

Python로 배우는 외부데이터 수집과 정제

[Part1] 파이썬 기초

파이썬 기초

1. 파이썬 소개
2. 파이썬 개발환경
3. 파이썬 기초
4. 파이썬 자료구조

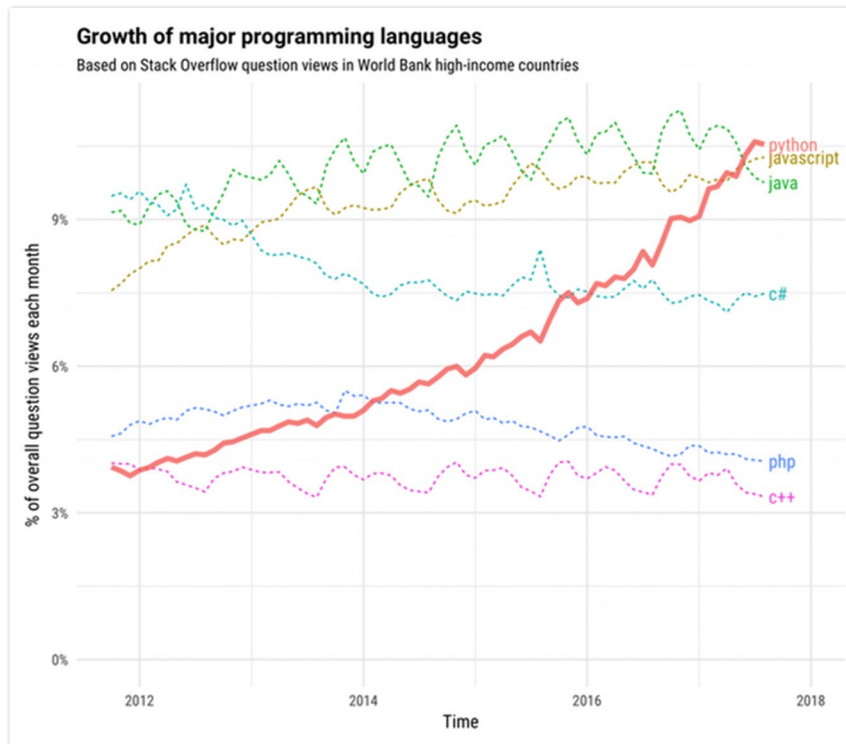
파이썬 소개

- 1991년 Guido Van Rossum*이 발표한 프로그래밍 언어
 - Guido는 구글에 근무한 적이 있으며, 현재는 DropBox에서 근무중
- 단순하고 최소화된 언어, 쉬운 문법 체계, 가독성, 간결한 코드
- FLOSS (Free/Libre and Open Source Software)
- 메모리 관리 등이 불필요한 고수준 언어
- 멀티 패러다임 언어(절차형, 객체지향, 함수형 모두 지원)
- 인터프리터 언어(jupyter와 같은 대화형 환경 구성)
- 유니코드 지원(한글변수 사용 가능)
- 동적 타이핑(실행 시간에 자료형을 검사) 언어
- 방대한 규모의 라이브러리(2020. 6월 현재 242,825 projects, 1,922,208 releases)



파이썬 소개

- 파이썬 언어 습득을 위한 진입장벽이 높지 않아 파이썬 언어 사용자는 매년 증가하는 추세
- 머신러닝/딥러닝 분야에서 많이 활용



출처 : <https://data-flair.training/blogs/why-should-i-learn-python/>

출처 : <https://www.quora.com/What-are-the-programming-languages-most-popular-in-2019>

파이썬 개발환경



Python : <https://www.python.org/downloads/>
Python 기본 개발환경



Anaconda : <https://www.anaconda.com/distribution/>
Python 기본 개발환경과 여러 패키지들을 쉽게 설치/관리할 수 있는 배포판
Jupyter notebook, Jupyter lab을 이용한 대화형 개발/분석 도구 제공



Pycharm : <https://www.jetbrains.com/pycharm>
디버깅 등의 기능을 제공하는 통합 IDE 개발환경
Community 버전은 Free, Open Source로 제공



Google colab : <https://colab.research.google.com>
설치가 필요 없으며 완전히 클라우드에서 실행되는 Jupyter notebook과 유사한 환경을 통해 코드를 작성 및 실행, 강력한 컴퓨팅 리소스를 이용

Anaconda 설치

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



Products ▾

Pricing

Solutions ▾

Resources ▾

Blog

Company ▾

Get Started



Individual Edition

Your data science toolkit

With over 20 million users worldwide, the open-source Individual Edition (Distribution) is the easiest way to perform Python/R data science and machine learning on a single machine. Developed for solo practitioners, it is the toolkit that equips you to work with thousands of open-source packages and libraries.

클릭

Download

Anaconda 설치

Anaconda Installers

Windows 

Python 3.8

64-Bit Graphical Installer (466 MB)

32-Bit Graphical Installer (397 MB)


클릭

MacOS 

Python 3.8

64-Bit Graphical Installer (462 MB)

64-Bit Command Line Installer (454 MB)

Linux 

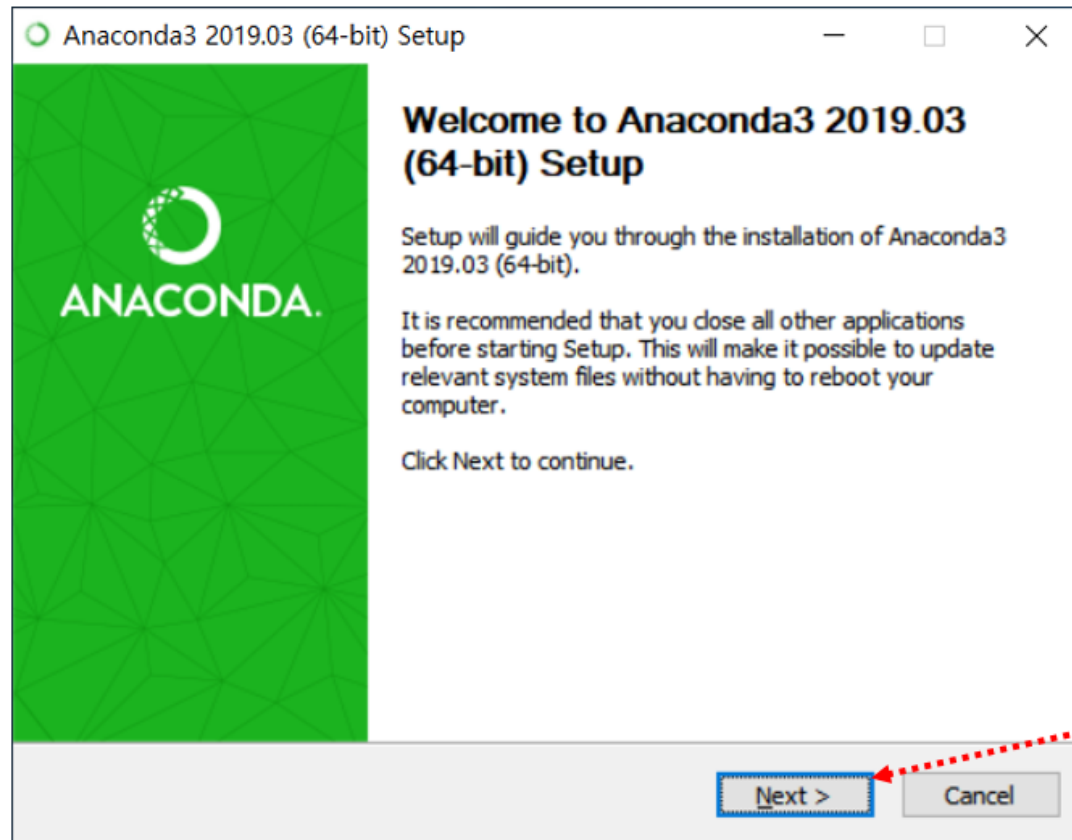
Python 3.8

64-Bit (x86) Installer (550 MB)

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

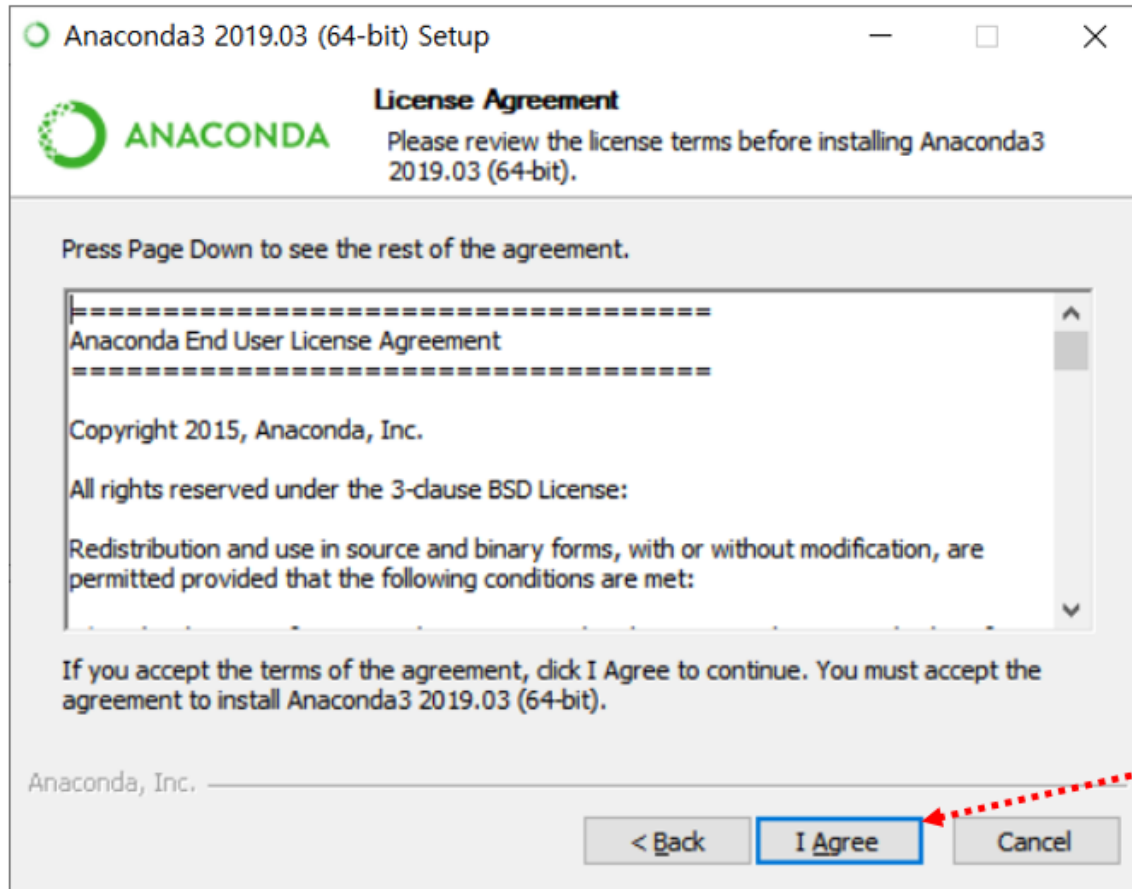
Anaconda 설치

PC에 다운로드된 아나콘다 설치 파일 실행 후 Next 버튼 클릭



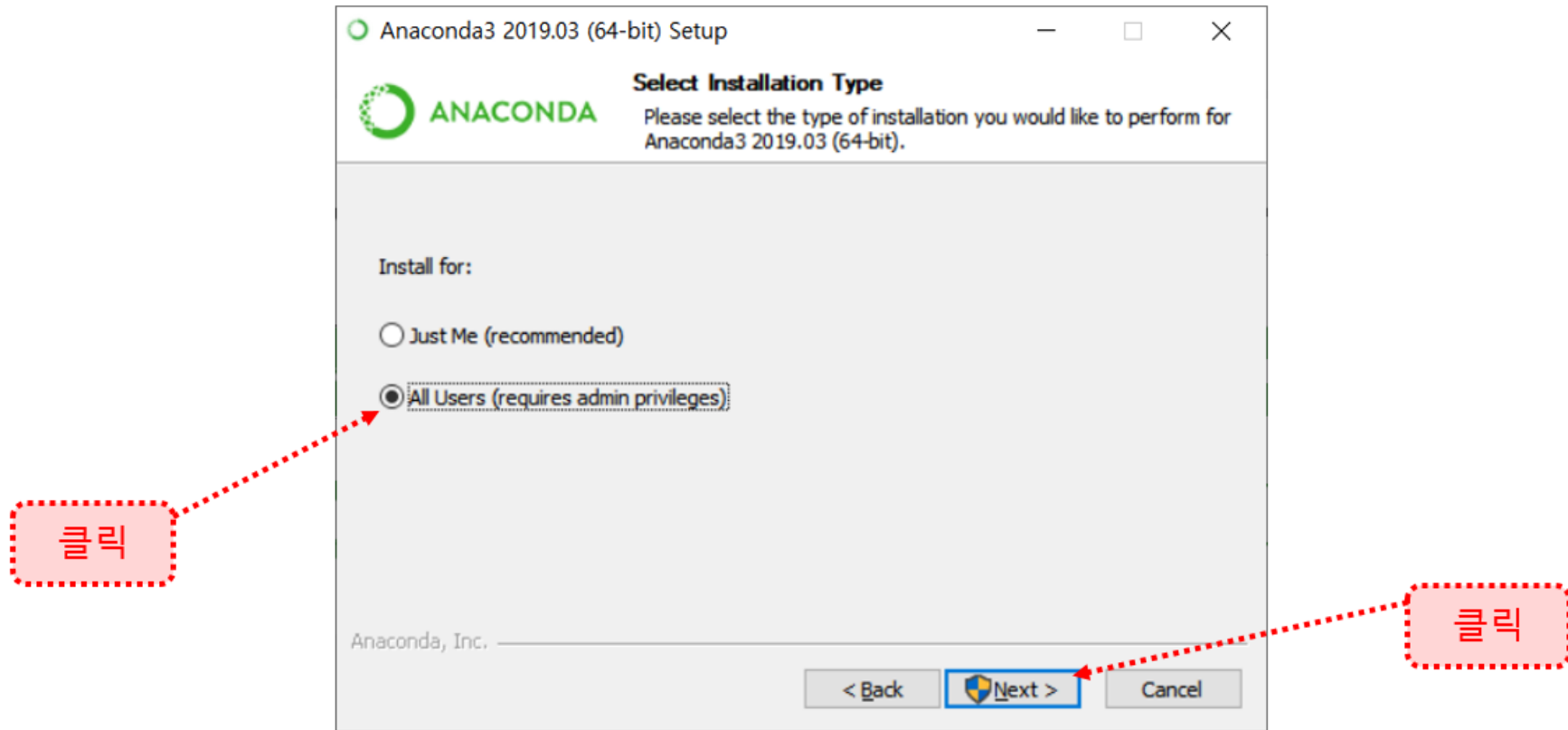
Anaconda 설치

아나콘다 License Agreement 동의 화면에서 I Agree 버튼 클릭



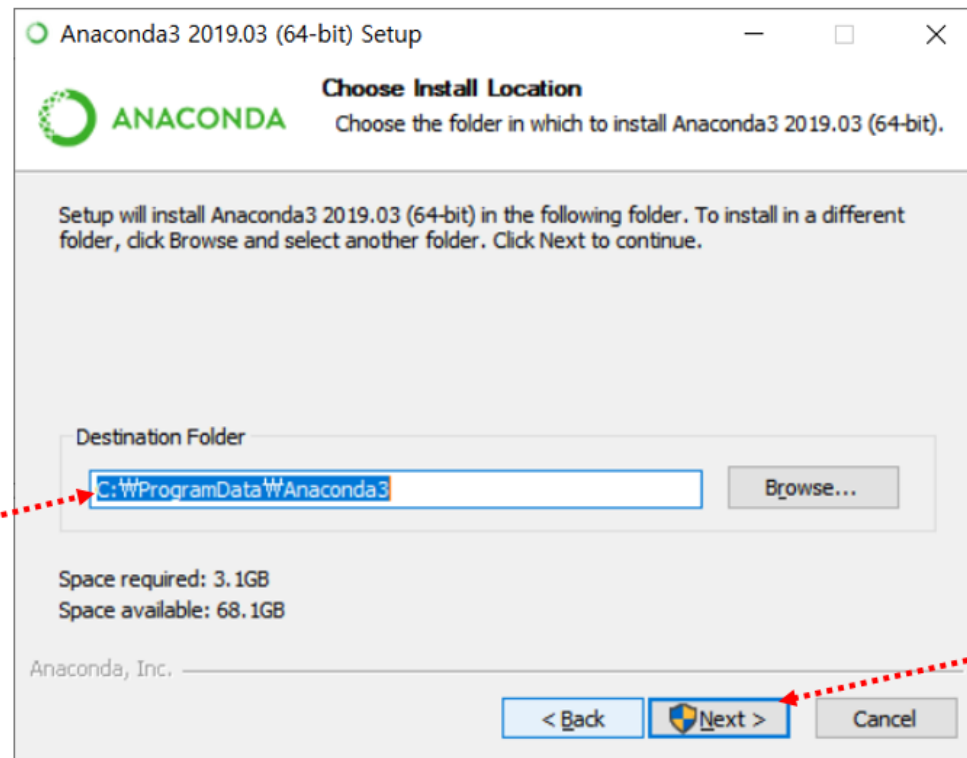
Anaconda 설치

All Users를 선택한 후 Next 버튼 클릭



Anaconda 설치

설치디렉토리를 확인한 후 Next 버튼 클릭

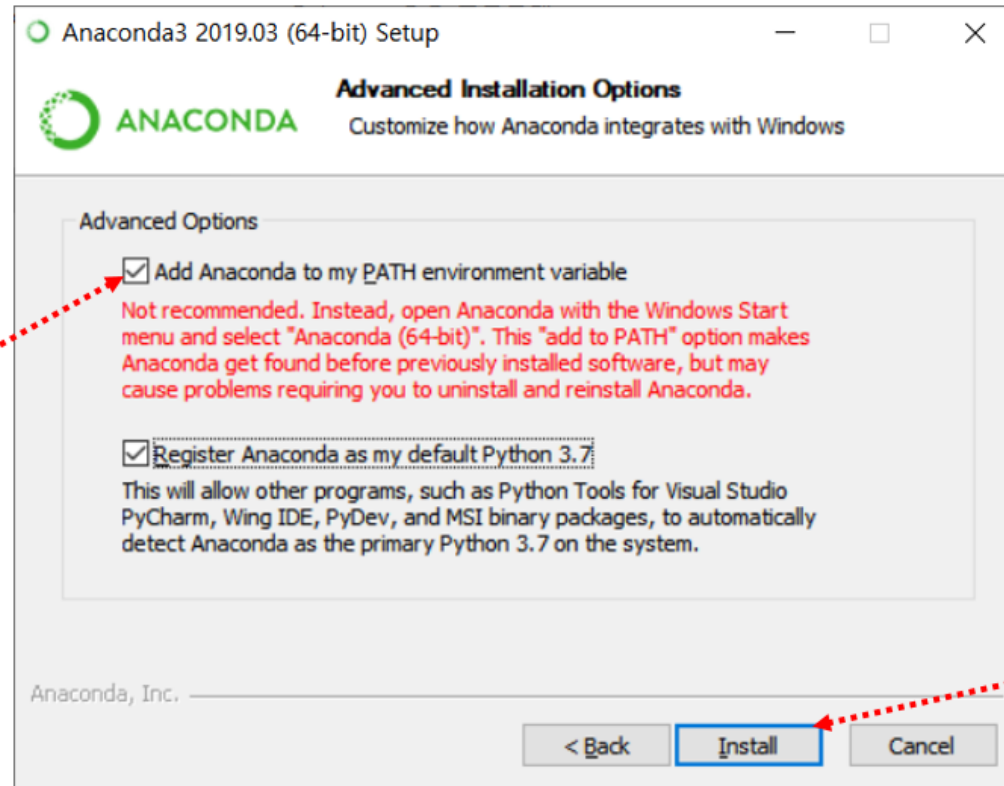


설치 디렉토리 확인

클릭

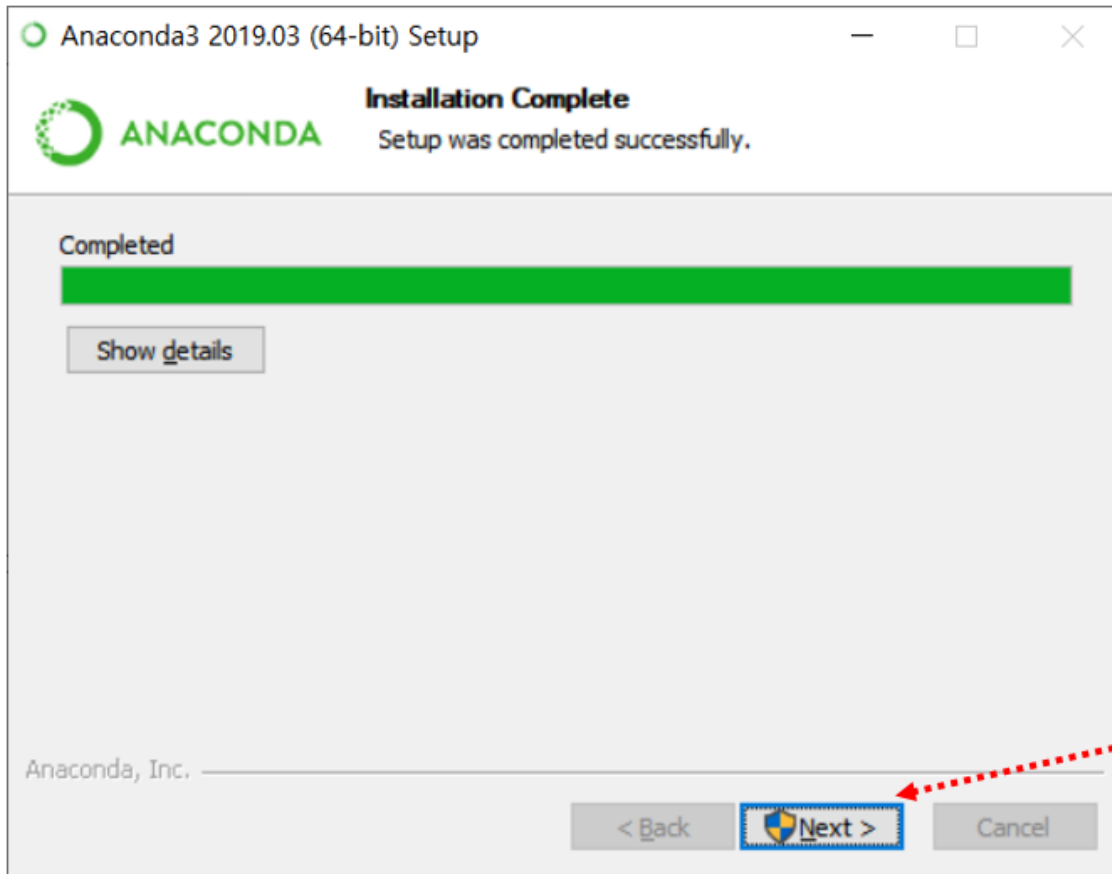
Anaconda 설치

Windows 운영체제 환경변수에 python.exe의 path 추가를 위해 체크박스 체크 후 Next 버튼 클릭



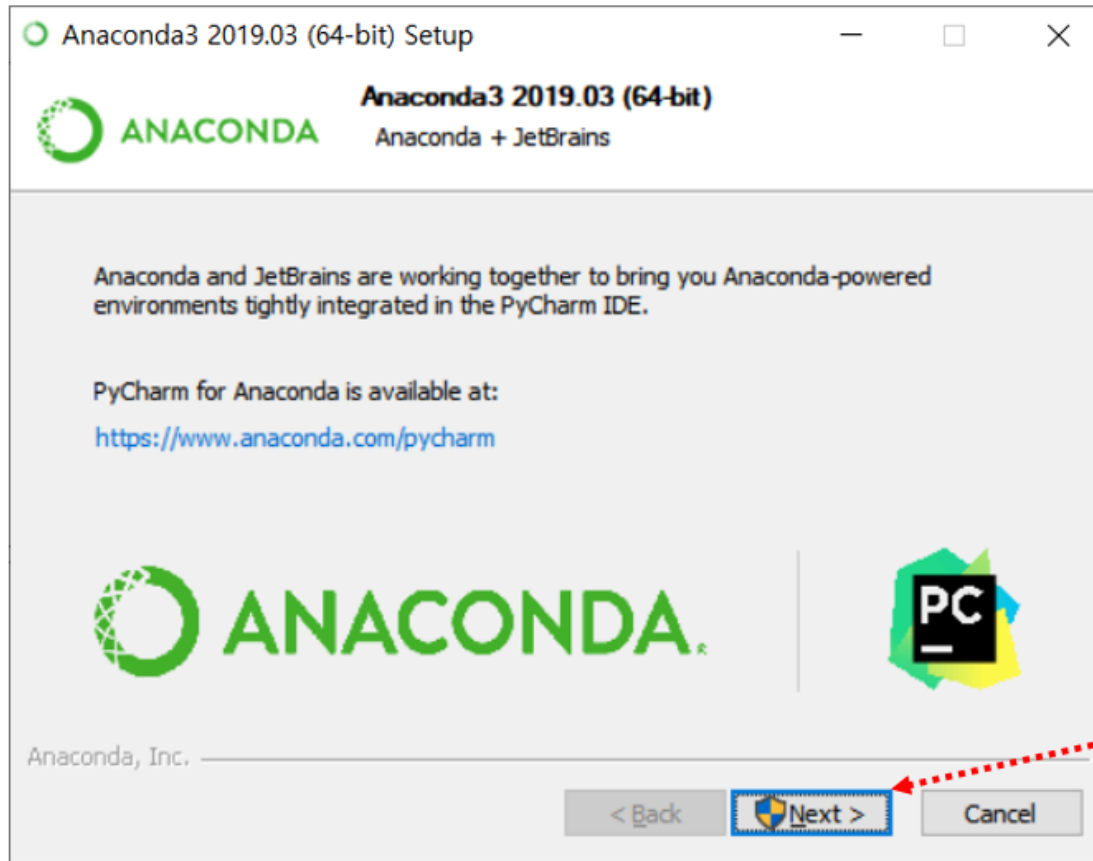
Anaconda 설치

설치 progress bar의 completed 확인 후 Next 버튼 클릭



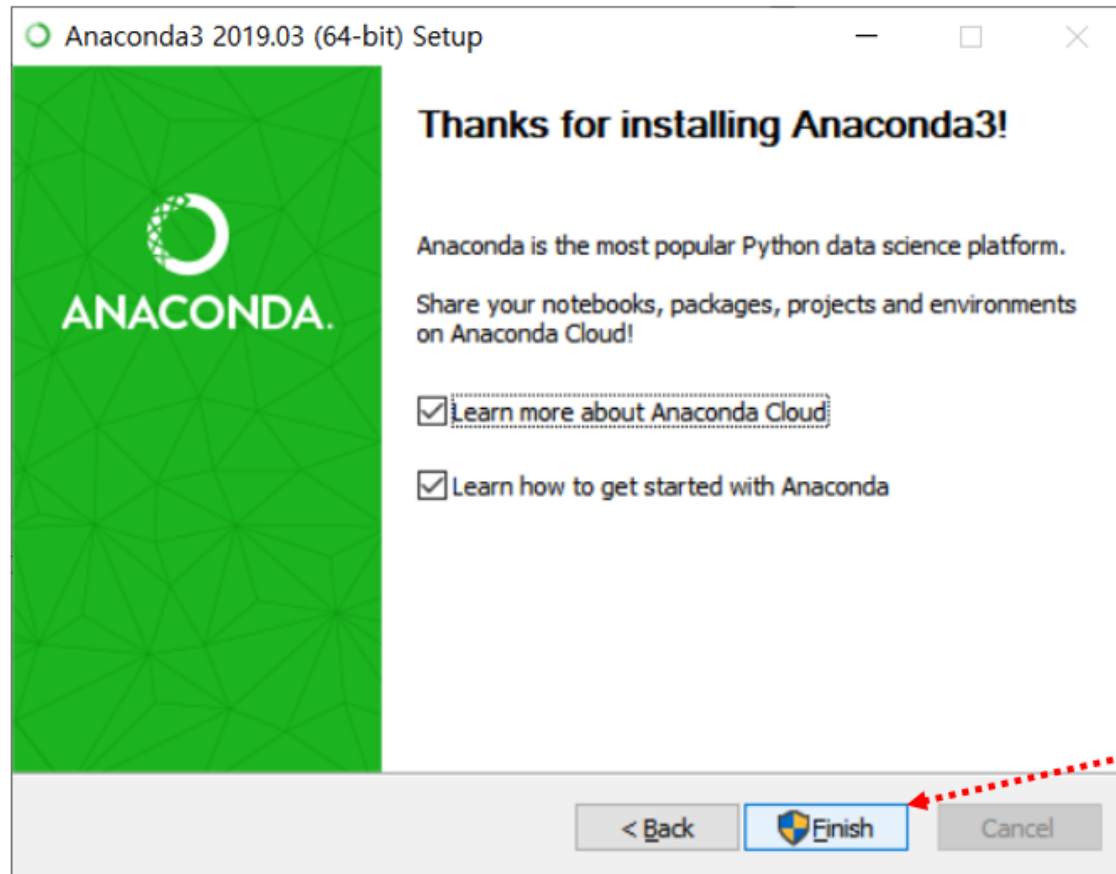
Anaconda 설치

Next 버튼 클릭



Anaconda 설치

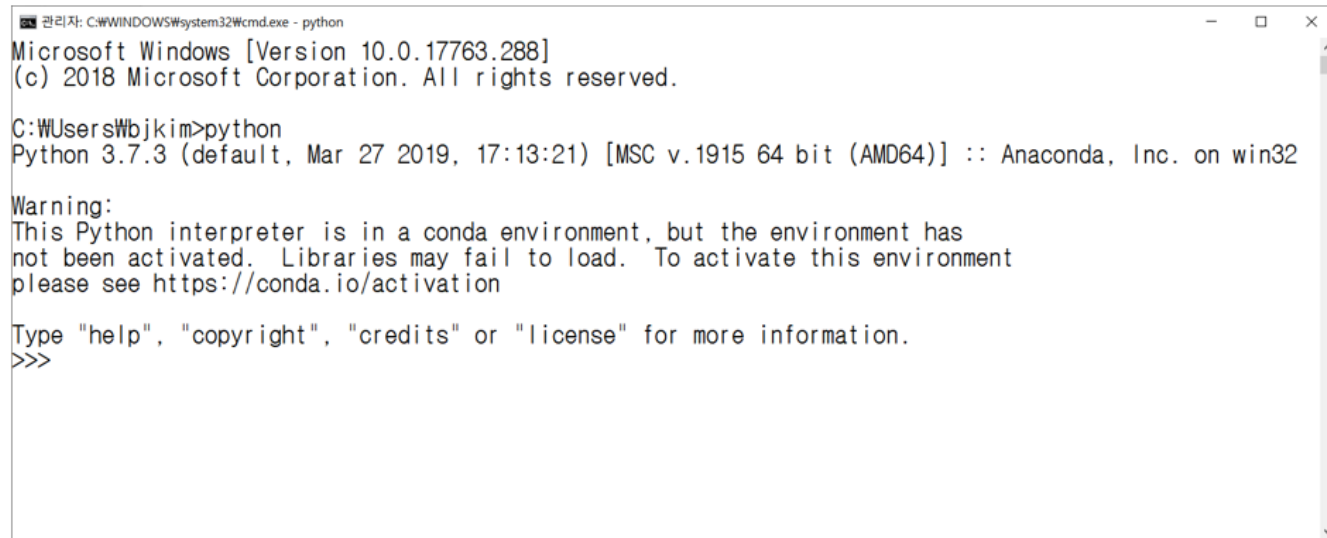
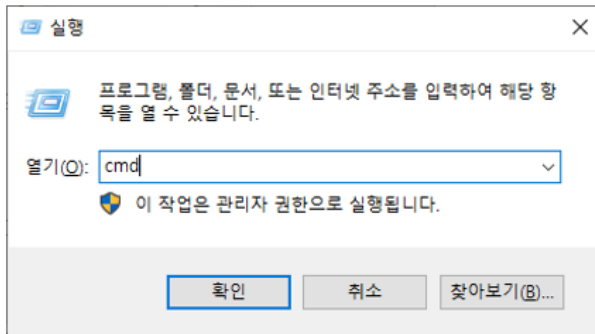
아나콘다 설치 마지막 화면으로 설치완료를 위해 Finish 버튼 클릭



클릭

Anaconda 설치

파이썬 및 아나콘다 정상 설치여부 확인을 위해 키보드에서 **Windows키 + R키**
실행창에서 **cmd** 입력 후 **Enter 키**
Command 창에서 **python** 입력 후 **Enter 키**
python 프롬프트 상태가 되면 정상 설치된 것임



Anaconda 설치 및 설치 확인 실습

파이썬 개발환경 구성

- 파일시스템 C 드라이브나 D 드라이브에 python 디렉토리 생성
- 문서편집기(notepad, notepad++ 등) 열고

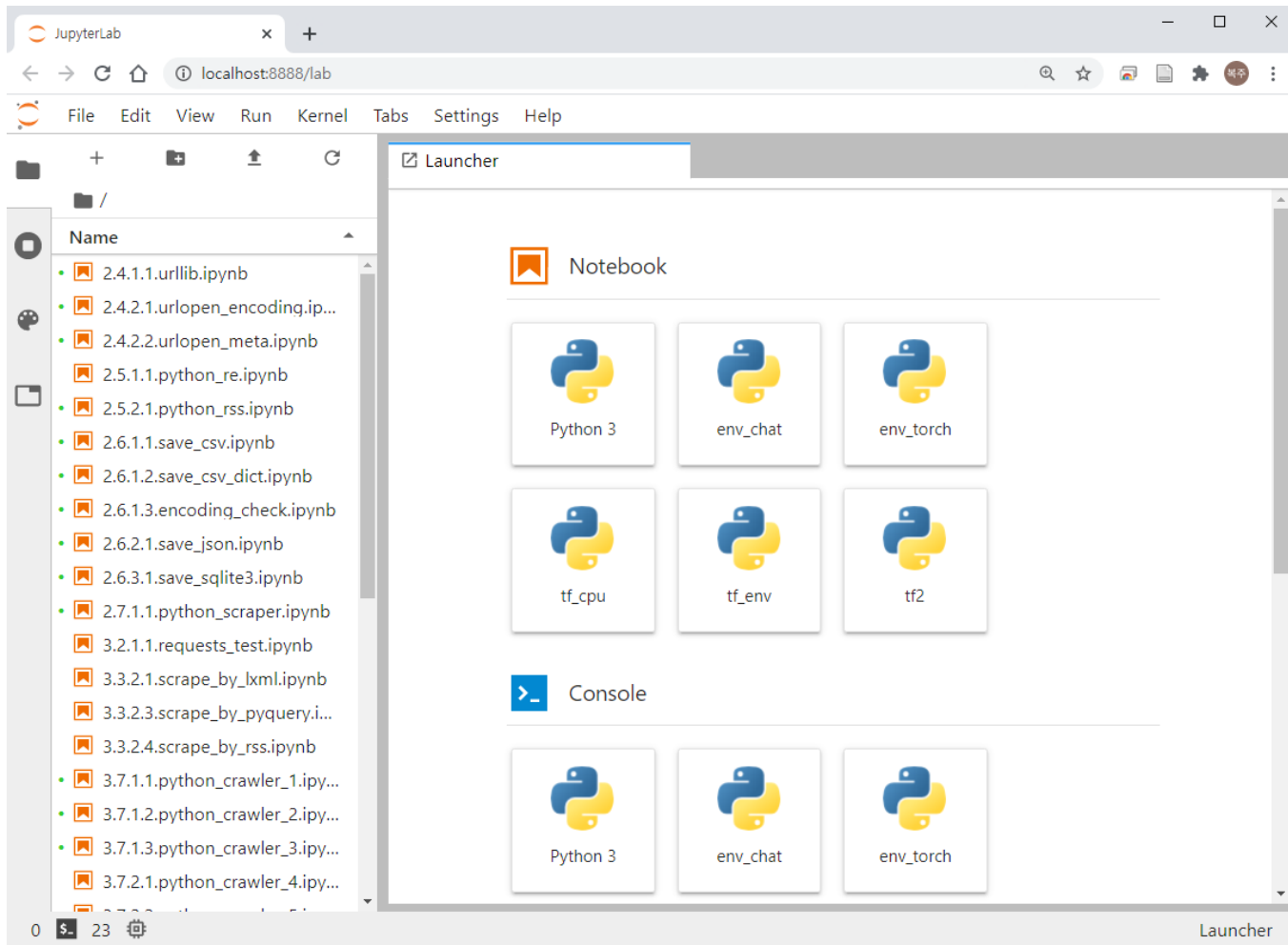
jupyter lab 입력 후 jupyter_lab.bat 파일명으로 저장

(notepad 사용 시 "jupyter_lab.bat" 와 같이 double quote 를 붙여서 저장)

- jupyter_lab.bat를 더블 클릭하여 jupyter lab 실행

파이썬 개발환경 구성 실습

Jupyter lab 사용가이드



Jupyter lab 사용 실습

Python Indentation

- 들여쓰기는 함수 몸체, 조건문 루프, 클래스 등 다양한 코드 블록을 표현
- 들여쓰기로 탭을 사용할 수 있지만 권장하지 않으며, 스페이스를 이용하여 보통 4칸을 권장

```
a = True
if a:
    print("True")
else:
    print("False")
```

```
a = True
if a:
    print("True")
else:
    print("False")
```

Code Block

- 코드 블록의 시작은 콜론(:)
- 들여쓰기를 기준으로 들여쓰기가 없는 곳이 코드 블록의 끝

```
a = True
if a:
    print("True")
    print("True End ...")
else:
    print("False")
print('End...')
```

```
// Java
boolean a = true;
if(a){
    System.out.println("True");
    System.out.println("True End...");
}else{
    System.out.println("False");
}
    System.out.println("End...");
```

Python 연산자

더하기

```
2 + 4
```

6

빼기

```
4 - 2
```

2

곱하기

```
2 * 4
```

8

나누기

```
4 / 2
```

2.0

제곱

```
4 ** 2
```

16

몫

```
7 / 2
```

3.5

정수몫

```
7 // 2
```

3

나머지

```
7 % 2
```

1

Python 주석문 및 표준 출력

- 주석처리/해제 : CTRL +

/

파이썬 주석문

```
# ctrl + /
```

```
# 파이썬 주석문
```

```
# 파이썬  
# 주석문
```

```
...  
파이썬  
블럭  
주석문  
...  
print(1)
```

- print() 함수 사용

파이썬 표준출력

```
print('Hello Python World')
```

```
Hello Python World
```

```
print('a', 1)
```

```
a 1
```

```
print('a', 1, sep=':')
```

```
a : 1
```

- * Jupyter lab 에서는 cell 변수명만 입력해도
출력 가능(제일 마지막 한 행만 유효, 코드 블럭 내 사용불가)

- 변수(Variable)는 값을 저장하기 위한 메모리 영역
- 변수를 생성은 숫자나 문자열 등과 같은 데이터를 저장할 수 있는 메모리 공간을 확보하고 해당 메모리 주소에 대한 명명
- 파이썬의 변수는 알파벳 대소문자, 숫자, under bar(_) 그리고 한글, 한자 등 사용가능
- 첫 글자는 숫자로 시작불가
- 공백이나 특수문자, 문장 부호는 변수명으로 사용 불가
- 대소문자 구별

Python 변수

```
a = 10  
b = 5
```

```
print(a, b)
```

10 5

```
한글변수 = 10
```

```
한글변수
```

10

```
a = b = 10
```

```
a
```

10

```
b
```

10

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

```
a
```

2

```
b
```

4

```
a = 10; b = 5
```

```
temp = a; a = b; b = temp
```

```
a, b = b, a
```

```
a
```

10

```
b
```

5

Python 동적 타입 & 객체

파이썬 동적 타입

```
price = 1000  
rate = 0.05  
num_years = 5
```

```
type(rate)
```

float

```
type(price)
```

int

```
price = price * (1+rate)  
price
```

1050.0

```
price
```

1050.0

파이썬 객체

```
type("a")
```

str

```
type(1)
```

int

```
a = 1
```

```
a.bit_length()
```

1

```
st = "hello"
```

Python 문자열

파이썬 문자열

```
a = '파이썬'
```

```
a
```

```
'파이썬'
```

```
b = "문자열"
```

```
c = """  
죽는 날까지 하늘을 우러러  
한 점 부끄럼 없기를  
잎새에 이는 바람에도  
나는 괴로워 했다  
"""
```

```
c
```

```
'\n죽는 날까지 하늘을 우러러 \n한 점 부끄럼 없기를\n잎새에 이는 바람에도\n나는 괴로워 했다\n'
```

Python 문자열

'파이썬' + 3

[illegible]

```
<ipython-input-16-ff25ad6d14ce> in <module>
```

----> 1 '파이썬' + 3

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

```
d = '파이썬' + str(3)
```

d

‘파이썬3’

```
e = '*' * 10
```

e

Python 문자열 포매팅

```
name, age, phone = '홍길동', 25, '010-111-2222'
```

```
소개 = "이름은 {} 이고, 나이는 {}세 이며, 전화번호는 {} 입니다.".format(name, age, phone)
```

소개

'이름은 홍길동 이고, 나이는 25세 이며, 전화번호는 010-111-2222 입니다. '

```
소개 = "이름은 {0} 이고, 나이는 {2}세 이며, 전화번호는 {1} 입니다.".format(name, phone, age)
```

소개

'이름은 홍길동 이고, 나이는 25세 이며, 전화번호는 010-111-2222 입니다. '

```
소개 = "이름은 {a} 이고, 나이는 {b}세 이며, 전화번호는 {c} 입니다.".format(c = phone, a = name, b = age)
```

소개

'이름은 홍길동 이고, 나이는 25세 이며, 전화번호는 010-111-2222 입니다. '

Python 문자열 포매팅

```
jan, dec = 1, 12  
print("한 해의 시작은 {:02d}월".format(jan))  
print("한 해의 마지막은 {:02d}월".format(dec))
```

한 해의 시작은 01월
한 해의 마지막은 12월

```
val = 123456789  
money = "{:,}"  
money.format(val)
```

'123,456,789'

```
'{}, {:.f}, {:.1f}, {:.2f}, {:.2%}'.format(3, 3, 3, 3.1475, 1/3)
```

'3, 3.000000, 3.0, 3.15, 33.33%'

Python 문자열 indexing & slicing

```
a = 'HelloWorld!'
```

H	e	l	l	o	W	o	r	l	d	!
---	---	---	---	---	---	---	---	---	---	---

[0] [1] [2] [3] [4] [5] [6] [7] [8] [9] [10]

[-11] [-10] [-9] [-8] [-7] [-6] [-5] [-4] [-3] [-2] [-1]

Python 문자열 indexing & slicing

H	e	l	l	o	W	o	r	l	d	!
---	---	---	---	---	---	---	---	---	---	---

[0] [1] [2] [3] [4] [5] [6] [7] [8] [9] [10]

[-11] [-10] [-9] [-8] [-7] [-6] [-5] [-4] [-3] [-2] [-1]

```
a[0]
```

```
'H'
```

```
a[5]
```

```
'W'
```

```
a[-1]
```

```
'!'
```

```
a[-1]
```

```
'!'
```

```
a[-2]
```

```
'd'
```

```
a[:5] # a[0:5]
```

```
'Hello'
```

```
a[5:]
```

```
'World!'
```

Python 문자열 indexing & slicing

문자열 변수[start : end -1 : step]

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

```
b = '1-2-3-4-5'
```

```
b
```

```
'1-2-3-4-5'
```

```
b[0::2]
```

```
'12345'
```

```
b[::-1] # 건너뛰는 값이 음수일 때는, 기본 시작점은 -1
```

```
'5-4-3-2-1'
```

```
b[::-2]
```

```
'54321'
```


Python 문자열 함수

함수명	설명
join	지정된 문자로 문자열들을 연결
split	지정된 문자로 분할하여 리스트로 반환
strip	지정된 문자열을 제거
replace	특정 문자열을 지정된 문자열로 대체
startswith	지정된 문자열로 시작하는지 검사, True/False로 반환
endswith	지정된 문자열로 끝나는지 검사, True/False로 반환
count	특정 문자의 갯수
index	특정 문자의 위치 index
find	특정 문자열의 시작 위치 index
capitalize	단어의 첫 글자만 대문자로 변환
upper	모든 문자열을 대문자로 변환
lower	모든 문자열을 소문자로 변환

Python 문자열 함수

```
'-'.join('HelloWorldPython')
```

```
'H-e-l-l-o-W-o-r-l-d-P-y-t-h-o-n'
```

```
'-'.join('12345')
```

```
'1-2-3-4-5'
```

```
'Hello-World-Python'.split('-')
```

```
['Hello', 'World', 'Python']
```

```
'서울시 마포구 상암동 1585'.split()
```

```
['서울시', '마포구', '상암동', '1585']
```

```
text = '\t 문자열 정리 \n'  
text.strip()
```

```
'문자열 정리'
```

```
생일 = '2016/08/30'
```

```
생일.replace('/', '-')
```

```
'2016-08-30'
```

```
'Hello World Python'.startswith('Hello')
```

```
True
```

```
'Hello World Python'.startswith('Python')
```

```
False
```

```
'Python' in 'Hello World Python'
```

```
True
```

```
'Java' in 'Hello World Python'
```

```
False
```

Python 문자열 함수

```
text = 'Hello World Python'  
text.count('o')
```

3

```
text = 'Hello World Python, Welcome to Python World'  
text.count('Python')
```

2

```
text.index('o') # text.index('o', 5)
```

4

```
text.find('Python')
```

12

```
text.find('Python', 20)
```

31

```
'Hello World'.capitalize()
```

'Hello world'

```
'Hello World'.lower()
```

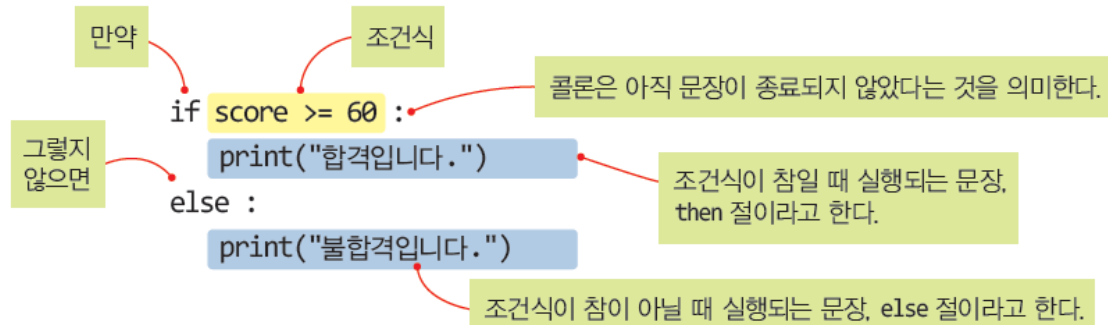
'hello world'

```
'Hello World'.upper()
```

'HELLO WORLD'

Python 조건문

if-else 문



```
a = 3  
b = 2
```

```
if a > b:  
    print('TRUE')
```

TRUE

```
if a < b:  
    print("Yes")  
    print("a")  
else:  
    print("No")  
    print("b")
```

No
b

```
suffix = "png"  
  
if suffix == "htm":  
    content = "text/html"  
  
elif suffix == "jpg":  
    content = "image/jpeg"  
  
elif suffix == "png":  
    content = "image/png"  
  
print(content)
```

image/png

■ 관계 연산자

연산	의미
<code>x == y</code>	x와 y가 같은가?
<code>x != y</code>	x와 y가 다른가?
<code>x > y</code>	x가 y보다 큰가?
<code>x < y</code>	x가 y보다 작은가?
<code>x >= y</code>	x가 y보다 크거나 같은가?
<code>x <= y</code>	x가 y보다 작거나 같은가?

■ 논리 연산자

연산	의미
<code>x and y</code>	AND 연산, x와 y가 모두 참이면 참, 그렇지 않으면 거짓
<code>x or y</code>	OR 연산, x나 y중에서 하나만 참이면 참, 모두 거짓이면 거짓
<code>not x</code>	NOT 연산, x가 참이면 거짓, x가 거짓이면 참

Python 반복문

```
a = 0
while a < 5:
    print(a, '반복합니다.')
    a += 1 # a = a + 1
```

0 반복합니다.
1 반복합니다.
2 반복합니다.
3 반복합니다.
4 반복합니다.

```
for a in range(0, 5):
    print(a, '반복합니다.')

```

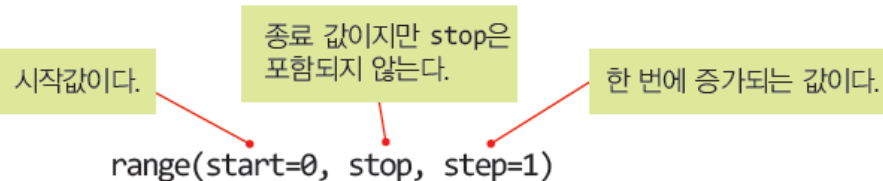
0 반복합니다.
1 반복합니다.
2 반복합니다.
3 반복합니다.
4 반복합니다.

```
list(range(0, 5))
```

[0, 1, 2, 3, 4]

Python 반복문

range() 함수



```
list(range(10))
```

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

```
list(range(0, 10))
```

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

```
list(range(0, 10, 2))
```

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

```
list(range(1, 10, 2))
```

```
[1, 3, 5, 7, 9]
```

```
list(range(10, 0))
```

```
[]
```

```
list(range(10, 0, -1))
```

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

```
list(range(10, 0, -2))
```

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

```
list(range(10, 0, -3))
```

```
[10, 7, 4, 1]
```

Python 반복문

```
num = list(range(1, 11))
```

```
for i in num:  
    print(i)
```

1
2
3
4
5
6
7
8
9
10

```
num = 3
```

```
for i in range(1, 10):  
    answer = num * i  
    print(f'{num} * {i} = {answer}')
```

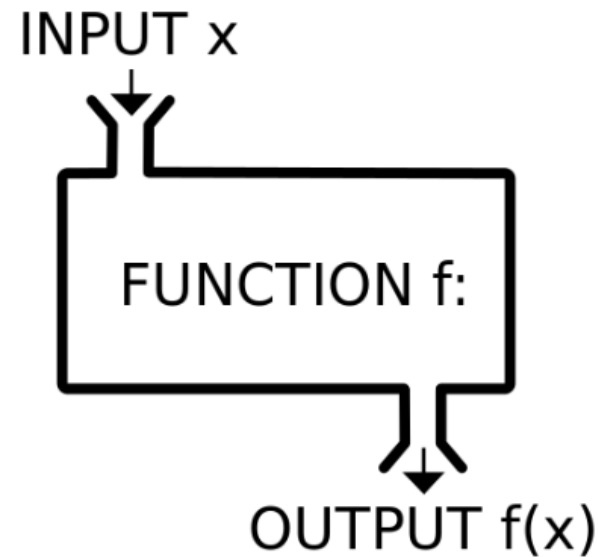
3 * 1 = 3
3 * 2 = 6
3 * 3 = 9
3 * 4 = 12
3 * 5 = 15
3 * 6 = 18
3 * 7 = 21
3 * 8 = 24
3 * 9 = 27

1.1.1.1.python_basic.ipynb 실습

Python 함수

함수는 호출해야 실행되는 Callable Object

`def 함수명(parameter):`



Input : parameter, **default value**

Output : return, **multiple return**

Python 함수

- 함수 정의

```
def add1():  
    print("더하기 함수입니다.")
```

- 함수 호출

```
add1()
```

더하기 함수입니다.

- 함수 정의

```
def add2(x, y):  
    print(x + y)
```

- 함수 호출

```
add1(1, 2)
```

3

Python 함수

```
def add3(x, y):  
    return x + y
```

```
result = add3(1, 2)  
result
```

3

```
def square(x, y):  
    x = x ** 2  
    y = y ** 2  
    return x, y
```

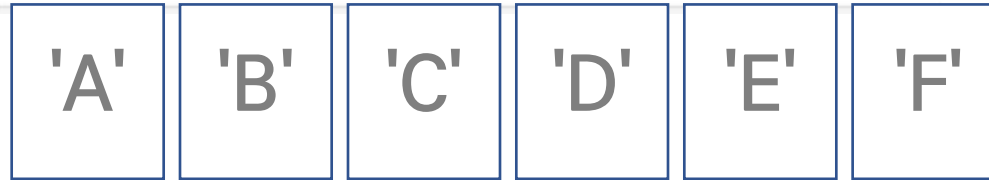
```
a, b = square(2, 3)  
print(a)  
print(b)
```

4

9

Python 자료구조 – List

```
letters = ['A', 'B', 'C', 'D', 'E', 'F']
```



[0] [1] [2] [3] [4] [5]

```
letters
```

```
['A', 'B', 'C', 'D', 'E', 'F']
```

```
letters[0]
```

```
'A'
```

```
letters[1]
```

```
'B'
```

```
letters[5]
```

```
'F'
```

Python 자료구조 – List 함수

```
letters.append('a')  
letters
```

```
['A', 'B', 'C', 'D', 'E', 'F', 'a']
```

```
letters.count('a')
```

```
1
```

```
letters.insert(2, 'z')  
letters
```

```
['A', 'B', 'z', 'C', 'D', 'E', 'F', 'a']
```

```
letters.pop(2)
```

```
'z'
```

```
letters
```

```
['A', 'B', 'C', 'D', 'E', 'F', 'a']
```

```
letters.remove('a')  
letters
```

```
['A', 'B', 'C', 'D', 'E', 'F']
```

```
letters.sort(reverse=True)
```

```
letters
```

```
['F', 'E', 'D', 'C', 'B', 'A']
```

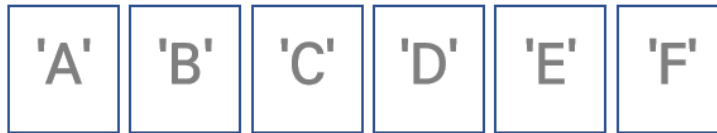
```
letters.sort()
```

```
letters
```

```
['A', 'B', 'C', 'D', 'E', 'F']
```

Python 자료구조 – List indexing & slicing

```
letters = ['A', 'B', 'C', 'D', 'E', 'F']
```



[0] [1] [2] [3] [4] [5]

[-6] [-5] [-4] [-3] [-2] [-1]

```
letters[-1]
```

'F'

```
letters[-2]
```

'E'

```
letters[0:3]
```

['A', 'B', 'C']

```
letters[3:]
```

['D', 'E', 'F']

```
letters[:4]
```

['A', 'B', 'C', 'D']

```
letters[:]
```

['A', 'B', 'C', 'D', 'E', 'F']

```
letters[::2]
```

['A', 'C', 'E']

```
letters[::-1]
```

['F', 'E', 'D', 'C', 'B', 'A']

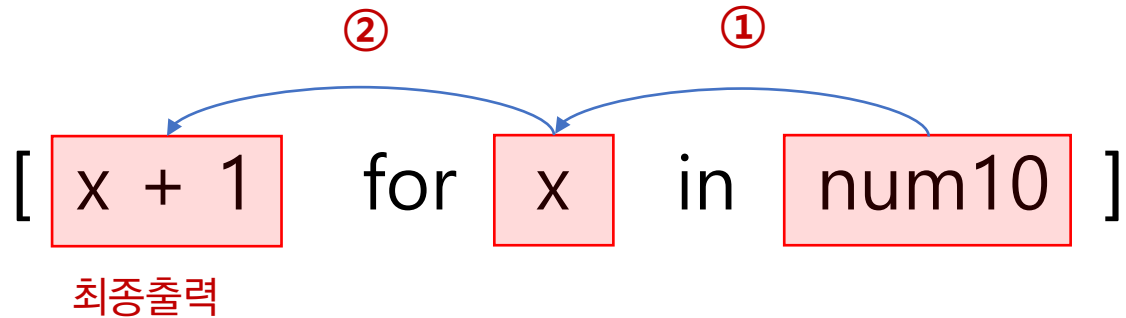
Python 자료구조 – List Comprehension

```
list(range(1,11))
```

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

```
num10 = list(range(1, 11))  
[ x + 1 for x in num10 ]
```

```
[2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
```



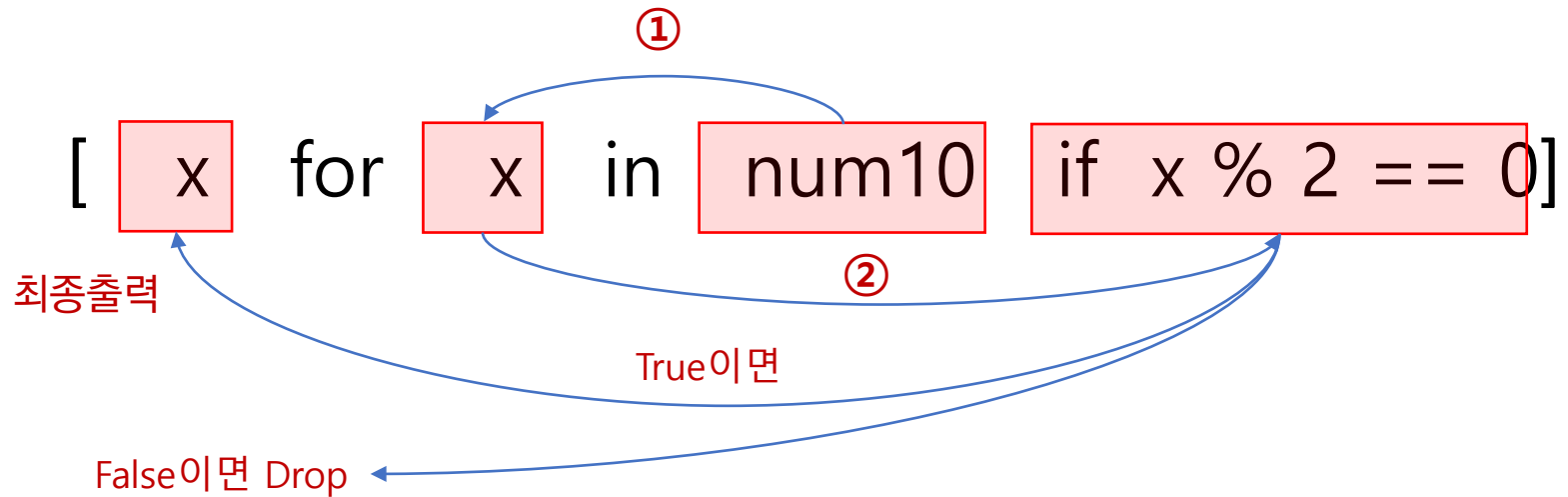
Python 자료구조 – List Comprehension

```
for x in num10:  
    if x % 2 == 0:  
        print(x)
```

2
4
6
8
10

```
[x for x in num10 if x % 2 == 0]
```

[2, 4, 6, 8, 10]



Python 자료구조 – Tuple

```
tuple1 = (1, 2, 3,)
tuple1
```

```
(1, 2, 3)
```

```
tuple1[0] = 4
```

```
-----
TypeError                                 Traceback (most recent call last)
<ipython-input-72-9e1c1167873d> in <module>
----> 1 tuple1[0] = 4

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

```
list1 = list(tuple1)
list1
```

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

```
list1.append(4)
```

```
list1
```

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

```
tuple1 = tuple(list1)
```

```
tuple1
```

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

Python 자료구조 – Dictionary

```
중간고사 = {  
    "수학": 100,  
    "영어": 90,  
}  
중간고사
```

```
{'수학': 100, '영어': 90}
```

```
중간고사['국어'] = 85  
중간고사
```

```
{'수학': 100, '영어': 90, '국어': 85}
```

```
중간고사['영어']
```

```
90
```

```
list(중간고사.keys())
```

```
['수학', '영어', '국어']
```

```
list(중간고사.values())
```

```
[100, 90, 85]
```

```
for 과목, 점수 in 중간고사.items():  
    print('{} 과목점수는 {} 점'.format(과목, 점수))
```

```
수학 과목점수는 100 점
```

```
영어 과목점수는 90 점
```

```
국어 과목점수는 85 점
```

```
중간고사['영어'] = 95  
중간고사
```

```
{'수학': 100, '영어': 95, '국어': 85}
```

1.2.1.1.python_basic2.ipynb 실습



Thank you