

Python_Basic

류영표

목 차



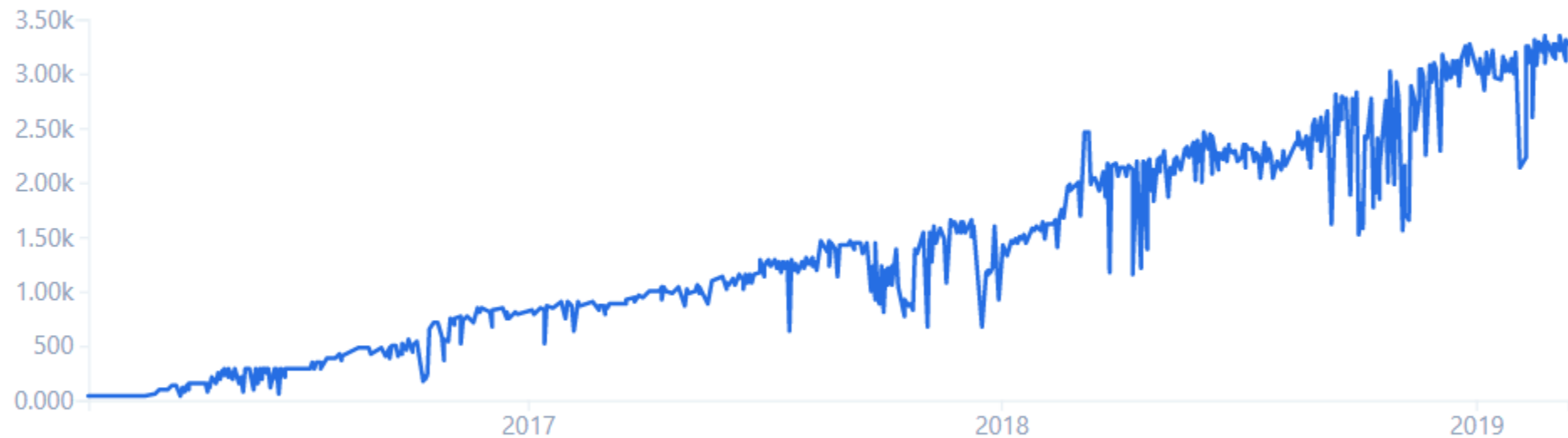
1. 파이썬 소개
2. 파이썬의 기본
3. 파이썬의 자료형
4. 조건문
5. 반복문
6. 파일 읽기/ 쓰기
7. 함수



파이썬 소개

Job Trends

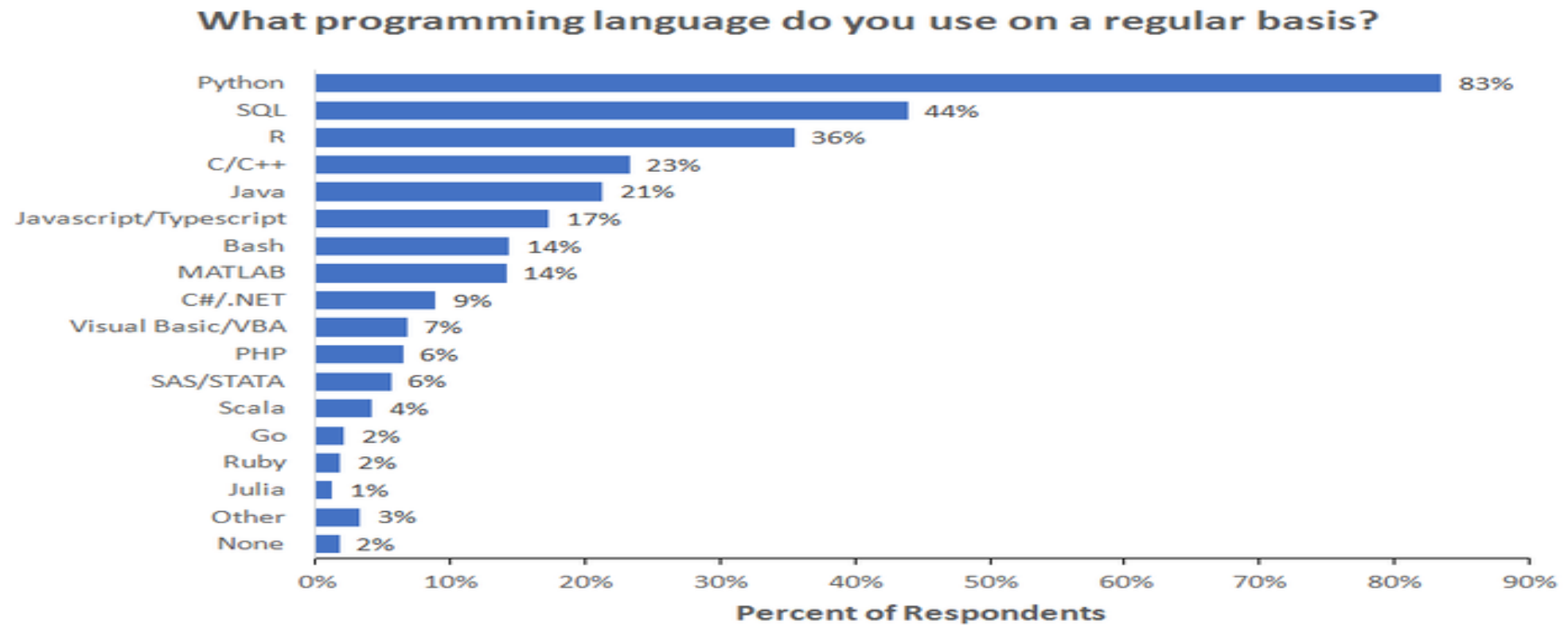
Data Scientist job openings at the world's top companies



Data from Thinknum - [Open dataset](#)

● Title (Count)

Programming Language Ranking



What is programming?



코딩 교육

Fig. 1: 코딩교육은 왜 필요할까요?(코딩으로 준비하는 미래)



“세상의 모든 사람은 컴퓨터 프로그래밍을 배워야 합니다.
프로그래밍은 생각하는 법을 가르쳐 주기 때문입니다.”

“Everybody in this country should learn to program a computer,
because it teaches you how to think”

- 스티브 잡스 -



“프로그램은 사고력과 문제해결력을 향상시킵니다.”

“Learning to write programs stretches your mind,
and helps you think better.”

- 빌 게이츠 -



“코딩 교육은 당신의 미래일 뿐만 아니라 조국의 미래다.”

“Learning these skills isn't just important for your future,
it's important for our country's future.”

- 버락 오바마 -

파이썬 이란?

- 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

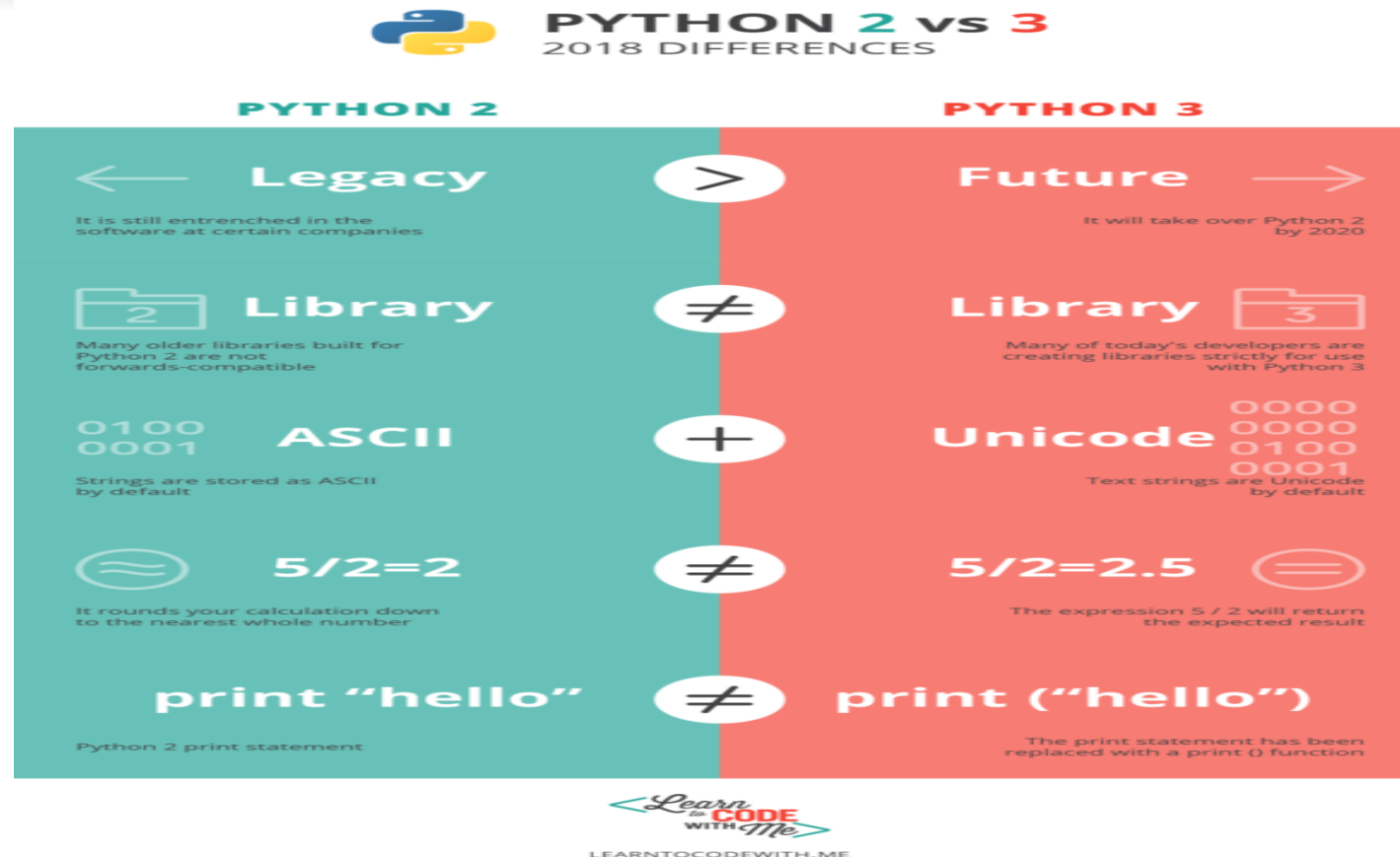
- 2. 널리쓰인다.

- 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

Python 2 VS python 3



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 |

Why 파이썬?

간결하고 쉬운 문법

(1) 간결함

| C | JAVA | Python |
|--|---|--------------------------|
| <pre>int array[2]; array[0] = 1; array[1] = 2;</pre> | <pre>int[] array = new int[2]; array[0] = 1; array = 2;</pre> | <pre>array = [1,2]</pre> |

(2) 들여쓰기(Indentation)

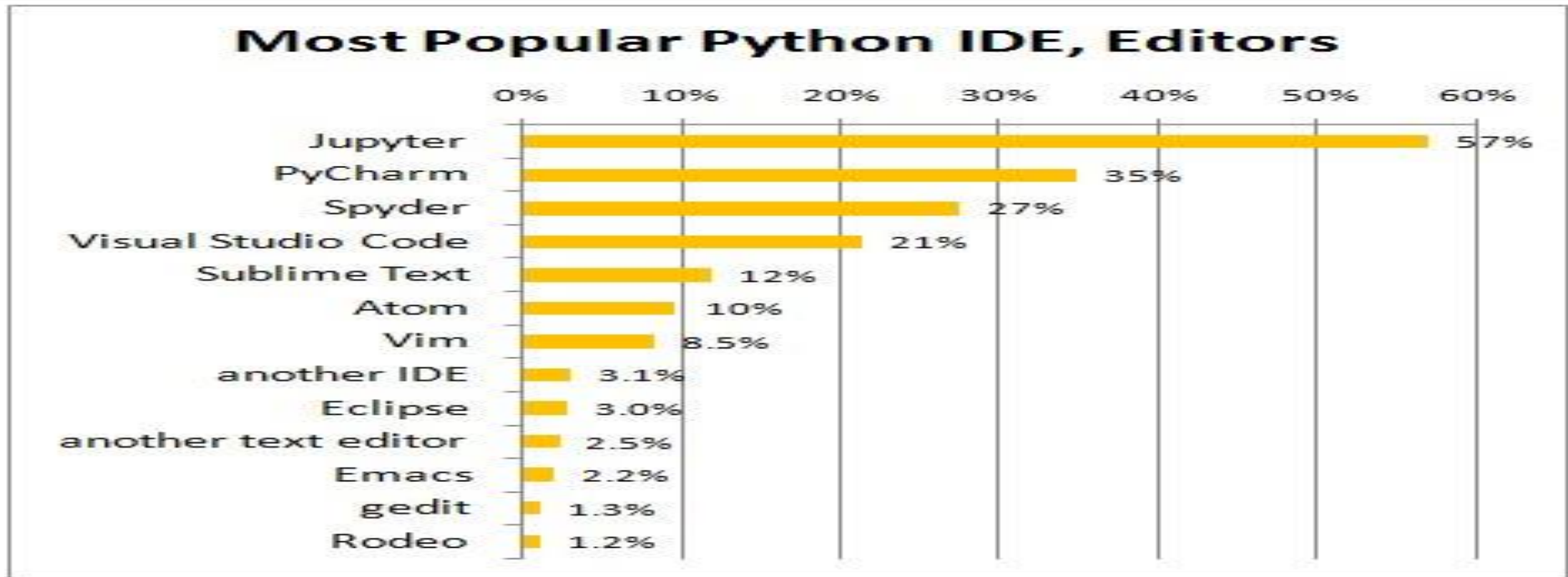
| C | JAVA | Python |
|--|--|--|
| <pre>for (int a=0; a<3; a++) { print(a) }</pre> | <pre>for (int a=0; a<3; a++){ System.out.println(a) }</pre> | <pre>for a in range(3): print(a)</pre> |

파이썬 쓰는 회사

The Google logo, consisting of the word "Google" in its multi-colored sans-serif font, is centered within a light blue circular background.The Netflix logo, with the word "NETFLIX" in red, bold, sans-serif capital letters, is centered within a light blue circular background.The Nexon logo, featuring a stylized blue and green cube icon to the left of the word "NEXON" in black, bold, sans-serif capital letters, is centered within a light blue circular background.The SK Telecom logo, featuring a stylized orange and red arrow icon to the left of the text "SK telecom" in orange, is centered within a light blue circular background.The Naver logo, with the word "NAVER" in green, bold, sans-serif capital letters, is centered within a light blue circular background.The YouTube logo, featuring a red play button icon to the left of the word "YouTube" in black, bold, sans-serif capital letters, is centered within a light blue circular background.The Kakao logo, with the word "kakao" in yellow, lowercase, sans-serif letters, is centered within a light blue circular background.The Coupang logo, with the word "coupang" in multi-colored lowercase letters, is centered within a light blue circular background.

출처 : <https://www.infllearn.com/roadmaps/38>

파이썬 개발환경



2020.01월 기준

출처: <https://wakestand.tistory.com/135>

파이썬 개발환경



- 손쉬운 패키지 설치
- 가상 환경을 통한 패키지 버전관리가 용이

파이썬 개발환경



- web을 통한 접근 가능
- Coding 결과를 실시간으로 확인 가능
- 자동완성 기능
- 다양한 언어 지원
- markdown을 통해 문서화 가능

ANACONDA 설치

<https://www.anaconda.com/products/individual>

Anaconda Installers

Windows 

Python 3.8

64-Bit Graphical Installer (466 MB)

32-Bit Graphical Installer (397 MB)

MacOS 

Python 3.8

64-Bit Graphical Installer (462 MB)

64-Bit Command Line Installer (454 MB)

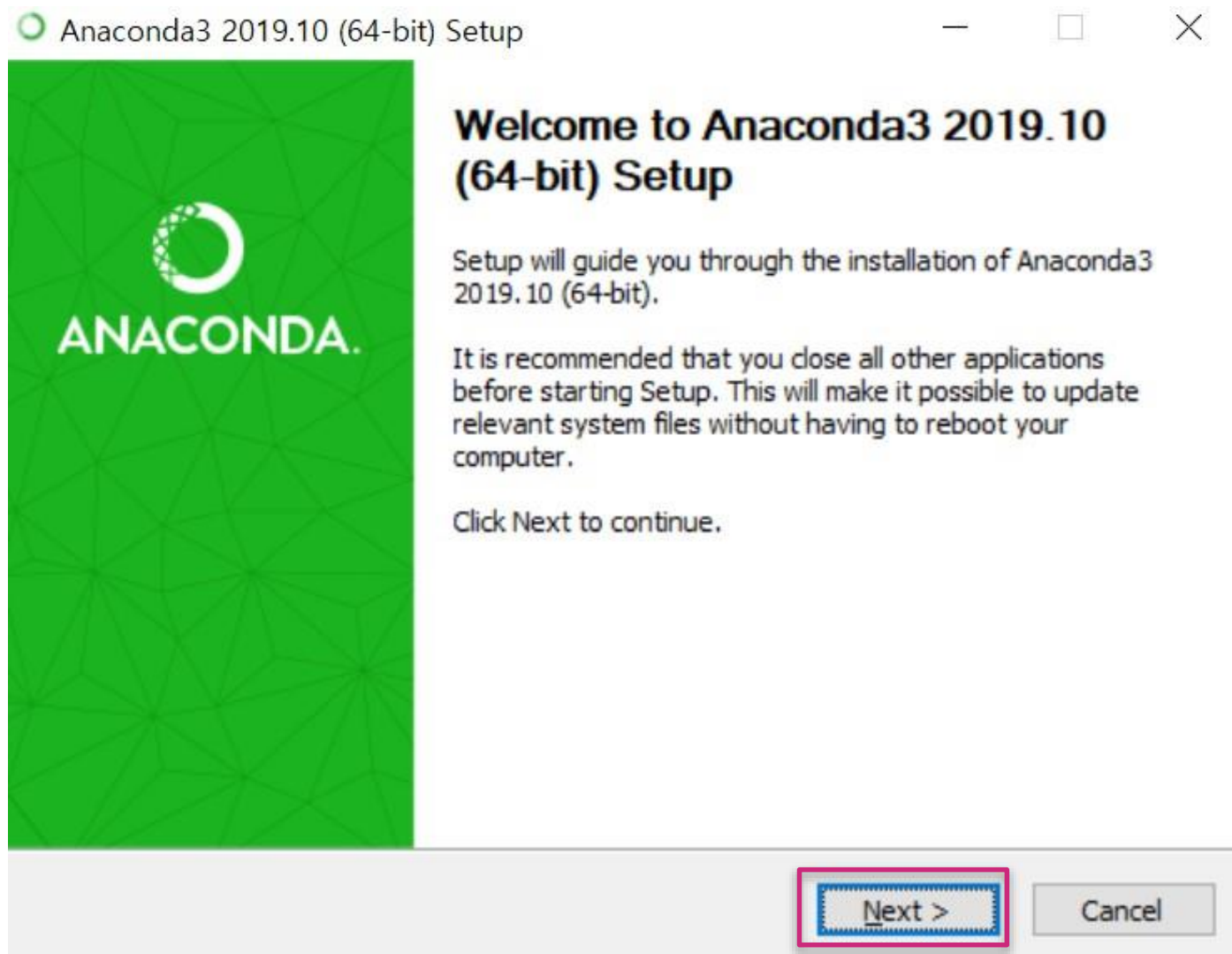
Linux 

Python 3.8

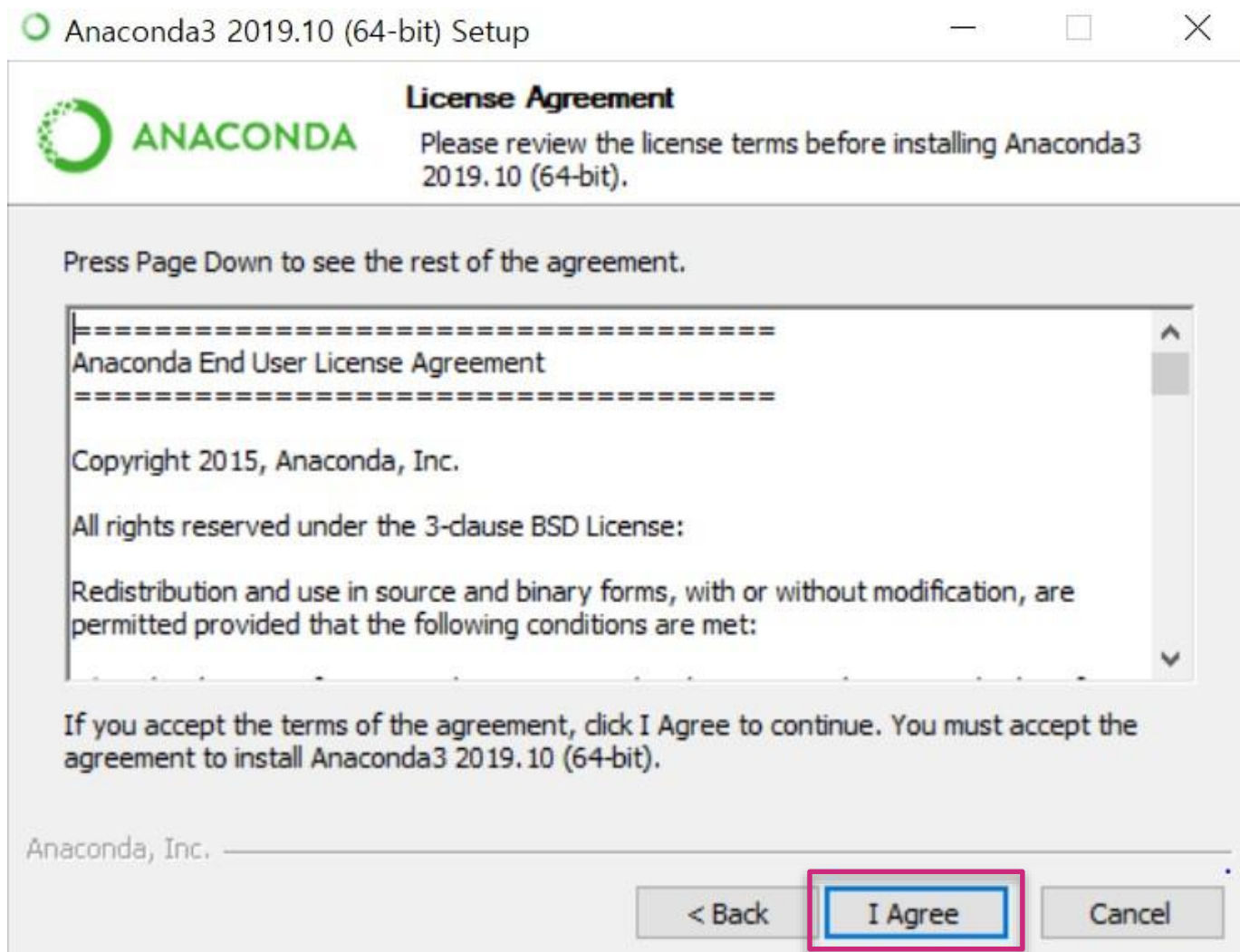
64-Bit (x86) Installer (550 MB)

64-Bit (Power8 and Power9) Installer (290 MB)

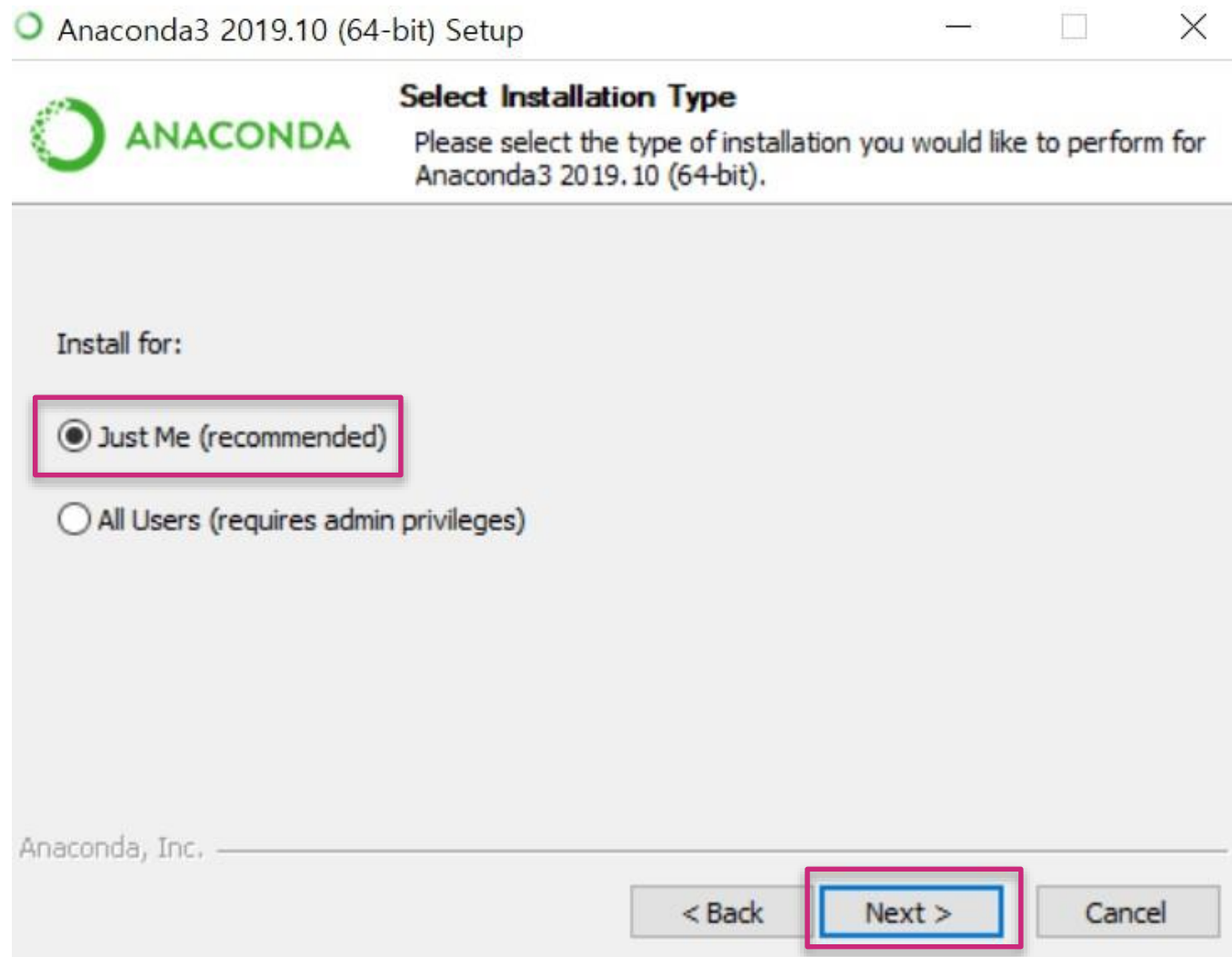
ANACONDA 설치



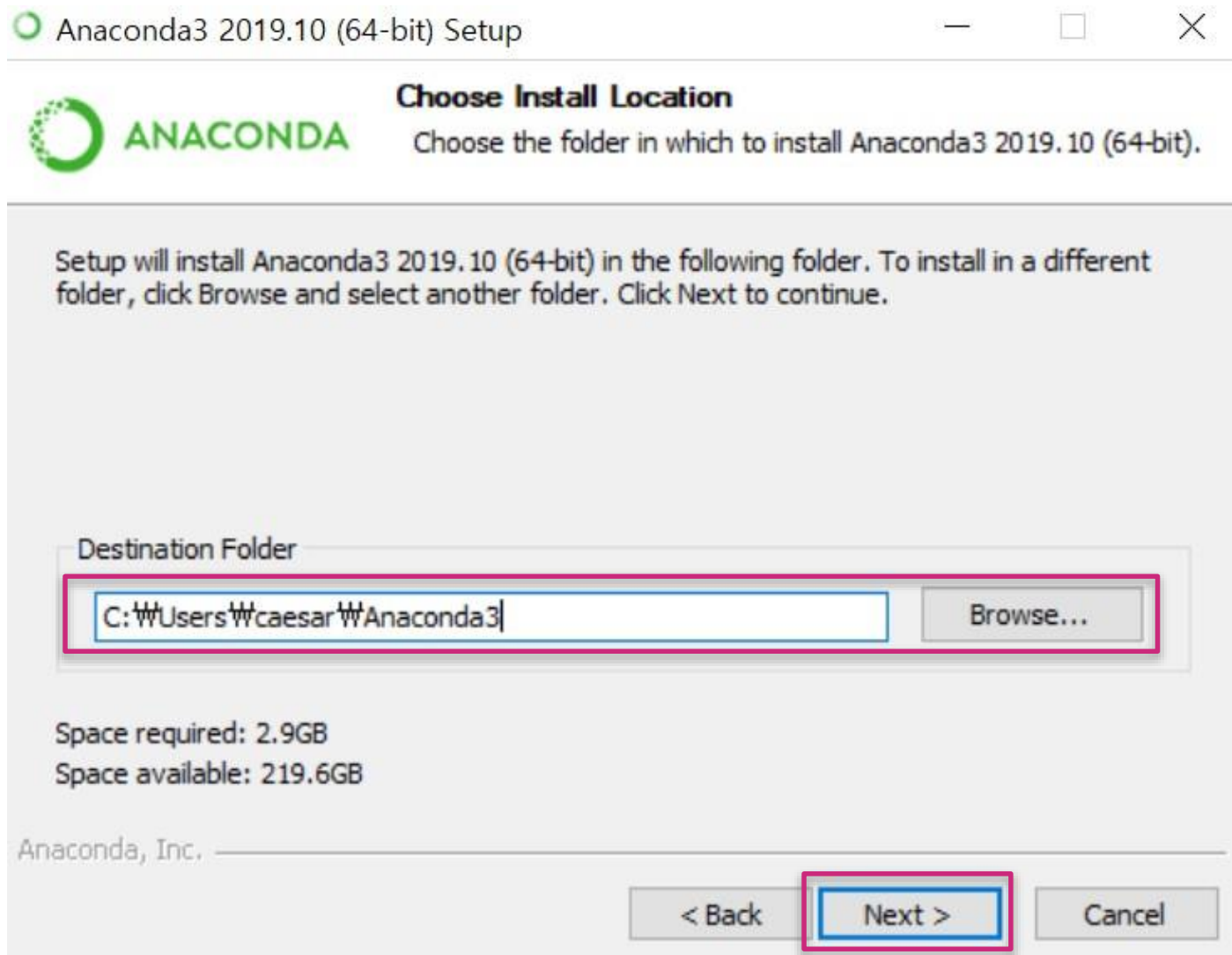
ANACONDA 설치



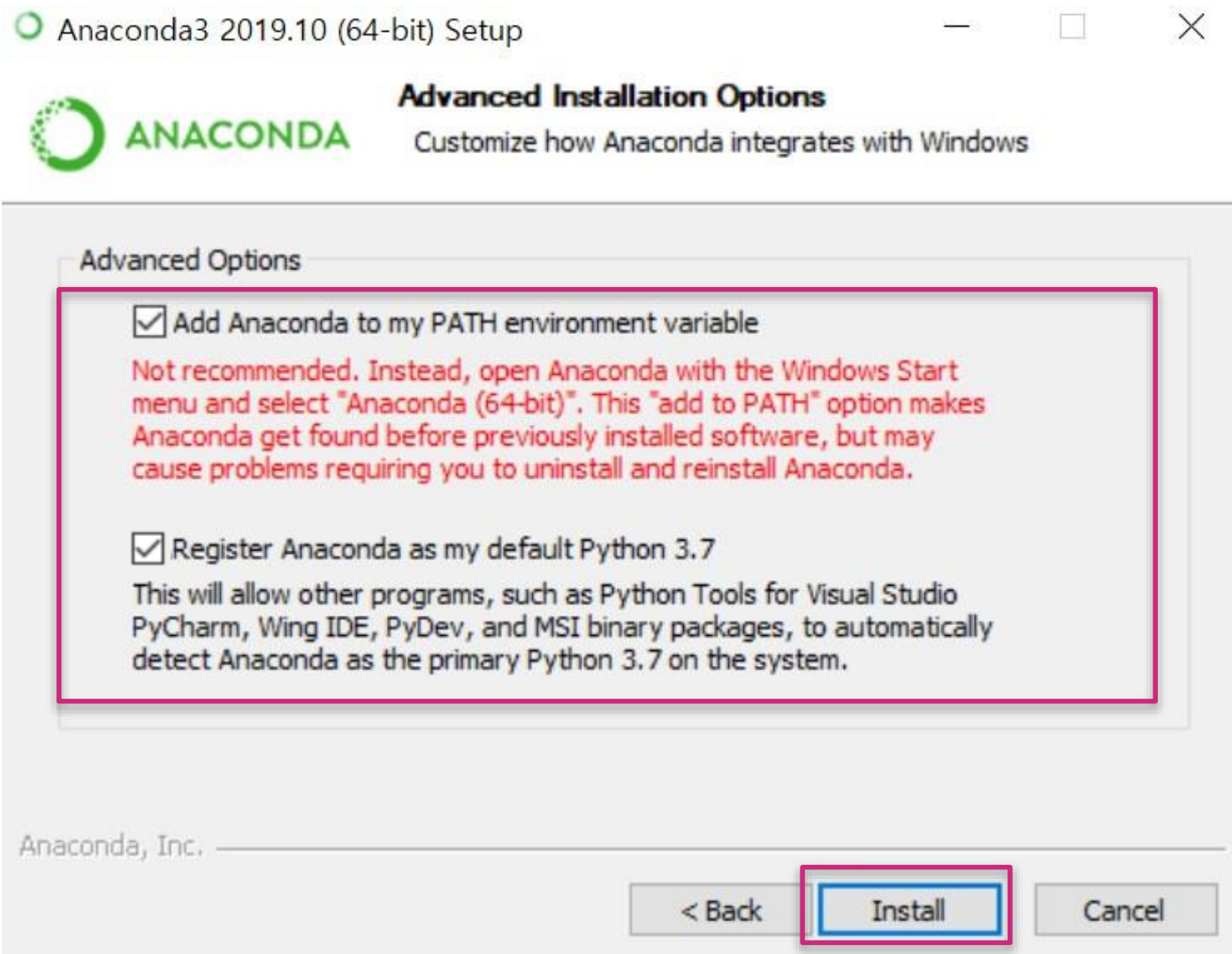
ANACONDA 설치



ANACONDA 설치



ANACONDA 설치



Jupyter Notebook 사용하기

- 아나콘다 네비게이터 실행



Jupyter Notebook 사용하기

- Anaconda Prompt 실행 후 아래 명령어 실행
→ jupyter notebook

```
Last login: Wed Aug  5 15:41:48 on ttys000  
➔ ~ jupyter notebook
```


Jupyter Notebook 사용하기

- 새로운 노트북 생성하기



Quit

Logout

Files

Running

Clusters

Select items to perform actions on them.

Upload

New ▾



☐ 0 ▾

📁 / sample

Name ▾

Last Modified

File size

📁 ..

seconds ago

☐

📁 sample1

seconds ago



jupyter

Quit

Logout

Files

Running

Clusters

Select items to perform actions on them.

Upload

New ▾



☐ 0 ▾

📁 / sample

Name ▾

Notebook:

Python 3

Other:

Text File

Folder

Terminal

Jupyter Notebook 사용하기

- 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
```

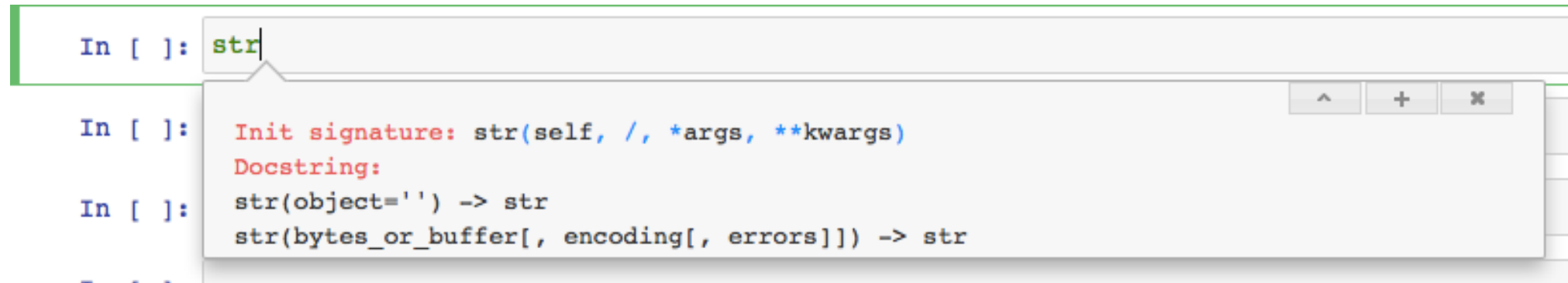
```
In [ ]: # 선택한 셀을  
        # 복사하기 : c  
        # 잘라내기 : x
```

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

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

Jupyter Notebook 사용하기

- 유용한 단축키
 - 함수나 변수에 대한 설명1: Shift + Tab



```
In [ ]: str
```

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

- 함수나 변수에 대한 설명2: ? 셀을 실행 해야함



```
In [1]: str?
```

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



Google colab 사용법

- 풀 네임은 Google Colaboratory
- Google Drive + Jupyter Notebook
 - Google Drive처럼 협업 가능(동시에 수정 가능)
- <https://colab.research.google.com/>로 접속시 사용 가능
- 컴퓨터 사양(21년 01월 기준)
 - Ubuntu 18.04.5 LTS
 - CPU 제논 2.3GHz
 - 메모리 13G
 - GPU : K80 또는 T4 :
 - TPU도 사용 가능
- GPU 사용시 최대 12시간 (Colab은 GPU를 무료로 사용가능)
- Github의 소스 코드를 Colab에서 사용 가능

Google colab

1. OS 확인

- `!cat /etc/issue.net`

하드웨어 사양

1. CPU

- `!cat /proc/cpuinfo`

2. Memory

- `!cat /proc/meminfo`

3. Disk

- `!df -h`

4. GPU

- `!nvidia-smi`

Python IDE Pycharm 이란?

- Pycharm은 Python 언어에 사용되는 JetBrains사에 의해 개발된 IDE입니다.
- IDE는 컴파일, 코드 편집기, 디버거, 배포 등의 모든 작업을 하나의 프로그램에서 할 수 있도록 해주는 통합 개발 환경

Pycharm 의 특징

1. 코드 분석 및 코딩 지원
2. 프로젝트 및 코드 탐색
3. 파이썬
4. 웹 프레임 워크 지원(Professional 전용, 유료버전)
5. 통합 파이썬 디버거
6. 라인 단위 테스트
7. Google App Engine python(Professional 전용, 유료버전)
8. 과학적 도구 지원(Professional 전용, 유료버전)

파이썬의 기본

- 산술 연산자

1. 더하기 : +

[] 1 + 2



3

[] 250.7 + 300.2



550.9

▼ 2. 빼기 : -

[] 5 - 3



2

[] 250.5 - 100.3



150.2

▼ 3. 곱하기 : *

[] 5 * 3



15

[] 20.5 * 2



41.0

▼ 4. 나누기 : /

[] 10 / 2



5.0

[] 7 / 3




2.3333333333333335

파이썬의 기본

- 산술 연산자

▼ 4. 나누기 : /

```
[ ] 10 / 2
```

 5.0

```
[ ] 7 / 3
```

 2.3333333333333335

▼ 5. //

```
[ ] 10 // 2
```

 5

```
[ ] 7 // 3
```

 2

/의 연산 결과 : 소수점 표시

→ 실수형

//의 연산 결과 : 소수점 없음(나누기의 몫)

→ 정수형

파이썬의 기본

- 산술 연산자

▼ 6. 나머지 : %

```
[ ] 10 % 2
```

 0

```
[ ] 7 % 3
```

 1

▼ 7. 그 밖에 연산자

제곱 : **

```
[ ] 2 ** 3
```

 8

```
[ ] 5 ** 3
```

 125

파이썬의 기본

- 연산자의 우선순위

사칙연산의 연산자 우선순위와 같음
괄호()가 있다면 가장 우선 처리

[] 5 + 3 * 4

 17

[] (8 - 3) * 2

 10

연습문제 1

1번

10의 제곱을 출력해보자

```
] : ▶ # 정답을 적어주세요  
out[1] : 100
```

```
] : ▶ # 정답을 적어주세요  
out[3] : 100
```

파이썬의 기본

- 할당 연산자(Assignment) : =

변수 = 연산/조건/함수

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

파이썬의 기본

● 할당 연산자(Assignment) : =

▪ 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

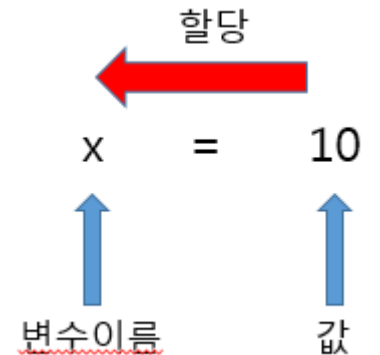


It just **changes the tag** to the new value in memory
Garbage collection free the memory automaticall

b = a

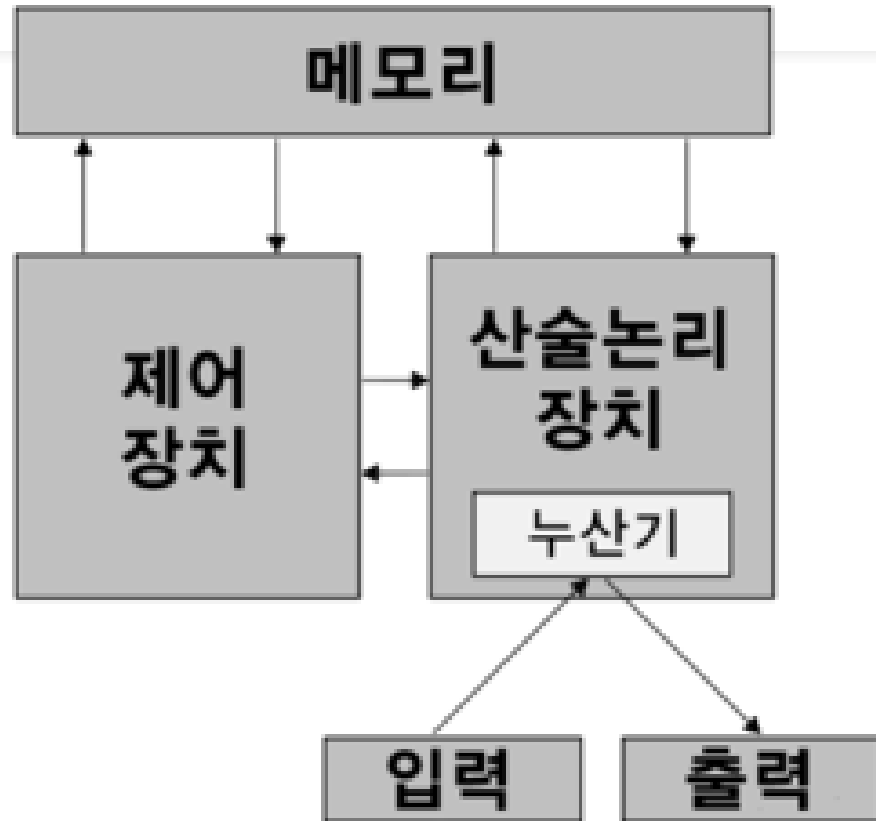


It makes a **new tag** bound to the same value.



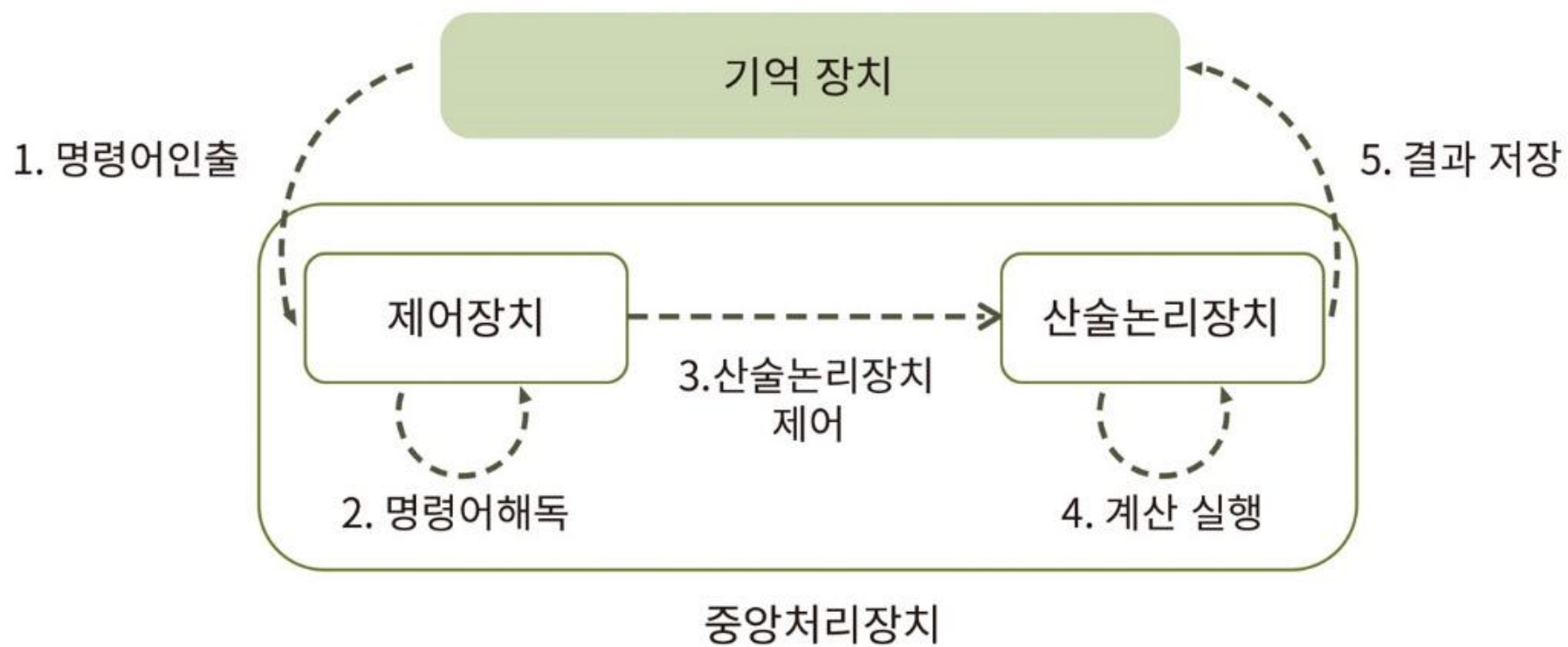
영문, 숫자 사용가능
대소문자 구분
숫자부터 시작할 수 없음.
특수문자 사용 불가(+, -, *, /, \$ 등)

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

중앙처리장치



파이썬의 기본

- 할당 연산자(Assignment) : **=**

```
In [ ]: ▶ a = 3
```

```
In [ ]: ▶ b = 5
```

```
In [ ]: ▶ print(a+b)
```

```
In [ ]: ▶ print("a+b")
```

```
In [ ]: ▶ a=8
```

```
In [ ]: ▶ print(a+b)
```

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

파이썬의 기본

- 할당 연산자(Assignment) : **=**

```
In [3]: ▶ a += 1  
a
```

```
Out[3]: 4
```

```
In [5]: ▶ a = a+1  
a
```

```
Out[5]: 4
```

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

파이썬의 기본

- 할당 연산자(Assignment) : =

```
In [6]: ▶ a += 1  
a
```

```
Out[6]: 5
```

```
In [7]: ▶ a -= 5  
a
```

```
Out[7]: 0
```

```
In [10]: ▶ a *= -1  
a
```

```
Out[10]: 0
```

```
In [11]: ▶ a /= 5  
a
```

```
Out[11]: 0.0
```

```
In [12]: ▶ a * = -1  
a
```

```
File "<ipython-input-12-0  
a * = -1  
^
```

```
SyntaxError: invalid syntax
```

```
In [13]: ▶ a / = 5  
a
```

```
File "<ipython-input-13-0  
a / = 5  
^
```

```
SyntaxError: invalid syntax
```

파이썬의 기본

- 할당 연산자(Assignment) : **=**

단순 산술 연산이 아닌 변수간의 연산도 가능

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

 12

주의 파이썬은 대소문자를 구분함

```
[ ] 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
```

파이썬의 기본

- 할당 연산자(Assignment) : =

파이썬은 들여쓰기(indent)에 굉장히 민감(jupyter 에서는 보정)

```
(base) C:\Users\USER>python
Python 3.7.3 (default, Apr 24 2019, 15:29:51) [MSC v.1915 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> 1+1
File "<stdin>", line 1
    1+1
    ^
IndentationError: unexpected indent
>>>
```

연습문제 2-1

1번

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

```
: ▶ # 정답을 적어주세요
```

```
: ▶ # 아래 주석을 풀고 실행시켜서 원하는 값이 나왔는지 확인하세요  
# total
```

```
[10]: 8
```

연습문제 2-2

2번

국어는 100점, 영어는 88점, 수학은 94점 일 때, 평균을 구하려고 한다.
각각의 점수를 kor, eng, math라는 변수에 저장한 후 평균을 구해 avg라는 변수에 할당해보자.

```
[ ]: ▶ # 정답을 적어주세요
```

```
[ ]: ▶ # 아래 주석을 풀고 실행시켜서 원하는 값이 나왔는지 확인하세요  
# avg
```

```
Out[13]: 94.0
```

파이썬의 자료형

1. 숫자(int, Float)
2. 문자(String)
3. 리스트(list)
4. 튜플(tuple)
5. 딕셔너리(dict)
6. 셋(set)
7. 불(bool)

1. 숫자(int, float)

- int와 float

```
[5] a = 3  
    type(a)
```

```
↳ int
```

```
[6] b = 5.0  
    type(b)
```

```
↳ float
```

int는 정수, float는 실수

type(변수) : 변수의 자료형을 알려주는 함수

1. 숫자(int, float)

- int와 float

```
[9] d = int(c)
    print(d)
    type(d)
```

```
↳ 8
   int
```

```
[10] e = float(5)
      print(e)
      type(e)
```

```
↳ 5.0
   float
```

숫자는 형 변환을 할 수 있음

1. 숫자의 자료형

| 구 분 | | 크기(byte) | 범위 |
|-----|-------------|----------|---|
| 정수형 | char | 1 | -128 ~ 127 |
| | short | 2 | -32768 ~ 32767 |
| | int | 4 | -2,147,483,648 ~ 2,147,483,647 |
| | long | 4 | -2,147,483,648 ~ 2,147,483,647 |
| | long long | 8 | -9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807 |
| 실수형 | float | 4 | $\pm 3.4 \times 10^{-37} \sim \pm 3.4 \times 10^{38}$ |
| | double | 8 | $\pm 1.7 \times 10^{-307} \sim \pm 3.4 \times 10^{308}$ |
| | long double | 8 이상 | double 이상 |

2. 문자(String)

- string

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

```
[11] f = '문자형 예시입니다.'  
      print(f)  
      type(f)
```

```
↳ 문자형 예시입니다.  
   str
```

2. 문자(String)

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

↳ 문장안에 따옴표를 쓰고 싶다면? "" 번갈아 쓰면 됩니다.

```
[13] escape = "escape 문자인 \" \\ \"(역슬래시 혹은 원화표시)도 사용가능"  
      print(escape)
```

↳ escape 문자인 " \ " (역슬래시 혹은 원화표시)도 사용가능

2. 문자(string)

```
[21] g = 3  
     h = '5'    따옴표가 있으므로 string
```

```
[22] type(g)
```

```
↳ int
```

```
[23] type(h)
```

```
↳ str
```

2. 문자(string)

```
[21] g = 3  
     h = '5'
```

```
[24] i = str(g)  
     print(i)  
     type(i)
```

```
↳ 3  
   str
```

```
[ ] j = int(h)  
     print(j)  
     type(j)
```

```
↳ 5  
   int
```

str와 int도 형 변환이 가능

2. 문자(string)

- string의 연산

```
[26] str_1 = '문자'  
     str_2 = '의 연산'  
     result = str_1 + str_2  
     result
```

☞ '문자의 연산'

```
[27] str_3 = '더하기'  
     result += str_3  
     result
```

☞ '문자의 연산더하기'

더하기 를 하면 두 문자열 을 붙여줌

2. 문자(string)

- string의 연산

```
[28] str_4 = '곱하기'  
      print(str_4 * 5)
```

↳ 곱하기곱하기곱하기곱하기곱하기

```
[29] str_5 = '반복'  
      print((str_4 + str_5)*2)
```

↳ 곱하기반복곱하기반복

곱하기는 그 수만큼 문자열을 반복함

2. 문자(string)

- string의 연산

단, 자료형이 다르다면 연산이 불가

```
int_1 = 8
str_6 = '문자'

error = str_6+int_1
```

```
-----
----
TypeError                                Traceback (most
<ipython-input-1-9ac98546a9bd> in <module>
      2 str_6 = '문자'
      3
----> 4 error = str_6+int_1

TypeError: can only concatenate str (not "int") to str
```

2. 문자(string)

- string의 indexing(인덱싱)

인덱싱? 순서가 있는 변수의 특정 위치에 접근 할 수 있게 해줌

변수[위치]

주의 파이썬의 인덱싱은 0부터 시작

문자(string)

- string의 indexing(인덱싱)

인덱싱? 순서가 있는 변수의 특정 위치에 접근 할 수 있게 해줌

변수[위치]

```
[47] str_5 = '인덱싱을 하기 위한 string입니다.'  
      len(str_5)
```

len(변수)는 변수의 길이를 재는 함수

```
↳ 21
```

```
[48] str_5[2]
```

```
↳ '식'
```

2. 문자(string)

- string의 slicing(슬라이싱)

슬라이싱? 변수의 일정 부분에 접근하여 잘라옴

변수[시작 : 끝 : 간격]

주의

- 슬라이싱은 시작부터 끝 -1까지 가져옴
- 간격이 1이라면 생략해도 됨

2. 문자(string)

- string의 slicing(슬라이싱)

슬라이싱? 변수의 일정 부분에 접근하여 잘라옴

변수[시작 : 끝 : 간격]

```
[49] str_5 = '인덱싱을 하기 위한 string입니다.'  
      str_5[2:7]
```

↳ '싱을 하기'

2. 문자(string)

- string의 slicing(슬라이싱)

슬라이싱? 변수의 일정 부분에 접근하여 잘라옴

변수[시작 : 끝 : 간격]

```
[50] str_6 = '안녕하세요. 홍길동 고객님!'
```

```
[52] str_6[7:10]
```

```
↳ '홍길동'
```

2. 문자(string)

- string의 slicing(슬라이싱)

```
[54] str_6 = '안녕하세요. 홍길동 고객님!'  
      print(str_6[:2])
```

☞ 안녕

```
[57] print(str_6[11:])
```

☞ 고객님!

```
[58] print(str_6[:])
```

☞ 안녕하세요. 홍길동 고객님!

→ 시작점이 없다면 처음부터

→ 끝이 없다면 끝까지

→ 시작과 끝이 없다면 전체

2. 문자(string)

- string의 slicing(슬라이싱)

파이썬은 마이너스 인덱싱을 지원함

```
str_6 = '안녕하세요. 홍길동 고객님!'
```

```
print(str_6[-1])
```

!

```
print(str_6[-4:])
```

고객님!

```
print(str_6[:-2])
```

안녕하세요. 홍길동 고객

→ -1은 끝에서부터 거꾸로

2. 문자(string)

- string의 slicing(슬라이싱)

파이썬은 마이너스 인덱싱을 지원함

```
[65] str_6 = '안녕하세요. 홍길동 고객님!'  
      print(str_6[::-1])
```

↳ !님객고 동길홍 .요세하녕안

→[::-1]은 str전체를 뒤집음

2. 문자(string)

- string의 method

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

1. 대소문자 변환: lower, upper

```
[66] str_7 = 'Alphabet'  
      print(str_7.lower())
```

```
↳ alphabet
```

```
[67] print(str_7.upper())
```

```
↳ ALPHABET
```

2. 문자(string)

- string의 method

2. 해당 문자의 갯수 세기: count

```
[68] str_7 = 'Alphabet'  
      str_7.count('a')
```

```
↳ 1
```

→ count는 대소문자를 구분함

Quiz) str_7에서 모든 것을 소문자 바꾼 다음에 a 문자의 갯수를 세려면?

```
[71] str_7.lower().count('a')
```

```
↳ 2
```

2. 문자(string)

- string의 method

3. 문자열의 위치 찾기

1) find

→ 해당하는 문자열의 위치(인덱스)를 반환

```
[72] str_7 = 'Alphabet'  
     str_7.find('t')
```

```
↳ 7
```

2) index

```
[73] str_7.index('t')
```

```
↳ 7
```

2. 문자(string)

- string의 method

3. 문자열의 위치 찾기

1) find

```
[74] str_7 = 'Alphabet'  
      str_7.find('z')
```

☞ -1

→ 문자열에 없으면 -1 반환

2) index

```
[75] str_7.index('z')
```

☞ -----
ValueError: substring not found

→ 문자열에 없으면 error

2. 문자(string)

- string의 method

4. 문자열 바꾸기:replace

```
[82] str_8 = 'Life is C between B and D'
      str_8.replace('C', 'Chicken')
```

```
↳ 'Life is Chicken between B and D'
```

5. 문자열 나누기:split

```
[83] str_8.split(' ')
```

```
↳ ['Life', 'is', 'C', 'between', 'B', 'and', 'D']
```

→따옴표 ' ' 사이에 기준이 되는 문자를 넣기
생략 시 기준은 띄어쓰기

2. 문자(string)

- string의 method

6. 문자열 삽입: join

```
[84] str_9 = 'abcd'  
      ','.join(str_9)
```

```
↳ 'a,b,c,d'
```

→ 따옴표 ' ' 사이에 삽입할 문자를 넣기

7. join과 split

```
[85] str_10 = str_8.split()
```

```
[86] ' '.join(str_10)
```

```
↳ 'Life is C between B and D'
```

→ split한 결과에 join을 사용하면
다시 문자열로 만들 수 있음

2. 문자(string)

- string의 method

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

```
[21] print(dir(str_8))
```

```
['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__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']
```

3. 리스트(list)

- list

[]을 이용하여 리스트(list)임을 선언

```
[1] list_1 = [3,2,5]  
    list_2 = list()
```

```
[2] type(list_1)
```

```
[>] list
```

3. 리스트(list)

- list의 특징

리스트는 모든 자료형을 담을 수 있음

리스트는 삽입, 수정, 삭제가 자유로움

리스트는 순서가 있는 자료형 → 인덱싱과 슬라이싱이 가능

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

```
[5] list_3[3]
```

```
↳ ['이중 리스트', '가능']
```

```
[6] list_3[4][1]
```

```
↳ '튜플'
```

주의 python의 인덱스는 0부터 시작

3. 리스트(list)

- list의 인덱싱

```
[7] list_4 = [[1,2,3,4],  
              [2,4,6,8],  
              [3,6,9,12],  
              [4,8,12,16],  
              [5,10,15,20]]
```

가장 바깥쪽부터 괄호부터

```
[9] list_4[1][3]
```

앞에서부터 순서대로

↳ 8

3. 리스트(list)

- list의 인덱싱

```
[ ] list_5 = [[[1,2,3],[4],[5,[6,7,8,9]]],  
              [10,11,[12,13,[14]]]]
```

```
[ ] list_5[1][2][2]
```

```
↳ [14]
```

3. 리스트(list)

- list의 슬라이싱

```
[7] list_4 = [[1,2,3,4],  
              [2,4,6,8],  
              [3,6,9,12],  
              [4,8,12,16],  
              [5,10,15,20]]
```

```
[ ] list_4[1][:3]
```

```
↳ [2, 4, 6]
```

```
[ ] list_4[:2][1]
```

```
↳ [2, 4, 6, 8]
```

3. 리스트(list)

- list의 슬라이싱

```
[9] list_5 = [[1, 2, 3], [4], [5, [6, 7, 8, 9]]],  
             [10, 11, [12, 13, [14]]]]
```

```
[13] list_5[:2][0]
```

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

```
[14] list_5[0][:2]
```

```
↳ [[1, 2, 3], [4]]
```

3. 리스트(list)

- list의 연산

```
[15] list_6 = ['a', 'b', 'c']
```

```
[16] list_7 = ['가', '나', '다']
```

```
[17] list_6 + list_7
```

```
↳ ['a', 'b', 'c', '가', '나', '다']
```

```
[18] list_6 * 3
```

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


3. 리스트(list)

- list의 메소드

리스트는 삽입, 수정, 삭제가 자유로움

1. 삽입: append, insert

```
[19] list_8 = ['파이썬', 'C', 'java']
```

```
[20] list_8.append('R')  
list_8
```

```
↳ ['파이썬', 'C', 'java', 'R']
```

→ append는 리스트 제일 뒤에

→ insert는 원하는 위치에

```
[21] list_8.insert(2, 'C++')  
list_8
```

```
↳ ['파이썬', 'C', 'C++', 'java', 'R']
```

```
[22] list_8.insert(5, 'C')  
list_8
```

```
↳ ['파이썬', 'C', 'C++', 'java', 'R', 'C']
```

3. 리스트(list)

- list의 메소드

2. 수정: 덮어쓰기 → 인덱스나 슬라이싱을 통해 수정

```
[23] list_8[0] = 'Python'  
list_8
```

```
☞ ['Python', 'C', 'C++', 'java', 'R', 'C']
```

3. 리스트(list)

- list의 메소드

3. 삭제: remove

```
[24] list_8
```

```
↳ ['Python', 'C', 'C++', 'java', 'R', 'C']
```

```
[25] list_8.remove('C')  
list_8
```

```
↳ ['Python', 'C++', 'java', 'R', 'C']
```

→ remove는 지우고자 하는 요소를 명시해야함

→ 가장 앞에 있는 요소가 삭제됨

3. 리스트(list)

- list의 메소드

4. 꺼내오기: pop

```
[26] list_8.pop()
```

```
↳ 'c'
```

```
[28] list_8
```

```
↳ ['Python', 'C++', 'java', 'R']
```

→ pop은 괄호()안 인덱스에 해당하는 요소를 반환

→ 인덱스를 생략하면 가장 마지막 요소를 꺼내옴

3. 리스트(list)

- list의 메소드

5. 정렬 : sort

```
[29] list_9 = [1,5,7,2,6]
```

```
[30] list_9.sort()  
list_9
```

→ 리스트 자체를 정렬

```
↳ [1, 2, 5, 6, 7]
```

list.sort(key=None, reverse=False)

- **key** : 정렬해주고 싶은 기준, 기본적으로 < 비교에 의해 정렬(오름차순)
- **reverse** : **True**로 설정하게 되면 > 비교에 의해 정렬(내림차순)

3. 리스트(list)

- list의 메소드

5. 정렬: sort

```
[32] list_10 = [1,47,12,6]  
      list_10.sort(reverse=True)  
      list_10
```

```
↳ [47, 12, 6, 1]
```

3. 리스트(list)

- list의 메소드

6. 뒤집기: reverse

```
[37] list_9
```

```
↳ [1, 2, 5, 6, 7]
```

```
[38] list_9.reverse()  
list_9
```

```
↳ [7, 6, 5, 2, 1]
```

→ 현재 리스트 요소 순서를 뒤집음

→ 리스트 자체를 뒤집음

3. 리스트(list)

- list의 메소드

6. 뒤집기: reverse

```
[32] list_10 = [1,47,12,6]  
      list_10.sort(reverse=True)  
      list_10
```

```
↳ [47, 12, 6, 1]
```

```
[41] list_10.reverse()  
      list_10
```

```
↳ [1, 6, 12, 47]
```


연습문제 3

1번

다음과 같은 출력값을 만들어보자

```
In [ ]: ▶ a = '아메리카노를'  
        b = '좋아한다'  
        c = '많이'
```

```
In [ ]: ▶ # 정답을 적어주세요
```

```
Out[2]: '아메리카노를많이많이많이좋아한다'
```

연습문제 3

2번

좋아하는 음식 5개를 food라는 리스트에 담아보자

```
: ▶ # 정답을 적어주세요
```

3번

리스트 food의 가장 앞에 파이썬이, 가장 뒤에는 집이라는 단어가 오도록 수정해보자

```
In [ ]: ▶ # 정답을 적어주세요
```

```
In [ ]: ▶ # 아래의 주석을 풀고 실행시켜보세요  
# food
```

4번

pop을 이용하여 파이썬을 삭제해보자

```
In [ ]: ▶ # 정답을 적어주세요
```

```
In [ ]: ▶ # 아래의 주석을 풀고 실행시켜보세요  
# food
```

5번

remove를 이용하여 집을 삭제해보자

4. 튜플(tuple)

- tuple

()을 이용하여 튜플(tuple)임을 선언

```
[42] tuple_1 = ()  
     tuple_2 = tuple()
```

```
[43] 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

4. 튜플(tuple)

- tuple

인덱싱과 슬라이싱이 가능

리스트와 달리 요소를 **삽입/삭제/수정** 할 수 없음

```
[44] tuple_3 = (1, 2)
      tuple_3[0] = 3
```

```
↳ -----
TypeError                                Traceback (most recent call last)
<ipython-input-44-c8372825bae8> in <module>()
      1 tuple_3 = (1, 2)
----> 2 tuple_3[0] = 3

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

SEARCH STACK OVERFLOW

4. 튜플(tuple)

- tuple의 연산

```
[45] tuple_3 = (1, 2)
      tuple_4 = (3, )
      tuple_3 + tuple_4
```

☞ (1, 2, 3)

```
[46] tuple_4 * 2
```

☞ (3, 3)

5. 딕셔너리(dictionary)

- dictionary

{ }을 이용하여 딕셔너리(dictionary)임을 선언

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

- 딕셔너리는 **key**와 **value** 값으로 이뤄진 자료형
- 순서가 있는 자료형이 아니며, **key**를 통해 **value**값 에 접근이 가능
- **key**는 고유한 값으로 중복될 수 없음
- **value**는 중복 가능

5. 딕셔너리(dictionary)

- dictionary 추가/삭제/수정

1. 요소 추가하기: 새로운 **key**에 **value**를 할당

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

```
[49] dict_3['슈퍼맨'] = 300  
dict_3
```

```
↳ {'슈퍼맨': 300, '홍계월': 200, '홍길동': 100}
```

주의 인덱스 대신 **key** 사용

2. 요소 수정하기: 덮어쓰기

```
[51] dict_3['홍길동'] = 1  
dict_3
```

```
↳ {'홍계월': 200, '홍길동': 1}
```

5. 딕셔너리(dictionary)

- dictionary 추가/삭제/수정

3. 요소 삭제하기

```
[50] del dict_3['슈퍼맨']  
dict_3
```

```
↳ {'홍계월': 200, '홍길동': 100}
```

주의 인덱스 대신 **key** 사용

del : 파이썬 자체의 명령어로 메모리 자체에서 삭제

5. 딕셔너리(dictionary)

- key와 value에 접근하기

1. key에 접근하기: keys()

```
[1] dict_3 = {'홍길동' : 100, '홍계월' : 200, '슈퍼맨' : 150, '배트맨' : 250}  
print(dict_3.keys())  
print(type(dict_3.keys()))
```

```
↳ dict_keys(['홍길동', '홍계월', '슈퍼맨', '배트맨'])  
<class 'dict_keys'>
```

2. value에 접근하기: values()

```
[2] print(dict_3.values())  
print(type(dict_3.values()))
```

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

5. 딕셔너리(dictionary)

- key와 value에 접근하기

3. key와 value에 함께 접근하기

```
[3] print(dict_3.items())  
    print(type(dict_3.items()))
```

```
↳ dict_items([('홍길동', 100), ('홍계월', 200), ('슈퍼맨', 150), ('배트맨', 250)])  
<class 'dict_items'>
```

주의

리스트 처럼 보이나 실제로는 **dict**의 개체 이므로
리스트의 메소드를 사용할 수 없음

리스트처럼 사용하고 싶다면 리스트로 **형 변환**을 해야함

```
[4] list(dict_3.keys())[0]
```

```
↳ '홍길동'
```

5. 딕셔너리(dictionary)

- key와 value에 접근하기

4. key로 value에 접근하기

- 1) 인덱싱처럼 키값을 이용하여 바로 접근
- 2) 메소드 `get`을 사용

```
[5] dict_3
```

```
↳ {'배트맨': 250, '슈퍼맨': 150, '홍계월': 200, '홍길동': 100}
```

```
[6] print(dict_3['배트맨'])  
     print(dict_3.get('배트맨'))
```

```
↳ 250  
   250
```

5. 딕셔너리(dictionary)

- key와 value에 접근하기

4. key로 value에 접근하기

```
[7] print(dict_3['펍수'])
```

```
-----  
Traceback (most recent call last):  
  <ipython-input-7-43368d90b0e9> in <module>()  
    ----> 1 print(dict_3['펍수'])
```

```
KeyError: '펍수'
```

SEARCH STACK OVERFLOW

→ 키값을 이용하여 접근 할 경우, 해당 키가 없으면 **오류** 발생

```
[8] print(dict_3.get('펍수'))
```

```
None
```

→ get을 통해 접근할 경우, **None**을 반환

6. 셋(집합)

- set

`{ }`을 이용하여 `set`을 선언

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



```
[10] set_2
```

```
↳ {'H', 'e', 'l', 'o'}
```

```
[11] set_3
```

```
↳ {1, 2, 3}
```

```
[12] set_4
```

```
↳ {3, 5, 'hi'}
```

중복을 허용하지 않음
순서가 없음

6. 셋(집합)

- set의 연산

1. 교집합

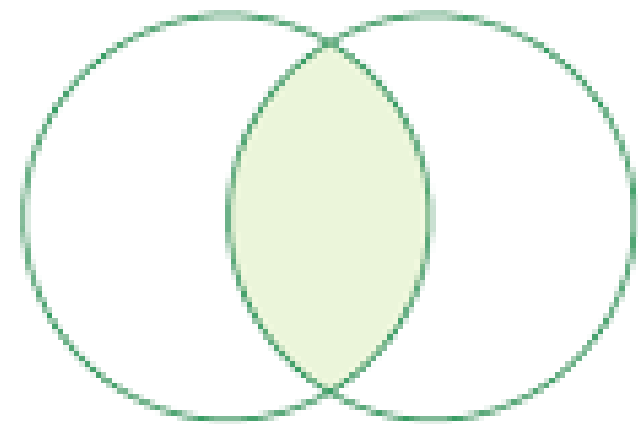
```
[13] set_5 = {1,2,3,4}  
      set_6 = {4,5,6,7}
```

```
[14] set_5 & set_6
```

```
↳ {4}
```

```
[15] set_5.intersection(set_6)
```

```
↳ {4}
```



집합 1 집합 2

(a) 교집합

6. 셋(집합)

- set의 연산

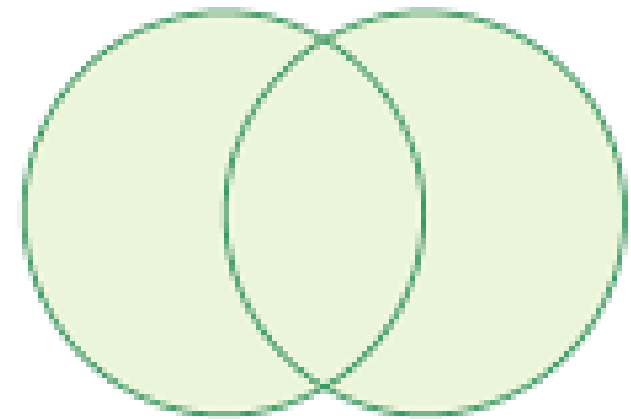
2. 합집합

```
[16] set_5 | set_6
```

```
↳ {1, 2, 3, 4, 5, 6, 7}
```

```
[17] set_5.union(set_6)
```

```
↳ {1, 2, 3, 4, 5, 6, 7}
```



집합 1 집합 2

(b) 합집합

6. 셋(집합)

- set의 연산

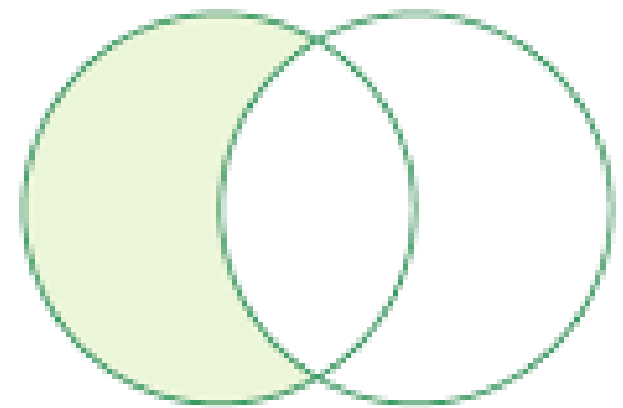
3. 차집합

```
[18] set_5 - set_6
```

```
↳ {1, 2, 3}
```

```
[19] set_5.difference(set_6)
```

```
↳ {1, 2, 3}
```



집합 1 집합 2

(c) 차집합

6. 셋(집합)

- set의 메소드

1. 한 개 추가하기 : add

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

```
↳ {4, 5, 6, 7, 8}
```

2. 여러 개 추가하기 : update

```
[23] set_6.update([1, 2, 3])  
     set_6
```

```
↳ {1, 2, 3, 4, 5, 6, 7, 8}
```

6. 셋(집합)

- set의 메소드

3. 한 개 지우기 : remove

```
[24] set_6.remove(3)  
      set_6
```

```
↳ {1, 2, 4, 5, 6, 7, 8}
```

7. 불(bool)

- bool : 참과 거짓을 나타내는 자료형

참 : True

거짓 : False

```
[26] True
```

```
↳ True
```

```
[27] type(False)
```

```
↳ bool
```

```
[28] 1 == 1
```

```
↳ True
```

```
[29] 1 > 2
```

```
↳ False
```

→ 비교 연산자와 조건문의 반환값으로 불자료형이 사용 됨

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

7. 불(bool)

- bool : 참과 거짓을 나타내는 자료형

자료형에도 참/거짓이 있음

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

```
[30] bool(1)
```

```
↳ True
```

```
[31] bool('')
```

```
↳ True
```

```
[32] bool('')
```

```
↳ False
```

bool()함수를 통해 해당 자료형의 참/거짓을 확인할 수 있음

연습문제 4

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

연습문제 4

1번

국어, 영어, 수학 점수가 키로 하는 딕셔너리를 만들어보자.
각각의 점수는 다음과 같다.

국어: 87

영어: 88

수학: 92

▶ # 정답을 적어주세요

2번

다음 연산들의 값을 예측해보자

```
] ▶ # (1)  
3 <= 1
```

```
] ▶ # (2)  
6 % 3 == 0
```

```
] ▶ s1 = {'a', 'b', 'c'}  
s2 = {'b', 'e', 'f', 'g'}  
  
# (3)  
s1 & s2 == 'b'
```

```
] ▶ # (4)  
s1 & s2 == {'b'}
```

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
```

조건문

조건문

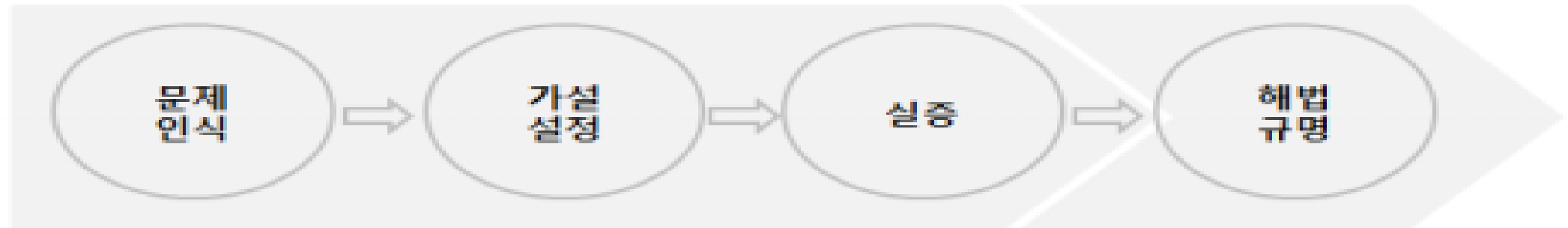
1. if와 else
2. elif
3. 조건문 만들기
4. 조건문 중첩하기
5. pass

문제해결 프로세스

과학적 문제해결 프로세스

PART1.
문제발생의 원인 규명 과정

PART2.
해법의 수립과 적용 과정

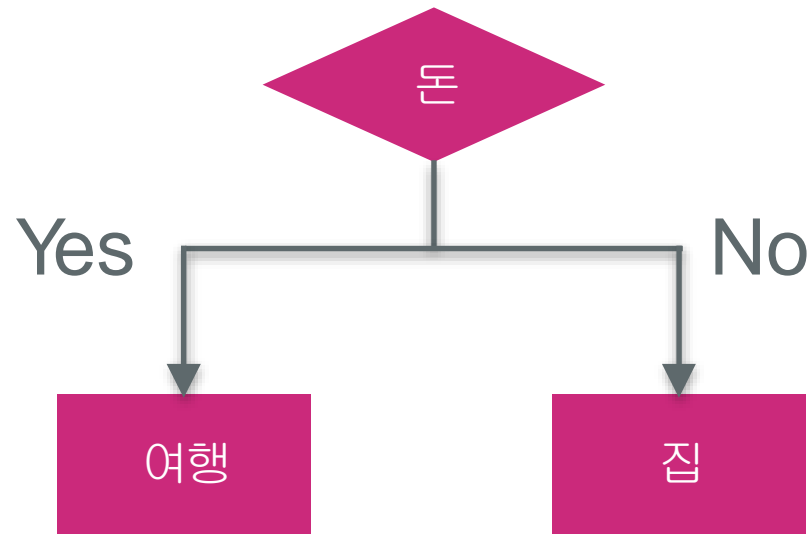


조건문이란 ?

- 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.

조건문

- 1. if / else



if 조건식:

코드

들여쓰기 or 탭

공식 4칸

```
[ ] money = True

if money :
    print("여행을 간다")
else :
    print("집에서 쉰다")
```



여행을 간다

조건문

- 1. if / else

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

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



여행을 간다

주의

콜론(:)과 들여쓰기에 주의할 것

조건문

- 1. if / elif / else

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

```
[ ] 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


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

 True

```
[ ] c = 3  
    d = 3  
  
    c <= d
```

 True

```
[ ] c != d
```

 False


조건문

- 조건문?


조건문 : 참과 거짓을 판단하게 하는 문장

2. 논리연산자 : and / or / not

```
[ ] True and True
```

 True

```
[ ] True and False
```

 False

1) and

비교하는 대상이 모두 참이어야만
True반환


조건문

- 조건문?


조건문 : 참과 거짓을 판단하게 하는 문장

2. 논리연산자: and / or / not


```
[ ] True or True
```

 True

```
[ ] True or False
```

 True

```
[ ] False or False
```

 False

2) or

비교하는 대상 중 하나만 참이여도
True 반환


조건문

- 조건문?


조건문 : 참과 거짓을 판단하게 하는 문장

2. 논리연산자: and / or / not

```
[ ] not True
```

 False

```
[ ] not False
```

 True

3) not

참이면 False를, 거짓이면 True를 반환

조건문


- 조건문?

조건문 : 참과 거짓을 판단하게 하는 문장


3. 요소인지 파악하기: in / not in

```
[ ] e = [1, 3, 5, 7]
```

```
[ ] 0 in e
```

 False


```
[ ] 1 in e
```

 True


1) **x in s**

x가 s의 요소인가

```
[ ] 2 not in e
```

 True

```
[ ] 3 not in e
```

 False

2) **x not in s**

x가 s의 요소가 아닌가

조건문

- if문 안에 if문

조건 안에 조건을 만들 수 있음

```
[ ] money = 20000  
    card = True
```

```
[ ] 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?

```
[ ] money = 1000  
    card = False
```

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

연습문제 5-1

b로 a를 나눈 나머지가 3 초과면 실패, 3이면 무승부, 3 미만이면 성공이 출력되도록 만들어 보자

```
▶ a = 34  
b = 4
```

```
▶ # 정답을 적어주세요
```

성공

연습문제 5-2

사용자로 점수를 3개 입력받아

**모든 점수가 65점보다 클 경우 합격 아닐경우 불합격을 출력하
세요**

**단, 0~100 이 아닌 숫자가 입력된경우 잘못된 "잘못된 점수가
입력되었습니다" 를 출력하세요**

연습문제 5-3

[홀수 짝수 판별기]

사용자로부터 정수를 하나 입력받아
입력한 정수가 홀수인지 짝수인지 판별하여라.
(** 0은 짝수라 하자.)

[출력결과]

정수를 입력해주세요: 5
입력하신 5는 홀수입니다.

짝수를 입력해주세요: 10
입력하신 10은 짝수입니다.

반복문

반복문

- while 문
- for 문

반복문 - while

조건문이 참인 동안 반복해서 문장을 수행함

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

```
[ ] # 10이하의 짝수 프린트하기
```

```
i = 1
```

→ 조건문에 쓰일 변수

```
while i <= 10 :
```

→ 조건문

```
    if i % 2 == 0 :
```

```
        print(i)
```

```
    i += 1
```

→ 변수를 증감



2

4

6

8

10

반복문 - while

- break

반복문에서 빠져나오고 싶다면?

```
[ ] # 100번째 방문자 찾기  
i = 90
```

```
while i :  
    i += 1  
    if i == 100 :  
        print("축하합니다. %d번째 방문자입니다." % i)  
        break  
print("감사합니다. 이벤트가 종료되었습니다.")
```

→ 변수는 0이 아니면 항상 참 (무한 반복)

→ 반복문이 종료되었으므로 수행됨



```
축하합니다. 100번째 방문자입니다.  
감사합니다. 이벤트가 종료되었습니다.
```

break를 사용하면 반복문이 더 이상 작동하지 않고 멈춤

반복문 - while

- continue

반복문의 첫부분으로 돌아가고 싶다면?

```
[ ] i = 0

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



```
2
4
8
10
```

→ 6일 때는 **continue**를 사용해 반복문의 가장 처음으로 가도록 함

→ 6일 때는 짝수임에도 불구하고 프린트가 안됨

반복문 - for

- for문

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

주의

콜론(:)과 들여쓰기에 주의할 것

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




0
1
2
3
4

반복문 - for

- for문

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



0
1
2
3
4



변수의 범위 지정하기

- 1) 숫자로 범워주기
- 2) 자료형으로 범워주기

반복문 - for

- 변수의 범위 지정하기

1) 숫자로 범위주기: range()

range(시작, 끝, 간격)

- 시작부터 끝 - 1 까지
- 간격이 1이라면 생략가능
- 0부터 시작하여 1씩 증가한다면 시작과 간격 생략가능

반복문 - for

- 변수의 범위 지정하기

1) 숫자로 범위주기: range()

연습문제 7! 5에서부터 0까지 카운트 다운을 세어 볼까요?

```
[ ] # 카운트 다운
```

```
for count in range(5, -1, -1):  
    print(count)
```

→ 하나씩 작아지므로 간격은 -1

→ 0까지 출력해야하므로 -1이 끝 인덱스가 되어야 함




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

반복문 - for

- 변수의 범위 지정하기


2) 자료형으로 범위

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



H
e
l
l
o
!

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

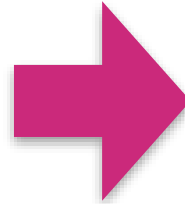


2
4
6
8

반복문 - for

- 이중으로 반복문을 사용하기

```
[ ] # 구구단 2단과 3단을 출력하기  
  
[5] for i in range(2,4) :  
    print('===', i, '단 ===')  
    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
```

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

0 1 2 19

continue

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

결과

1 3 5 7 29

Toy example

- 1부터 10까지 합을 구하시오. (range 함수를 이용하시오)

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

**

*

연습문제 3

- (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

정수를 입력해주세요: 5

$5 \times 1 = 5$

$5 \times 2 = 10$

$5 \times 3 = 15$

...

...

연습문제4

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

연습문제 5

1-100 까지 숫자중

3과 5의 공배수일 경우 “3과 5의 공배수”

나머지 숫자중 3의배수일 경우 “3의배수”

나머지 숫자중 5의배수일 경우 “5의배수”

모두 해당되지 않을 경우 그냥숫자

를 출력하세요

심화 : 1-입력한숫자까지의 숫자중

연습문제 6

사용자로부터 숫자를 계속 입력받다가
s or S 를 입력하면 합계출력

값을 입력해주세요30

값을 입력해주세요20

값을 입력해주세요50

값을 입력해주세요40

값을 입력해주세요30

값을 입력해주세요s

합계는 ? 170

파일

파일

- 파일 쓰기
- 파일 읽기

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)

```
[ ] new_file = open('/content/drive/My Drive/Colab Notebooks/python/toy_data/new.txt', 'w')
```

```
[ ] print(new_file)
```

```
[ ] <_io.TextIOWrapper name='./drive/My Drive/Colab Notebooks/day2/toy_data/new.txt' mode='w' encoding='UTF-8'>
```

*** 주의 *** open으로 연 파일은 파일객체를 반환함

파일 쓰기

- open

- 1. file path

- file path(파일 경로)는 string
 - 파일은 두 가지 방식으로 표현할 수 있음
 - 1) 절대 경로 : 파일이 가지고 있는 고유한 경로
 - 2) 상대 경로 : 현재 내 위치에서부터 해당 파일까지의 경로

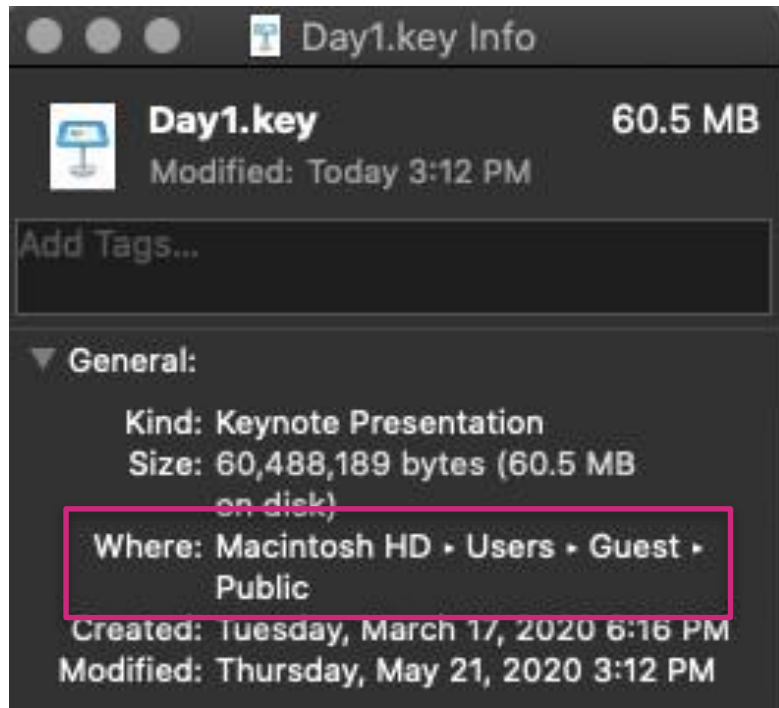
파일 쓰기

- open

1. file path

1) 절대 경로 : 파일이 가지고 있는 고유한 경로

현재 위치와 무관하게 항상 해당 파일에 접근 가



→ 'Users/Guest/Public/Day1.key'

파일 쓰기

- open

1. file path

- 2) 상대 경로 : 지금 내가 작업하고 있는 위치에서부터 해당 파일까지의 경로
내 위치에 따라 달라질 수 있음

| | |
|--------------|-----|
| 현재 파일의 위치 | ./ |
| 현재 파일의 상위 폴더 | ../ |

파일 쓰기

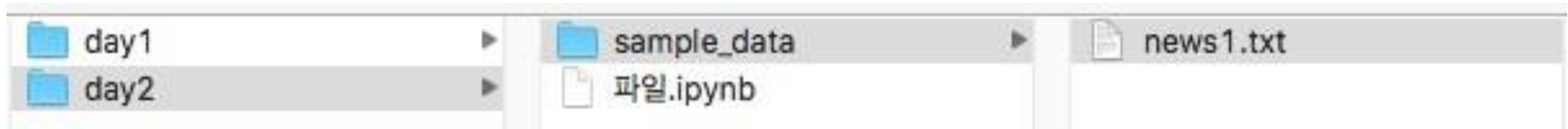
- open

1. file path

2) 상대 경로 : 지금 내가 작업하고 있는 위치에서부터 해당 파일까지의 경로
내 위치에 따라 달라질 수 있음

| | |
|--------------|-----|
| 현재 파일의 위치 | ./ |
| 현재 파일의 상위 폴더 | ../ |

Quiz. 현재 '파일.ipynb'에서 작업 중이다. 'new1.txt' 파일을 열기 위해 필요한 상대 경로는?



→ './sample_data/news1.txt'

파일 쓰기

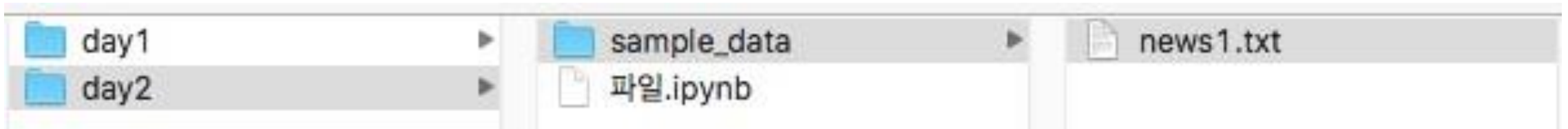
- open

1. file path

2) 상대 경로 : 지금 내가 작업하고 있는 위치에서부터 해당 파일까지의 경로
내 위치에 따라 달라질 수 있음

| | |
|--------------|-----|
| 현재 파일의 위치 | ./ |
| 현재 파일의 상위 폴더 | ../ |

Quiz. 현재 '파일.ipynb'에서 작업 중이다. 'day1' 디렉토리(폴더)에 있는 'quiz.txt' 파일을 열기 위해 필요한 상대 경로는?



→ '../day1/quiz.txt'

파일 쓰기

- open

1. file path

- file path(파일 경로)는 **string**
- 윈도우 환경에서 파일 열 때 주의사항
윈도우 환경에서는 'U'가 파일 경로에 포함된 경우 안 열리는 경우
(유니코 드 에러)가 존재함.
→이럴 땐, **r**'파일경로'를 사용하거나 ****를 **두번씩** 써주면 됨.

경로 : 'Users/Guest/Public/Day1.key'

→ **r'Users/Guest/Public/Day1.key'**

→ **'Users\\Guest\\Public\\Day1.key'**

파일 쓰기

- open

2. mode

| 파일 모드 | 설명 | 비고 |
|-------|------------|----------------------------------|
| r | 읽기 모드 | 파일에 새로운 내용을 쓰거나 수정이 불가능 |
| 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

- CP949 : 현재 윈도우 커널에는 유니코드가 적용되었지만, 한글 윈도우의 명령 프롬프트가 사용하는 기본 인코딩은 여전히 **CP949**

파일 쓰기

- write

파일.write(내용)

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

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

```
[7] new_file.write(data)
```

```
↳ 29
```

→파일 안의 문자의 갯수를 반환

*** 주의 *** 파일을 열고, 작업을 마쳤다면 반드시 닫을 것

```
[8] new_file.close()
```

파일 쓰기

- with

with open(file_path, mode=) **as** 변수 :

수행할 문장 1

수행할 문장 2

- with문을 사용하면 with블록을 벗어나는 순간 파일을 자동으로 close해 줌
- **as** 변수: open으로 연 파일 객체를 변수로 받아줌

```
[ ] with open('/content/drive/My Drive/Colab Notebooks/python/toy_data/new.txt', 'w') as f :  
    data = '이렇게도 작성이 가능합니다. close를 따로 안해도 되요.'  
    f.write(data)
```

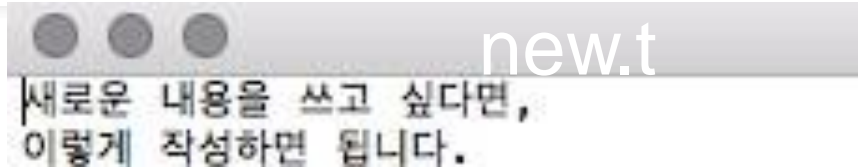
파일 읽기

- read / readline / readlines

| 함수 | 설명 |
|-----------|------------------------------------|
| read | 파일 전체를 한번에 읽어서 string 으로 반환 |
| readline | 파일을 한 줄만 읽어서 string 으로 반환 |
| readlines | 파일을 줄단위로 모두 읽어서 list 로 반환 |

파일 읽기

- read / readline / readlines



1. read

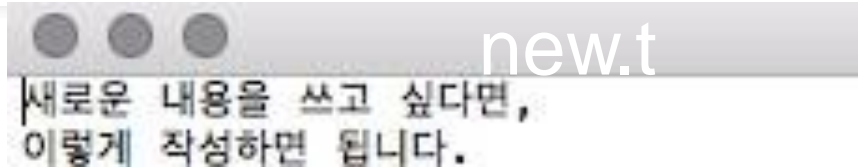
```
[ ] with open('/content/drive/My Drive/Colab Notebooks/python/toy_data/new.txt', 'r') as f:  
    lines = f.read()  
    print(lines)  
    print(type(lines))
```

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

→ 파일 전체를 **string**으로 불러옴

파일 읽기

- read / readline / readlines



2. readline

```
[ ] with open('/content/drive/My Drive/Colab Notebooks/python/toy_data/new.txt', 'r') as f:
    line = f.readline()
    print(line)
    print(type(line))
```

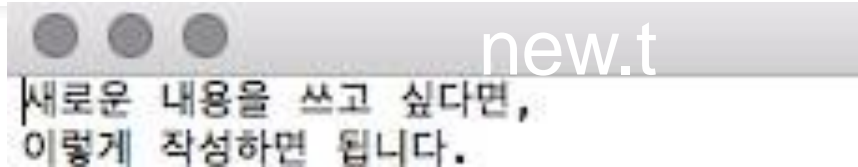
↳ 새로운 내용을 쓰고 싶다면,

<class 'str'>

→ 파일에서 한 줄만 string으로 불러옴

파일 읽기

- read / readline / readlines



2. readline

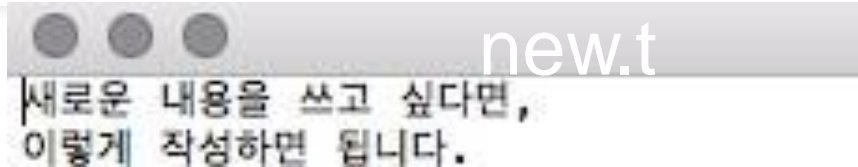
(참고) readline + while문으로 전체파일을 불러올 수 있음

```
[ ] with open('/content/drive/My Drive/Colab Notebooks/python/toy_data/new.txt', 'r') as f:
    while True :
        line = f.readline()
        if not line : break
        print(line)
```

↳ 새로운 내용을 쓰고 싶다면,
이렇게 작성하면 됩니다.

파일 읽기

- read / readline / readlines



2. readlines

```
[ ] with open('/content/drive/My Drive/Colab Notebooks/python/toy_data/new.txt', 'r') as f:
    lines = f.readlines()
    print(lines)
    print(type(lines))
```

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

→ 파일에서 줄단위로 읽어서 list로 불러옴

→ 개행문자(\n)가 남아 있음

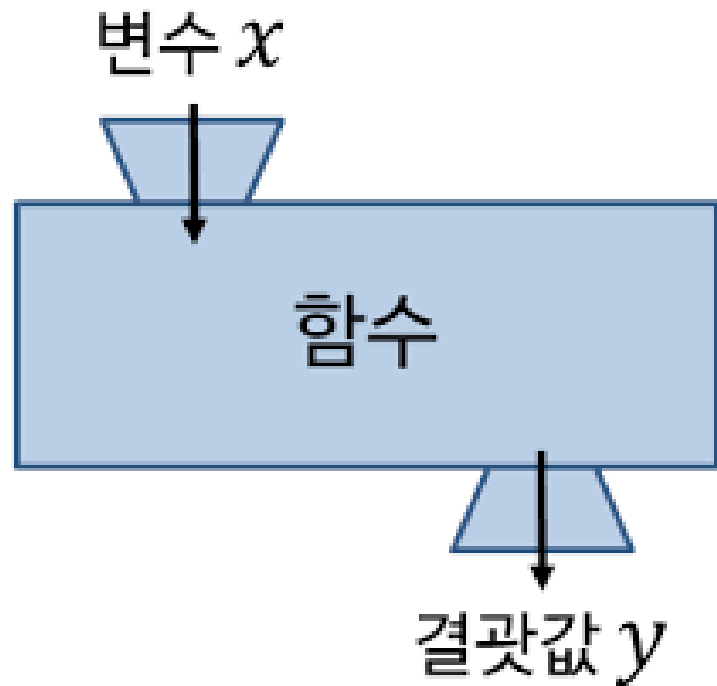
함수(Function)

함수

- 함수란?
- 함수 정의하기
- 함수와 변수

함수(Function)

- 함수란?



```
[ ] def 함수이름 (함수인자) :  
    수행할 문장1  
    수행할 문장2  
    return 반환값
```

일정한 프로세스를 할 수 있도록 하는 것

함수(Function)

- 함수 정의하기

```
def 함수이름(매개변수):  
    함수의 내용  
    return 반환값
```

- **함수이름**: 사용자가 정의하는 함수 이름, 기존에 사용되는 함수나 예약어 들을 제외하고 사용
- **매개변수**: 함수 안에서 사용 할 변수들(생략 가능)
- **return**: 함수 안에서 모든 연산을 마친 후 반환할 값(생략 가능)

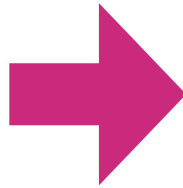
함수(Function)

- 함수 정의하기

pop quiz! 두 개의 input를 받으면 더하기를 하여 값을 돌려주는 함수를 만들어봅시다.

```
[ ] def add(a, b) :  
    result = a + b  
    return result
```

→ 함수 정의



```
[ ] c = add(3, 5)  
c
```

 8

→ 함수 호출

함수(Function)

- 함수 정의하기

1. 매개변수와 return값이 모두 있는 함수

```
[ ] def add(a, b) :  
    result = a + b  
    return result
```

```
[ ] c = add(3, 5)  
    c
```



8

함수(Function)

- 함수 정의하기

2. return값이 비어있는 함수

```
[ ] def sub(a, b) :  
    print('뽕섬의 결과는 %d입니다.' % (a-b))  
    return
```

```
[ ] d = sub(1, 2)
```

 뽕섬의 결과는 -1입니다.

```
[ ] print(d)
```

 None

함수(Function)

- 함수 정의하기

3. return이 없는 함수

```
[ ] def mul(a, b) :  
    print('%d 와 %d의 곱은 %d입니다.' % (a, b, a*b))
```

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



3 와 2의 곱은 6입니다.

함수(Function)

- 함수 정의하기

4. 매개변수와 return값이 모두 없는 함수

```
[ ] def start():  
    print("Hello World!")
```

```
[ ] start()
```



Hello World!

연습문제

함수를 만들어 보자.

- 이 함수는 두 개의 숫자를 input으로 받으면 앞의 숫자를 뒤의 숫자로 나누는 함수이다.
- 나눗셈을 한 후에는 몫과 나머지 순으로 된 튜플 값을 반환한다.

```
def div(a, b):  
    # 정답을 적어주세요
```

```
div(10, 3)
```

```
(3, 1)
```

```
div(3, 5)
```

```
(0, 3)
```

```
div(1, 10)
```

```
(0, 1)
```

연습문제

함수를 만들어보자.

- 이 함수는 두 개의 숫자를 input으로 받으면 작은 수로 큰 수를 나눈 몫과 나머지를 반환하는 함수이다.
- 반환 값은 튜플로 되어 있으며 몫, 나머지 순으로 되어 있다.
- 단, 0으로 나누는 것은 불가하기 때문에 두 수 중에 작은 수가 0이라면 화면에 '0은 사용할 수 없습니다.'를 출력하고 종료되어야한다.

```
div2(3, 5)
```

```
(1, 2)
```

```
div2(0, 5)
```

```
0은 사용할 수 없습니다.
```

```
div2(10, 1)
```

```
(10, 0)
```

함수(Function)

- 더 알아보기

1. 매개변수의 초기값 미리 설정하기

```
[ ] def function(a, n=2) :  
    print("%d의 제곱은 %d 입니다." % (a, a**n))
```

```
[ ] function(4)
```



4의 제곱은 16 입니다.

→ 초기값을 설정하는 경우, 매개변수의 순서에 주의!
초기값 없는 변수, 초기값 있는 변수 순서로 배치

함수(Function)

- 더 알아보기

2. 매개변수가 몇 개가 필요할지 모를 때

```
[3] def test(*args) :  
    print(args)
```

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

```
↳ (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
```

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

```
↳ ([1, 2], 3)
```

→ 매개변수에 `*args`를 붙이면 }
함수가 받은 input 모두를 tuple로
묶어줌

Quiz! 왜 tuple로 묶어줄까요?

함수(Function)

- 더 알아보기

2. 매개변수가 몇 개가 필요할지 모를 때

```
def add(**kwargs):  
    print(kwargs)
```

```
add(a =1, b=2, c=3)
```

```
{ 'a' : 1, 'b' : 2, 'c' : 3 }
```

→ 매개변수에 *kwargs를 붙이면
함수가 받은 input 모두를 dict로
묶어줌

Quiz! 왜 dict로 묶어줄까요?

연습문제

어떠한 string을 받으면 일정한 단위로 끊어서 화면에 출력하는 함수를 짜보자.
끊는 단위는 따로 정하지 않으면 2로 설정해보자.

Hint : input을 string 과 unit = 2 로 받고, while을 사용하고, 길이는 len 함수를 사용하도록 하자.

```
: func('테스트를 위한 문장입니다.')
```

```
테스  
트를  
위  
한  
문장  
입니  
다.
```

```
: func('테스트를 위한 문장입니다.', 4)
```

```
테스트를  
위한  
문장입니  
다.
```

연습문제

`add_all` 함수를 짜봅시다

```
add_all(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
```

55

Hint : *args를 input으로 받으세요.

함수(Function)

- 함수와 변수

Quiz! 함수 안에 선언한 변수를 함수 밖에서도 사용할 수 있을까?

NO!

함수 안의 블록은 함수 고유의 영역임.

따라서, 함수 안에서 선언한 변수는 함수 밖에서 사용할 수 없음.

함수(Function)

- 함수와 변수

예시 1)

```
[ ] def myfunction() :  
    in_val = '함수 안의 변수'
```

```
[ ] myfunction()
```

```
[ ] in_val
```



```
-----  
NameError                                Traceback (most recent call last)  
<ipython-input-30-a23adeea90d3> in <module>  
----> 1 in_val  
  
NameError: name 'in_val' is not defined
```

함수(Function)

- 함수와 변수

예시 2)

```
[ ] a = 1
```

```
[ ] def myfunction2(a) :  
    a += 1  
    print(a)
```

```
[ ] myfunction2(a)
```

 2

```
[ ] a
```

 1

함수 실행 시 출력값은 무엇일까요?

a는 어떤 값일까요?

함수(Function)

- 함수와 변수

함수 안의 블록은 함수 고유의 영역임.

따라서, 함수 안에서 선언한 변수는 함수 밖에서 사용할 수 없음.

```
[ ] def 함수이름(함수인자) :  
    수행할 문장1  
    수행할 문장2  
    return 반환값
```

함수(Function)

2. global 사용하기 ← 권장하지 않음

- 함수와 변수
- 함수 안에 있는 변수를 밖에서도 쓰고 싶다면?
 1. return 사용하기

함수(Function)

- 함수와 변수

함수 안에 있는 변수를 밖에서도 쓰고 싶다면?

1. return 사용하기

```
a = 1  
def myfunction3(a) :  
    a += 1  
    print(a)  
    return a
```

함수(Function)

- 함수와 변수

함수 안에 있는 변수를 밖에서도 쓰고 싶다면?

2. global 사용하기

```
[ ] a = 1
```

```
[ ] def myfunction4():  
    global a  
    a += 1  
    print(a)
```

```
[ ] myfunction4()
```

 2

```
[ ] a
```

 2

함수(Function)

- 함수와 변수

return 사용하기

```
[21] for _ in range(5) :  
      c = myfunction3(a)  
      print('함수의 결과 : %d' %c )  
      print('a의 값 : %d' %a)
```

```
↳ 3  
   함수의 결과 : 3  
   a의 값 : 2  
   3  
   함수의 결과 : 3  
   a의 값 : 2  
   3  
   함수의 결과 : 3  
   a의 값 : 2  
   3  
   함수의 결과 : 3  
   a의 값 : 2  
   3  
   함수의 결과 : 3  
   a의 값 : 2
```

global 사용하기

```
[25] for _ in range(5) :  
      myfunction4()  
      print('a의 값 : %d' %a)
```

```
↳ 2  
   a의 값 : 2  
   3  
   a의 값 : 3  
   4  
   a의 값 : 4  
   5  
   a의 값 : 5  
   6  
   a의 값 : 6
```

← 함수 밖의 변수(global variable)이
함수의 작동에 따라 그 값이 변함