.14159*(2*self.radius)
27     def __lt__(self, other):
28        return self.radius < other.radius
Line 1, Column 1 Tab Size: 4 Python
```

- Sublime Text (<http://www.sublimetext.com/>)
 - A cross-platform source code editor
 - Natively support many programming languages and markup languages
 - community-built and maintained under free-software licenses



Sublime Text

PYTHON 기초 REVIEW

기초문법

사칙연산

In [12]: `1 + 2`

Out[12]: 3

In [3]: `3 / 2.4`

Out[3]: 1.25

In [4]: `3 * 9`

Out[4]: 27

Ctrl-
Enter:
RUN

Shift-
Enter:
RUN &
Next

변수 및 대입연산

In [5]: `a = 1
b = 2
a + b`

Out[5]: 3

기초문법

출력

```
In [6]: a = "Python"  
print(a)
```

Python

조건문

```
In [7]: a = 3  
if a > 1:  
    print("a is greater than 1")
```

a is greater than 1

기초문법

반복문

```
In [8]: for a in [1, 2, 3]:  
        print(a)
```

1
2
3

```
In [10]: i = 0  
        while i < 3:  
            i=i+1  
            print(i)
```

1
2
3

함수

```
In [11]: def sum(a, b):  
        return a+b  
  
        print(sum(3,4))
```

7

PYTHON DATA TYPES

숫자형

정수형, 실수형 상수 및 연산

```
In [14]: a = 123  
a = -178  
a = 0  
  
a = 1.2  
a = -3.45
```

```
In [15]: a = 3  
b = 4  
a + b
```

Out[15]: 7

```
In [16]: a = 3  
b = 4  
a * b
```

Out[16]: 12

```
In [17]: a = 3  
b = 4  
a / b
```

Out[17]: 0.75

숫자형

In [18]: # x의 y 제곱

```
a = 3  
b = 4  
a ** b
```

Out[18]: 81

In [19]: # 나머지
7 % 3

Out[19]: 1

In [20]: # 몫
7 // 4

Out[20]: 1

문자열

```
In [29]: food = "Python's favorite food is perl"  
print(food)
```

Python's favorite food is perl

```
In [30]: say = "Python is very easy." he says.'  
print(say)
```

"Python is very easy." he says.

```
In [31]: food = 'Python\'s favorite food is perl'  
print(food)  
  
say = "\"Python is very easy.\" he says."  
print(say)
```

Python's favorite food is perl

"Python is very easy." he says.

문자열

```
In [33]: # 여러줄 출력
multiline = "Life is too short\nYou need python"
print(multiline)
```

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

```
In [35]: # 여러줄 출력
multiline1='''
Life is too short
You need python
'''

print(multiline1)

multiline2="""
Life is too short
You need python
"""

print(multiline2)
```

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

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

문자열 연산

```
In [37]: # 문자열 더하기
head = "Python"
tail = " is fun!"
print(head + tail)
```

Python is fun!

```
In [39]: # 문자열 곱하기
a = "python"
print(a * 2)

print("=" * 50)
print("My Program")
print("=" * 50)
```

pythonpython

=====

My Program

=====

Indexing and slicing

인덱싱

```
In [41]: # 인덱스는 0부터 시작  
a = "Life is too short, You need Python"  
print(a[3])
```

e

```
In [44]: # 음수 인덱스는 뒤부터 시작. -1이 가장 마지막 문자를 의미  
a = "Life is too short, You need Python"  
print(a[-1])  
print(a[-2])
```

n

o

Indexing and slicing

슬라이싱

```
In [45]: a = "Life is too short, You need Python"
print(a[0:4]) # 마지막 5번 인덱스는 포함되지 않음
print(a[5:7])
```

```
Life
is
```

```
In [46]: a = "Life is too short, You need Python"

print(a[19:]) # 끝을 지정하지 않으면 마지막까지를 의미함
print(a[:17]) # 시작을 생략하면 처음부터
print(a[19:-7])
```

```
You need Python
Life is too short
You need
```

```
In [47]: a = "20010331Rainy"
date = a[:8]
weather = a[8:]
print(date)
print(weather)
```

```
20010331
Rainy
```



0 <= a < 4

Indexing and slicing

- 다음과 같은 연산은 에러발생
- 문자열 요소는 수정이 불가능함

```
In [48]: a = "Pithon"
a[1] = 'y'
```

```
-----
TypeError                                Traceback (most recent call last)
<ipython-input-48-a877f06250fd> in <module>()
      1 a = "Pithon"
----> 2 a[1] = 'y'

TypeError: 'str' object does not support item assignment
```

문자열 formatting

문자열 포매팅

```
In [50]: print("I eat %d apples." % 3)
          print("I eat %s apples." % "five")

          number = 10
          day = "three"
          print("I ate %d apples. so I was sick for %s days." % (number, day))
```

I eat 3 apples.

I eat five apples.

I ate 10 apples. so I was sick for three days.

문자열 관련 함수들

```
In [51]: # 특정문자 갯수 세기
a = "hobby"
print(a.count('b'))
```

2

```
In [41]: # 특정문자 위치찾기 (find와 index)
a = "Python is best choice"

print(a.find('b')) # 처음 아온 위치 (0부터 인덱싱)
print(a.find('k')) # 존재하지 않으면 -1 리턴

# find와 다른 점은 존재하지 않을 경우 에러 발생
print(a.index('b'))
print(a.index('k')) # 존재하지 않으면 에러 발생
```

10

-1

10

```
-----
ValueError                                Traceback (most recent call last)
<ipython-input-41-b06219fb9213> in <module>
      7 # find와 다른 점은 존재하지 않을 경우 에러 발생
      8 print(a.index('b'))
----> 9 print(a.index('k')) # 존재하지 않으면 에러 발생

ValueError: substring not found
```


문자열 관련 함수들

```
In [57]: # 문자열 삽입(join)
a = ","
b = a.join('abcd')
print(b)

# 대소문자 바꾸기(upper, lower)
a = "hi"
b = a.upper()
print(b)

# 공백지우기(strip, lstrip, rstrip)
a = " hi "
b = a.strip()
print(b)

# 문자열 바꾸기(replace)
a = "Life is too short"
b = a.replace("Life", "Your leg")
print(b)

# 문자열 나누기(split)
a = "Life is too short"
b = a.split() # 기본은 공백(스페이스, 탭, 엔터등)을 기준으로 분리
print(b)

a = "a:b:c:d"
b = a.split(':')
print(b)

a,b,c,d
HI
hi
Your leg is too short
['Life', 'is', 'too', 'short']
['a', 'b', 'c', 'd']
```

기타 문자열 formatting

```
In [59]: print("I eat {0} apples".format(3))
          print("I eat {0} apples".format("five"))

          number = 3
          print("I eat {0} apples".format(number))

          number = 10
          day = "three"
          print("I ate {0} apples. so I was sick for {1} days.".format(number, day))
```

```
I eat 3 apples
I eat five apples
I eat 3 apples
I ate 10 apples. so I was sick for three days.
```

리스트(list)

- 다차원 데이터 표현
- 형식
 - 리스트명 = [요소1, 요소2, 요소3, ...]
- 예제
 - odd = [1, 3, 5, 7, 9]
 - a = []
 - b = [1, 2, 3]
 - c = ['Life', 'is', 'too', 'short']
 - d = [1, 2, 'Life', 'is']
 - 혼용 가능하고, 형식이 통일되지 않고 그대로 저장됨
 - e = [1, 2, [' Life ' , ' is ']]
 - 리스트 내부의 리스트

리스트(list)

```
In [60]: d = [1, 2, 'Life', 'is']  
d
```

```
Out[60]: [1, 2, 'Life', 'is']
```

인덱싱

```
In [61]: # 문자열 인덱싱과 동일  
a = [1, 2, 3]  
print(a)  
print(a[0])  
print(a[0] + a[2])  
  
print(a[-1])
```

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

리스트(list)

```
In [62]: # 이중 리스트
a = [1, 2, 3, ['a', 'b', 'c']]
print(a[3])
print(a[3][0])

#삼중 리스트
a = [1, 2, ['a', 'b', ['Life', 'is']]]
print(a[2][2][0])
```

```
['a', 'b', 'c']
a
Life
```

슬라이싱

```
In [63]: # 문자열 슬라이싱과 동일
a = [1, 2, 3, 4, 5]
print(a[0:2])
print(a[:2])
print(a[2:])
```

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

리스트 연산자

```
In [ ]: a = [1, 2, 3]
b = [4, 5, 6]
print(a + b)    [1, 2, 3, 4, 5, 6]
print(a * 3)    [1, 2, 3, 1, 2, 3, 1, 2, 3]
```

리스트 수정, 변경, 삭제

```
In [70]: a = [1, 2, 3]
a[2] = 4
print(a)

# 1과 2사이의 서브리스트를 새로운 리스트로 대체
a = [1, 2, 3]
a[1:2] = ['a', 'b', 'c']
print(a)

# 인덱스 1번 요소를 대체
a = [1, 2, 3]
a[1] = ['a', 'b', 'c']
print(a)

# 삭제
a = [1, 2, 3]
a[1:3] = [ ]
print(a)

a = [1, 2, 3]
del a[1]
print(a)

[1, 2, 4]
[1, 'a', 'b', 'c', 3]
[1, ['a', 'b', 'c'], 3]
[1]
[1, 3]
```

리스트 관련 함수

```
In [71]: # 요소추가
a = [1, 2, 3]
a.append(4)
print(a)

# 정렬
a = [1, 4, 3, 2]
a.sort()
print(a)

# reverse
a = ['a', 'c', 'b']
a.reverse()
print(a)

# 위치반환 (주어진 요소가 위치한 인덱스를 반환. 없으면 오류발생)
a = [1, 2, 3]
print(a.index(3))
print(a.index(1))

[1, 2, 3, 4]
[1, 2, 3, 4]
['b', 'c', 'a']
2
0
```

리스트 관련 함수

```
In [76]: # 요소 삽입(insert(a, b)는 리스트의 a번째 위치에 b를 삽입)
a = [1, 2, 3]
a.insert(0, 4)
print(a)

# 요소 제거(remove(x)는 리스트에서 첫 번째로 나오는 x를 삭제)
a = [1, 2, 3, 1, 2, 3]
a.remove(3)
print(a)

# 갯수 세기(count(x)는 리스트 내에 x가 몇 개 있는지 조사하여 그 개수를 돌려주는 함수)
a = [1, 2, 3, 1]
print(a.count(1))

# 리스트 확장(extend(x)에서 x에는 리스트만 올 수 있으며 원래의 a 리스트에 x 리스트를 더함)
a = [1, 2, 3]
a.extend([4, 5])
print(a)

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


튜플(tuple)

- 다음의 두 가지 점을 제외하고 리스트와 동일함(일종의 상수 리스트와 같은 개념)
 - 리스트는 [과]으로 둘러싸지만 튜플은 (과)으로 둘러쌘
 - 리스트는 그 값의 생성, 삭제, 수정이 가능하지만 튜플은 그 값을 바꿀 수 없음
- 예
 - `t1 = ()`
 - `t2 = (1,)` # 요소가 하나라도 ','를 반드시 삽입
 - `T3 = (1, 2, 3)`
 - `T4 = 1, 2, 3` # 괄호 생략 가능
 - `t5 = ('a', 'b', ('ab', 'cd'))`
- 내용을 바꾸는 연산은 불가능하지만 인덱싱, 슬라이싱, `+`, `*` 연산은 리스트와 동일함

딕셔너리(dictionary)

- Key와 Value를 한 쌍으로 갖는 자료형
- 순차적으로(sequential) 해당 요소 값을 구하지 않고 Key를 통해 Value를 얻음
- 구성
 - {Key1:Value1, Key2:Value2, Key3:Value3 ...}
 - dic = {'name':'pey', 'phone':'0119993323', 'birth': '1118'}
 - a = {1: 'hi'}
 - a = { 'a': [1,2,3]} # value에 리스트도 가능

딕셔너리 요소 삽입과 삭제

요소 구성 및 추가

```
In [58]: a = {1:'a'}  
a[2] = 'b'  
print(a)  
  
a['name'] = 'pey'  
print(a)  
  
a[3] = [1,2,3]  
print(a)  
  
{1: 'a', 2: 'b'}  
{1: 'a', 2: 'b', 'name': 'pey'}  
{1: 'a', 2: 'b', 'name': 'pey', 3: [1, 2, 3]}
```

요소 삭제

```
In [78]: del a[1]  
print(a)  
  
{2: 'b', 'name': 'pey', 3: [1, 2, 3]}
```

Key로 value 가져오기

key로 value 얻기

```
In [79]: grade = {'pey': 10, 'julliet': 99}
          print(grade['pey'])
          print(grade['julliet'])
```

```
10
99
```

```
In [80]: dic = {'name': 'pey', 'phone': '0119993323', 'birth': '1118'}
          print(dic['phone'])
```

```
0119993323
```

단, key는 리스트로 정의할 수 없다! (튜플은 가능)

딕셔너리 함수들

```
In [85]: # Key 리스트 만들기(dict_keys라는 객체 생성)
a = {'name': 'pey', 'phone': '0119993323', 'birth': '1118'}
print(a.keys())

# Value 리스트 만들기(dict_values라는 객체 생성)
print(a.values())

# Key, Value 쌍 얻기(items)
# key와 value의 쌍을 튜플로 묶은 값을 dict_items 객체로 돌려준다.
print(a.items())

# Key로 Value얻기(get)
# a['name']과 결과가 동일하나 존재하지 않을 경우 get은 None을 반환하고, a['name']은 오류 발생
a = {'name': 'pey', 'phone': '0119993323', 'birth': '1118'}
print(a.get('name'))

# 해당 Key가 딕셔너리 안에 있는지 조사하기(in)
print('name' in a)
print('email' in a)

dict_keys(['birth', 'phone', 'name'])
dict_values(['1118', '0119993323', 'pey'])
dict_items([('birth', '1118'), ('phone', '0119993323'), ('name', 'pey')])
pey
True
False
```

집합(set)

- 순서와 중복이 없음

```
In [86]: # 집합 생성
s1 = set([1,2,3])
print(s1)

s2 = set("Hello")
print(s2)
```

```
{1, 2, 3}
{'e', 'l', 'H', 'o'}
```

```
In [87]: # set을 리스트나 튜플로 변환 가능
s1 = set([1,2,3])

l1 = list(s1)
print(l1)

t1 = tuple(s1)
print(t1)
```

```
[1, 2, 3]
(1, 2, 3)
```

집합(set)

```
In [88]: # 집합연산
s1 = set([1, 2, 3, 4, 5, 6])
s2 = set([4, 5, 6, 7, 8, 9])

print(s1 & s2) # 교집합
print(s1 | s2) # 합집합
print(s1 - s2) # 차집합

{4, 5, 6}
{1, 2, 3, 4, 5, 6, 7, 8, 9}
{1, 2, 3}
```

```
In [89]: # 관련 함수
# 원소 1개 추가
s1 = set([1, 2, 3])
s1.add(4)
print(s1)

# 여러 원소 추가
s1 = set([1, 2, 3])
s1.update([4, 5, 6])
print(s1)

# 원소제거
s1 = set([1, 2, 3])
s1.remove(2)
print(s1)

{1, 2, 3, 4}
{1, 2, 3, 4, 5, 6}
{1, 3}
```

참고(변수와 객체)

- 파이썬에서 사용하는 변수는 객체임
 - 예) $a = 3$
 - 3이라는 값을 가지는 정수 자료형 "객체"가 자동으로 메모리에 생성
 - 변수 a 는 객체가 저장된 메모리의 위치를 가리키는 레퍼런스(Reference)
- 파이썬의 모든 자료형은 객체로 생성

- 예)

```
In [77]: a = 3
         id(a)

Out[77]: 4333774000
```

```
In [78]: b = a
         id(b)

Out[78]: 4333774000
```

```
>>> a = 3
>>> b = 3
>>> a is b
True
```

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

- 동일한 내용의 리스트를 다른 객체로 복사하려면 `copy` 모듈을 이용해야 함

```
>>> from copy import copy
>>> b = copy(a)
```


END