

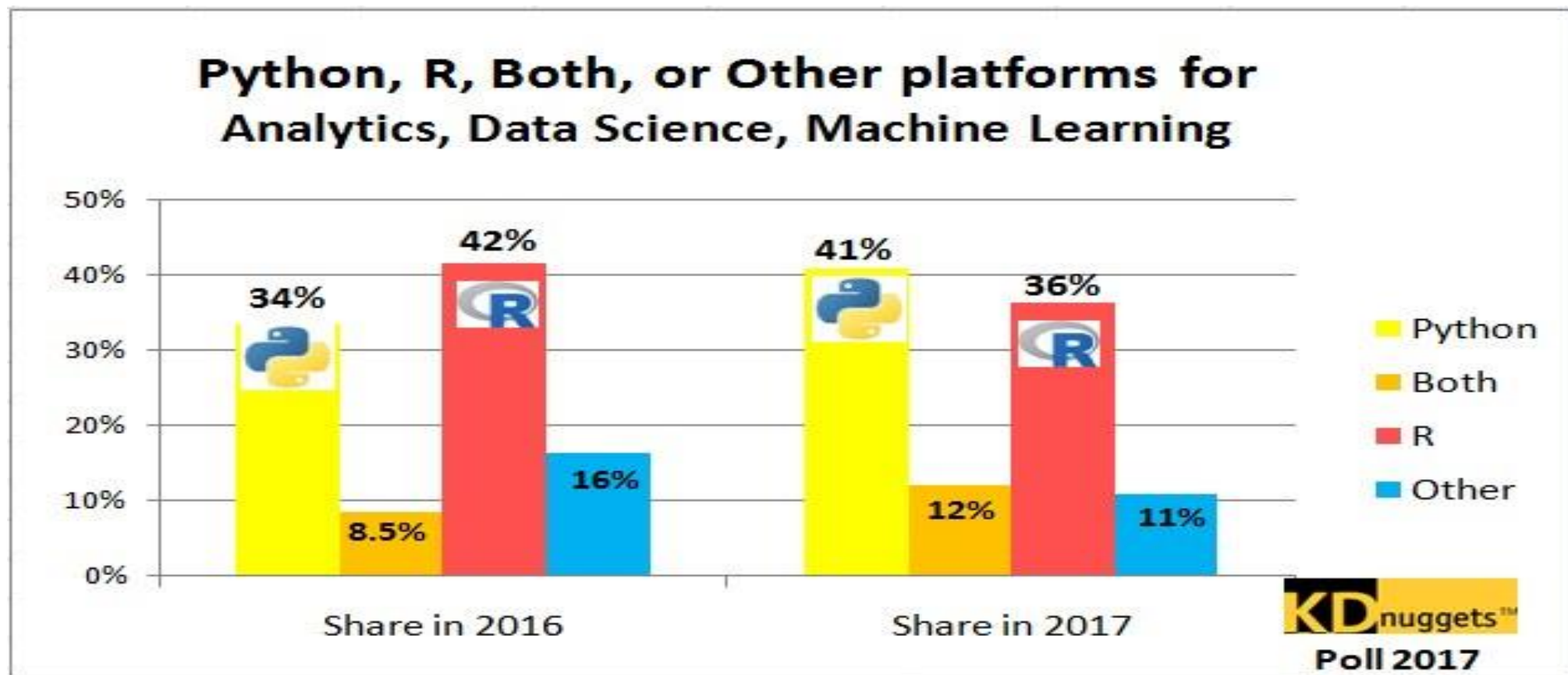
파이썬 기초

youngpyoryu@dongguk.edu

Job Trends



선호 프로그래밍 언어.



파이썬이란?

- 1991년 네덜란드의 귀도 반 로섬(Guido van Rossum)이 개발
- C, C++, 자바 등 어떤 컴퓨터 프로그래밍 언어보다 배우기 쉬움
- 직관적이고 이해하기 쉬운 문법
- 객체 지향의 고수준 언어
- 앱(App)과 웹(WEB) 프로그램 개발 목적
- 웹 서버, 과학 연산, 사물 인터넷(IOT), 인공지능, 게임 등의 프로그램 개발하는 강력한 도구



Python의 특징

"Life is too short, You need python." (인생은 너무 짧으니 파이썬이 필요해.)

- **1. 직관적이고 쉽다.**

- 아주 간단한 영어 문장을 읽듯이 보고 쉽게 이해할 수 있도록 구성

- **2. 널리쓰인다.**

- 구글, 아마존, 핀터레스트, 인스타그램, IBM, 디즈니, 야후, 유튜브, 노키아, NASA 등과 네이버, 카카오톡의 주력 언어 중 하나

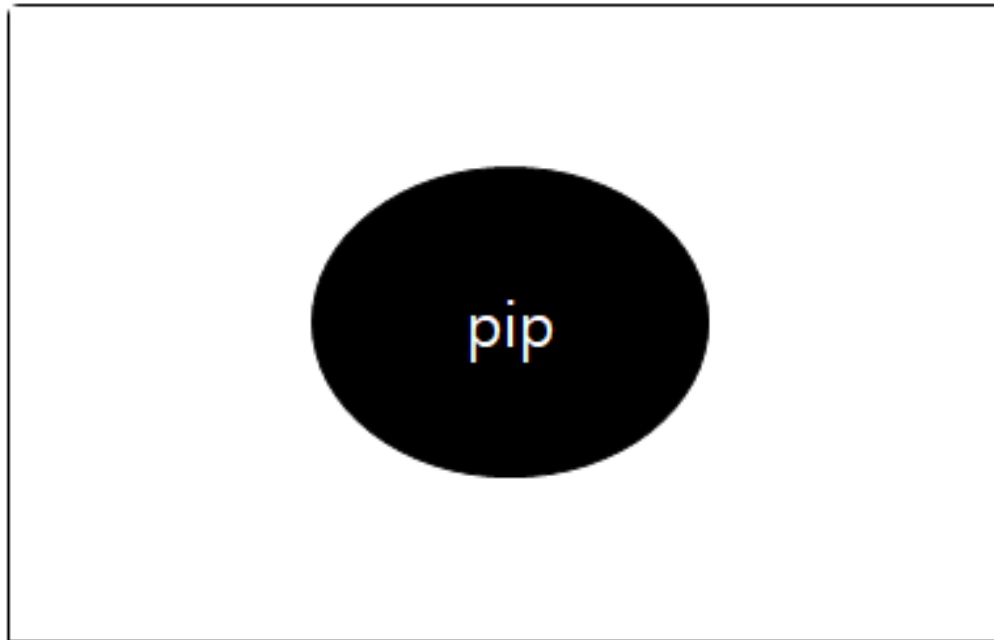
- **3. 개발 환경이 좋다.**

- 게임, 인공지능, 수치해석 등 다양한 라이브러리와 커뮤니티 활성화

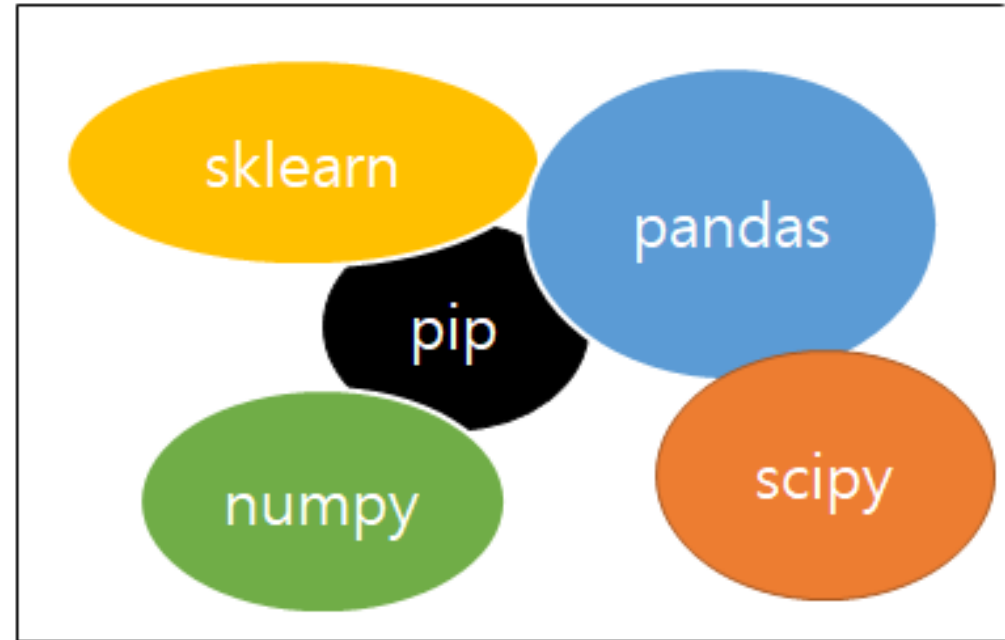
컴파일러 vs 인터프리터

특징 \ 방식	컴파일러	인터프리터
번역 방법	프로그램 전체 번역	실행되는 줄(라인) 단위 번역
장점	한 번 컴파일한 후에는 매번 빠른 시간 내에 전체 실행 가능	번역 과정이 비교적 간단하고 대화형 언어에 편리함
단점	프로그램의 일부를 수정하는 경우에도 전체 프로그램을 다시 컴파일해야 함	실행할 때마다 매번 기계어로 바꾸는 과정을 다시 수행해야 하므로 항상 인터프리터가 필요함
출력물	목적 코드	즉시 실행
언어 종류	FORTTRAN, COBOL, C 등	BASIC 등

Python VS Anaconda



Python



Anaconda

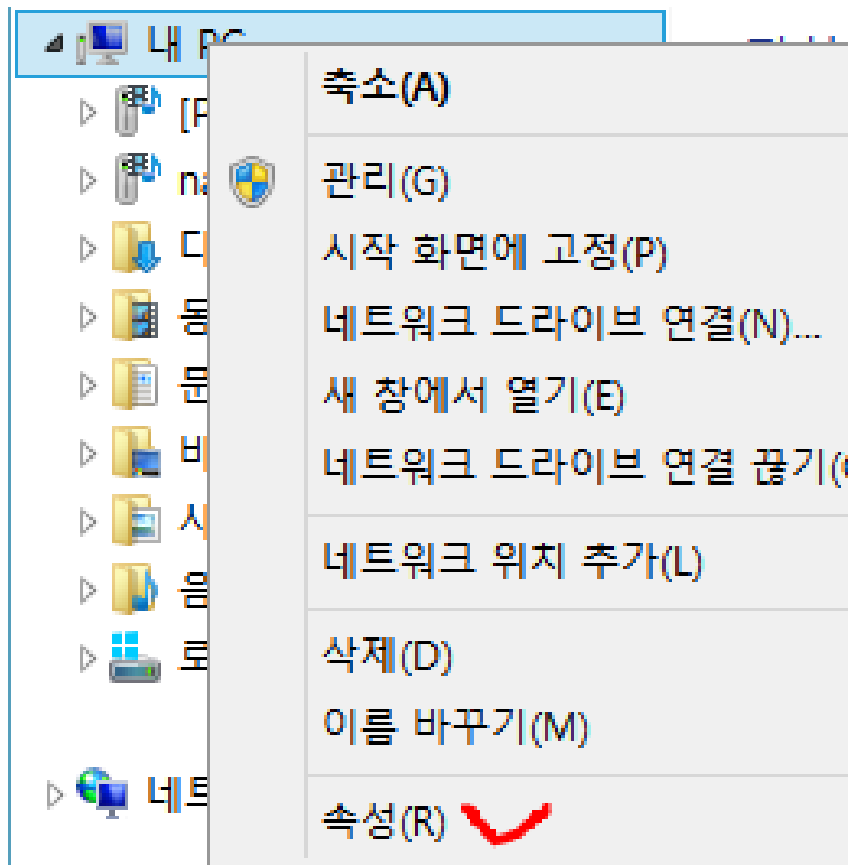
아나콘다 설치 전 확인

- 1. OS

- 내 컴퓨터가 Mac, Windows Check.

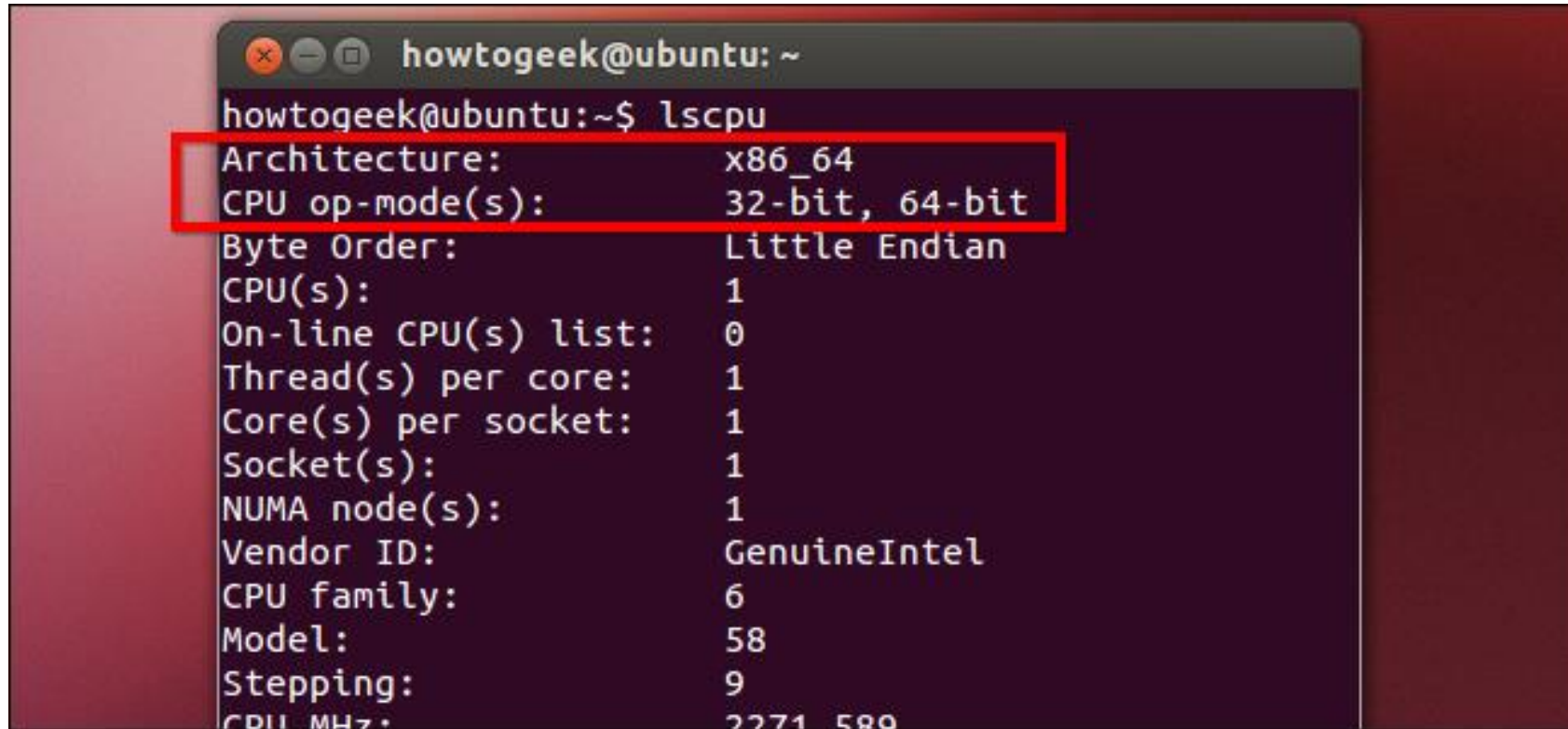
- 2. 운영체계가 32bit , 64 bit check

아나콘다 설치 전 확인(windows)



프로세서:	Intel(R) Core(TM) i7-7700K CPU @ 4.20GHz 4.20 GHz
설치된 메모리(RAM):	44.0GB(43.9GB 사용 가능)
시스템 종류:	64비트 운영 체제, x64 기반 프로세서
펜 및 터치:	이 디스플레이에 사용할 수 있는 펜 또는 터치식 입력이 없습니다.

아나콘다 설치 전 확인(MAC)



```
howtogeek@ubuntu: ~  
howtogeek@ubuntu:~$ lscpu  
Architecture:          x86_64  
CPU op-mode(s):        32-bit, 64-bit  
Byte Order:            Little Endian  
CPU(s):                 1  
On-line CPU(s) list:    0  
Thread(s) per core:     1  
Core(s) per socket:     1  
Socket(s):              1  
NUMA node(s):           1  
Vendor ID:              GenuineIntel  
CPU family:             6  
Model:                  58  
Stepping:               9  
CPU MHz:                2271.580
```

아나콘다 설치

(<https://www.anaconda.com/distribution/>)



Windows



macOS



Linux

Anaconda 2019.03 for macOS Installer

Python 3.7 version

Download

64-Bit Graphical Installer (637 MB)

64-Bit Command Line Installer (542 MB)

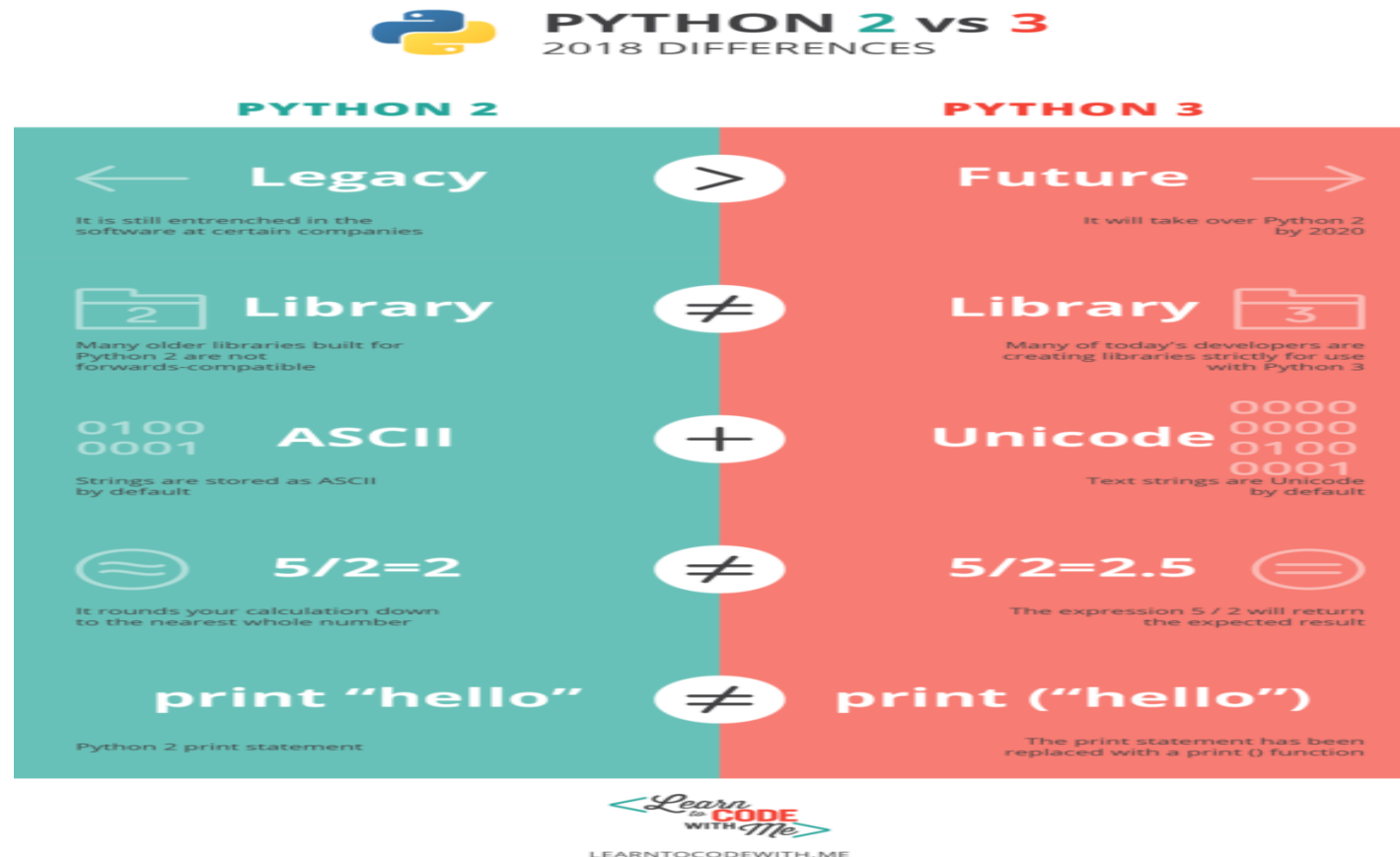
Python 2.7 version

Download

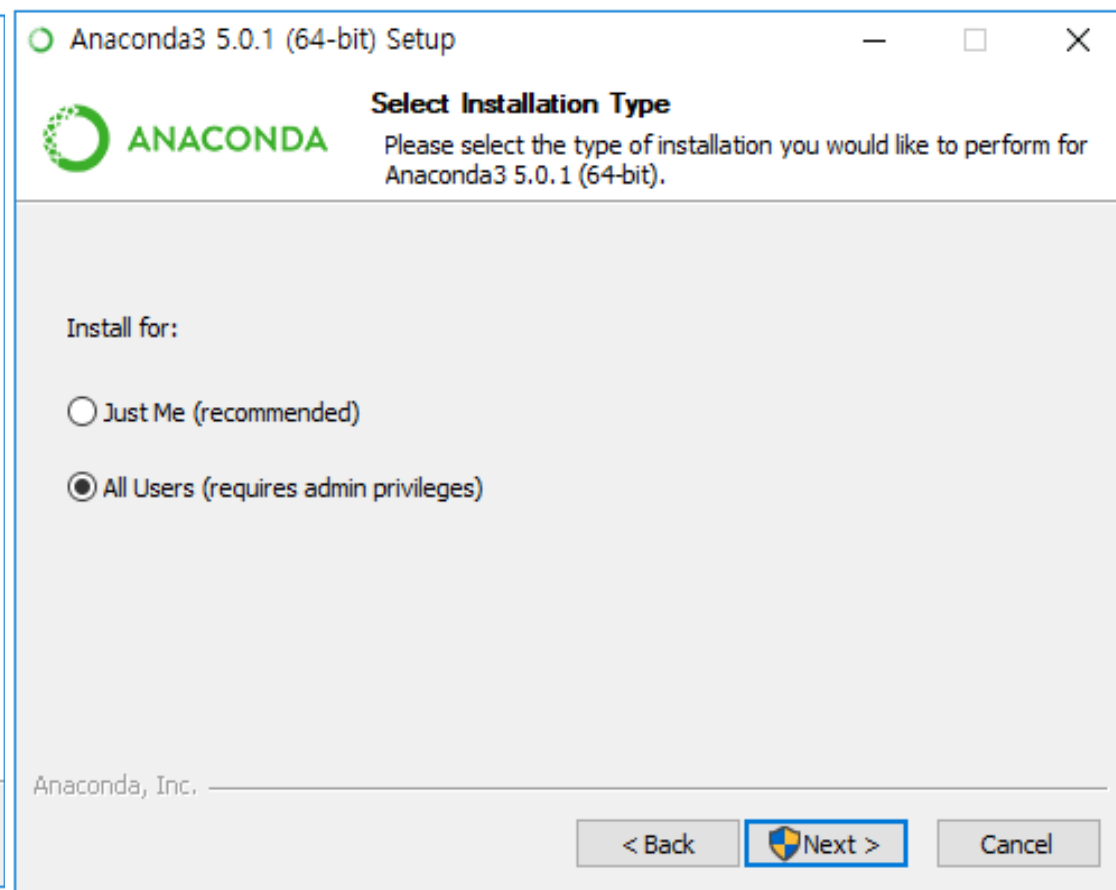
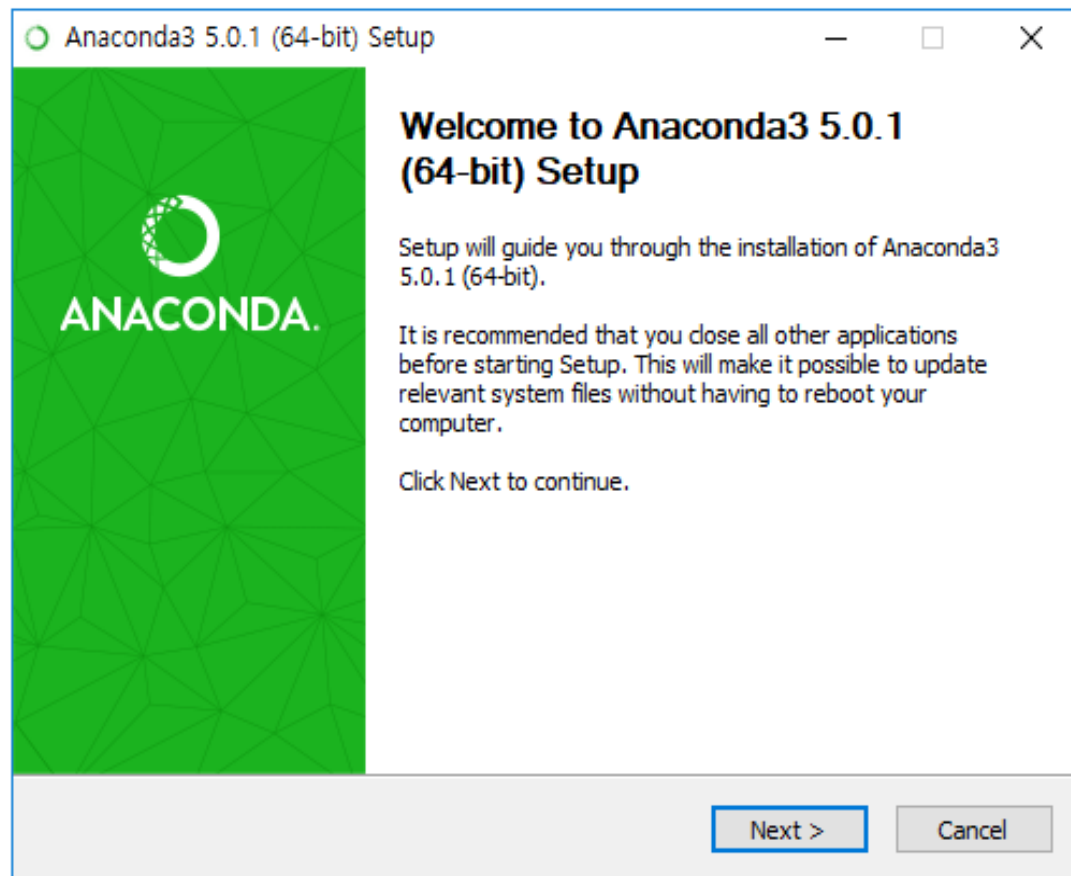
64-Bit Graphical Installer (624 MB)

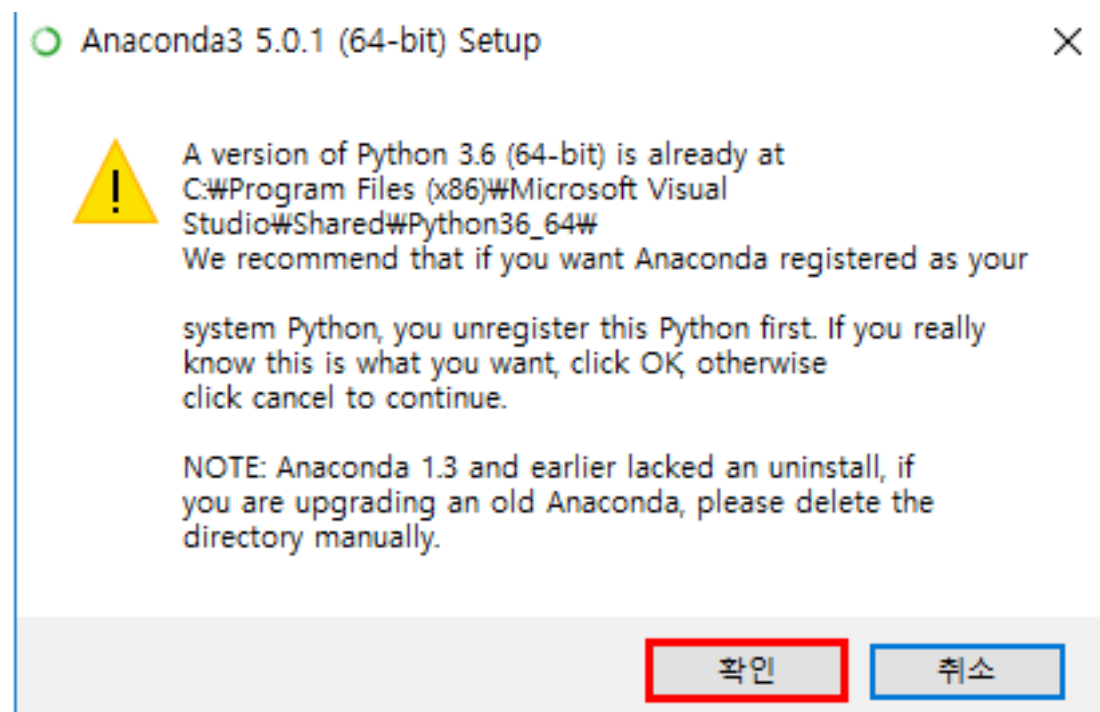
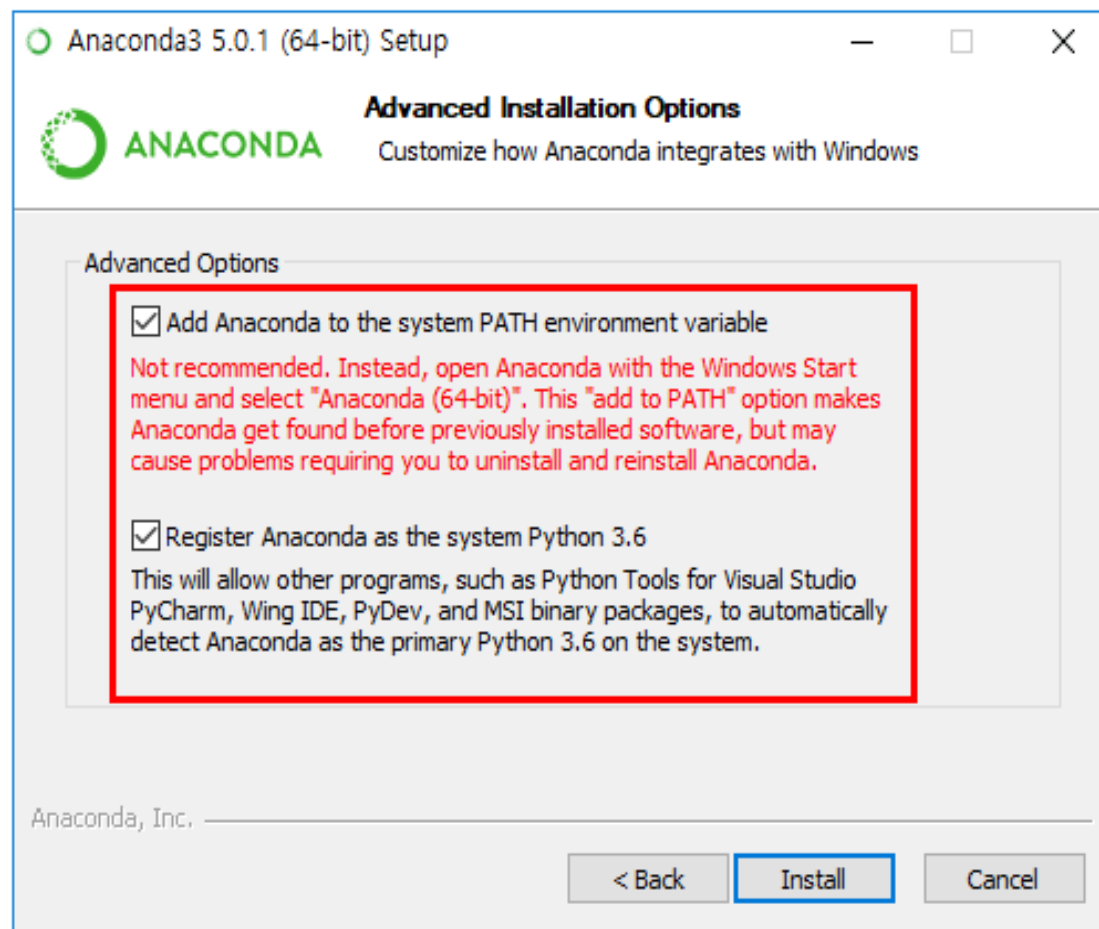
64-Bit Command Line Installer (530 MB)

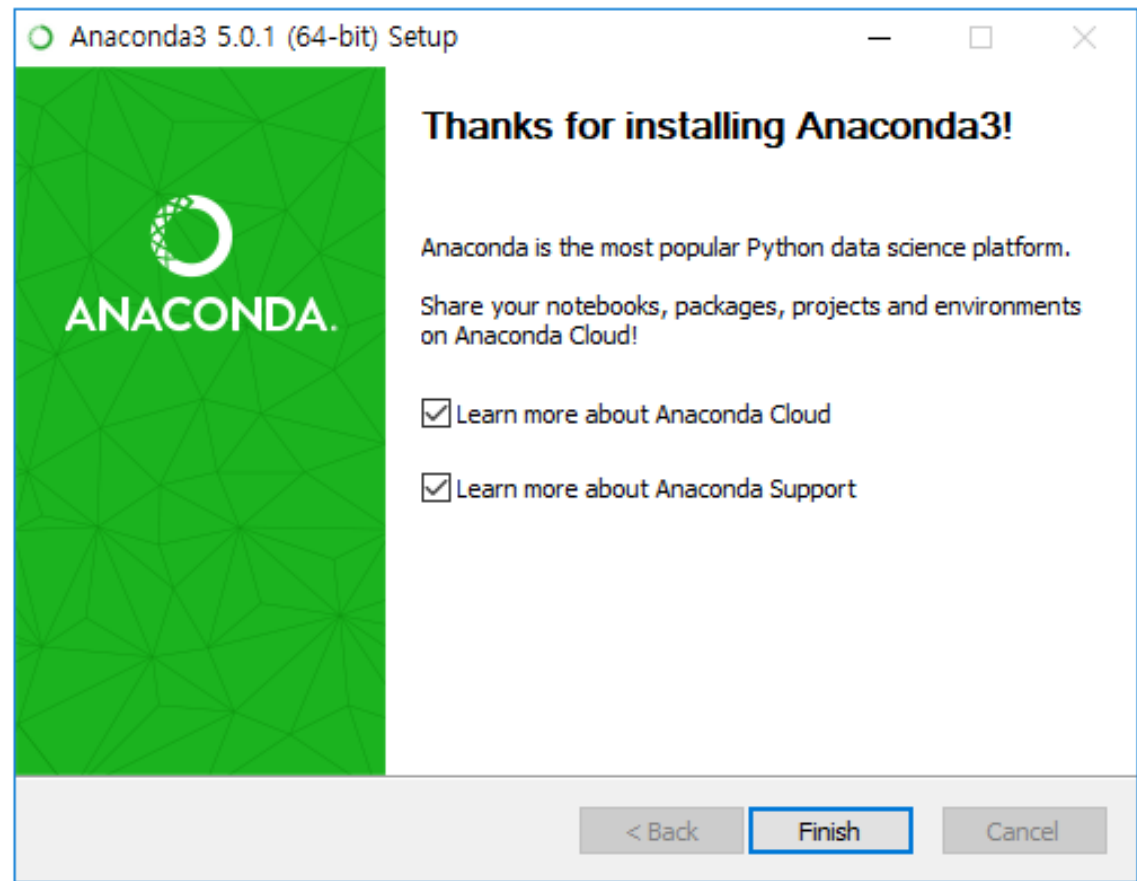
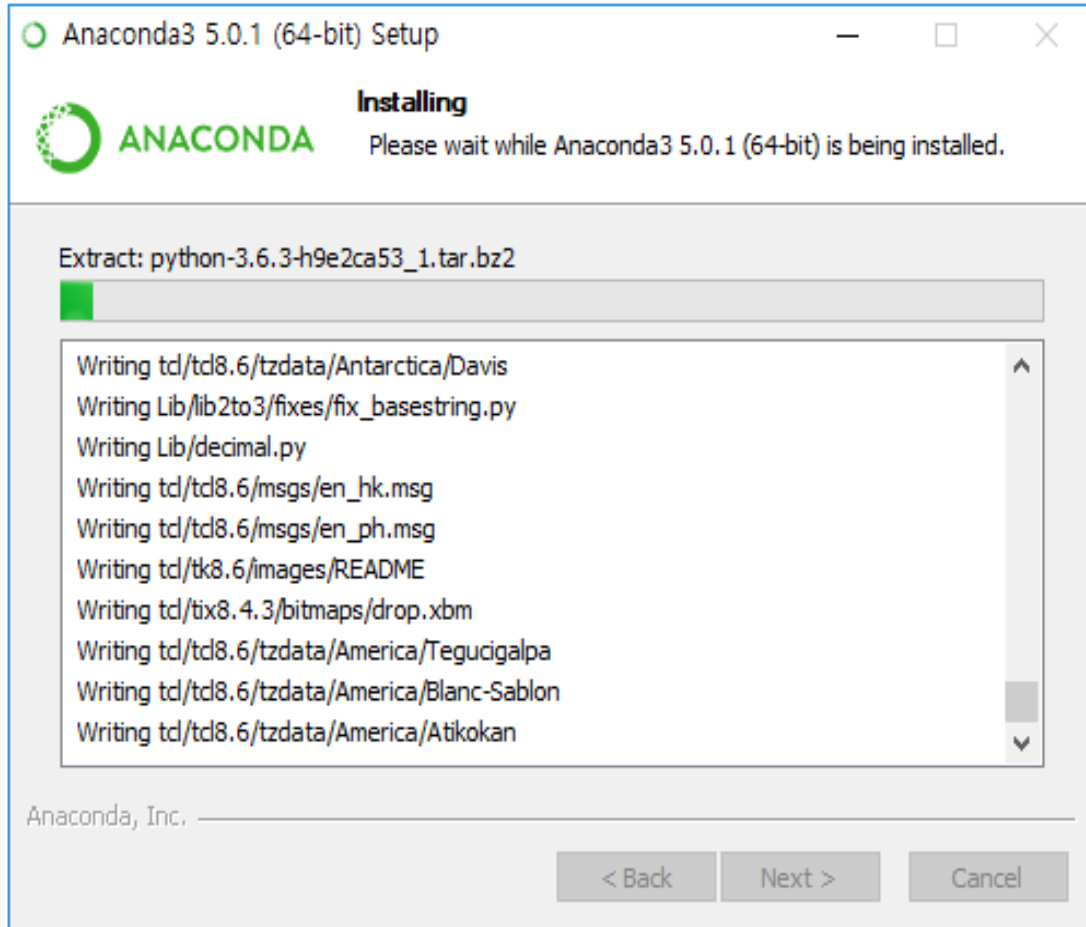
Python 2 VS python 3



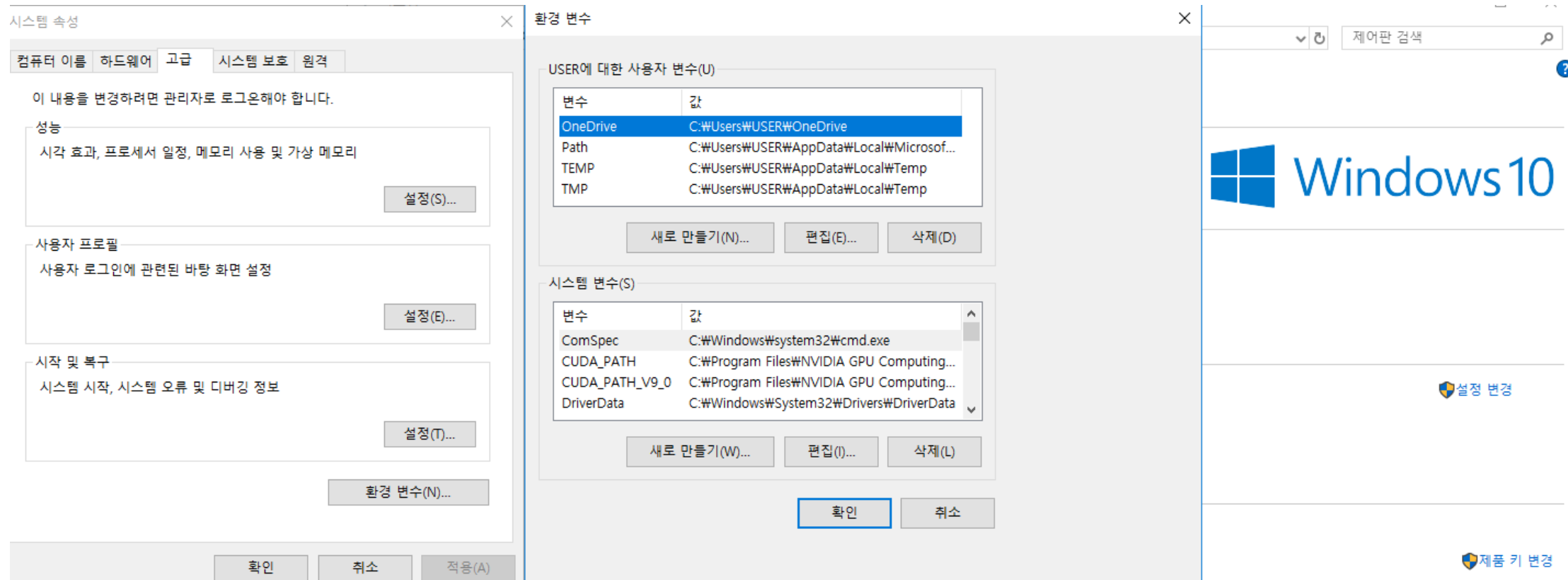
아나콘다 설치



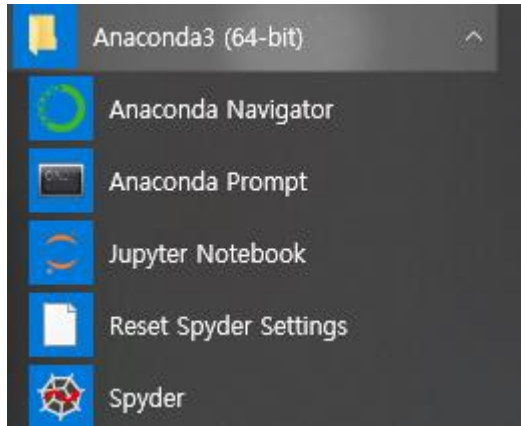




환경 변수 Check



Anaconda3 check (window key)



```
선택 C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.16299.192]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users>python
Python 3.6.3 |Anaconda, Inc.| (default, Oct 15 2017, 03:00:00)
Type "help", "copyright", "credits" or "license()" for more
>>> exit()

C:\Users>python --version
Python 3.6.3 :: Anaconda, Inc.
```

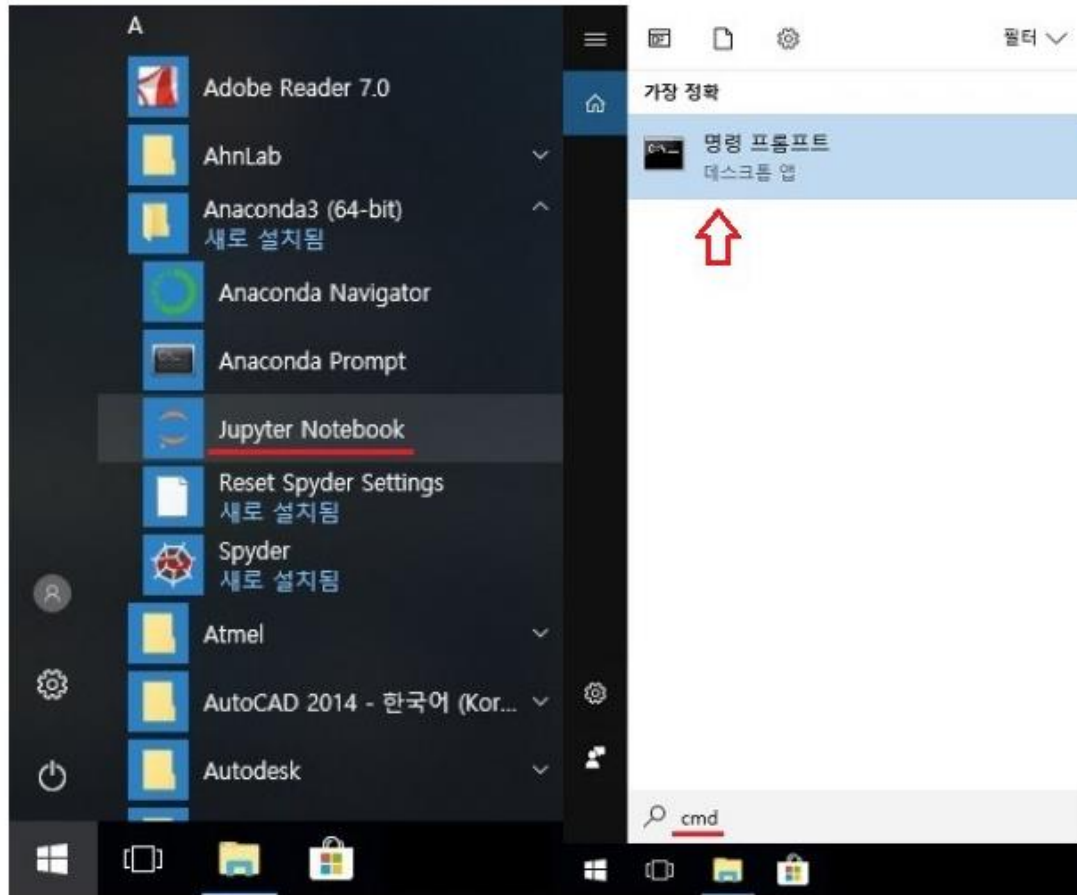
버전 Check

Windows key -> CMD -> python --version

Jupyter notebook

- - Matplotlib, Seaborn 등 다양한 시각화 라이브러리 사용 가능
 - - Markdown 등을 통한 보고서 작성
 - - 사용법을 찾기 쉽고 커뮤니티 활성화가 많이 됨
 - - 인터페이스가 친숙함.
-

Jupyter notebook 들어가는 법.



Jupyter notebook 로고를 클릭하여 들어가면.

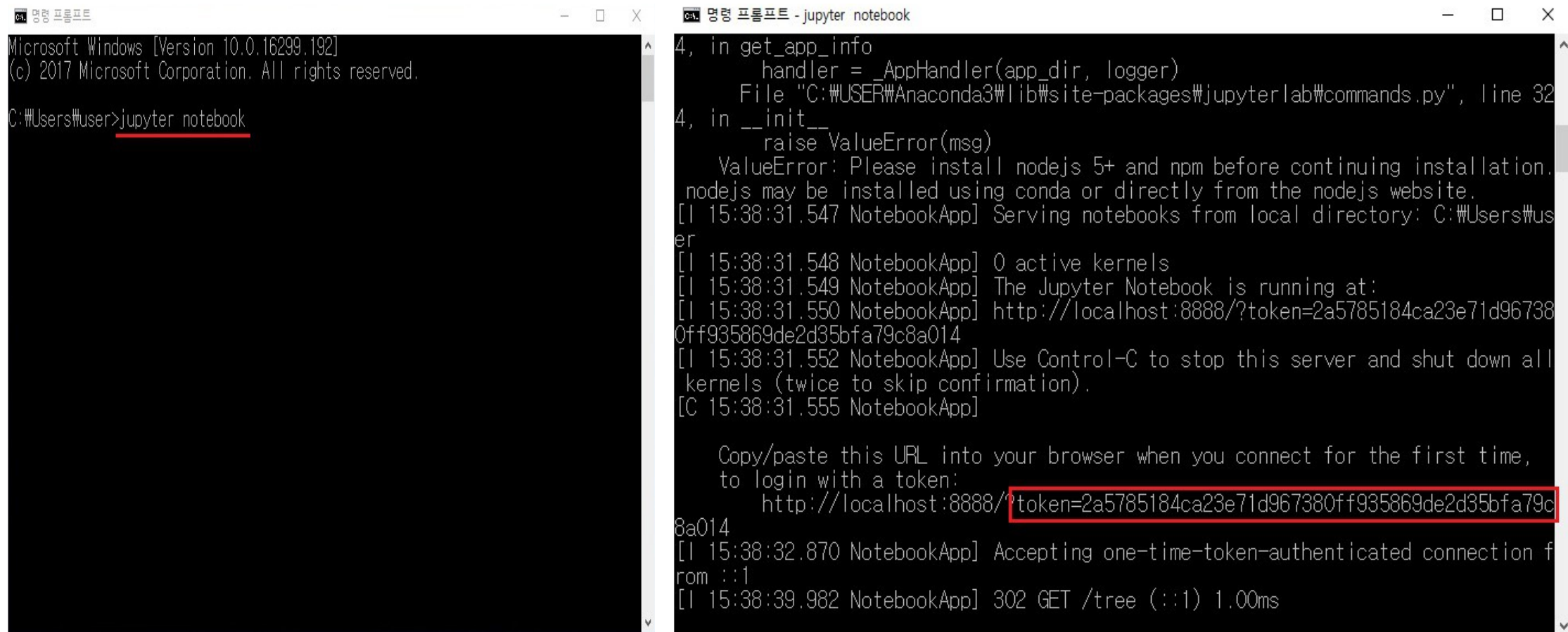
```
Jupyter Notebook
File "C:\USER#Anaconda3\lib\site-packages\jupyterlab\commands.py", line 24
4, in get_app_info
    handler = _AppHandler(app_dir, logger)
File "C:\USER#Anaconda3\lib\site-packages\jupyterlab\commands.py", line 32
4, in __init__
    raise ValueError(msg)
ValueError: Please install nodejs 5+ and npm before continuing installation.
nodejs may be installed using conda or directly from the nodejs website.
[I 14:33:32.613 NotebookApp] Serving notebooks from local directory: C:\Users#us
er
[I 14:33:32.613 NotebookApp] 0 active kernels
[I 14:33:32.615 NotebookApp] The Jupyter Notebook is running at:
[I 14:33:32.616 NotebookApp] http://localhost:8888/?token=8cfc0a0ec45a61ac3fcec2
5ceade213ac2925e225d3209ab
[I 14:33:32.618 NotebookApp] Use Control-C to stop this server and shut down all
kernels (twice to skip confirmation).
[C 14:33:32.621 NotebookApp]

Copy/paste this URL into your browser when you connect for the first time,
to login with a token:
http://localhost:8888/?token=8cfc0a0ec45a61ac3fcec25ceade213ac2925e225d3
209ab
[I 14:33:44.352 NotebookApp] Accepting one-time-token-authenticated connection f
rom ::1
```

Jupyter notebook 활성화



2. 명령 프롬프트의 경우(Jupyter notebook실행)




```
Microsoft Windows [Version 10.0.16299.192]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\User>jupyter notebook

4, in get_app_info
    handler = _AppHandler(app_dir, logger)
    File "C:\Users\User\Anaconda3\lib\site-packages\jupyterlab\commands.py", line 32
4, in __init__
    raise ValueError(msg)
ValueError: Please install nodejs 5+ and npm before continuing installation.
nodejs may be installed using conda or directly from the nodejs website.
[I 15:38:31.547 NotebookApp] Serving notebooks from local directory: C:\Users\User
[I 15:38:31.548 NotebookApp] 0 active kernels
[I 15:38:31.549 NotebookApp] The Jupyter Notebook is running at:
[I 15:38:31.550 NotebookApp] http://localhost:8888/?token=2a5785184ca23e71d967380ff935869de2d35bfa79c8a014
[I 15:38:31.552 NotebookApp] Use Control-C to stop this server and shut down all
kernels (twice to skip confirmation).
[C 15:38:31.555 NotebookApp]

Copy/paste this URL into your browser when you connect for the first time,
to login with a token:
http://localhost:8888/?token=2a5785184ca23e71d967380ff935869de2d35bfa79c8a014
[I 15:38:32.870 NotebookApp] Accepting one-time-token-authenticated connection f
rom ::1
[I 15:38:39.982 NotebookApp] 302 GET /tree (::1) 1.00ms
```

Jupyter notebook 활성화

 jupyter

Password or token:

Token authentication is enabled

If no password has been configured, you need to open the notebook server with its login token in the URL, or paste it above. This requirement will be lifted if you [enable a password](#).

The command:

```
jupyter notebook list
```

will show you the URLs of running servers with their tokens, which you can copy and paste into your browser. For example:

```
Currently running servers:  
http://localhost:8888/?token=c8de56fa... :: /Users/you/notebooks
```

or you can paste just the token value into the password field on this page.

See [the documentation on how to enable a password](#) in place of token authentication, if you would like to avoid dealing with random tokens.

Cookies are required for authenticated access to notebooks.

Jupyter notebook 접속



The screenshot displays the JupyterLab web interface. At the top left is the Jupyter logo, and at the top right is a 'Logout' button. Below the logo are three tabs: 'Files', 'Running', and 'Clusters'. The 'Files' tab is active, showing a message 'Select items to perform actions on them.' and a toolbar with 'Upload', 'New', and a refresh icon. The file browser shows a list of folders: '3D Objects', 'ansel', 'Contacts', 'Desktop', 'Documents', 'Downloads', and 'e-Postbank'. The 'Documents' folder is selected, and a dropdown menu is open, showing options: 'Notebook: Python 3' (highlighted with a red underline), 'Other: Text File', 'Folder', and 'Terminal'. The right side of the interface shows a partial view of the file list with dates: '11월 전', '하루 전', and '3달 전'.

jupyter

Logout

Files Running Clusters

Select items to perform actions on them.

Upload New ↕ ↻

Notebook:
Python 3

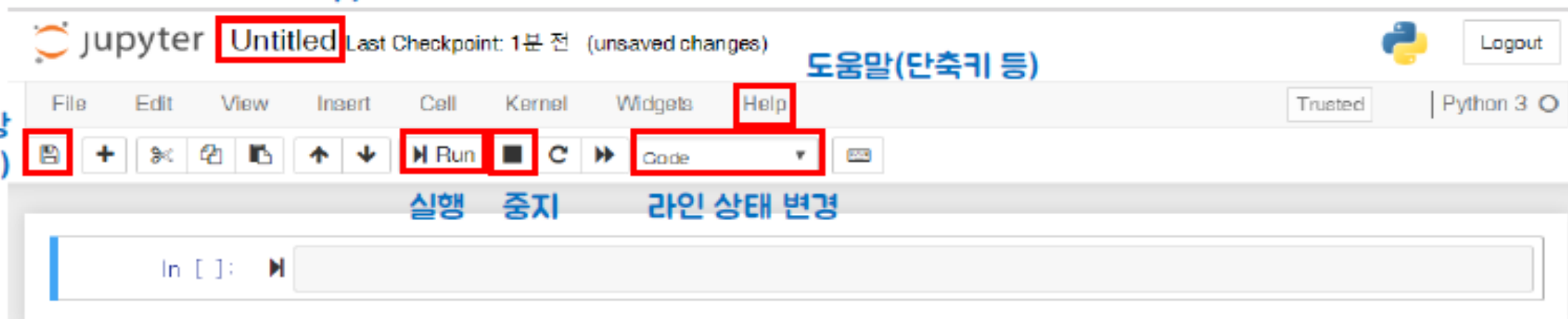
Other:
Text File
Folder
Terminal

11월 전
하루 전
3달 전

쥬피터 노트북 사용법

파일명.ipynb -> 쥬피터 노트북 파일 확장자

저장
(컨트롤+S)



도움말(단축키 등)

실행 중지 라인 상태 변경

<많이 쓰는 단축키> (Help -> Keyboard Shortcuts)

코드 실행 : 컨트롤+엔터
 : 쉬프트+엔터

위에 라인 추가 : a
아래에 라인 추가 : b
라인 제거 : dd

라인 나누기 : 컨트롤+쉬프트+'-'
라인 합치기 : 쉬프트+M
저장 : 컨트롤+S

1. 변수의 자료형

변수의 자료형

변수 (variable)

Variables in C: Call-by-value

```
int a = 1;
```



Putting the value in a box with the variable name.

```
a = 2;
```



If you change the value of the variable, the box will be updated with the new value.

```
int b = a;
```



Assigning one variable to another makes a copy of the value and put that value in the new box.

Names in Python: Call-by-object

```
a = 1
```



It **tags** the value with the variable name.

```
a = 2
```



It just **changes the tag** to the new value in memory

Garbage collection free the memory automatical

```
b = a
```



It makes a **new tag** bound to the same value.

영문, 숫자 사용가능

대소문자 구분

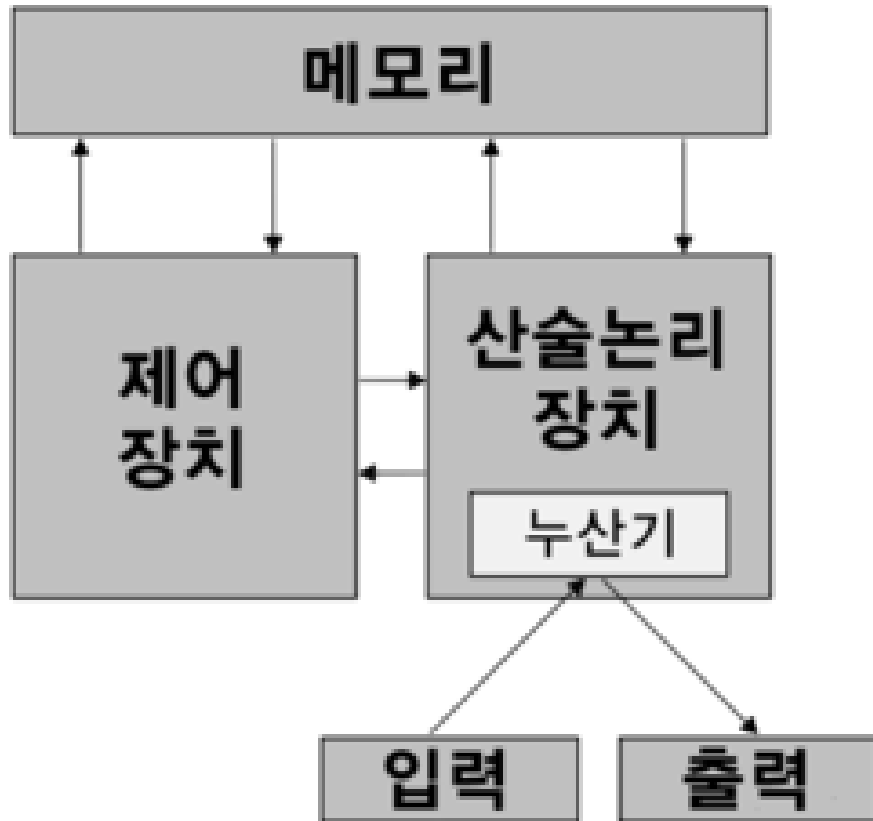
숫자부터 시작할 수 없음.

특수문자 사용 불가(+, -, *, /, \$ 등)

문제

- teacher = “Youngpyo Ryu” 의 의미는 ?
 - 1. teacher 의 이름은 Youngpyo Ryu 이다.
 - 2. teacher 는 Youngpyo Ryu 이다.
 - 3. teacher와 Youngpyo Ryu는 같다.
 - 4. teacher 에 Youngpyo Ryu를 넣어라.
-

컴퓨터의 구조 - 폰노이만 아키텍처.



폰 노이만 아키텍처에서는 사용자가 컴퓨터에 값을 입력하거나 프로그램을 실행하는 경우, 그 **정보를 먼저 메모리에 저장시키고 CPU가 순차적으로** 그 정보를 해석하고 계산하여 사용자에게 결과값 전달

변수(Variable)란?

- 프로그램에서 사용하기 위한 특정한 값을 저장하는 공간
- 선언 되는 순간 메모리 특정영역에 공간이 할당됨
- 변수에는 값이 할당되고 해당 값은 메모리에 저장됨.
- $A = 8$ 의 의미는 “A는 8이다”가 아닌 A라는 이름을 가진 메모리 주소에 8을 저장하라 이다.

변수의 자료형

기본 자료형(Data Type)

int	정수 (양/음의 정수)
float	실수 (소수점이 포함된 실수)
bool	True/False
str	문자열 (따옴표 “, ‘이 들어가는 있는 문자형)
list	리스트
tuple	튜플형
set	집합
dict	딕셔너리(dictionary)

숫자 자료형

Int

정수

Float

실수

```
>>> type(1)
```

```
int
```

```
>>> Type(1.0)
```

```
float
```

연산자

+

덧셈

```
>>> 1+1
```

```
2
```

```
>>> a, b = 1, 2.0
```

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

```
3.0
```

-

뺄셈

```
>>> 3-5
```

```
-2
```

```
>>> a, b = 1, 2.0
```

```
>>> a-b
```

```
-1.0
```

*

곱셈

```
>>> 3*5
```

```
15
```

```
>>> a, b = 3, 5.0
```

```
>>> a*b
```

```
15.0
```

/

나눗셈

```
>>> 3/5
```

```
0.6
```

```
>>> a, b = 3, 5.0
```

```
>>> a/b
```

```
0.6
```


연산자

제곱

```
>>> 3**3
```

```
27
```

```
>>> a, b = 4, 2
```

```
>>> a**b
```

```
16
```

//

몫

```
>>> 10//3
```

```
3
```

```
>>> a, b = 10, 3
```

```
>>> a//b
```

```
3
```

%

나머지 연산

```
>>> 10%3
```

```
1
```

```
>>> a, b = 10, 3
```

```
>>> a%b
```

```
1
```

연산자 우선순위

```
>>> 10+2*3
```

```
16
```

```
>>> (10+2)*3
```

```
36
```

숫자 자료형(정수 & 실수)

연산자 (+, -, *, /, **, //, %)

정수

```
>>> a = 5
>>> b = 3
>>> print(a+b) # 덧셈
8
>>> print(a-b) # 뺄셈
2
>>> print(a*b) # 곱셈
15
>>> print(a/b) # 나눗셈
1.6666666666666667
>>> print(a**b) # 제곱
125
>>> print(a//b) # / 후 몫
1
>>> print(a%b) # / 후
나머지
2
```

실수

```
>>> c = 10.0
>>> d = 4.0
>>> print(c+d) # 덧셈
14.0
>>> print(c-d) # 뺄셈
6.0
>>> print(c*d) # 곱셈
40.0
>>> print(c/d) # 나눗셈
2.5
>>> print(c**d) # 제곱
10000.0
>>> print(c//d) # / 후 몫
2.0
>>> print(c%d) # / 후
나머지
2.0
```

연산자 우선순위

```
>>> a = 5
>>> b = 2
>>> c = 10
>>> d = a+b*c
>>> print(d)
?
>>> e = (a+b)*c
>>> print(e)
?
```

자료형

int	정수
float	실수
bool	True/False
str	문자열
list	리스트
tuple	튜플형
set	집합
dict	dictionary

연습문제 1

아래와 같이 $a = 7$, $b = 3.0$ 로 초기화 하고
 a 를 b 로 나누었을 때 몫과 나머지를 각각 c 와 d 라는
변수에 초기화한 후 출력하라.
그리고 e 라는 변수에 a 의 b 제곱을 초기화하고 출력하라.

$a, b = 7, 3.0$

code 시작

$c = ?$

$d = ?$

$e = ?$

code 종료

형 변환(type cast)

묵시적 형변환

```
>>> a, b = 7, 3.0
>>> c = a*b
a 형변환
>>> d = a/b # a 형변환
```

```
>>> type(c)
float
>>> print(c)
21.0
```

```
>>> type(d)
float
>>> print(d)
2.3333333333333335
```

명시적 형변환

```
#
>>> c = int(c)
>>> type(c)
int
>>> print(c)
21

>>> d = int(d)
>>> type(d)
2

>>> c = float(d)
>>> type(d)
float
>>> print(d)
2.0
```

자료형 변환 함수

```
int 정수 변환
float 실수 변환
bool bool 변환
str 문자열 변환
list 리스트 변환
tuple 튜플형 변환
set 집합 변환
dict dictionary 변환
```

=

자료형

```
int 정수 변환
float 실수 변환
bool bool 변환
str 문자열 변환
list 리스트 변환
tuple 튜플형 변환
set 집합 변환
dict dictionary 변환
```

Boolean 자료형

True

참

False

거짓

```
>>> True
```

```
True
```

```
>>> False
```

```
False
```

```
>>> Type(True)
```

```
bool
```

비교연산자

>

크다

>=

크거나같
다

<

작다

<=

작거나같다

==

같다

!=

같지않다

연산결과는 Boolean

```
>>> print(3 > 1)
```

```
True
```

```
>>> print(10 != 10)
```

```
False
```

```
>>> type(10==10)
```

```
bool
```

논리 연산자

and

모두가 True 면 True

```
>>> True and True
```

True

```
>>> True and False
```

False

```
>>> False and False
```

False

or

하나만 True 면 True

```
>>> True or True
```

True

```
>>> True or False
```

True

```
>>> False or False
```

False

not

반대

```
>>> not True
```

False

```
>>> not False
```

True

비교 + 논리 연산자

5 > 10 and 20 == 20

False

True

False

5 == 5 or 10 != 10

True

False

True

Boolean 형변환 (other → bool)

숫자 자료형은 0이면 False
나머지는 True

```
>>> a = 0
>>> bool(a)
False
```

```
>>> b = -10
>>> bool(b)
True
```

str, list, tuple, set, dict 등 요소가 있는 자료형은
요소가 있으면 True, 요소가 없으면 False

```
>>> a = "" # str
>>> bool(a)
False
```

```
>>> a = "hello python" # str
>>> bool(a)
True
```

```
>>> bool([]) # list
False
```

```
>>> bool([1,2,3]) # list
True
```

```
>>> a = () # tuple
>>> bool(a)
True
```

```
>>> a = {1, 2} # set
>>> bool(a)
True
```

```
>>> a = {} # dict
>>> bool(a)
False
```

자료형

int	정수
float	실수
bool	True/False
str	문자열
list	리스트
tuple	튜플형
set	집합
dict	dictionary

연습문제 2

사용자로 점수를 3개 입력받아
모든 점수가 65점보다 클 경우 True 아닐 경우
False 를 출력하세요

비어있는 변수(Null)

None

아무값도 없는

```
>>> x = None
```

```
>>> print(x)
```

```
None
```

```
>>> type(x)
```

```
None Type
```

문자열 자료형

변수의 자료형

문자열

first = “Hello, World”

**multi = “”“Hello,World
Hello, Python”””**

문자열 작은 따옴표 (') 포함시키기

```
Python's favorite food is perl
```

```
>>> food = "Python's favorite food is perl"
```

큰따옴표(")가 아닌 작은따옴표(')로 문자열을 둘러싼 후 다시 실행해 보자.

```
>>> food = 'Python's favorite food is perl'  
File "<stdin>", line 1  
    food = 'Python's favorite food is perl'  
            ^  
SyntaxError: invalid syntax
```

문자열 연산하기

문자열 더해서 연결하기(Concatenation)

```
>>> head = "Python"
>>> tail = " is fun!"
>>> head + tail
'Python is fun!'
```

문자열 곱하기

```
>>> a = "python"
>>> a * 2
'pythonpython'
```

이스케이프 코드

\n
개행

```
>> print("hi \nfriend") >> print("hi \tfriend")  
hi  
friend          hi      friend
```

\t
개행

\\
개행

```
>> print("hi  
\\friend")  
hi \friend
```

\"
개행

```
>> print("hi  
\"friend")  
hi "friend
```

```
>> print('hi  
"friend")  
hi "friend
```

\'
개행

```
>> print('hi  
\'friend')  
hi 'friend
```

```
>> print("hi  
'friend")  
hi 'friend
```


문자열 인덱싱

s[i]

i번째 항.

```
>>> s = "hello world"
```

```
>>> s[0]  
'h'
```

```
>>> s[6]  
'w'
```

s[i:j]

i번째 항이상 j번째 미만

```
>>> s = "hello world"
```

```
>>> s[:5]  
'hello'
```

```
>>> s[6:]  
'world'
```

```
>>> s[3:8]
```

```
'lo wo'
```

```
>>> s[:]  
'hello world'
```

s[i:j:k]

i번째 항 이상 j번째 미만
k만큼 건너뛰면서

```
>>> s = "hello world"
```

```
>>> s[::2]
```

```
'hlowrd'
```

```
>>> s[1:6:3]
```

```
'eo'
```

```
>>> s[:5:2]
```

```
'hlo'
```

s[-i]

-index

```
>>> s = "hello world"
```

```
>>> s[-1]  
d
```

```
>>> s[:-1]
```

```
'hello worl'
```

```
>>> s[1:-4:3]  
'eo'
```

문자열 포매팅	기능
%d, %x, %o	십진수, 16진수, 8진수(복소수는 출력이 안 됨)
%f %.숫자f	실수를 출력 (복소수는 출력이 안 됨.) 표시할 소수점 아래 자리수를 명시한다.
%s	문자열 출력
%%	'%' 문자 자체를 출력

서식지정자 #1

“문자열 **%s** 문자열” % “추가문자”

```
>>> name = "tom"
```

```
>>> print("I am " + name + "!") # +사용
```

I am tom!

```
>>> print("I am %s!" % name) # 서식지정자사용
```

I am tom!

%(공백 level)s - 공백추가

```
>>> print("I am %10s!" % name)
```

I am tom!

```
>>> print("I am %-10s!" % name)
```

I am tom !

여러개의 추가문자

```
>>> f1, f2 = "apple", "banana"
```

```
>>> print("I like %s, %s!!" % (f1, f2))
```

I like apple, banana

서식 지정자 #2

“문자열 **%s** 문자열” **%** “추가문자”

```
>>> name = "tom"
```

```
>>> print("I am " + name + "!") # +사용
```

```
I am tom!
```

```
>>> print("I am %s!" % name) # 서식지정자사용
```

```
I am tom!
```

%s - 문자, **%d** - 정수, **%f** - 실수

```
>>> n1, n2 = 3, 3.2323
```

```
>>> print("n1 = %d, n2 = %f" % (n1, n2))
```

```
n1 = 3, n2 = 3.2323
```

```
>>> print("n2 = %.2f" % n2)
```

```
n2 = 3.23
```

Format 함수 변수사용

“문자열 {0}, {1} 문자열”.format(값, 값)

```
>>> “I like {0}, {1} !”.format(“apple”, “banana”)
```

I like apple, banana

```
>>> “Number {0} {2} {1}”.format(1,2,3)
```

Number 1 3 2

```
>>> “Number {0} {0} {1}”.format(1,2,3)
```

Number 1 1 2

```
>>> “Number {} {} {}”.format(1,2,3)
```

Number 1 2 3

```
>>> f1, f2 = “apple”, “banana”
```

```
>>> “I like {0}, {1} !”.format(f1, f2)
```

I like apple, banana

{0:(숫자)<(숫자)} - 공백추가

```
>>> “Number {0:>4}!”.format(1)
```

Number 1!

```
>>> “Number {0:<4}!”.format(1)
```

Number 1 !

```
>>> “Number {0:==<4}!”.format(1)
```

Number ===1!

```
>>> “Number {0:-^5}!”.format(1)
```

Number --1--!

f문자열 (python >=3.6)

f“문자열 {변수} 문자열”

```
>>> math = 95  
>>> english = 65  
>>> science = 40
```

```
>>> print(f"수학점수: {math}\n영어점수: {english:^10}\n과학점수: {science}")
```

```
수학점수 95  
영어점수 -----65-----  
과학점수 40
```

```
>>> print(f"수학점수 {math+5}\n영어점수 {english}\n과학점수 {science}")
```

```
수학점수 100
```

```
영어점수 65
```

```
과학점수 40
```

연습문제 4

`a="20190505chicken19000"`는 잘못 쓰여진 변수이다.

2019는 `year`이라는 변수에, 그 다음 0505를 `day`라는 변수에,

`chicken`을 `menu`라는 변수에, 19000을 `money`라는 변수에

인덱싱을 사용하여 각각 저장하고 출력하시오.

```
a = "20190505chicken19000"
```

```
# code 작성
```

```
year = ?
```

```
day = ?
```

```
menu = ?
```

```
money = ?
```

```
# code 종료
```

연습 문제 5

a="90:30:80"이고 :을 기준으로 각각 수학점수, 영어점수, 과학점수이다.

문자열 인덱싱을 이용해 각 점수를 각각 math, english, science 변수에 저장하고, average 란 변수에 평균값을 저장하고 출력하시오.

```
a = "90:30:80"
```

```
# code 작성
```

```
math = ?
```

```
english = ?
```

```
science = ?
```

```
average = ?
```

```
# code 종료
```

문자열 함수 소개 #1

len(값)

문자 길이

```
>>> word = "Apple banana"
```

```
>>> len(word) #문자길이
```

index(찾을문자)

Index 찾기

```
>>> word.index("p")
```

```
1
```

upper() / lower()

소문자 바꾸기

```
>>> word.upper()  
'APPLE BANANA'
```

```
>>> word.lower()  
'apple banana'
```

replace(바꿀문자, 새문자)

문자열 바꾸기

```
>>> word.replace("banana","Orange")  
'Apple orange'
```


문자열 함수소개 #2

count(target 문자)

target 문자 개수

```
>>> word = "Apple Orange"
```

```
>>> count('A')
```

```
1
```

```
>>> count("p")
```

```
2
```

strip() / lstrip() / rstrip()

공백 제거

```
>>> word =  
" Apple Orange "
```

```
>>> word.lstrip()  
'Apple Orange '
```

```
>>> word.rstrip()  
' Apple Orange'
```

```
>>> word.strip()  
'Apple Orange'
```

split(기준 문자)

문자열 분리하기

```
>>> scores = "10:20:50:80:88"
```

```
>>> scores.split(":")  
['10', '20', '50', '80', '88']
```

```
>>> word = "Apple Orange"
```

```
>>> word.split()  
['Apple', 'Orange']
```

join

리스트에서 문자열로

```
>>> score_list = ['10', '20',  
'50', '80', '88']
```

```
>>> ",".join(score_list)  
'10,20,30,40,50'
```

연습문제 6

a=" 90:30:80 "이고 :을 기준으로 각각 수학점수, 영어점수, 과학점수이다.

문자열 함수를 이용해 공백을 제거하고,
각 점수를 각각 math, english, science 변수에 저장한 후,
average 란 변수에 평균값을 저장하고 출력하시오.

hint: strip, split을 이용한다.

```
a = " 90:30:80 "  
  
# code 작성  
math, english, science = ?  
  
average = ?  
  
# code 종료
```

리스트와 튜플



리스트와 튜플 선언

시퀀스 자료형 - 연속된 여러값들을 한 변수에 저장

리스트 = [값, 값, 값 ...]

수정 및 추가 가능

```
>>> a = [1, 2, 3, 4, 5, 6, 7]
```

```
>>> a = [1, "apple", 3.14, False]
```

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

```
<class 'list'>
```

```
>>> a = [] # 빈값생성
```

```
>>> a = list() # 빈값생성
```

튜플 = (값, 값, 값)

수정 및 추가 불가능

```
>>> b = (1, 2, 3, 4, 5, 6, 7)
```

```
>>> b = (1, "apple", 3.14, False)
```

```
>>> type(b)
```

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

```
>>> b = () #빈값생성
```

```
>>> b = tuple()
```

```
>>> c = 1, # 요소가 1인 튜플
```

Range 함수

- 연속된 숫자를 생성하는 함수

range(횟수)

```
>>> a = list(range(10))
```

```
[0,1 ,2 ,3 ,4 ,5, 6, 7, 8, 9]
```

```
>>> a = list(range(1,11))
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

```
>>> a= tuple(range(10))
```

```
(0,1,2,3,4,5,6,7,8,9)
```

range(시작, 끝, 증가폭)

```
>>> a = list(range(-10, 10, 2))
```

```
[-10, -8, -6, -4, 2, 0, 2, 4, 6, 8]
```

```
>>> a = list(range(5, 0 ,-1))
```

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

```
>>> a= tuple(range(10))
```

```
(0,1,2,3,4,5,6,7,8,9)
```

리스트, 튜플 형 변환

list(튜플)

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

```
>>> tuple(a)
```

```
(1, 2, 3, 4, 5)
```

tuple(리스트)

```
>>> a = (1, 2, 3, 4, 5)
```

```
>>> list(a)
```

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

인덱스 접근 #1

기본접근 및 변경

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

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

```
3
```

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

```
5
```

```
>>> a[3] = 8
```

```
>>> a
```

```
[1, 2, 3, 8, 5]
```

슬라이스

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

```
>>> a[0:4]
```

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

```
>>> a[3:5]
```

```
[4, 5]
```

```
>>> a[1:-2]
```

```
[2, 3]
```

인덱스 접근 #2

증가폭 변경

```
>>> a = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

```
>>> a[2: 8: 2]
```

```
[3, 5, 7]
```

끝 인덱스까지 가져오기

```
>>> a = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

```
>>> a[:3]
```

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

```
>>> a[3:]
```

```
[4, 5, 6, 7, 8, 9, 10]
```

슬라이스 요소할당

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

```
>>> a[1:4] = ["a", "b", "c"]
```

```
>>> a
```

```
[1, a, b, c, 5]
```

```
>>> a[1:4] = ["d", "e"]
```

```
>>> a
```

```
[1, d, e, 5]
```

슬라이스 삭제

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

```
>>> del a[1:4]
```

```
[1, 5]
```


리스트와 튜플 기능 #1

특정값 있는지 확인 in

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

```
>>> 1 in a
```

```
True
```

```
>>> 6 in a
```

```
False
```

연결하기 +

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

```
>>> b = [6, 7, 8, 9]
```

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

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

반복하기 *

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

```
>>> a * 3
```

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

요소개수 구하기 len()

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

```
>>> len(a)
```

```
3
```

리스트 함수 #1

요소 추가하기 `append(값)`, `extend(리스트)`

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

```
>>> a.append(6)
```

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

```
>>> a.extend([7, 8])
```

```
[1, 2, 3, 4, 5, 6, 7, 8]
```

특정 인덱스에 요소추가 `insert(인덱스, 값)`

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

```
>>> a.insert(2, 100)
```

```
[1, 2, 100, 3]
```

리스트 요소삭제 `pop(인덱스)`

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

```
>>> a.pop(0)
```

```
[2, 3]
```

```
>>> a.pop()
```

```
[2]
```

리스트 특정값을 찾아삭제 `remove(값)`

```
>>> a = [100, 200, 300]
```

```
>>> a.remove(200)
```

```
[100, 300]
```

리스트 함수 #2

특정값의 인덱스 구하기 index(값)

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

```
>>> a.index(3)
```

²특정값의 개수구하기 count(값)

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

```
>>> a.count(2)
```

³순서 뒤집기 reverse()

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

```
>>> a.reverse()
```

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

정렬하기(오름차순) sort(), sort(reverse=False)

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

```
>>> a.sort()
```

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

정렬하기(내림차순) sort(reverse=True)

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

```
>>> a.sort(reverse=True)
```

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

리스트 할당과 복사

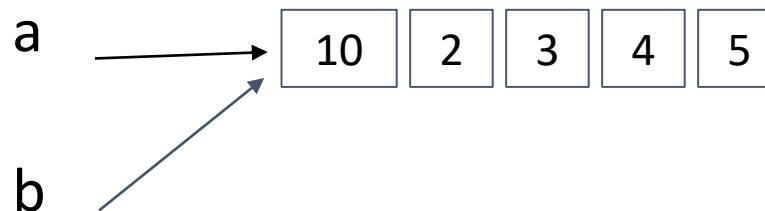
실제 값들의 복사가 일어나지 않음

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

```
>>> b = a
```

```
>>> a is b
```

```
True
```



```
>>> b[0] = 10
```

```
>>> print(a)
```

```
[10, 2, 3, 4, 5]
```

리스트 할당과 복사

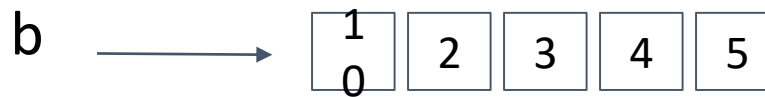
copy() 함수로 실제 값들을 복사

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



```
>>> b = a.copy()
```

```
>>> a is b
```



```
False
```

```
>>> b[0] = 10
```

```
>>> print(a)
```

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

다차원 리스트와 튜플

리스트 = [[값, 값], [값, 값] ...]

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

튜플 = ((값, 값), (값, 값) ...)

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

접근시 리스트[큰인덱스][작은인덱스]

```
>>> a[0][1]
```

```
2
```

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

```
3
```

import *module*

- import 사용 방법

- `import module_name`
- `module` : 함수나 변수 또는 클래스 들을 모아 놓은 파일

- sys module

- Python interpreter가 제공하는 변수들과 함수들을 직접 제어할 수 있게 해주는 module
- `stdin`, `stdout`, `stderr` 등의 attribute 포함

다차원 복사

2차원 이상의 리스트는 `copy.deepcopy()` 를 이용 복사

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

```
>>> b = a.copy()
```

```
>>> b[0][0] = 10
```

```
>>> a
```

```
[[10, 2], [3, 4], [5, 6]]
```

```
>>> import copy
```

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

```
>>> b = copy.deepcopy(a) #깊은복사
```

```
>>> b[0][0] = 10
```

```
>>> print(a)
```

```
>>> print(b)
```

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

```
[[10, 2], [3, 4], [5, 6]]
```


덕셔너리

딕셔너리

딕셔너리 = {키1: 값1, 키2: 값2}

score = {"name": "Tom", "math": 80, "english": 70}

```
>>> score = {"name": "tom", "math": 80, "english": 70}
```

```
>>> score["name"]
```

tom

```
>>> score["name"] = "Mike"
```

```
>>> score["name"]
```

Mike

```
>>> type(score)
```

dict

Dict

딕셔너리 = dict(키1=값1, 키2=값2)

score = dict(name="Tom", math=80, english=70)

```
>>> score = dict(name="tom", math=80, english=70)
```

```
>>> score["name"]
```

```
tom
```

```
>>> score = dict() #비어있는 딕셔너리
```

```
>>> score = {} #비어있는 딕셔너리
```

딕셔너리 기능 #1

키가 있는지 확인

```
>>> score = {"name": "tom", "math": 80,  
             "english": 70}
```

```
>>> "math" in score
```

```
True
```

```
>>> "age" in score
```

```
False
```

키의개수

```
>>> len(score)
```

```
3
```

키-값 쌍 추가하기 setdefault(키, 값)

```
>>> score.setdefault("age", 20)
```

```
{"name": "tom", "math": 80, "english": 70, "age": 20}
```

키-값 수정하기 update({키: 값})

```
>>> score.update({"math": 90})
```

```
{"name": "tom", "math": 90, "english": 70, "age": 20}
```

딕셔너리 기능 #2

키로 딕셔너리 항목삭제 pop(키,기본값)

```
>>> score = {"name": "tom", "math": 80, "english": 70}
```

```
>>> score.pop("name") #삭제된 키의 값 반환
```

```
tom
```

```
>>> score
```

```
{"math":80, "english":70}
```

```
>>> score.pop("age", 0)
```

```
0
```

모든 값 삭제 clear()

```
>>> score.clear()
```

```
>>> score
```

```
{}
```

모든 키, 값 가져오기

```
>>> score.keys()
```

```
dict_keys(["math", "english"])
```

```
>>> score.values()
```

```
dict_values([80, 70])
```

```
>>> score.items()
```

```
dict_items([("math",80), ("english",70)])
```

딕셔너리 할당과 복사

딕셔너리 복사 `copy()`

```
>>> a = {"a": 0, "b": 1}
```

```
>>> b = a.copy() #리스트와 마찬가지로
```

중첩 딕셔너리의 경우 `deepcopy()`

```
>>> import copy
```

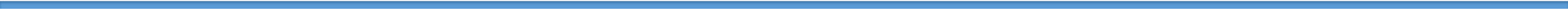
```
>>> a = {"a": {"c": 0, "d": 0}, "b": {"e": 0, "f": 0}}
```

```
>>> b = copy.deepcopy(a)
```

```
>>> print(b)
```

```
{"a": {"c": 0, "d": 0}, "b": {"e": 0, "f": 0}}
```

집합 (Set)



세트

세트 = {값1, 값2, 값3, 값4}

animal = {"dog", "cat", "monkey", "horse"}

```
>>> animal = {"dog", "cat", "monkey", "horse"}
```

```
>>> type(score)
```

```
<class 'set'>
```

세트의 기능

세트에 특정값 확인

```
>>> animal = {"dog", "cat", "monkey", "horse"}
```

```
>>> "cat" in animal
```

```
True
```

set을 사용하여 세트 만들기

```
>>> a = set("animal")
```

```
>>> a
```

```
{"a", "n", "i", "m", "a", "l"}
```

```
>>> b = set(range(5))
```

```
>>> b
```

```
{0, 1, 2, 3, 4}
```

집합의 연산 #1

합집합 |, set.union

```
>>> a = {1, 2, 3}
```

```
>>> b = {3, 4, 5}
```

```
>>> a | b
```

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

```
>>> set.union(a, b)
```

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

교집합 &, set.intersection

```
>>> a = {1, 2, 3}
```

```
>>> b = {3, 4, 5}
```

```
>>> a & b
```

```
{3}
```

```
>>> set.intersection(a, b)
```

```
{3}
```

집합의 연산 #2

차집합 -, set.difference

```
>>> a = {1, 2, 3}
```

```
>>> b = {3, 4, 5}
```

```
>>> a - b
```

```
{1, 2}
```

```
>>> set.difference(a, b)
```

```
{1, 2}
```

대칭차집합 ^, set.symmetric_difference

```
>>> a = {1, 2, 3}
```

```
>>> b = {3, 4, 5}
```

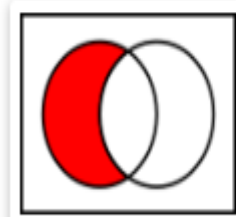
```
>>> a ^ b
```

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

```
>>> set.symmetric_difference(a, b)
```

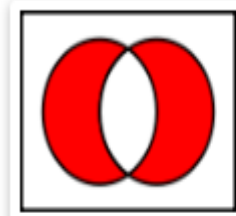
```
{1, 2, 4, 5}
```

4. 차집합 [편집]



```
complement = set1 - set2  
# set(['A', 'B'])
```

5. 대칭차집합 [편집]



```
sym_diff = set1 ^ set2  
# set(['A', 'B', 'E', 'F'])
```

부분집합, 상위집합 확인

부분집합 \leq , `issubset(다른세트)`

```
>>> a = {1, 2, 3, 4}
```

```
>>> a <= {1, 2, 3, 4, 5}
```

True

```
>>> a.issubset({1, 2, 3, 4, 5})
```

True

```
>>> a <= {1, 2, 3}
```

False

```
>>> a.issubset({1, 2, 3})
```

False

상위집합 \geq , `issuperset(다른세트)`

```
>>> a = {1, 2, 3, 4}
```

```
>>> a >= {1, 2, 3}
```

True

```
>>> a.issuperset({1, 2, 3})
```

True

```
>>> a >= {1, 2, 3, 4, 5}
```

False

```
>>> a.issuperset({1, 2, 3, 4, 5})
```

False

겹치는 요소확인

isdisjoint(다른세트)

```
>>> a = {1, 2, 3, 4}
```

```
>>> a.isdisjoint({5, 6, 7, 8}) #겹치는 요소 없음
```

```
True
```

```
>>> a.isdisjoint({2, 3, 4, 5}) #2, 3, 4 겹침
```

```
False
```

세트 조작하기

추가하기 add(요소)

```
>>> a = {1, 2, 3, 4}
```

```
>>> a.add(5)
```

```
>>> a
```

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

삭제하기 remove(요소), discard(요소)

```
>>> a = {1, 2, 3, 4}
```

```
>>> a.remove(1)
```

```
>>> a
```

```
{2, 3, 4}
```

```
>>> a.discard(2)
```

```
{3, 4}
```

2. 제어문

조건문 if

조건문이란 ?

- 조건에 따라 특정한 동작을 하게하는 명령어
- 프로그램 예시 in 생활
 - 지하철 앞차 간격이 10M 이하면 속도를 10km 이하로 낮춰라
 - 사용자가 20세 이하면 VOD를 플레이 하지 마라
 - 휴대폰 패턴이 5회 틀리면 20초 동안 대기 상태로 만들어라
- 조건문은 조건을 나타내는 기준과 실행해야 할 명령으로 구성됨
- 조건의 참, 거짓에 따라 실행해야 할 명령이 수행되거나 되지 않음
- 파이썬은 조건문으로 if, else, elif 등의 명령 키워드를 사용함.

if

조건문은 특정 조건일 때 코드를 실행하는 문법

if 조건식:

코드

들여쓰기 or 탭

공식 4칸

```
>>> x = 10
```

```
>>> if x == 10:
```

```
>>>     print("x 가 10 입니다")
```

if

if 문 코드 생략

```
>>> x = 10  
  
>>> if x == 10:  
    pass
```

if 문 들여쓰기

```
>>> x = 10  
  
>>> if x == 10:  
    >>> print("x 가 10 입니다")  
    >>> print("x 가 12 가 아닙니다")  
    >>> print("x 가 10 입니다") #error  
    >>> print("x가 10입니다") #error  
    >>> print("if문 밖") #if 문과는 상관없음
```

if

다양한 조건들

```
>>> if x > 3:
```

```
>>> print("x 가 3보다 크다")
```

```
>>> if x > 2 and x < 10:
```

```
>>> print("x가 2보다 크고 10보다 작다")
```

```
>>> 0 < x < 20:
```

```
>>> print("x는 0보다 크고 20보다 작다")
```

중첩 if 문

```
>>> if x >= 10:
```

```
>>>     if x <= 20:
```

```
>>>         print("10이상 20이하")
```

```
>>>     elif x <= 30:
```

```
>>>         print("20초과 30이하")
```

if

```
print('Tell me your age?')
myage = int(input()) #나이를 입력받아 mypage 변수에 할당
if myage < 30 : #mypage가 30미만일때,
    print('Welcome to the Club')
else :
    print("Oh, No. You are not accepted.")
```

elif와 else

if 조건문의 분기를 위한 문법

if 조건식: #조건1

코드 #조건1 True

elif 조건식: #조건2

코드 #조건1 False, 조건2 True

elif 조건식: #조건3

코드 #조건1 False, 조건2 False, 조건3 True

else:

코드 #모든 조건식이 False

```
>>> if x == "A":
```

```
>>>     print("x 는 A")
```

```
>>> elif x == "B":
```

```
>>>     print("x 는 B")
```

```
>>> elif x == "C":
```

```
>>>     print("x 는 C")
```

```
>>> else:
```

```
>>>     print("x 는 A, B, C 가 아님")
```

연습문제 1

사용자로 점수를 3개 입력받아

모든 점수가 65점보다 클 경우 합격 아닐 경우 불합격을 출력하세요

단, 0~100 이 아닌 숫자가 입력된 경우 잘못된 "잘못된 점수가 입력되었습니다" 를 출력하세요

연습문제 2

[홀수 짝수 판별기]

사용자로부터 정수를 하나 입력받아
입력한 정수가 홀수인지 짝수인지 판별하여라.

[출력결과] (** 0은 짝수라 하자.)

정수를 입력해주세요: 5
입력하신 5는 홀수입니다.

짝수를 입력해주세요: 10
입력하신 10은 짝수입니다.

반복문 for

for

어떠한 코드를 반복해야할때 사용

지정된 범위만큼 (주로 반복횟수가 정해져있을때 사용)

for 변수 in [iterator]:

코드

들여쓰기 or 탭

공식 4칸

```
>>> for i in range(0,10):  
>>>     print("현재값 : ", i)
```

for

for문과 range()

```
>>> for i in range(0, 10, 2):
```

```
>>>     print(i)
```

```
for i in range(10,0,-1):
```

```
    print(i)
```

for문과 iterator

```
>>> a = [1, 2, 3, 4, 5, 6, 7]
```

```
>>> for i in a :
```

```
>>>     print(i)
```

```
>>> for i in "Orange":
```

```
>>>     print(i, end=" ")
```

```
>>> a = {"name": "tom", "math": 80, "english": 70}
```

```
>>> for i in a:
```

```
>>>     print(i, end=" ")
```

```
>>>     print(a[i])
```

연습문제 1

- (1) 사용자로부터 정수를 입력받아, 해당 정수만큼 “안녕”을 출력하세요.
- (2) 사용자로부터 정수를 입력받아, 입력된 정수 만큼 별 찍기

(1) [출력결과]

```
정수를 입력해주세요: 5
안녕
안녕
안녕
안녕
안녕
```

(2) [출력결과]

```
정수를 입력해주세요: 5
*
**
***
****
*****
```

연습문제 2

- (3) 사용자로부터 정수를 입력받아, 입력된 정수 만큼 별 찍기(역순)

(3) [출력결과]

정수를 입력해주세요: 5

**

*

for

enumerate 사용하여 index 접근

```
>>> a = ['a', 'b', 'c', 'd', 'e', 'f', 'g']  
>>> for idx, val in enumerate(a):  
>>>     print(idx, val, sep=", ")
```

for문 중첩

```
>>> for i in range(0, 10, 2)  
>>>     print("*****", i)  
>>>     for j in range(0, 10, 2)  
>>>         print("j : ", j)
```

연습문제 2

- (1) $x = [3, 6, 9, 20, -7, 5]$ 의 값의 모든 요소에 10을 곱한뒤 출력하세요
- 심화 : 출력과 리스트 x 의 값에도 모두 10을 곱해주세요
- (2) $y = \{\text{"math": 70, "science": 80, "english": 20}\}$ 의 값의 모든 요소에 10을 더한뒤 출력하세요

심화 : 출력과 딕셔너리 y 의 값에도 모두 10을 더해주세요

- (3) 숫자를 입력받고 입력받은 정수의 구구단을 출력하세요

(3) [출력결과]

정수를 입력해주세요: 5

$5 * 1 = 5$

$5 * 2 = 10$

$5 * 3 = 15$

...

...

연습문제3

- (1) word = ["school", "game", "piano", "science", "hotel", "mountain"]
중 글자수가 6글자 이상인 문자를 모아 새로운 리스트를 생성하세요
- (2) 구구단을 1단부터 9단까지 출력하세요

연습문제 4

1-100 까지 숫자중

3과 5의 공배수일경우 "3과 5의 공배수"

나머지 숫자중 3의배수일경우 "3의배수"

나머지 숫자중 5의배수일경우 "5의배수"

모두 해당되지 않을경우 그냥숫자
를 출력하세요

심화 : 1-입력한숫자까지의 숫자중

반복문 while

반복문 while

어떠한 코드를 반복해야할때 사용

조건에 따라 반복 (주로 반복횟수가 정해져있지 않을때)

while 조건식:

코드

들여쓰기 or 탭

공식 4칸

```
>>> i = 0
```

```
>>> while i < 10:
```

```
>>>     print("현재값 : ", i)
```

```
>>>     i += 1
```

While 응용

입력한 횟수만큼 반복하기

```
>>> count = int(input("반복횟수?"))  
  
>>> i = 0  
  
>>> while i < count:  
  
>>>     print("입력한 횟수만큼 반복")  
  
>>>     i += 1
```

입력조건이 맞을때까지 반복하기

```
>>> i = 0  
  
>>> while i != 5:  
  
>>>     i = int(input("5를 입력하면 반복이 중단됩니다."))  
  
>>> print("중단!")
```

연습문제 5

사용자로부터 숫자를 계속 입력받다가

s or S 를 입력하면 합계출력

값을 입력해주세요30

값을 입력해주세요20

값을 입력해주세요50

값을 입력해주세요40

값을 입력해주세요30

값을 입력해주세요s

합계는 ? 170

연습문제6 (가위바위보 게임만들기)

random 함수 사용방법 - 랜덤값호출

```
>>> import random #random 모듈을 가져온다
```

```
>>> random.random()
```

```
0.00202302032
```

```
>>> random.randint(1,3)
```

```
3
```

가위(1), 바위(2), 보(3) 을 입력해주세요! : 3

유저 : 보, 컴퓨터 : 보

가위(1), 바위(2), 보(3) 을 입력해주세요! : 2

유저 : 바위, 컴퓨터 : 보

가위(1), 바위(2), 보(3) 을 입력해주세요! : 1

유저 : 가위, 컴퓨터 : 보

가위(1), 바위(2), 보(3) 을 입력해주세요! : 4

게임종료 (전체:3 ,승리:1)

1~3 을 입력하면 게임진행 이외의 숫자를 입력하면 게임종료

심화 : 비긴숫자도 출력하세요 (전체:3, 승리:1, 비김:1)

리스트 응용

연습문제 7

randint함수를 사용하여 정수(1~1000)로 이루어진 길이가 100개인
리스트 a를 만들고

가장 큰 수를 구하는 코드를 작성하시오.

예) a=[32, 45, 2, 5, 76] 일때, 출력결과는 76이 나와야 함.

리스트의 가장 큰 수 가장 작은 수 구하기

```
>>> a = [32, 45, 2, 5, 76]
```

```
>>> small = a[0]
```

```
>>> for i in a:
```

```
>>>     if i < small:
```

```
>>>         small = i
```

```
>>> large = a[0]
```

```
>>> for i in a:
```

```
>>>     if i > large:
```

```
>>>         large = i
```

```
>>> a = [32, 45, 2, 5, 76]
```

```
>>> a.sort()
```

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

```
>>> a.sort(reverse=True)
```

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

```
>>> a = [32, 45, 2, 5, 76]
```

```
>>> min(a)
```

```
>>> max(a)
```

연습문제8

randint함수를 사용하여 정수(1~1000)로 이루어진 길이가 100개인
리스트 a를 만들고

리스트의 모든 요소의 합계를 구하는 코드를 작성하시오.

예) a=[32, 45, 2, 5, 76] 일때, 출력결과는 160이 나와야 함.

합계 구하기

```
>>> a = [32, 45, 2, 5, 76]
```

```
>>> b = 0
```

```
>>> for i in a:
```

```
>>>     b += i
```

```
>>> a = [32, 45, 2, 5, 76]
```


```
>>> sum(a)
```

리스트 컴프리헨션(list comprehension)

list[식 for 변수 in 리스트]

```
>>> a = [i for i in range(10)]
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```



c = [i + 5 for i in range(10)]

```
>>> a = [i + 5 for i in range(10)]
```

```
[5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
```

```
>>> a = [i * 3 for i in range(10)]
```

```
[0, 3, 6, 9, 12, 15, 18, 21, 24, 27]
```

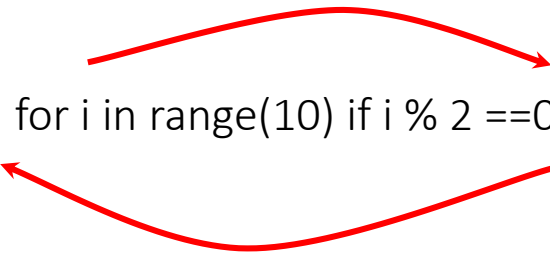
리스트 컴프리헨션(list comprehension)

list[식 for 변수 in 리스트 if 조건]

```
>>> a = [i for i in range(10) if i % 2 == 0]  
[0, 2, 4, 6, 8]
```

```
>>> a = [i for i in range(10) if i % 2 == 1]  
[1, 3, 5, 7, 9]
```

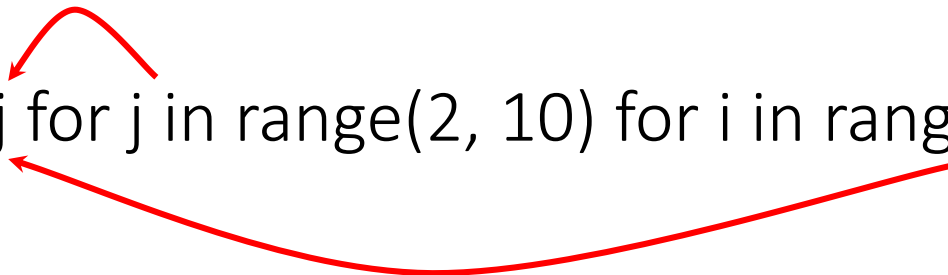
c = [i for i in range(10) if i % 2 == 0]



리스트 컴프리헨션(list comprehension)

list[식 for 변수 in 리스트 for 변수 in 리스트]

c = [i * j for j in range(2, 10) for i in range(1, 10)]



```
>>> a = [i * j for j in range(2, 10) for i in range(1, 10)]
```

```
a
```

```
[2, 4, 6, 8, 10 ..... 81]
```

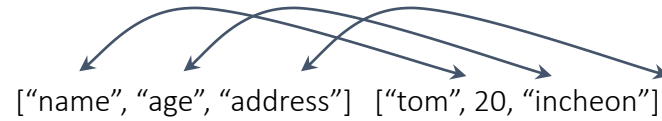
리스트 컴프리헨션(list comprehension)

리스트를 딕셔너리로 변경

```
>>> keys = ["name", "age", "address"]  
>>> users = ["tom", 20, "incheon"]  
>>> dicdic = { keys[i] : users[i] for i in range(0,3) }  
>>> dicdic  
{'name': 'tom', 'age': 20, 'address': 'incheon'}
```

ZIP

zip(리스트1, 리스트2)



zip(*iterable)은 동일한 개수로 이루어진 자료형을 묶어 주는 역할을 하는 함수이다.

```
>>> keys = ["name", "age", "address"]
```

```
>>> users = ["tom", 20, "incheon"]
```

```
>>> dic = dict(zip(keys, users))
```

```
>>> print(dic)
```

```
{'name': 'tom', 'age': 20, 'address': 'incheon'}
```

```
>>> lis = list(zip(keys, users))
```

```
>>> print(lis)
```

```
[('name', 'tom'), ('age', 20), ('address', 'incheon')]
```


연습문제9

(1) word = ["school", "game", "piano", "science", "hotel", "mountn"]
중 글자수가 6글자 이상인 문자를 모아 새로운 리스트를 생성하세요

(리스트 컴프리헨션을 사용해주세요)

(2) word = ["school", "game", "piano", "science", "hotel", "mountian"]
리스트의 글자수가 들어가있는 새로운 리스트를 생성하세요

(리스트 컴프리헨션을 사용해주세요)

2차원 리스트 응용

2차원 리스트 선언

```
>>> a = [[10, 20], [30, 40], [50, 60]]
```

```
>>> a = [[10, 20],
```

```
        [30, 40],
```

```
        [50, 60]]
```

```
>>> a
```

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

```
>>> a[0][1]
```

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

2차원 리스트 값추가

```
>>> a = [[10, 20], [30, 40], [50, 60]]
```

```
>>> a[0].append(10)
```

```
>>> a[1].append(20)
```

```
>>> a[2].extend([1,2])
```

```
>>> print(a)
```

```
>>> pprint(a)
```

For문을 이용하여 2차원리스트 값 꺼내기

for 반복문 한번사용

```
>>> a = [[10, 20], [30, 40], [50, 60]]
```

```
>>> for x, y in a:
```

```
>>>     print(x, y)
```

```
10 20
```

```
30 40
```

```
50 60
```

for 문을 두번사용

```
>>> a = [[10, 20], [30, 40], [50, 60]]
```

```
>>> for i in a:
```

```
>>>     for j in i:
```

```
>>>         print(j, end=' ')
```

```
>>> print()
```

```
10 20
```

```
30 40
```

```
50 60
```

for문을 이용하여 2차원리스트 값 꺼내기

for 와 range 사용

```
>>> a = [[10, 20], [30, 40], [50, 60]]
```

```
>>> for i in range(len(a)):
```

```
>>>     for j in range(len(a[i])):
```

```
>>>         print(a[i][j], end=" ")
```

```
>>>     print()
```

```
10 20
```

```
30 40
```

```
50 60
```

for 와 enumerate 사용

```
>>> a = [[10, 20], [30, 40], [50, 60]]
```

```
>>> for idx, val in enumerate(a):
```

```
>>>     for idx2, val2 in enumerate(val):
```

```
>>>         print(idx, idx2, val2)
```

```
0 0 10
```

```
0 1 20
```

```
1 0 30
```

```
1 1 40
```

```
2 0 50
```

```
2 1 60
```

While 문을 이용하여 2차원리스트 값 꺼내기

while 반복문 한번사용

```
>>> a = [[10, 20], [30, 40], [50, 60]]

>>> i = 0

>>> while i < len(a):

>>>     x, y = a[i]

>>>     print(x, y)

>>>     i += 1
```

while 문을 두번사용

```
>>> a = [[10, 20], [30, 40], [50, 60]]

>>> i = 0

>>> while i < len(a):

>>>     j = 0

>>>     while j < len(a[i]):

>>>         print(a[i][j], end=' ')

>>>         j += 1

>>>     print()

>>>     i += 1
```

연습문제9

(1) 아래 두 리스트를 각 요소들을 곱해 새로운 리스트 c 를 만드세요

$a = [[10, 20], [30, 40], [50, 60]]$

$b = [[2, 3], [4, 5], [6, 7]]$

- 심화 2X3 리스트 a, b를 사용자로부터 입력받아 두 값을 곱해 새로운 리스트 c 를 만드세요

for 반복문을 사용 2차원 리스트 만들기

```
>>> a = []  
>>> for i in range(3):  
>>>     line = []  
>>>     for j in range(2):  
>>>         line.append(0)  
>>>     a.append(line)  
>>> print(a)  
[[0, 0],[0, 0],[0, 0]]
```

연습문제 10

(1) 아래의 학습코드를 가지고 a 리스트가 [[1,2],[3,4],[5,6]] 와 같이 만들어지도록 수정하세요

```
>>> a = []
```

```
>>> for i in range(3):
```

```
>>>     line = []
```

```
>>>     for j in range(2):
```

```
>>>         line.append(0)
```

```
>>>     a.append(line)
```

```
>>> print(a)
```

break, Continue

break, continue

for, while 에서 제어흐름을 벗어나기 위해사용

break

for, while 을 완전히 중단

continue

처음으로 돌아가 다음반복 수행

break, continue

for 문에서의 예제

break

```
>>> for i in range(5):
```

```
>>>     if(i == 3):
```

```
>>>         break
```

```
>>>     print(i, end=" ")
```

결과

012

continue

```
>>> for i in range(5):
```

```
>>>     if(i == 3):
```

```
>>>         continue
```

```
>>>     print(i, end=" ")
```

결과

0124

break, continue

while 문에서의 예제

break

```
>>> i = 0  
  
>>> while i < 30:  
  
>>>     if i == 20 :  
  
>>>         break  
  
>>>     print(i, end=" ")  
  
>>>     i += 1
```

결과

012.....19

continue

```
>>> i = 0  
  
>>> while i < 30:  
  
>>>     i += 1  
  
>>>     if i % 2 == 0:  
  
>>>         continue  
  
>>>     print(i, end=" ")
```

결과

1 3 5 7 29

연습문제11

- (1) while 문과 break 문을 사용하여 사용자가 입력한 숫자만큼 반복하여 'hi' 출력하기
 - (2) for 문과 continue 문을 사용하여 사용자가 입력한 숫자까지의 짝수를 출력하기
-

4. 함수

함수

특정한 기능을 반복해서 사용해야할때



함수이름 인자
def func(a, b):

print("함수입니다.")

return a + b

코드블록

반환값

func_test = func(1, 2)

print(func_test)

이미지 출처

<https://terms.naver.com/entry.nhn?docId=2039077&cid=47308&categoryId=47308>

함수

인자 x / 반환값 x

```
>>> def func():  
  
>>> print("Hello, Function")
```

인자 o / 반환값 x

```
>>> def func(name):  
  
>>> print("Hello, " + name)
```

인자 o / 반환값 o

```
>>> def func(a, b):  
  
>>> return a + b
```

인자 x / 반환값 o

```
>>> def func():  
  
>>> return 3, 5
```


변수의 유효범위

전역변수

지역변수

Scope

변수의 유효범위

```
1 # 전역변수 a, b
2 a = 1
3 b = 2
4 c = 5
```

```
1 def func1():
2     a = 3
3     b = 4
4     print(a, b, c)
5     return
6
7
```

```
1 func1()
```

전역 영역

a = 1
b = 2
c = 5

func1()
영역

a = 3
b = 4

변수의 유효범위

```
1 def func2():  
2     # 지역변수 a, b  
3     a = 5  
4     b = 6  
5  
6     # func1 호출  
7     func1()  
8  
9     print(a, b)
```

```
1 func2()
```

전역 영역

a = 1
b = 2
c = 5

func2()
영역

a = 5
b = 6

func1() 영역

a = 3
b = 4

변수의 유효범위

```
a = 5 # 전역변수
```

```
def func1():
```

```
    a = 1 # func1에서만 사용
```

```
    print(a)
```

```
func1() # 출력값: ?
```

```
print(a) # 출력값: ?
```

```
def func2():
```

```
    print(a) # 전역변수 사용
```

```
func2() # 출력값: ?
```

```
def func3():
```

```
    global a # 전역변수 사용
```

```
    a = 1 # 전역변수 변경
```

```
    print(a)
```

```
func3() # 출력값: ?
```

```
print(a) # 출력값: ?
```

변수의 유효범위

a = 5 # 전역변수

def func1():

 a = 1 # func1에서만 사용

 print(a)

func1() # 1 출력

print(a) # 5 출력

def func2():

 print(a) # 전역변수 사용

func2() # 5 출력

def func3():

 global a # 전역변수 사용

 a = 1 # 전역변수 변경

 print(a)

func3() # 1 출력

print(a) # 1 출력

연습문제1

사칙연산 함수만들기

ex)

`calc(5, 6)`

덧셈: 11, 뺄셈: -1, 곱셈: 30, 나눗셈: 0.83333333333333333334

`a= calc(5,6)`

a가 tuple형(11, -1, 30 , 0.8333)

연습문제 2

해당값을 모두 찾아 위치를 리턴해주는 함수만들기

ex)

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

```
>>> allindex(lis, 1)
```

```
[0, 3, 6]
```

```
def allindex(a, b):
```

lambda 함수

이름 없는 함수
(lambda <인자> : <코드>) (전달인자)

```
>>> (lambda x : x**2) (3)
```

9

```
>>> temp_func = lambda x,y : x*y
```

```
>>> temp_func(3,9)
```

27

함수응용 #1

인자값에 리스트 사용 (언패킹)

```
>>> def func(a, b, c):  
  
>>> print(a, b, c)  
  
>>> x = [1, 2, 3]  
  
>>> func(*x)
```

¹²³가변인수

```
>>> def func(*args):  
  
>>> for arg in args:  
  
>>>     print(arg)  
  
>>> func(1)  
  
>>> func(1, 2, 3, 4, 5)
```

가변인수와 고정인수 같이사용

```
>>> def func(a, *args):  
  
>>> print(a, end="")  
  
>>> for arg in args:  
  
>>>     print(arg, end="")  
  
>>> func(100, 1, 2, 3, 4, 5)  
  
100 1 2 3 4 5
```

함수 응용 #2

키워드 인수 & 딕셔너리 언패킹

```
>>> def func(email, name):  
  
>>>     print("이메일 : ", email)  
  
>>>     print("이름 : ", name)  
  
  
  
>>> func(email = "aa@aa.com", name = "tom")  
  
>>> x = {"email": "aa@aa.com", name: "tom"}  
  
>>> func(**x)
```

가변 키워드인수

```
>>> def func(**kwargs):  
  
>>>     print("이메일 : ", kwargs["email"])  
  
>>>     print("이름 : ", kwargs["name"])  
  
  
  
>>> func(email = "aa@aa.com", name="tom")
```

함수응용 #3

매개변수 초기값

```
>>> def func(email, name, age=20):
```

```
>>>     print("이메일 : ", email)
```

```
>>>     print("이름 : ", name)
```

```
>>>     print("나이 : ", age)
```

```
>>> func(email = "aa.aa.com", name = "tom")
```

```
>>> func(email = "aa.aa.com", name = "tom", age=18)
```

연습문제3

가변인수와 고정인수를 사용해 모든값을 더하거나 빼거나 곱하는 함수
작성

ex)

```
>>> calc("+", [1, 2, 3, 4, 5]) # 1 + 2 + 3 + 4 + 5
```

15

```
>>> calc("-", [1, 2, 3, 4, 5]) # 1 - 2 - 3 - 4 - 5
```

-13

```
>>> calc("*", [1, 2, 3, 4, 5]) # 1 * 2 * 3 * 4 * 5
```

120

연습문제4

가변인수를 이용 가장 높은값, 낮은값, 평균값을 구하는 함수 작성

ex)

>>> calc(1,2,3,4,5)

최대값 : 5, 최소값 : 1, 평균값 : 3.0

1. 객체(Object)

◆ 파이썬에서 모든 데이터는 객체이다.

```
>>> a = 10; b = 3.5; c = 2+10j; d = True
>>> type(a); type(b); type(c); type(d)
<class 'int'>
<class 'float'>
<class 'complex'>
<class 'bool'>
```

```
>>> L = [1,2,3]; type(L)
<class 'list'>
>>> T = (1,2,3); type(T)
<class 'tuple'>
>>> S = 'hello'; type(S)
<class 'str'>
>>> A = {1,2,3}; type(A)
<class 'set'>
>>> D = {1:20, 2:25, 3:22}; type(D)
<class 'dict'>
```

2. 클래스

◆ 클래스 (class)

- 객체(object)를 만들기 위한 도구
- 클래스를 이용하면 현실 세계의 모든 물체들을 객체로 만들 수 있다.
- 클래스의 구성

속성	객체를 구성하는 데이터
메소드	속성에 대해 어떤 기능을 수행하는 함수
생성자, 소멸자	객체 생성과 소멸 시에 자동 호출되는 특별한 메소드
연산자 중복	연산자(+, - 등) 기호를 이용하여 표현할 수 있도록 함

생성자는 `def __init__(self,...):` 으로 정의함

소멸자는 `def __del__(self,...):` 으로 정의함

2 클래스

◆ 강아지를 클래스를 이용하여 객체로 표현하기

class Dog :

""" 강아지를 이름과 나이로 표현하는 클래스 """

""" 속성은 강아지 객체를 구성하는 데이터임. """

def **__init__**(**self**, name, age):

""" 강아지 객체를 생성하는 생성자 메소드 """

self.name = name

self.age = age

속성

생성자

def bark(**self**):

print(self.name, 'is barking')

메소드 (method)

2. 클래스

```
class Dog :
```

```
    def __init__(self, name, age):  
        self.name = name  
        self.age = age
```

```
    def bark(self):  
        print(self.name, 'is barking')
```

```
x = Dog('Jack', 3)  
y = Dog('Daisy', 2)  
x.bark()  
y.bark()  
print(x.name, 'is', x.age, 'years old.')  
print(y.name, 'is', y.age, 'years old.')
```

x

name : Jack
age : 3

y

name : Daisy
age : 2

Jack is barking
Daisy is barking
Jack is 3 years old.
Daisy is 2 years old.

2. 클래스

◆ 원(circle)을 클래스를 이용하여 객체로 표현하기

```
class circle :
```

```
    def __init__(self, radius):  
        self.radius = radius
```

```
    def area(self):  
        a = 3.14 * pow(self.radius, 2)  
        return a
```

```
    def perimeter(self):  
        p = 2 * 3.14 * self.radius  
        return p
```

```
c1 = circle(5)  
c2 = circle(10)
```

c1 radius : 5

c2 radius : 10

```
print('c1 area :', c1.area())  
print('c1 perimeter : %.2f' % c1.perimeter())
```

```
print('c2 area :', c2.area())  
print('c2 perimeter : %.2f' % c2.perimeter())
```

```
c1 area : 78.5  
c1 perimeter : 31.40  
c2 area : 314.0  
c2 perimeter : 62.80
```

2. 클래스

◆ list 클래스

```
>>> dir(list)
['__add__', '__class__', '__contains__', '__delattr__',
 '__delitem__', '__dir__', '__doc__', '__eq__',
 '__format__', '__ge__', '__getattribute__',
 '__getitem__', '__gt__', '__hash__', '__iadd__',
 '__imul__', '__init__', '__iter__', '__le__', '__len__',
 '__lt__', '__mul__', '__ne__', '__new__', '__reduce__',
 '__reduce_ex__', '__repr__', '__reversed__',
 '__rmul__', '__setattr__', '__setitem__', '__sizeof__',
 '__str__', '__subclasshook__', 'append', 'clear',
 'copy', 'count', 'extend', 'index', 'insert', 'pop',
 'remove', 'reverse', 'sort']
```

```
>>> L1 = [1,3,5]
>>> L2 = list([2,4,6])

>>> L1.append(7)
>>> print(L1)
[1, 3, 5, 7]
>>> L2.pop()
6
>>> print(L2)
[2, 4]
```

3. 연산자 중복

◆ int 클래스

```
>>> dir(int)
['__abs__', '__add__', '__and__', '__bool__', '__ceil__',
 '__class__', '__delattr__', '__dir__', '__divmod__',
 '__doc__', '__eq__', '__float__', '__floor__',
 '__floordiv__', '__format__', '__ge__',
 '__getattr__', '__getnewargs__', '__gt__',
 '__hash__', '__index__', '__init__', '__int__',
 '__invert__', '__le__', '__lshift__', '__lt__', '__mod__',
 '__mul__', '__ne__', '__neg__', '__new__', '__or__',
 '__pos__', '__pow__', '__radd__', '__rand__',
 '__rdivmod__', '__reduce__', '__reduce_ex__',
 '__repr__', '__rfloordiv__', '__rlshift__', '__rmod__',
 '__rmul__', '__ror__', '__round__', '__rpow__',
 '__rrshift__', '__rshift__', '__rsub__', '__rtruediv__',
 '__rxor__', '__setattr__', '__sizeof__', '__str__',
 '__sub__', '__subclasshook__', '__truediv__',
 '__trunc__', '__xor__', 'bit_length', 'conjugate',
 'denominator', 'from_bytes', 'imag', 'numerator',
 'real', 'to_bytes']
```

```
>>> x = 10
>>> y = int(20)
>>> z = x.__add__(y) # z = x + y
>>> print(z)
30
>>> p,q,r = 100,100,200
>>> p.__eq__(q) # p == q
True
>>> p.__gt__(r) # p > r
False
```

3. 연산자 중복

◆ 수치 연산자 메소드

메소드	연산자
<code>__add__(self, other)</code>	<code>+</code>
<code>__sub__(self, other)</code>	<code>-</code>
<code>__mul__(self, other)</code>	<code>*</code>
<code>__truediv__(self, other)</code>	<code>/</code>
<code>__floordiv__(self, other)</code>	<code>//</code>
<code>__mod__(self, other)</code>	<code>%</code>
<code>__pow__(self, other[, modulo])</code>	<code>**</code>
<code>__gt__(self, other)</code>	<code>></code>
<code>__ge__(self, other)</code>	<code>>=</code>
<code>__lt__(self, other)</code>	<code><</code>
<code>__le__(self, other)</code>	<code><=</code>
<code>__eq__(self, other)</code>	<code>==</code>
<code>__ne__(self, other)</code>	<code>!=</code>

3. 연산자 중복

◆ 연산자 중복 예제 : 좌표 클래스 만들기

```
class Point:
    def __init__(self, x=0, y=0):
        self.x = x
        self.y = y

    def get(self):
        return "(" + str(self.x) + "," + str(self.y) + ")"

    def __add__(self, other):
        newX = self.x + other.x
        newY = self.y + other.y
        return Point(newX, newY)
```

```
p1 = Point(2,3)
p2 = Point(4,7)
p3 = p1 + p2
print(p1.get())
print(p2.get())
print(p3.get())
```

```
(2,3)
(4,7)
(6,10)
```

4. 클래스와 모듈

- ◆ 클래스가 저장된 파일을 모듈로 사용할 수 있다.

```
""" 나의 애완동물 클래스 """
```

```
class Dog:
```

```
    """ 강아지 클래스 """
```

```
    def __init__(self, name, age):  
        self.name = name  
        self.age = age
```

```
    def bark(self):  
        print(self.name, 'is barking')
```

```
class Cat:
```

```
    """ 고양이 클래스 """
```

```
    def __init__(self, name, color):  
        self.name = name  
        self.color = color
```

```
    def show_color(self):  
        print(self.name, 'is', self.color)
```

```
from pet import Dog, Cat
```

```
a = Dog('Jack', 3)  
b = Dog('Daisy', 2)
```

```
c = Cat('Kitty', 'white')  
d = Cat('Molly', 'black')
```

```
a.bark()  
b.bark()
```

```
c.show_color()  
d.show_color()
```

← pet.py

```
Jack is barking  
Daisy is barking  
Kitty is white  
Molly is black
```

2

6. 예외처리

예외처리

- 프로그램 사용할 때 일어나는 오류들
 - 주소를 입력하지 않고 배송 요청
 - 저장도 안 했는데 컴퓨터가 날아감
 - 게임 아이템 샀는데 게임에서 튕김
 - > 예상치 못한 많은 일(예외) 들이 생김
-

Exception

- 1) 예상 가능한 예외
 - 2) 예상이 불가능한 예외
-

예상 가능한 예외

- 발생 여부를 **사전에 인지**할 수 있는 예외
 - 사용자의 잘못된 입력, 파일 호출시 파일 없음
 - 개발자가 **반드시 명시적으로 정의**해야함.
-

예상 불가능한 예외

- 인터프리터 과정에서 발생하는 예외, 개발자 실수
 - 리스트 범위를 넘어가는 값 호출, 정수 0으로 나눔
 - 수행 불가시 인터프리터가 자동 호출
-

예외 처리 (Exception Handling)

- 예외가 발생할 경우 후속조치 등 대처 필요

- 1) 없는 파일 호출 -> 파일 없음을 알림

- 2) 게임 이상 종료 -> 게임 정보 저장

프로그램 = 제품, 모든 사항에 대처가 필요

Exception Handling

예외처리

```
For i in range(10):  
    try:  
        print( i, 10/i)  
    except ZeroDivisionError:  
        print("Not divided by 0")
```

Exception의 종류

- Built-in Exception : 기본적으로 제공하는 예외

Exception 이름	내용
IndexError	List의 Index 범위를 넘어갈때
NameError	존재하지 않은 변수를 호출할때
ZeroDivisionError	0으로 숫자를 나눌때
ValueError	변환할 수 없는 문자 / 숫자를 변환
FileNotFoundError	존재하지 않는 파일을 호출할때

특정예외만 처리

```
x = [1, 2, 3]
```

```
try:
```

```
    print(10 / 0)
```

```
    y[5]
```

```
except ZeroDivisionError as e:
```

```
    print("숫자를 0으로 나눌수 없음", e)
```

```
except IndexError as e:
```

```
    print("잘못된 인덱스", e)
```

예외처리 else와 finally

try:

print(10 / 0)

except:

print("예외오류발생")

else:

print("오류발생하지 않음")

finally:

print("무조건 실행")

else : 오류가 발생하지 않을때만 동작

finally : 오류발생여부 상관없이 무조건 동작