

DAY 1

목차

- 파이썬 소개
- 파이썬의 기본
- 파이썬의 자료형
- 조건문
- 반복문
- 파일쓰기
- 파일읽기

파이썬?

- 파이썬 소개
- 다른 언어와 비교
- 개발환경 구축하기

파이썬?



1991년 귀도 반 로섬(Guido van Rossum)이 발표

파이썬?

- 왜 파이썬이어야 할까?
 1. 모든 걸 할 수 있는(General-purpose) 프로그래밍 언어
 2. 간결하고 쉬운 문법
 3. 구글, 인스타그램, 드롭박스 등 현업에서 사용 중

파이썬?

1. General-purpose

- 시스템 유틸리티
- GUI 프로그래밍
- C/C++ 과의 결합
- 웹 프로그래밍
- 데이터 베이스
- 데이터 분석
- 사물인터넷

파이썬?

2. 간결하고 쉬운 문법

(1) 간결함

C	JAVA	Python
int array[2]; array[0] = 1; array[1] = 2;	int[] array = new int[2]; array[0] = 1; array = 2;	array = [1,2]

(2) 들여쓰기(Indentation)

C	JAVA	Python
for (int a=0; a<3; a++) { print(a) }	for (int a=0; a<3; a++){ System.out.println(a) }	for a in range(3) : print(a)

파이썬?

3. 현업에서 사용 중



Gmail



Dropbox

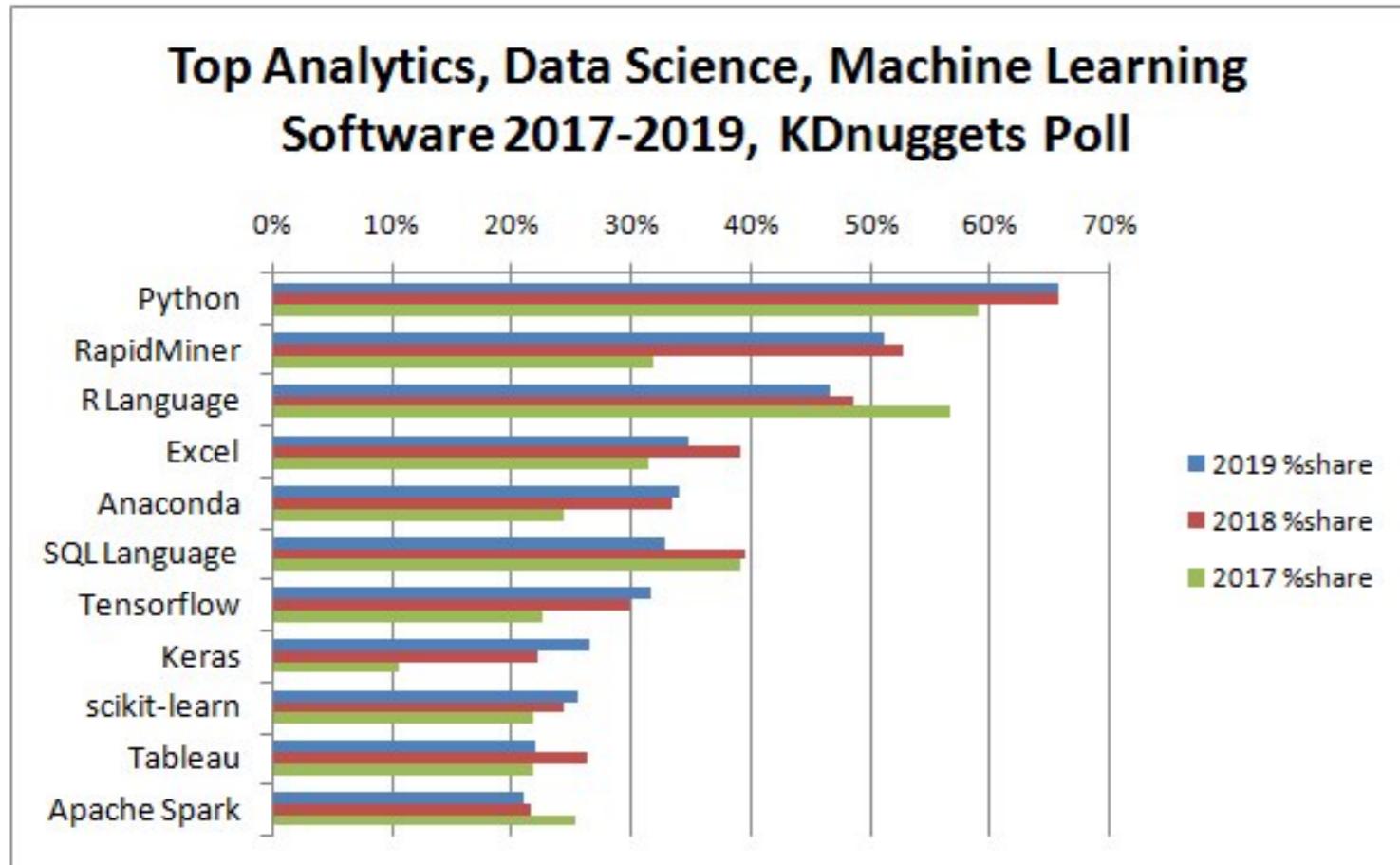


Instagram

- 구글, 드롭박스, 인스타그램 등 현업에서 사용 중
- 미국 Top 10 컴퓨터공학과 중 80%는 컴퓨터언어 입문강좌로 C가 아닌 Python을 채택

파이썬?

3. 현업에서 사용 중



Software	2019 % share	2018 % share	2017 % share
Python	65.8%	65.6%	59.0%
RapidMiner	51.2%	52.7%	31.9%
R Language	46.6%	48.5%	56.6%
Excel	34.8%	39.1%	31.5%
Anaconda	33.9%	33.4%	24.3%
SQL Language	32.8%	39.6%	39.2%
Tensorflow	31.7%	29.9%	22.7%
Keras	26.6%	22.2%	10.7%
scikit-learn	25.5%	24.4%	21.9%
Tableau	22.1%	26.4%	21.8%
Apache Spark	21.0%	21.5%	25.5%

파이썬 개발환경

- 아나콘다



손쉬운 패키지 설치
환경구축이 쉬움
가상 환경을 통한 파이썬 버전관리 가능

ANACONDA

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



Anaconda 2019.10 for Windows Installer

Python 3.7 version

[Download](#)

[64-Bit Graphical Installer \(462 MB\)](#)

[32-Bit Graphical Installer \(410 MB\)](#)

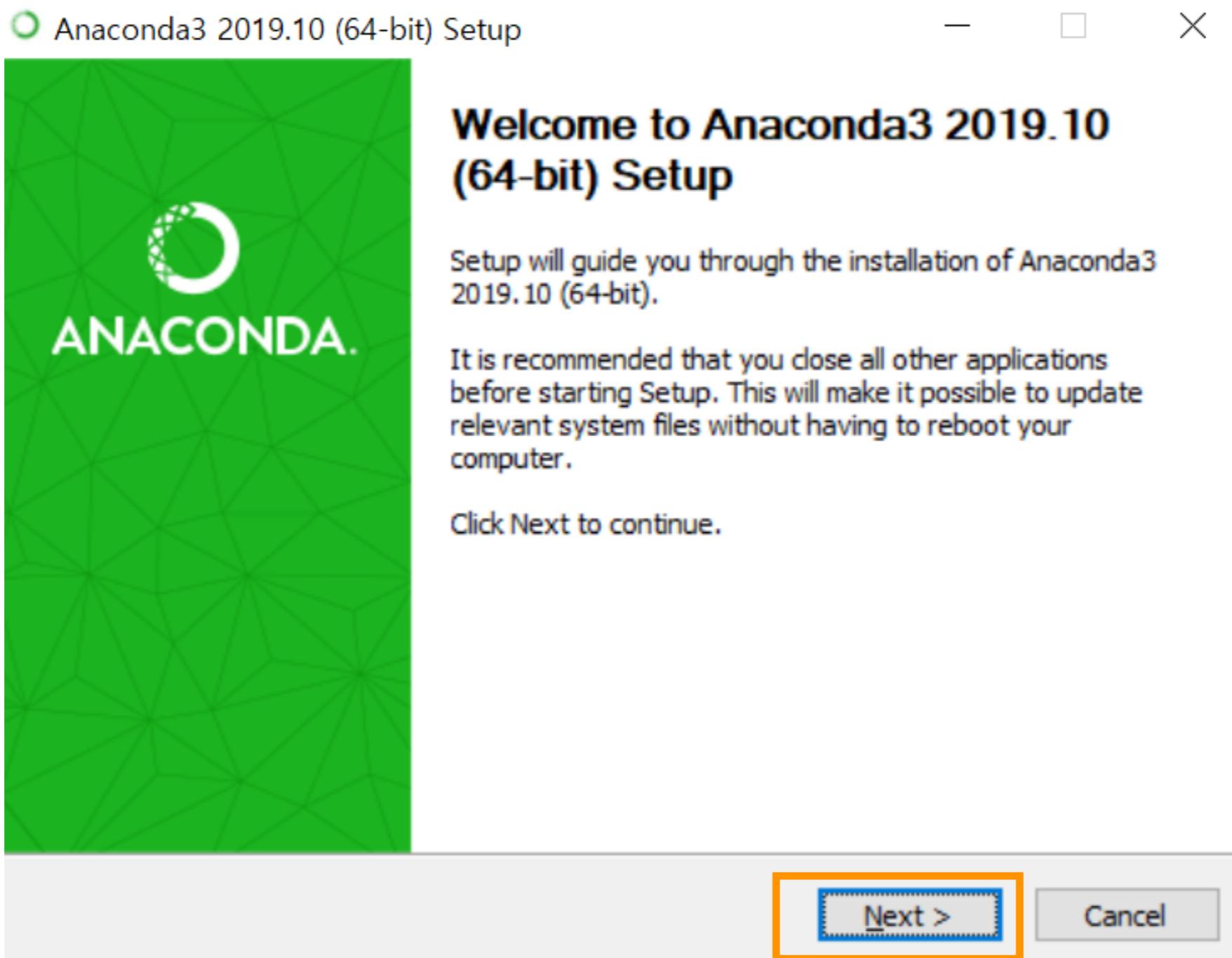
Python 2.7 version

[Download](#)

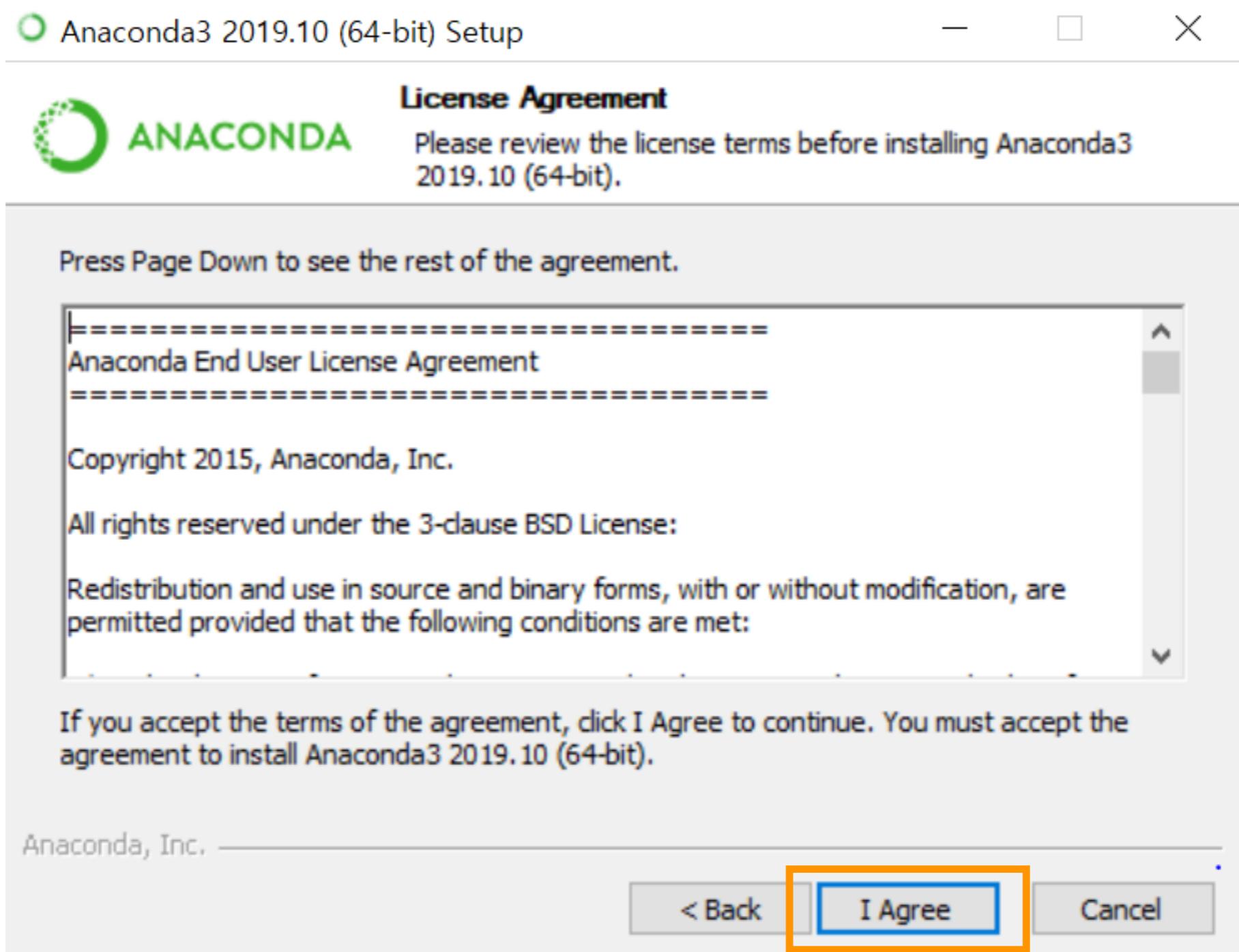
[64-Bit Graphical Installer \(413 MB\)](#)

[32-Bit Graphical Installer \(356 MB\)](#)

ANACONDA



ANACONDA



ANACONDA

○ Anaconda3 2019.10 (64-bit) Setup — □ ×

 **ANACONDA**

Select Installation Type
Please select the type of installation you would like to perform for Anaconda3 2019.10 (64-bit).

Install for:

Just Me (recommended)

All Users (requires admin privileges)

Anaconda, Inc.

< Back Next > Cancel

ANACONDA

○ Anaconda3 2019.10 (64-bit) Setup — X

 **ANACONDA** **Choose Install Location**
Choose the folder in which to install Anaconda3 2019.10 (64-bit).

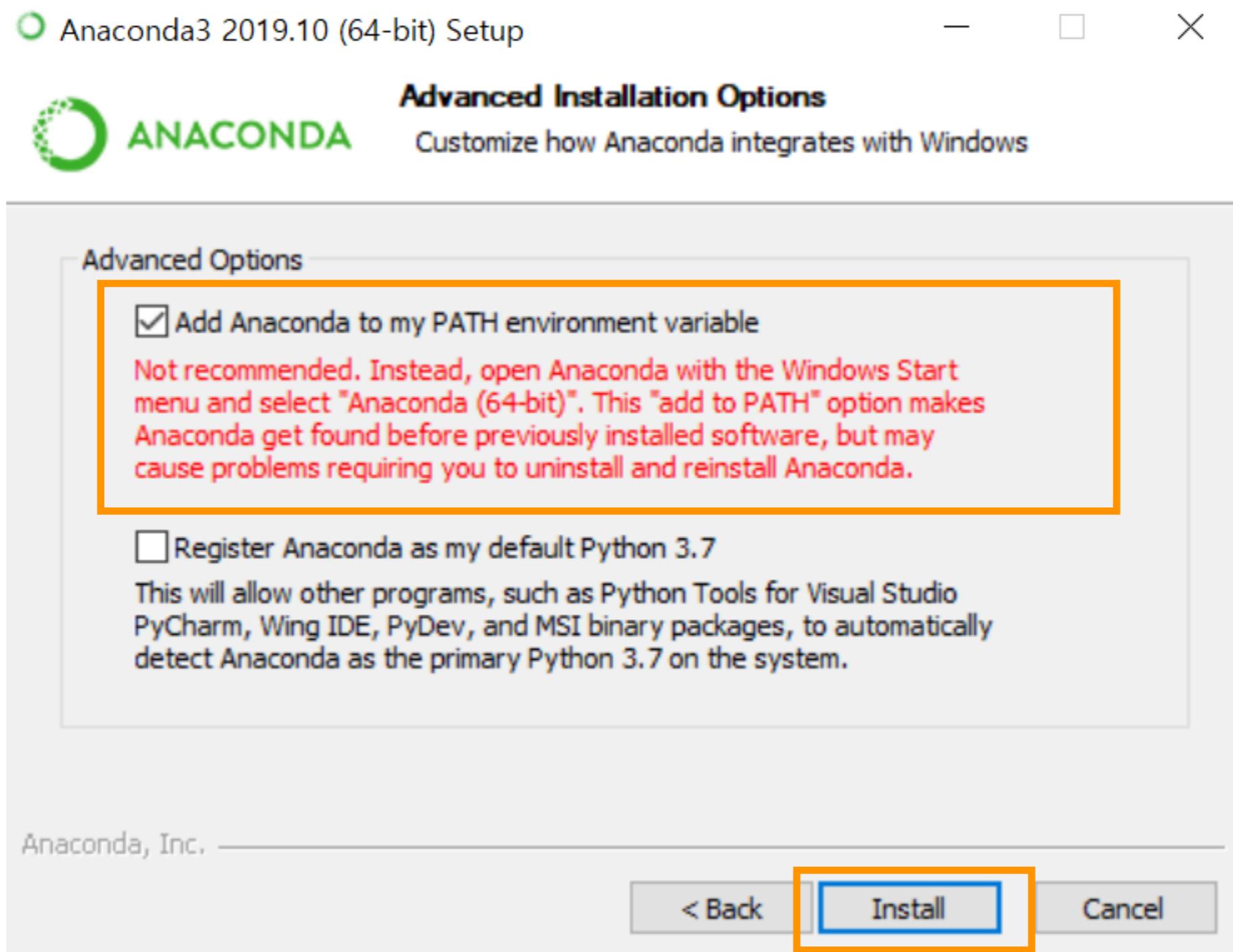
Setup will install Anaconda3 2019.10 (64-bit) in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.

Destination Folder

Space required: 2.9GB
Space available: 219.6GB

Anaconda, Inc.

ANACONDA



파이썬 개발환경

- Jupyter Notebook



Coding 결과를 실시간으로 확인 가능

자동완성 기능

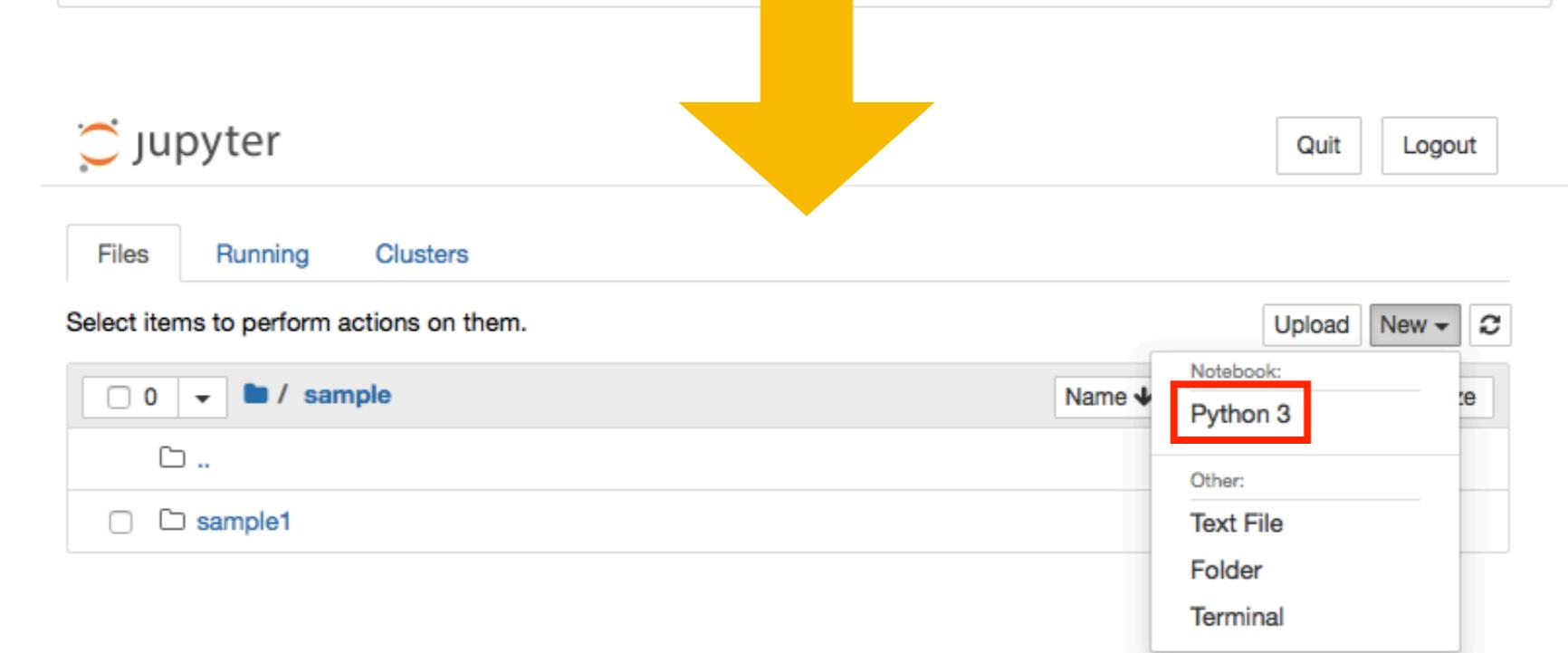
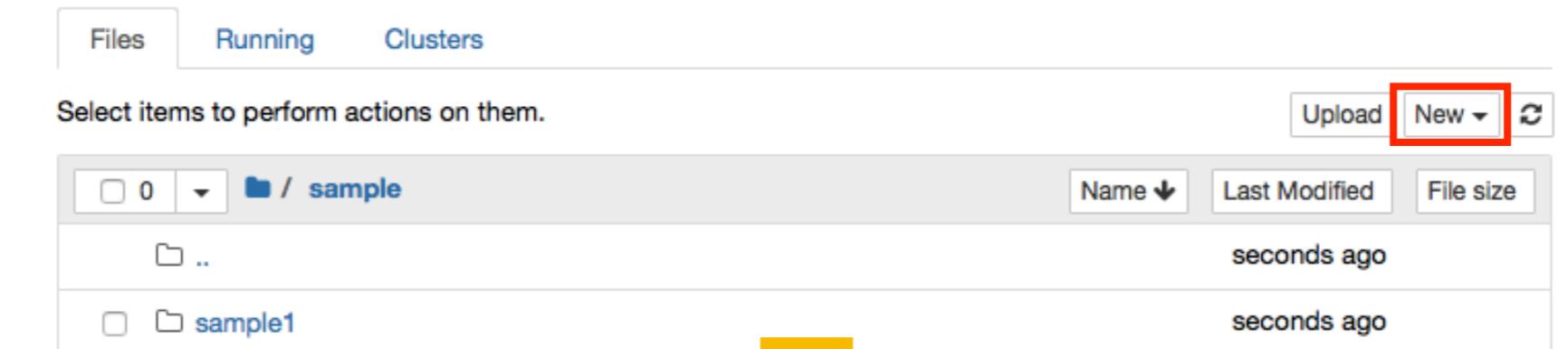
다양한 언어 지원

markdown을 통해 문서화 가능

web을 통한 접근 가능

Jupyter Notebook

1. 새로운 노트북 생성하기



Jupyter Notebook

2. 유용한 단축키

```
In [ ]: # Edit mode : Enter  
        # Command mode : ESC
```

```
In [ ]: # 셀 추가 (Command mode에서)  
        # 현재 셀 위로 : a  
        # 현재 셀 아래로 : b
```

```
In [ ]: # 셀 삭제 (Command mode에서) : dd
```

```
In [ ]: # 선택한 셀을  
        # 복사하기 : c  
        # 잘라내기 : x
```

```
In [ ]: # 셀 붙여넣기  
        # 선택한 셀 위로 : Shift + v  
        # 선택한 셀 아래로 : v
```

```
In [ ]: # 셀에 라인 추가하기 (Command mode에서) : l
```

Jupyter Notebook

2. 유용한 단축키

```
In [ ]: # 셀 변경하기 (Command mode에서)  
# code : y  
# markdown : m
```

```
In [ ]: # 해당 셀을 실행하기 : Ctrl + Enter
```

```
In [ ]: # 해당 셀을 실행하고 아래에 셀 선택 : Shift + Enter
```

```
In [ ]: # 코드 자동 완성 : Tab  
str.|
```

```
In [1]: str|  
capitaliz  
casefold  
center  
count
```

```
In [ ]: |  
encode  
endswith
```

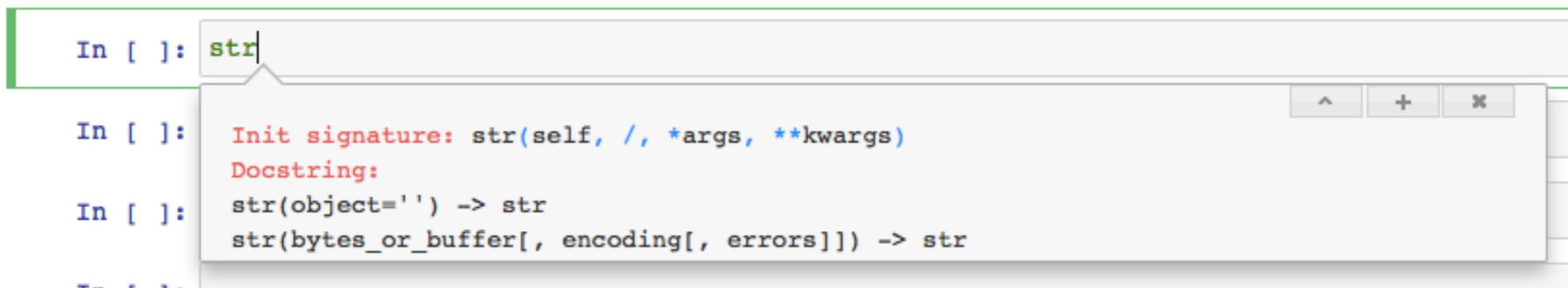
```
In [ ]: |  
expandtabs  
find  
format
```

```
In [ ]: |  
format_map
```

Jupyter Notebook

2. 유용한 단축키

- 함수나 변수에 대한 설명1 : Shift + Tab

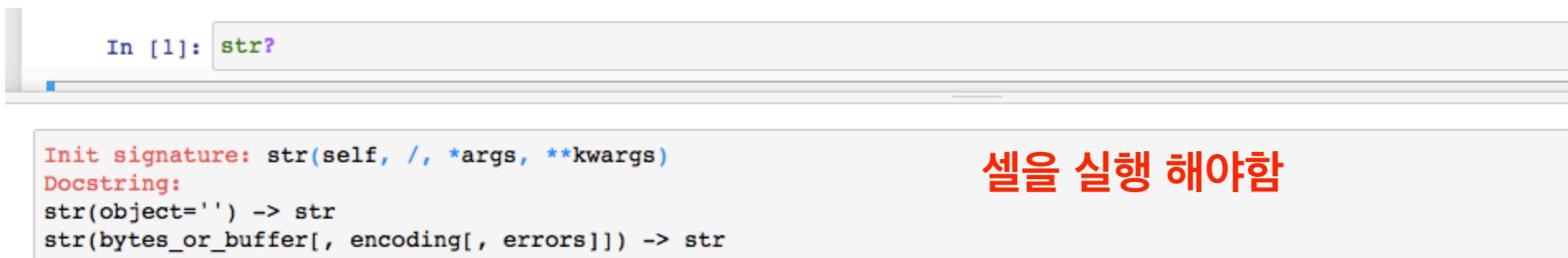


In []: str

In []: Init signature: str(self, /, *args, **kwargs)
Docstring:
In []: str(object='') -> str
str(bytes_or_buffer[, encoding[, errors]]) -> str

A screenshot of a Jupyter Notebook cell. The input field contains 'str'. A tooltip is displayed over the input field, showing the function signature 'Init signature: str(self, /, *args, **kwargs)', the docstring 'Docstring:', and two examples of the str function: 'str(object='') -> str' and 'str(bytes_or_buffer[, encoding[, errors]]) -> str'. The tooltip has a close button in the top right corner.

- 함수나 변수에 대한 설명2 : ?



In [1]: str?

Init signature: str(self, /, *args, **kwargs)
Docstring:
str(object='') -> str
str(bytes_or_buffer[, encoding[, errors]]) -> str

셀을 실행 해야함

A screenshot of a Jupyter Notebook cell. The input field contains 'str?'. A tooltip is displayed over the input field, showing the function signature 'Init signature: str(self, /, *args, **kwargs)', the docstring 'Docstring:', and two examples of the str function: 'str(object='') -> str' and 'str(bytes_or_buffer[, encoding[, errors]]) -> str'. To the right of the tooltip, the text '셀을 실행 해야함' (Run cell required) is displayed in red.

Jupyter Notebook

2. 유용한 단축키

Jupyter 맛보기

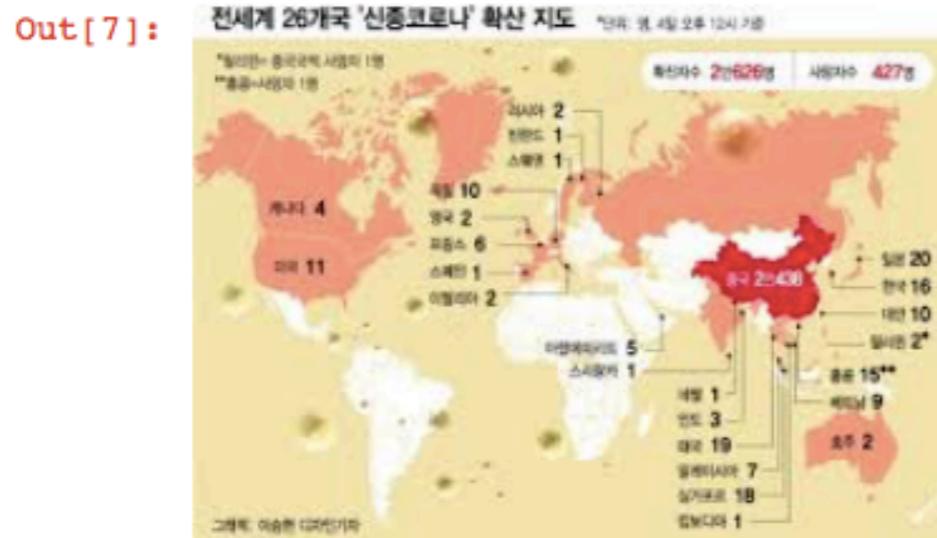
markdown

문서처럼 작성할 수도 있습니다.

```
In [2]: print("당연히 코딩도 할 수 있습니다.")
```

당연히 코딩도 할 수 있습니다.

```
In [7]: from IPython.display import Image  
Image(url="data:image/jpeg;base64,/9j/4AAQSkZJRgABAQAAAQABAAAD/2wCEAAkGBxMTEhUTEhWFhUWGBkYC
```



파이썬 개발환경

- Colab



구글 아이디로 연동

jupyter notebook + 구글 drive

<https://colab.research.google.com/>

파이썬의 기본

- 산술 연산자

1. 더하기 : +

```
In [1]: 1 + 2
```

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

```
In [2]: 250.7 + 300.2
```

```
Out[2]: 550.9
```

2. 빼기 : -

```
In [3]: 5 - 3
```

```
Out[3]: 2
```

```
In [4]: 250.5 - 100.3
```

```
Out[4]: 150.2
```

3. 곱하기 : *

```
In [5]: 5 * 3
```

```
Out[5]: 15
```

```
In [6]: 20.5 * 2
```

```
Out[6]: 41.0
```

4. 나누기 : /

```
In [7]: 10 / 2
```

```
Out[7]: 5.0
```

```
In [8]: 7 / 3
```

```
Out[8]: 2.333333333333335
```

파이썬의 기본

- 산술 연산자

5. //

```
In [9]: 10 // 2
```

```
Out[9]: 5
```

```
In [10]: 7 // 3
```

```
Out[10]: 2
```

/의 연산 결과 : 소수점 표시

→ 실수형

//의 연산 결과 : 소수점 없음(나누기의 몫)

→ 정수형

파이썬의 기본

- 산술 연산자

6. 나머지 : %

```
In [11]: 10 % 2
```

```
Out[11]: 0
```

```
In [12]: 7 % 3
```

```
Out[12]: 1
```

7. 그 밖에 연산자

제곱 : **

```
In [13]: 2 ** 3
```

```
Out[13]: 8
```

```
In [14]: 5 ** 3
```

```
Out[14]: 125
```

비트 연산자 : ^

```
In [15]: 4 ^ 2
```

```
Out[15]: 6
```

파이썬의 기본

- 연산자의 우선순위

사칙연산의 연산자 우선순위와 같음
괄호()가 있다면 가장 우선 처리

```
In [16]: 5 + 3 * 4
```

```
Out[16]: 17
```

```
In [17]: (8 - 3) * 2
```

```
Out[17]: 10
```

정리문제 1-1

1. 10의 제곱을 출력해보자
2. $2 \times 5 + 3$ 과 $2 \times (5 + 3)$ 을 각각 화면에 출력해보자

파이썬의 기본

- 할당 연산자(Assignment) : =

result = 1 + 2

할당 연산자
변수 연산

The diagram illustrates the components of an assignment statement. A red bracket labeled '할당 연산자' (Assignment Operator) points to the '=' symbol. Below the code, the word '변수' (variable) is written under the underlined 'result', and the word '연산' (operator) is written under the underlined '1 + 2'.

할당 연산자 우측을 연산을 한 후, 그 결과를 좌측에 있는 변수에 할당(저장)

파이썬의 기본

- 할당 연산자(Assignment) : =

```
In [18]: result = 1 + 2
```

```
In [19]: print(result)
```

```
3
```

```
In [20]: result
```

```
Out[20]: 3
```

할당 연산자 우측을 연산을 한 후, 그 결과를 좌측에 있는 변수에 할당(저장)

파이썬의 기본

- 할당 연산자(Assignment) : =

```
In [21]: result += 1  
result
```

```
Out[21]: 4
```

```
In [ ]: result = result + 1
```

산술 연산자와 할당 연산자를 함께 쓸 수 있음
코드를 더욱 간결하게 쓸 수 있음

파이썬의 기본

- 할당 연산자(Assignment) : =

더 알아보기

```
In [21]: result += 1  
result
```

```
Out[21]: 4
```

```
In [22]: result -= 5  
result
```

```
Out[22]: -1
```

```
In [23]: result *= -1  
result
```

```
Out[23]: 1
```

```
In [24]: result /= 5  
result
```

```
Out[24]: 0.2
```

파이썬의 기본

- 할당 연산자(Assignment) : =

단순 산술 연산이 아닌 변수간의 연산도 가능

```
In [25]: a = 5  
        b = 7  
        c = a + b  
        print(c)
```

12

주의 파이썬은 대소문자를 구분함

```
In [26]: c = A + B
```

```
-----  
NameError                                 Traceback (most recent call last)  
<ipython-input-26-b698ae2737d5> in <module>  
----> 1 c = A + B  
  
NameError: name 'A' is not defined
```

정리문제 1-2

1. 사과가 5개 오렌지가 3개 있을 때 총 과일의 갯수를 구해보자

사과는 apple이라는 변수, 오렌지는 orange라는 변수에 할당 한 후, 총 과일의 갯수를 total이라는 변수에 저장해보자

파이썬의 자료형

- 숫자(int, float)
- 문자(string)
- 불(bool)
- 리스트(list)
- 튜플(tuple)
- 딕셔너리(dict)
- 셋(set)

숫자

- int와 float

```
In [1]: a = 3  
       b = 3.0
```

```
In [2]: print('a의 타입은 {}'.format(type(a)))  
       print('b의 타입은 {}'.format(type(b)))
```

```
a의 타입은 <class 'int'>  
b의 타입은 <class 'float'>
```

```
In [3]: c = a + b  
       print('c의 타입은 {}'.format(type(c)))
```

```
c의 타입은 <class 'float'>
```

int는 정수, float는 실수

숫자

- int와 float

```
In [4]: d = int(c)
print(d)
print('d의 타입은 {}'.format(type(d)))
```

```
6
d의 타입은 <class 'int'>
```

```
In [5]: e = float(5)
print(e)
print('e의 타입은 {}'.format(type(e)))
```

```
5.0
e의 타입은 <class 'float'>
```

숫자는 형 변환을 할 수 있음

문자

- **string**

‘’(작은 따옴표) 혹은 “”(큰 따옴표)을 이용하여 문자형(string)임을 표시

```
In [6]: f = '문자형 예시입니다.'  
        print(f)  
        print('f의 타입은 {}'.format(type(f)))
```

문자형 예시입니다.
f의 타입은 <class 'str'>

```
In [7]: 따옴표 = '문장안에 따옴표를 쓰고 싶다면? "" 번갈아 쓰면 됩니다.'  
        print(따옴표)
```

문장안에 따옴표를 쓰고 싶다면? "" 번갈아 쓰면 됩니다.

```
In [8]: escape = "또 다른 방법은 escape 문자인 \" \\" (역슬래시 혹은 원화표시) 를 이용하는 것입니다."  
        print(escape)
```

또 다른 방법은 escape 문자인 " \" (역슬래시 혹은 원화표시) 를 이용하는 것입니다.

\”

문자

- string

```
In [9]: g = 3
        h = '3'
        print('g의 타입은 {}'.format(type(g)))
        print('h의 타입은 {}'.format(type(h)))
```

```
g의 타입은 <class 'int'>
h의 타입은 <class 'str'>
```

```
In [10]: i = str(g)
        j = int(i)
        print('i의 타입은 {}'.format(type(i)))
        print('j의 타입은 {}'.format(type(j)))
```

```
i의 타입은 <class 'str'>
j의 타입은 <class 'int'>
```

str와 int도 형 변환이 가능

문자

- string의 연산

```
In [11]: str_1 = '문자'  
str_2 = '의 연산'  
result = str_1 + str_2  
result
```

```
Out[11]: '문자의 연산'
```

```
In [12]: str_3 = '더하기'  
result += str_3  
result
```

```
Out[12]: '문자의 연산더하기'
```

```
In [13]: str_4 = '곱하기'  
print(str_4 * 5)
```

```
곱하기곱하기곱하기곱하기곱하기
```

문자도 연산이 가능

문자

- string의 연산

```
In [14]: int_1 = 8
         error = str_1 + int_1
-----
TypeError                                     Traceback (most recent call last)
<ipython-input-14-67de6265ddc4> in <module>
      1 int_1 = 8
----> 2 error = str_1 + int_1
TypeError: can only concatenate str (not "int") to str
```

단, 자료형이 다르다면 연산이 불가

문자

- string의 indexing(인덱싱)

인덱싱? 변수의 해당 위치에 접근 할 수 있게 해줌

```
In [15]: str_5 = '인덱싱을 하기 위한 string입니다.'  
len(str_5)
```

```
Out[15]: 21
```

```
In [16]: str_5[2]
```

```
Out[16]: '싱'
```

대괄호 []와 함께 사용할 시 인덱싱을 할 수 있음.

주의 파이썬의 인덱싱은 0부터 시작

문자

- string의 slicing(슬라이싱)

슬라이싱? 변수의 일정 부분에 접근하여 잘라옴

```
In [17]: str_5 = '인덱싱을 하기 위한 string입니다.'  
str_5[2:7]  
  
Out[17]: '싱을 하기'
```

Str_5 [2:7] ⇒ 2부터 6까지
I ↘
시작 끝

문자

- string의 slicing(슬라이싱)

```
In [18]: str_5 = '인덱싱을 하기 위한 string입니다.'  
print(str_5[:3])
```

인덱싱

→ 시작점이 없다면 처음부터

```
In [19]: print(str_5[18:])  
니다.
```

→ 끝이 없다면 끝까지

```
In [20]: print(str_5[:])  
인덱싱을 하기 위한 string입니다.
```

→ 시작과 끝이 없다면 전체

```
In [21]: print(str_5[-1])  
.
```

→ -1은 끝에서부터 거꾸로

```
In [22]: print(str_5[::-1])  
.다니입gnirts 한위 기하 을싱덱인
```

→[::-1]은 str전체를 뒤집음

문자

- string의 method

method? 파이썬의 객체가 가지고 있는 고유의 함수
.을 이용하여 사용

1. 대소문자 변환 : lower, upper

```
In [23]: str_6 = 'Alphabet'  
        print(str_6.lower())
```

```
alphabet
```

```
In [24]: print(str_6.upper())
```

```
ALPHABET
```

2. 해당 문자의 갯수 세기 : count

```
In [25]: str_6.count('a')
```

```
Out[25]: 1
```

→python은 대소문자를 구분함

문자

- string의 method

3. 문자열의 위치 찾기

1) find

```
In [26]: str_6 = 'Alphabet'  
str_6.find('t')
```

```
Out[26]: 7
```

```
In [27]: str_6.find('z')
```

```
Out[27]: -1
```

→문자열에 없으면 -1 반환

2) index

```
In [28]: str_6.index('t')
```

```
Out[28]: 7
```

```
In [29]: str_6.index('z')
```

```
-----  
ValueError  
<ipython-input-29-9046828b426d> in <module>  
----> 1 str_6.index('z')
```

→문자열에 없으면 error

```
ValueError: substring not found
```

문자

- string의 method

4. 문자열 삽입 : join

```
In [30]: str_7 = 'abcd'  
      ','.join(str_7)  
  
Out[30]: 'a,b,c,d'
```

→따옴표 ‘ ’사이에 삽입할 문자를
넣기

5. 문자열 바꾸기 : replace

```
In [31]: str_8 = 'Life is C between B and D'  
      str_8.replace('C', 'Chicken')  
  
Out[31]: 'Life is Chicken between B and D'
```

6. 문자열 나누기 : split

```
In [33]: str_8.split(' ')  
  
Out[33]: ['Life', 'is', 'C', 'between', 'B', 'and', 'D']
```

→따옴표 ‘ ’사이에 기준이
되는 문자를 넣기

문자

- string의 method

더 많은 메소드는 [python 공식문서](#)나 아래와 같은 방법으로 확인 가능

```
In [34]: print(dir(str_8))
```

```
['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__ge__attribute__', '__getitem__', '__getnewargs__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__iter__', '__le__', '__len__', '__lt__', '__mod__', '__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__rmod__', '__rmul__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__', 'capitalize', 'casefold', 'center', 'count', 'encode', 'endswith', 'expandtabs', 'find', 'format', 'format_map', 'index', 'isalnum', 'isalpha', 'isascii', 'isdecimal', 'isdigit', 'isidentifier', 'islower', 'isnumeric', 'isprintable', 'isspace', 'istitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip', 'maketrans', 'partition', 'replace', 'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split', 'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'translate', 'upper', 'zfill']
```

리스트

- list

[]을 이용하여 리스트(list)임을 선언

```
In [34]: list_1 = []
```

```
In [35]: list_2 = list()
```

```
In [36]: print('list_1의 타입은 ' + str(type(list_1)))
print('list_2의 타입은 ' + str(type(list_2)))
```

```
list_1의 타입은 <class 'list'>
```

```
list_2의 타입은 <class 'list'>
```

리스트

- **list**

리스트는 순서가 있는 자료형 → 인덱싱과 슬라이싱이 가능
리스트는 모든 자료형을 담을 수 있음

```
In [39]: list_3 = [1, 2, '문자', ['이중 리스트', '가능'], ('리스트 속', '튜플')]
```

```
In [40]: list_3[3]
```

```
Out[40]: ['이중 리스트', '가능']
```

```
In [42]: list_3[4][1]
```

```
Out[42]: '튜플'
```

주의 python의 인덱스는 0부터 시작

리스트

- list의 연산

```
In [40]: list_4 = ['a', 'b', 'c']
```

```
In [41]: list_5 = ['가', '나', '다']
```

```
In [42]: list_4 + list_5
```

```
Out[42]: ['a', 'b', 'c', '가', '나', '다']
```

```
In [43]: list_4 * 3
```

```
Out[43]: ['a', 'b', 'c', 'a', 'b', 'c', 'a', 'b', 'c']
```

리스트

- list의 method

리스트는 삽입, 삭제, 수정이 자유로움

1. 삽입 : append, insert

```
In [44]: list_6 = ['파이썬', 'C', 'java']
```

```
In [45]: list_6.append('R')  
list_6
```

```
Out[45]: ['파이썬', 'C', 'java', 'R']
```

→ append는 리스트 제일 뒤에

```
In [46]: list_6.insert(2, 'C++)  
list_6
```

→ insert는 원하는 위치에

```
Out[46]: ['파이썬', 'C', 'C++', 'java', 'R']
```

2. 수정 : replace

```
In [47]: list_6[0] = 'Python'  
list_6
```

→ 인덱스나 슬라이싱을 통해 수정

```
Out[47]: ['Python', 'C', 'C++', 'java', 'R']
```

리스트

- list의 method

3. 삭제 : remove

```
In [49]: list_6  
Out[49]: ['Python', 'C', 'C++', 'java', 'R', 'C']
```

```
In [50]: list_6.remove('C')  
list_6  
Out[50]: ['Python', 'C++', 'java', 'R', 'C']
```

→ remove는 지우고자 하는 요소를 명시해야함
→ 가장 앞에 있는 요소가 삭제됨

4. 꺼내오기 : pop

```
In [51]: list_6.pop()  
Out[51]: 'C'  
  
In [52]: list_6  
Out[52]: ['Python', 'C++', 'java', 'R']
```

→ pop은 괄호()안 인덱스에 해당하는 요소를 반환
→ 인덱스를 생략하면 가장 마지막 요소를 꺼내옴

리스트

- list의 method

5. 정렬 : sort

```
In [58]: list_7 = [1,5,7,2,6]
```

```
In [59]: list_7.sort()  
list_7
```

```
Out[59]: [1, 2, 5, 6, 7]
```

→ 리스트 자체를 정렬

6. 뒤집기 : reverse

```
In [60]: list_7.reverse()  
list_7
```

```
Out[60]: [7, 6, 5, 2, 1]
```

→ 리스트 자체를 뒤집음

정리문제 1-3

1. 좋아하는 음식 5개를 food라는 리스트에 담아보자
2. 리스트 food의 가장 앞에 파이썬이, 가장 뒤에는 집이라는 단어가 오도록 수정해보자
3. pop을 이용하여 파이썬을 삭제해보자
4. remove를 이용하여 집을 삭제해보자
5. pop과 remove의 차이점을 몸소 느껴보자
6. len() 함수를 이용하여 리스트의 길이를 확인해보자
7. len() 함수를 이용하여 리스트 첫번째 요소의 길이를 확인해보자

튜플

- tuple

()을 이용하여 튜플(tuple)임을 선언

```
In [57]: tuple_1 = ()  
tuple_2 = tuple()
```

```
In [61]: tuple_3 = (1, 2)  
tuple_4 = (3,)  
tuple_5 = (4, 5, (6, 7))  
tuple_6 = 8, 9, 10
```

한 개의 요소만 사용 할 때에는 콤마(,)를 반드시 사용해야함
괄호가 없어도 튜플로 선언할 수 있음

튜플

- tuple

인덱싱과 슬라이싱이 가능

리스트와 달리 요소를 삽입/삭제/수정 할 수 없음

```
In [62]: tuple_3 = (1, 2)
tuple_3[0] = 3
```

```
-----
TypeError                                     Traceback (most recent call last)
<ipython-input-62-c8372825bae8> in <module>
      1 tuple_3 = (1, 2)
----> 2 tuple_3[0] = 3

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

튜플

- tuple의 연산

```
In [64]: tuple_3 = (1, 2)
          tuple_4 = (3, )
          tuple_3 + tuple_4
```

```
Out[64]: (1, 2, 3)
```

```
In [65]: tuple_4 * 2
```

```
Out[65]: (3, 3)
```

딕셔너리

- **dictionary**

{ }을 이용하여 딕셔너리(dictionary)임을 선언

```
In [67]: dict_1 = {'key' : 'value'}  
dict_2 = dict()
```

- 딕셔너리는 key와 value 값으로 이루어진 자료형
- 순서가 있는 자료형이 아니며, key를 통해 value값에 접근이 가능
- key는 고유한 값으로 중복될 수 없음
- value는 중복 가능

딕셔너리

- dictionary 추가/삭제/수정

```
In [85]: dict_3 = {'홍길동': 100, '홍계월': 200}
```

```
In [86]: dict_3['리정혁'] = 300  
dict_3
```

```
Out[86]: {'홍길동': 100, '홍계월': 200, '리정혁': 300}
```

```
In [87]: del dict_3['리정혁']  
dict_3
```

```
Out[87]: {'홍길동': 100, '홍계월': 200}
```

```
In [88]: dict_3['홍길동'] = 1  
dict_3
```

```
Out[88]: {'홍길동': 1, '홍계월': 200}
```

주의 인덱스 대신 key 사용

딕셔너리

- key와 value에 접근하기

1. key에 접근하기

```
In [73]: dict_3 = {'홍길동' : 100, '홍계월' : 200, '리정혁' : 150, '윤세리' : 250}
          print(dict_3.keys())
          print(type(dict_3.keys()))
```

```
dict_keys(['홍길동', '홍계월', '리정혁', '윤세리'])
<class 'dict_keys'>
```

2. value에 접근하기

```
In [74]: print(dict_3.values())
          print(type(dict_3.values()))
```

```
dict_values([100, 200, 150, 250])
<class 'dict_values'>
```

딕셔너리

- key와 value에 접근하기

3. key와 value에 함께 접근하기

```
In [75]: print(dict_3.items())
print(type(dict_3.items()))
```

```
dict_items([('홍길동', 100), ('홍계월', 200), ('리정혁', 150), ('윤세리', 250)])
<class 'dict_items'>
```

주의

리스트처럼 보이나 실제로는 dict의 개체이므로
리스트의 메소드를 사용할 수 없음

리스트처럼 사용하고 싶다면 리스트로 형 변환을 해야함

```
In [76]: list(dict_3.keys())[0]
```

```
Out[76]: '홍길동'
```

딕셔너리

- key와 value에 접근하기

4. key로 value에 접근하기

```
In [96]: dict_3
```

```
Out[96]: {'홍길동': 100, '홍계월': 200, '리정혁': 150, '윤세리': 250}
```

```
In [97]: print(dict_3['윤세리'])  
print(dict_3.get('윤세리'))
```

```
250
```

```
250
```

- 1) 인덱싱처럼 키값을 이용하여 바로 접근
- 2) 메소드 get을 사용

딕셔너리

- key와 value에 접근하기

4. key로 value에 접근하기

```
In [96]: dict_3
```

```
Out[96]: {'홍길동': 100, '홍계월': 200, '리정혁': 150, '윤세리': 250}
```

```
In [98]: print(dict_3['펭수'])
```

```
-----  
KeyError                                                 Traceback (most recent call last)  
<ipython-input-98-43368d90b0e9> in <module>  
----> 1 print(dict_3['펭수'])  
KeyError: '펭수'
```

→ 키값을 이용하여 접근 할 경우, 해당 키가 없으면 오류 발생

```
In [99]: print(dict_3.get('펭수'))
```

None

→ get을 통해 접근할 경우, None을 반환

셋(집합)

- set

{ }을 이용하여 set을 선언

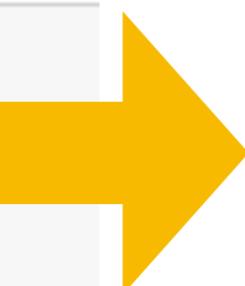
```
In [103]: set_1 = set()  
set_2 = set('Hello')  
set_3 = set([1, 2, 3])  
set_4 = {3,5, 'hi'}
```

셋(집합)

- set의 특징

중복을 허용하지 않음
순서가 없음

```
In [103]: set_1 = set()  
set_2 = set('Hello')  
set_3 = set([1, 2, 3])  
set_4 = {3, 5, 'hi'}
```



```
In [104]: set_2  
Out[104]: {'H', 'e', 'l', 'o'}
```

```
In [105]: set_3  
Out[105]: {1, 2, 3}
```

```
In [106]: set_4  
Out[106]: {3, 5, 'hi'}
```

셋(집합)

- set의 연산

1. 교집합

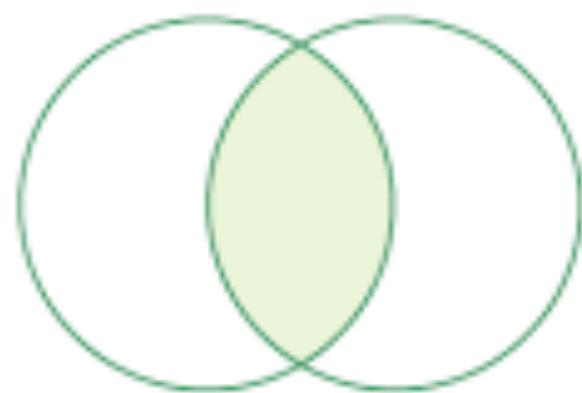
```
In [107]: set_5 = {1,2,3,4}  
set_6 = {4,5,6,7}
```

```
In [109]: set_5 & set_6
```

```
Out[109]: {4}
```

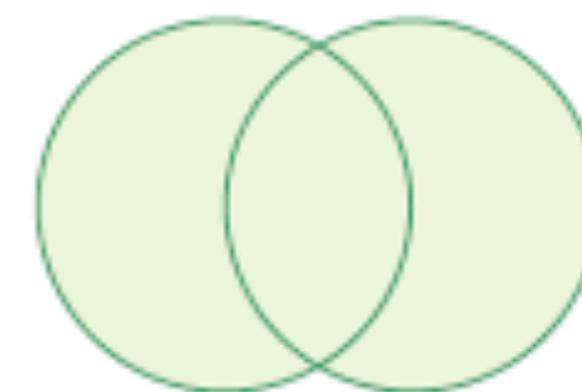
```
In [110]: set_5.intersection(set_6)
```

```
Out[110]: {4}
```



집합 1 집합 2

(a) 교집합



집합 1 집합 2

(b) 합집합

2. 합집합

```
In [113]: set_5 | set_6
```

```
Out[113]: {1, 2, 3, 4, 5, 6, 7}
```

```
In [114]: set_5.union(set_6)
```

```
Out[114]: {1, 2, 3, 4, 5, 6, 7}
```

셋(집합)

- set의 연산

3. 차집합

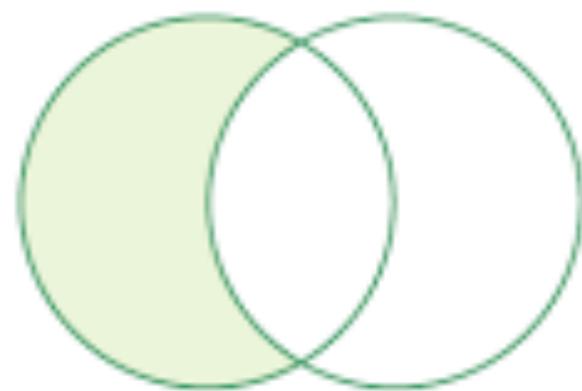
```
In [107]: set_5 = {1,2,3,4}  
set_6 = {4,5,6,7}
```

```
In [111]: set_5 - set_6
```

```
Out[111]: {1, 2, 3}
```

```
In [112]: set_5.difference(set_6)
```

```
Out[112]: {1, 2, 3}
```



집합 1 집합 2
(c) 차집합

셋(집합)

- set의 메소드

1. 한 개 추가하기 : add

```
In [116]: set_6 = {4, 5, 6, 7}  
set_6.add(8)  
set_6
```

```
Out[116]: {4, 5, 6, 7, 8}
```

2. 여러 개 추가하기 : update

```
In [118]: set_6.update([1,2,3])  
set_6
```

```
Out[118]: {1, 2, 3, 4, 5, 6, 7, 8}
```

3. 한 개 지우기 : remove

```
In [121]: set_6.remove(3)  
set_6
```

```
Out[121]: {1, 2, 4, 5, 6, 7, 8}
```

불

- bool : 참과 거짓

참 : True

거짓 : False

```
In [40]: True
```

```
Out[40]: True
```

```
In [41]: type(False)
```

```
Out[41]: bool
```

```
In [42]: 1 == 1
```

```
Out[42]: True
```

```
In [43]: 1 > 2
```

```
Out[43]: False
```

비교 연산자	의미
==	같음
!=	같지 않음
>, >=, <, <=	크거나 같음/ 작거나 같음

→ 비교
불자료

불

- bool : 참과 거짓

자료형에도 참/거짓이 있음

자료	참 / 거짓
'python'	참
“ ”	거짓
[1,2,3]	참
[]	거짓
()	거짓
{ }	거짓
1	참
0	거짓
None	거짓

정리 문제 1-4

1. 영희가 “겨울은 너무 추워.”라고 말했다. 를 출력하기
2. 리스트 [1, 3, 4, 5, 2]를 [5, 4, 3, 2, 1]로 만들어 출력하기
3. 주어진 리스트 ['Life', 'is', 'C', 'between', 'B', 'and', 'D']를 다음과 같은 문자열로 만들어보자
“Life is C between B and D”

조건문

- if / else

```
In [ ]: if 조건 :  
    수행 할 문장1  
    수행 할 문장2  
else :  
    수행 할 문장3  
    수행 할 문장4
```

주의 콜론(:)과 들여쓰기에
주의할 것

```
In [1]: money = True  
  
if money :  
    print("여행을 간다")  
else :  
    print("집에서 쉰다")
```

여행을 간다

조건문

- if / elif / else

```
In [ ]: if 조건1 :  
        수행 할 문장1  
    elif 조건2 :  
        수행 할 문장2  
    else :  
        수행 할 문장3
```

```
In [2]: money = 20000  
  
if money < 5000 :  
    print('라면을 먹는다')  
elif 5000 < money < 25000 :  
    print('치킨을 먹는다')  
elif 25000 < money < 50000 :  
    print('삼겹살을 먹는다')  
else :  
    print('소고기를 먹는다')
```

치킨을 먹는다

조건문

- 조건문?

조건문 : 참과 거짓을 판단하게 하는 문장

1. 비교연산자

```
In [3]: a = 3  
b = 5  
  
a < b
```

Out[3]: True

```
In [4]: c = 3  
d = 3  
  
c <= d
```

Out[4]: True

```
In [9]: c != d
```

Out[9]: False

조건문

- 조건문?

조건문 : 참과 거짓을 판단하게 하는 문장

2. 논리연산자 : and / or / not

```
In [18]: True and True
```

```
Out[18]: True
```

```
In [24]: True and False
```

```
Out[24]: False
```

1) and

비교하는 대상이 모두 참이여야만 True반환

조건문

- 조건문?

조건문 : 참과 거짓을 판단하게 하는 문장

2. 논리연산자 : and / or / not

```
In [25]: True or True
```

```
Out[25]: True
```

```
In [26]: True or False
```

```
Out[26]: True
```

```
In [27]: False or False
```

```
Out[27]: False
```

2) or

비교하는 대상 중 하나만 참
이여도 True 반환

조건문

- 조건문?

조건문 : 참과 거짓을 판단하게 하는 문장

2. 논리연산자 : and / or / not

```
In [28]: not True
```

```
Out[28]: False
```

```
In [29]: not False
```

```
Out[29]: True
```

3) not

참이면 False를, 거짓이면
True를 반환

조건문

- 조건문?

조건문 : 참과 거짓을 판단하게 하는 문장

3. 요소인지 파악하기 : in / not in

```
In [30]: e = [1, 3, 5, 7]
```

```
In [31]: 0 in e
```

```
Out[31]: False
```

```
In [32]: 1 in e
```

```
Out[32]: True
```

```
In [33]: 2 not in e
```

```
Out[33]: True
```

```
In [34]: 3 not in e
```

```
Out[34]: False
```

1) x in s

x가 s의 요소인가

2) x not in s

x가 s의 요소가 아닌가

조건문

- if문 안에 if문

조건 안에 조건을 만들 수 있음

```
In [40]: money = 20000  
card = True
```

```
In [41]: if card :  
    if money < 30000 :  
        print("삼겹살을 먹는다")  
    else :  
        print("소고기를 먹는다")  
else :  
    if money <= 1000 :  
        pass  
    else :  
        print("라면을 먹는다")
```

삼겹살을 먹는다

조건문

- pass

조건을 만족해도 아무 일도 일어나지 않게 하려면?

```
In [42]: money = 1000  
card = False
```

```
In [43]: if card :  
    if money < 30000 :  
        print("삼겹살을 먹는다")  
    else :  
        print("소고기를 먹는다")  
else :  
    if money <= 1000 :  
        pass  
    else :  
        print("라면을 먹는다")
```

반복문

- 반복문

똑같은 작업을 반복해서 할 수 있도록 하는 문법

- 1) for문
- 2) while 문

반복문 - for

- for문

```
In [ ]: for 변수 in range(변수가 속한 자료형, 혹은 변수의 범위) :  
        수행해야할 문장1  
        수행해야할 문장2
```

주의 콜론(:)과 들여쓰기에
주의할 것

```
In [1]: for x in range(0,5) :  
        print(x)
```

```
0  
1  
2  
3  
4
```

반복문 - for

- for문

```
In [1]: for x in range(0,5) :  
    print(x)
```

```
0  
1  
2  
3  
4
```



변수의 범위 지정하기

- 1) 숫자로 범위주기
- 2) 자료형으로 범위주기

반복문 - for

- 변수의 범위 지정하기

- 1) 숫자로 범위주기

range (0, 5, 1)
 ↑ ↑ →
 시작 끝 간격

- 시작부터 끝 - 1 까지
- 간격이 1이라면 생략가능
- 0부터 시작하여 1씩 증가한다면 시작 인덱스 생략가능

반복문 - for

- 변수의 범위 지정하기

- 1) 숫자로 범위주기

QUIZ! 5에서부터 0까지 카운트 다운을 세 볼까요?

```
In [2]: # 카운트 다운
for count in range(5, -1, -1):
    print(count)
```

```
5
4
3
2
1
0
```

→ 하나씩 작아지므로 간격은 -1
→ 0까지 출력해야하므로 -1이 끝
인덱스가 되어야 함

반복문 - for

- 변수의 범위 지정하기

2) 자료형으로 범위 주기

```
In [4]: word = 'Hello!'  
      for w in word:  
          print(w)
```

H
e
l
l
o
!

```
In [5]: for a, b in [(2,1), (2,2), (2,3), (2,4)]:  
          print(a*b)
```

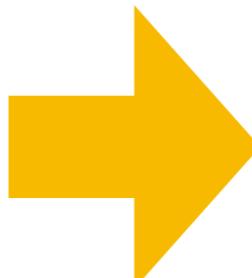
2
4
6
8

반복문 - for

- 이중으로 반복문을 사용하기

```
In [3]: # 구구단 2단과 3단을 출력하기
```

```
for i in range(2, 4) :  
    print("== %d 단 ==" % i)  
    for j in range(1, 10) :  
        print(i * j)
```



==== 2 단 ===

2
4
6
8
10
12
14
16
18

==== 3 단 ===

3
6
9
12
15
18
21
24
27

반복문 - while

- while문

조건문이 참인 동안 반복해서 문장을 수행함

```
In [ ]: while 조건 :  
        수행할 문장1  
        수행할 문장2
```

```
In [14]: # 100하의 짝수 프린트하기  
i = 1  
  
while i <= 10 :  
    if i % 2 == 0 :  
        print(i)  
    i += 1
```

→ 조건문에 쓰일 변수

→ 조건문

→ 변수를 증감

2
4
6
8
10

반복문 - while

- break

반복문에서 빠져나오고 싶다면?

```
In [17]: # 100번째 방문자 찾기  
i = 90
```

```
while i :
```

```
    i += 1
```

```
    if i == 100 :
```

```
        print("축하합니다. %d번째 방문자입니다." % i)
```

```
        break
```

```
print("감사합니다. 이벤트가 종료되었습니다.")
```

→ 변수는 0이 아니라면 항상 참
(무한 반복)

→ 반복문이 종료되었으

축하합니다. 100번째 방문자입니다.

므로 수행됨

감사합니다. 이벤트가 종료되었습니다.

break를 사용하면 반복문이 더 이상 작동하지 않고 멈춤

반복문 - while

- continue

반복문의 첫부분으로 돌아가고 싶다면?

```
In [18]: i = 0  
  
while i < 11 :  
    i += 1  
    if i == 6 :  
        continue  
    if i % 2 == 0 :  
        print(i)
```

→ 6일 때는 continue를 사용해 반복문의 가장 처음으로 가도록 함

2
4
8
10

→ 6일 때는 짝수임에도 불구하고
프린트가 안됨

정리하기

- 파이썬
- 파이썬의 자료형
 - 숫자(int, float)
 - 문자(string)
 - 불(bool)
 - 리스트(list)
 - 튜플(tuple)
 - 딕셔너리(dict)
 - 셋(set)
- 조건문
- 반복문

정리 문제 1-5

1. 20이하의 자연수 중 3으로 나눴을 때 나머지가 1인 숫자를 출력하기
2. 아래와 같이 출력하기

★

★★

★★★

★★★★

★★★★★

3. 고객의 개인정보보호를 위하여 이름을 비식별화하여 출력하기

예시) 홍길동 → 홍*동

파일 쓰기

- open

open은 파일을 여는데 사용하는 함수

```
open(file_path, mode=      , encoding=      )
```

- **file_path** : 파일의 주소
- **mode** : 파일을 열 때 필요한 파일 모드(읽기/쓰기/수정)
- **encoding** : 파일을 열 때 필요한 인코딩 (생략 가능)

```
In [58]: new_file = open('./sample_data/new.txt', 'w')
```

```
In [59]: print(new_file)
```

```
<_io.TextIOWrapper name='./sample_data/new.txt' mode='w' encoding='UTF-8'>
```

* 주의 * open으로 연 파일은 파일객체를 반환함

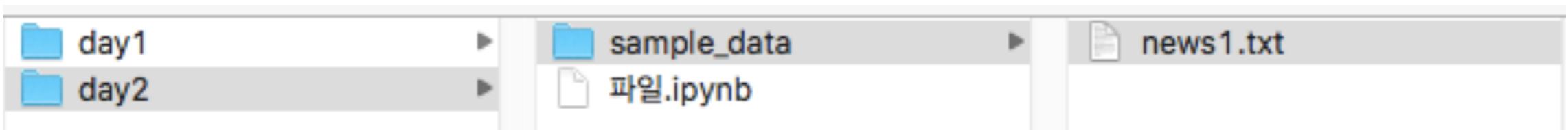
파일 쓰기

- open

1. file path

파일은 두 가지 방식으로 표현할 수 있음

- 1) 절대 경로 : 파일이 가지고 있는 고유한 경로로 현재 위치가 어디인지와 무관하게 항상 해당 파일에 접근 가능



→ `Users/Desktop/day2/sample_data/news1.txt`

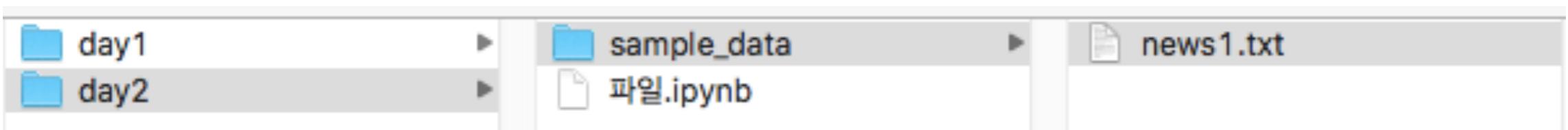
파일 쓰기

- open

1. file path

- 1) 절대 경로

- 2) 상대 경로 : 지금 내가 작업하고 있는 위치에서부터 해당 파일까지의 경로로, 달라질 수 있음



Quiz. 현재 ‘파일.ipynb’에서 작업 중이고 ‘new1.txt’파일의 상대 경로는?

→ `./sample_data/news1.txt`

현재 파일의 위치	/
현재 파일의 상위 폴더	../

파일 쓰기

- open

2. mode

파일 모드	설명	비고
r	읽기 모드	파일에 새로운 내용을 쓰거나 수정이 불 가능
w	쓰기 모드	파일에 새로운 내용을 덮어씀(기존의 내용 모두 삭제)
a	수정 모드	파일에 새로운 내용이 추가됨(기존의 내용 + 새로운 내용)
rb	바이너리 읽기 모드	바이너리 파일을 읽을 때 사용
wb	바이너리 쓰기 모드	바이너리 파일을 쓸 때 사용

파일 쓰기

- open
- 3. encoding

지정하지 않으면 플랫폼에 따르게 됨

- UTF-8 : 대표적인 조합형 유니코드 인코딩 방식

파일 쓰기

- **write**

파일.**write(내용)**

- 파일 : open으로 열었던 파일 객체
- 내용 : 파일에 쓰고 싶은 내용

```
In [60]: data = '새로운 내용을 쓰고 싶다면, \n이렇게 작성하면 됩니다.'
```

```
In [61]: new_file.write(data)
```

```
Out[61]: 29
```

* 주의 * 파일을 열고, 작업을 마쳤다면 반드시 닫을 것

```
In [62]: new_file.close()
```

파일 쓰기

- with

with open(file_path, mode=) as file :

수행할 문장 1

수행할 문장 2

with문을 사용하면 with블록을 벗어나는 순간 파일을 자동으로 close해줌

- **as file** : open으로 연 파일 객체를 변수로 받아줌

```
In [74]: with open('./sample_data/new2.txt', 'w') as f :  
    data = '이렇게도 작성이 가능합니다. close를 따로 안해도 되요.'  
    f.write(data)
```

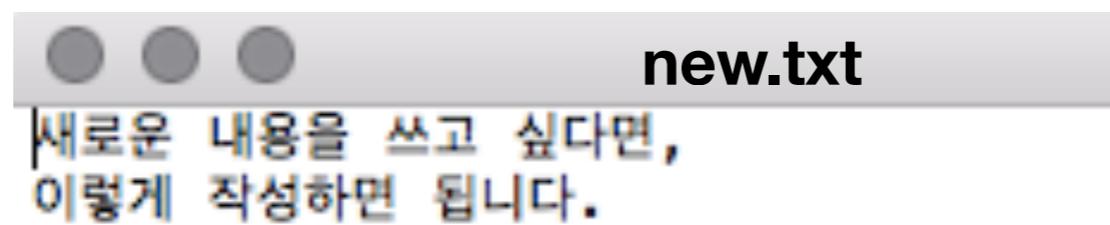
파일 읽기

- `read` / `readline` / `readlines`

함수	설명
<code>read</code>	파일 전체를 한번에 읽어서 string 으로 반환
<code>readline</code>	파일을 한 줄만 읽어서 string 으로 반환
<code>readlines</code>	파일을 줄단위로 모두 읽어서 list 로 반환

파일 읽기

- read / readline / readlines



1. read

```
In [84]: with open('./sample_data/new.txt', 'r') as f:  
    lines = f.read()  
    print(lines)  
    print(type(lines))
```

새로운 내용을 쓰고 싶다면,
이렇게 작성하면 됩니다.
<class 'str'>

2. readline

```
In [85]: with open('./sample_data/new.txt', 'r') as f:  
    line = f.readline()  
    print(line)  
    print(type(lines))
```

새로운 내용을 쓰고 싶다면,
<class 'str'>

3. readlines

```
In [87]: with open('./sample_data/new.txt', 'r') as f:  
    lines = f.readlines()  
    print(lines)  
    print(type(lines))
```

['새로운 내용을 쓰고 싶다면,\n', '이렇게 작성하면 됩니다.']
<class 'list'>

참고) readline으로 모든 내용 읽기

```
In [88]: with open('./sample_data/new.txt', 'r') as f:  
    while True :  
        line = f.readline()  
        if not line : break  
        print(line)
```

새로운 내용을 쓰고 싶다면,
이렇게 작성하면 됩니다.

final_quiz

오늘 배운 모든 것을 종합하는 문제들을 풀어봅시다.

앞으로 나올 문제들은 [초보를 위한 파이썬 300제](#)에서 발췌했습니다.

081

아래의 표에서, 아이스크림 이름을 키값으로, (가격, 재고) 리스트를 딕셔너리의 값으로 저장하라. 딕셔너리의 이름은 inventory로 한다.

이름	가격	재고
메로나	300	20
비비빅	400	3
죠스바	250	100

082

081의 inventory 딕셔너리에서 메로나의 가격을 화면에 출력하라.

실행 예시:

300 원

083

081의 inventory 딕셔너리에서 메로나의 재고를 화면에 출력하라.

실행 예시:

20 개

084

081의 inventory 딕셔너리에 아래 데이터를 추가하라.

이름	가격	재고
월드콘	500	7

실행 예시:

```
>> print(inventory)
{'메로나': [300, 20], '비비빅': [400, 3], '죠스바': [250, 100], '월드콘': [500, 7]}
```

085

다음의 딕셔너리에서 key 값으로만 구성된 리스트를 생성하라.

```
icecream = {'탱크보이': 1200, '풀라포': 1200, '빵빠레': 1800, '월드콘': 1500, '메로나': 1000}
```

086

다음의 딕셔너리에서 values 값으로만 구성된 리스트를 생성하라.

```
icecream = {'탱크보이': 1200, '풀라포': 1200, '빵빠레': 1800, '월드콘': 1500, '메로나': 1000}
```

087

icecream 딕셔너리에서 아이스크림 판매 금액의 총합을 출력하라.

```
icecream = {'탱크보이': 1200, '풀라포': 1200, '빵빠레': 1800, '월드콘': 1500, '메로나': 1000}
```

출력 예시:

6700

101

사용자로부터 입력받은 문자열을 두 번 출력하라. 아래는 사용자가 "안녕하세요"를 입력한 경우의 출력 결과이다.

```
>> 안녕하세요
```

```
안녕하세요안녕하세요
```

102

사용자로부터 하나의 숫자를 입력받고, 입력 받은 숫자에 10을 더해 출력하라.

```
>> 숫자를 입력하세요: 30
```

```
40
```

103

사용자로부터 하나의 숫자를 입력 받고 짹수/홀수를 판별하라.

```
>> 30
```

```
짝수
```