

Python 기본과정

inky4832@daum.net

1장. Python 개요 및 개발환경



Python 개요

Python 설치

PyCharm 프로그램 설치

1) Python 개요

python

- 네덜란드인 구도 반 로섬(Guido van Rossum)이 발표한 범용프로그램 언어.
- 순수한 프로그램 언어 기능 이외에 다른 언어로 만든 모듈들을 연결하는 접착언어(glue language) 기능 포함.
- 다음과 같은 다양한 파이썬 종류가 있다.

종 류	구 分
	Python(CPython) https://www.python.org/ C로 작성된 인터프리터. 파이썬이라 하면 보통 이를 일컫는다.
	Stackless Python https://bitbucket.org/stackless-dev/stackless/ C 스택을 사용하지 않는 인터프리터.
	Jython http://www.jython.org/ 자바 가상머신 용 인터프리터로 자바의 API를 사용할 수 있다.
	IronPython http://ironpython.net/ .NET 플랫폼 용 인터프리터.
	PyPy http://www.pypy.org/ 파이썬으로 작성된 인터프리터로 속도 개선이 목표.
	IPython http://ipython.org/ 주로 계산이나 통계처리를 위한 인터프리터.

2) Python 특징 및 활용

python

○ 특징

- 직관적이고 배우기 쉽다.
- 다양한 라이브러리 제공 (Numpy, Pandas, Matplotlib, PyQt, … .)
- 현재 가장 인기 있는 프로그램 언어 (구글, 아마존, 유튜브, 네이버, 카카오 등)

○ 활용 분야

- 빅데이터 및 데이터 분석 프로그래밍
- GUI 프로그래밍
- DB 연동 프로그래밍
- 시스템 유틸리티
- 수치연산 통계처리
- 네트워크 및 인터넷 프로그래밍
- 네트워크 장비 제어

3) Python 개발환경 구축 방법 3가지

python

가. Python 프로그램 설치후 PyCharm,Eclipse IDE 사용

<http://python.org> , <http://www.jetbrains.com>

나. Python 배포판: conda 설치

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

다. 구글 코랩 (구글 colaboratory 서비스)

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

코랩 노트북은 다음과 같은 장점이 있다.

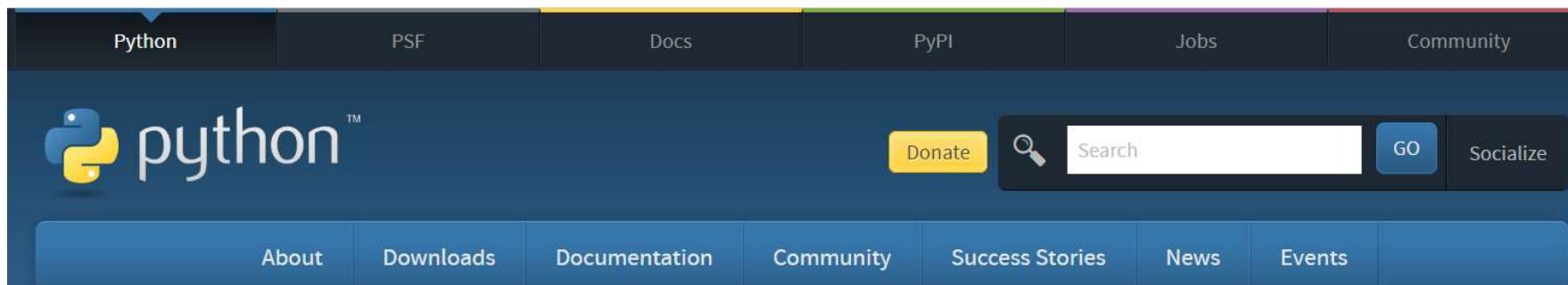
1. 별도의 파이썬 설치 없이 웹 브라우저 만을 이용해 주피터 노트북과 같은 작업을 할 수 있다.
2. 다른 사용자들과 공유가 쉬워 연구 및 교육용으로 많이 사용된다.
3. Tensorflow, keras, matplotlib, scikit-learn, pandas 등 데이터 분석에 많이 사용되는 패키지들이 미리 설치되어 있다.
4. 무료로 GPU를 사용할 수 있다.
5. 구글 드라이브나 구글 스프레드시트 등과 같은 식으로 공유와 편집이 가능하다. 만약 두 명 이상의 사람이 동시에 같은 파일을 수정 하더라도 변경사항이 모든 사람에게 즉시 표시된다.

4) Python 설치

python

가. 프로그램 다운로드

<http://python.org/downloads/windows/>



Python >> Downloads >> Windows

Python Releases for Windows

- [Latest Python 3 Release - Python 3.8.1](#)
- [Latest Python 2 Release - Python 2.7.17](#)

Stable Releases

- [Python 3.8.1 - Dec. 18, 2019](#)

Note that Python 3.8.1 cannot be used on Windows XP or earlier.

- [Download Windows help file](#)
- [Download Windows x86-64 embeddable zip file](#)
- [Download Windows x86-64 executable installer](#)
- [Download Windows x86-64 web-based installer](#)

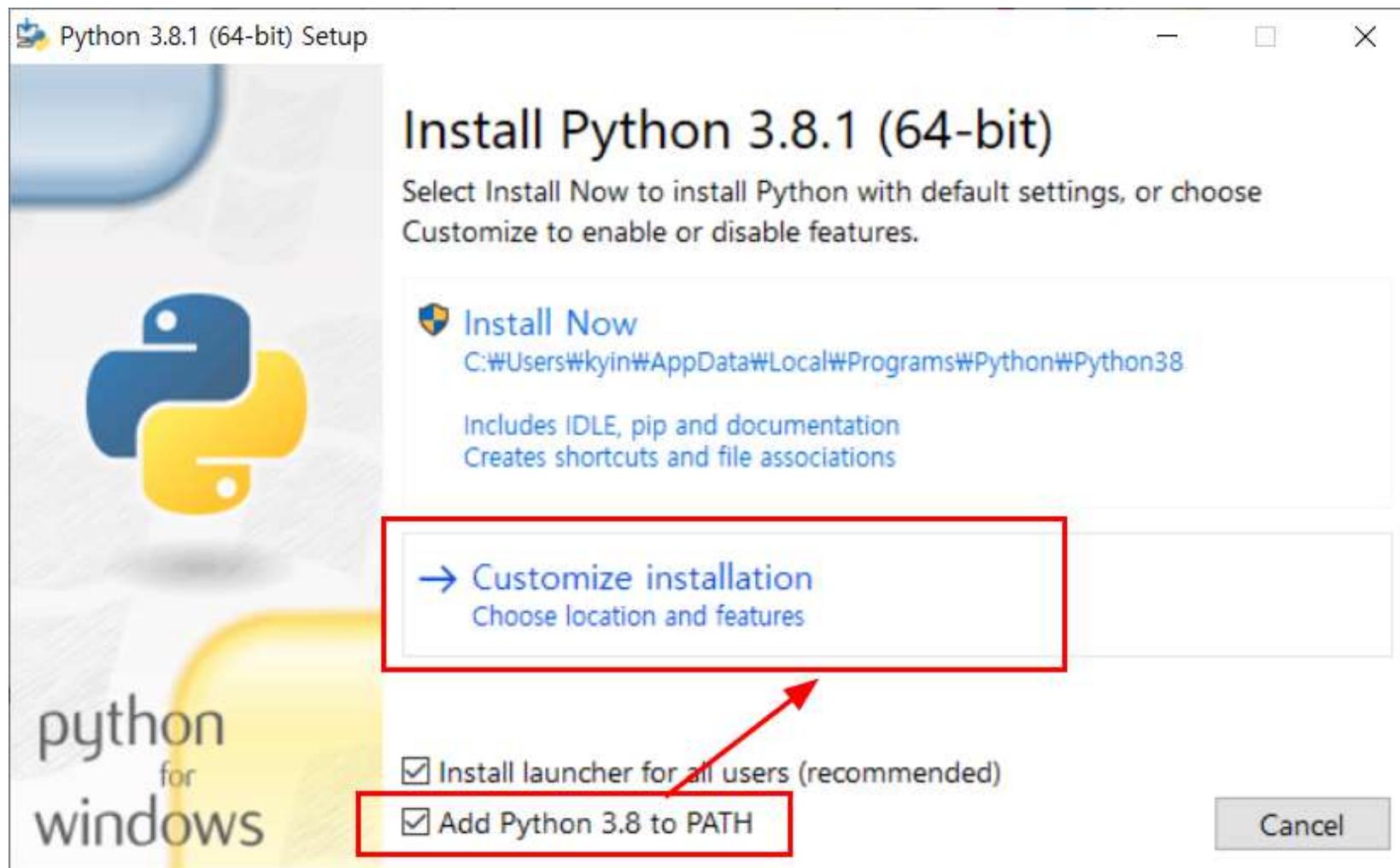
Pre-releases

- [Python 3.9.0a2 - Dec. 18, 2019](#)
 - [Download Windows help file](#)
 - [Download Windows x86-64 embeddable zip file](#)
 - [Download Windows x86-64 executable installer](#)
 - [Download Windows x86-64 web-based installer](#)
 - [Download Windows x86 embeddable zip file](#)
 - [Download Windows x86 executable installer](#)

4) Python 설치

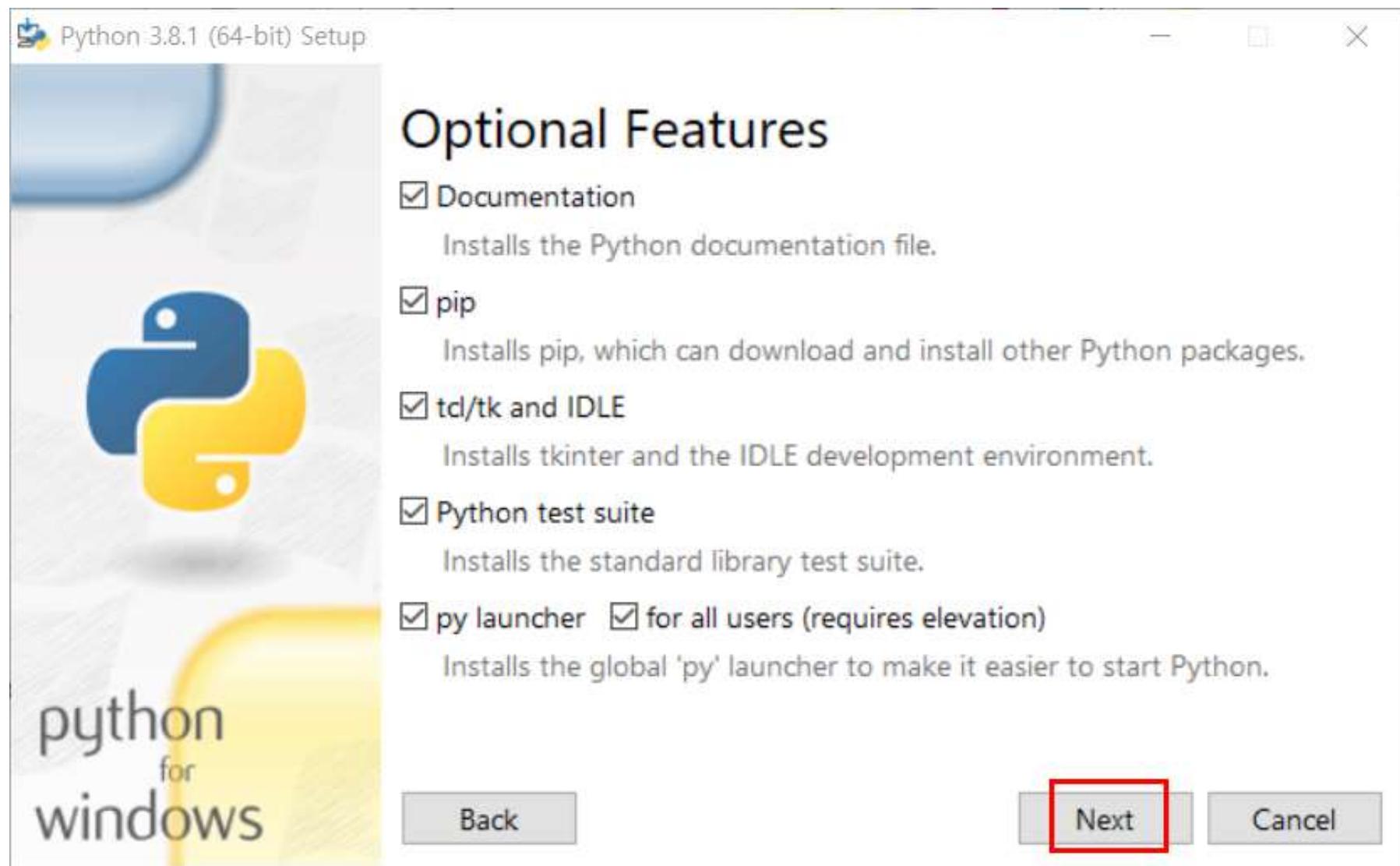
python

나. 설치 시작



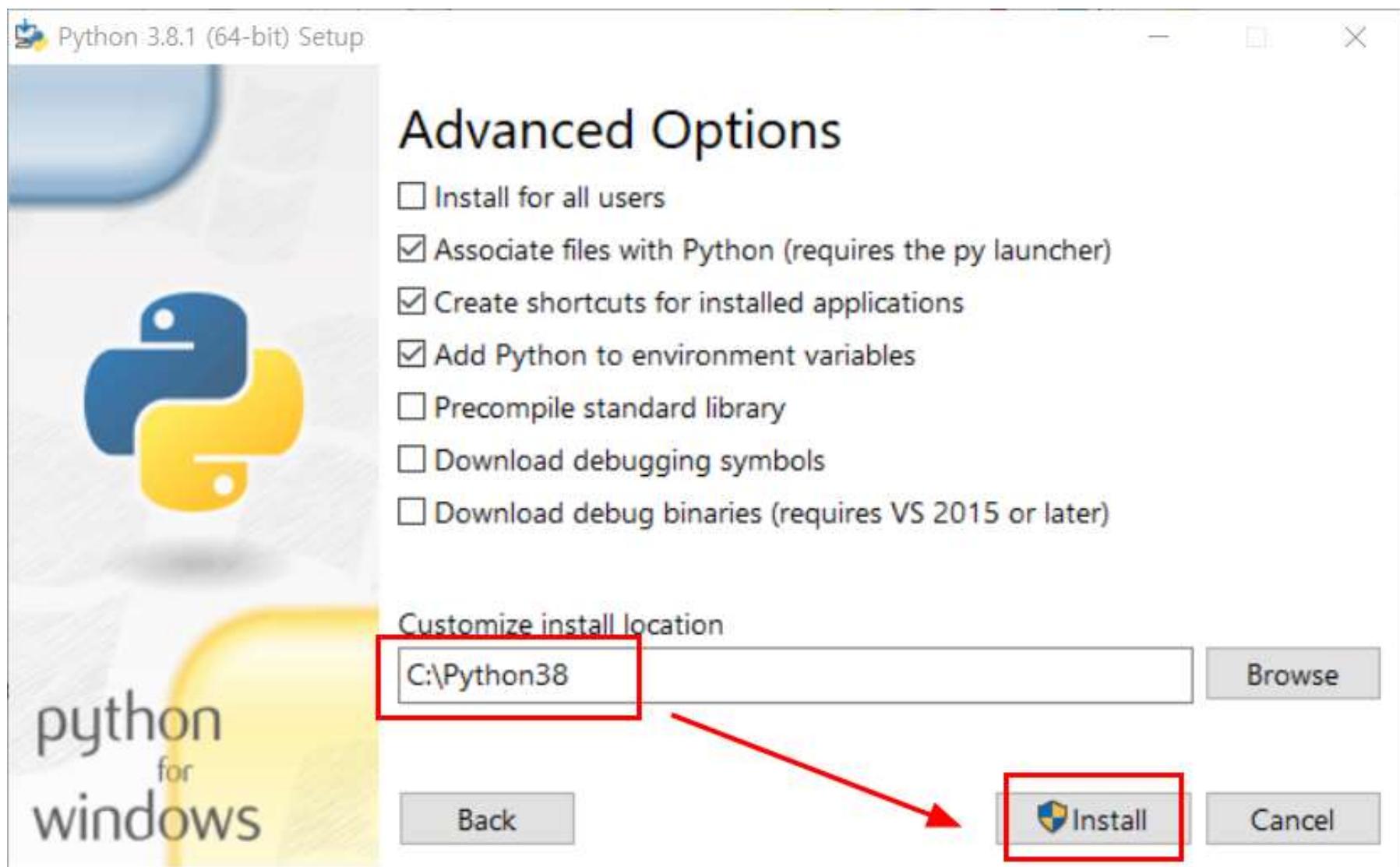
4) Python 설치

python



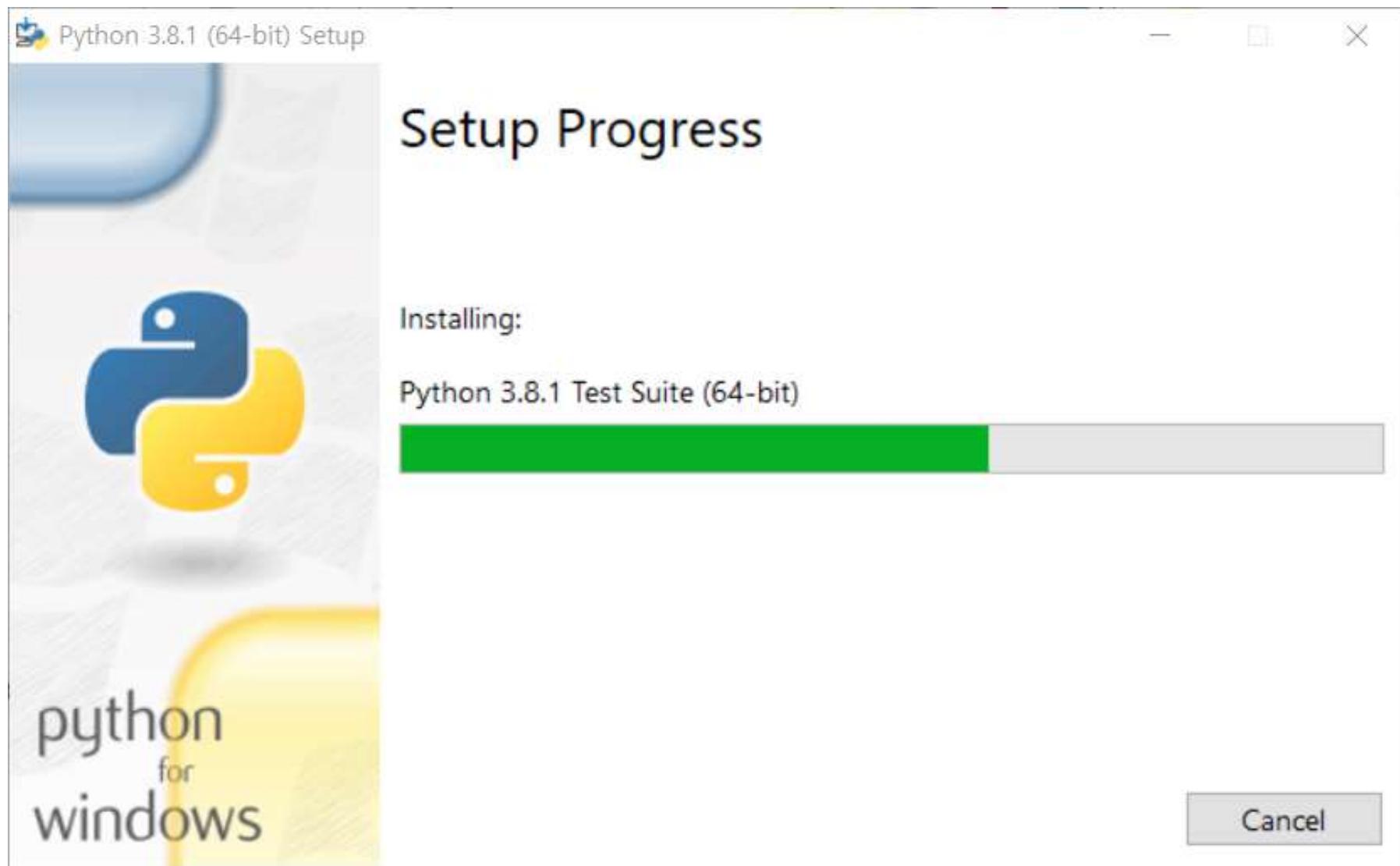
4) Python 설치

python



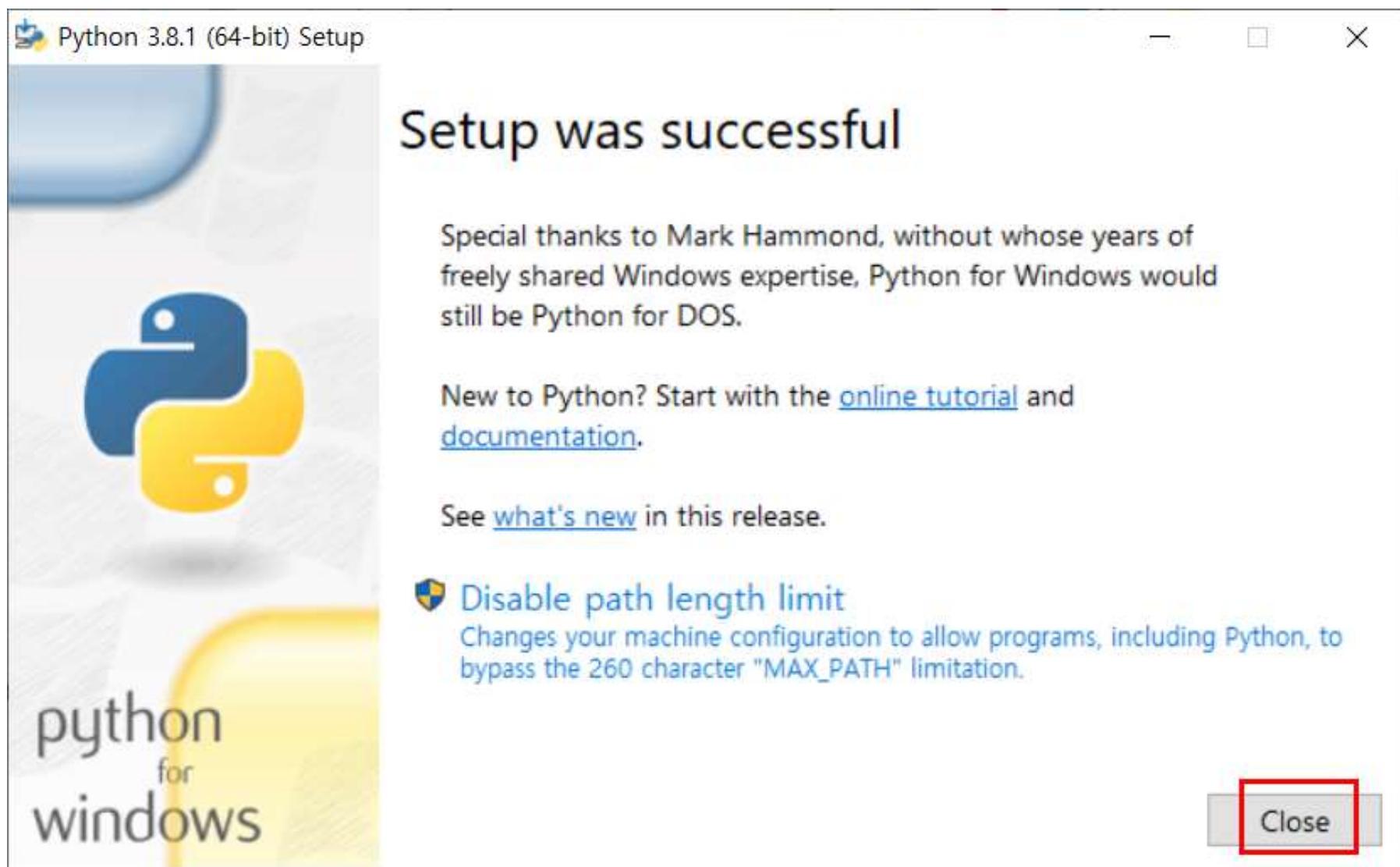
4) Python 설치

python



4) Python 설치

python

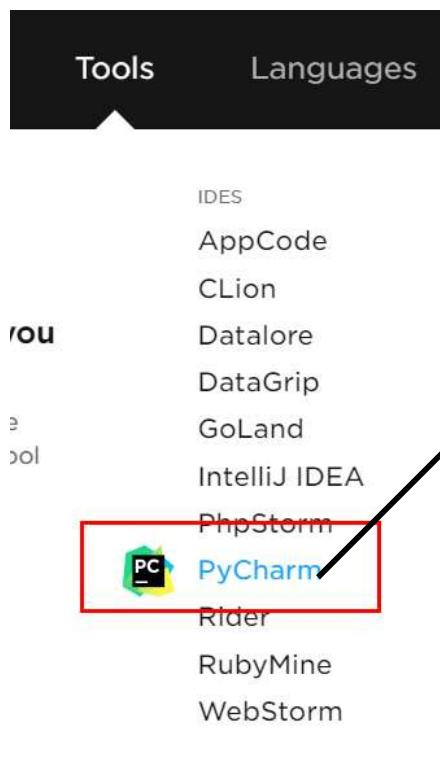


5) PyCharm 설치

python

가. 프로그램 다운로드

<https://www.jetbrains.com>



Download PyCharm

Windows macOS Linux

Professional

For both Scientific and Web Python development. With HTML, JS, and SQL support.

[DOWNLOAD](#)

Free trial

Community

For pure Python development

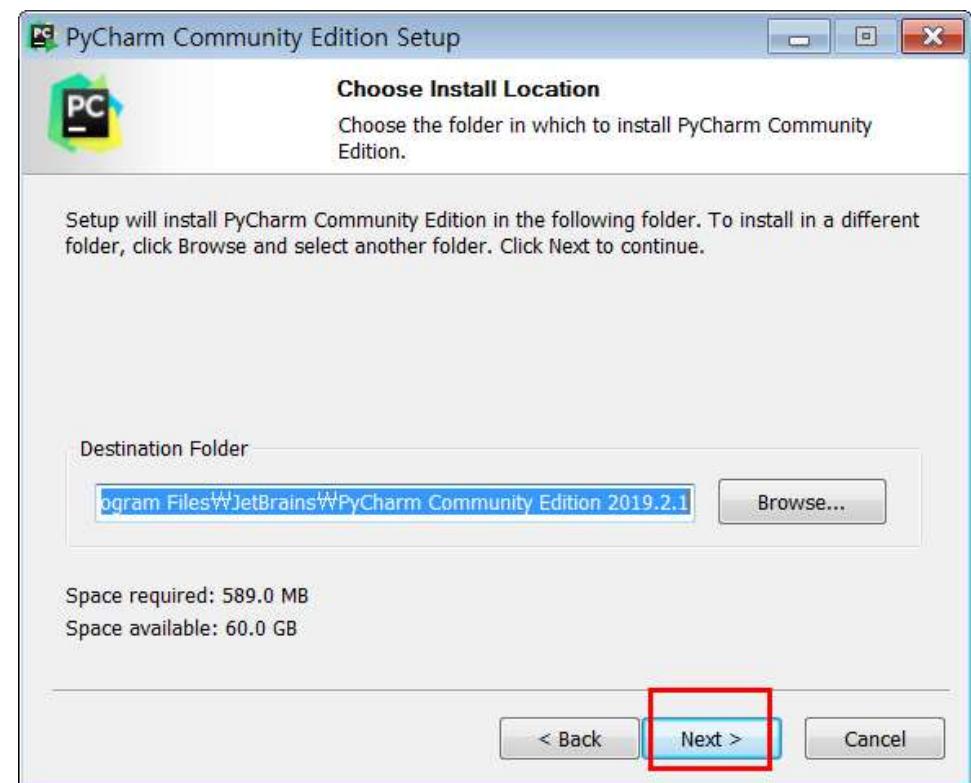
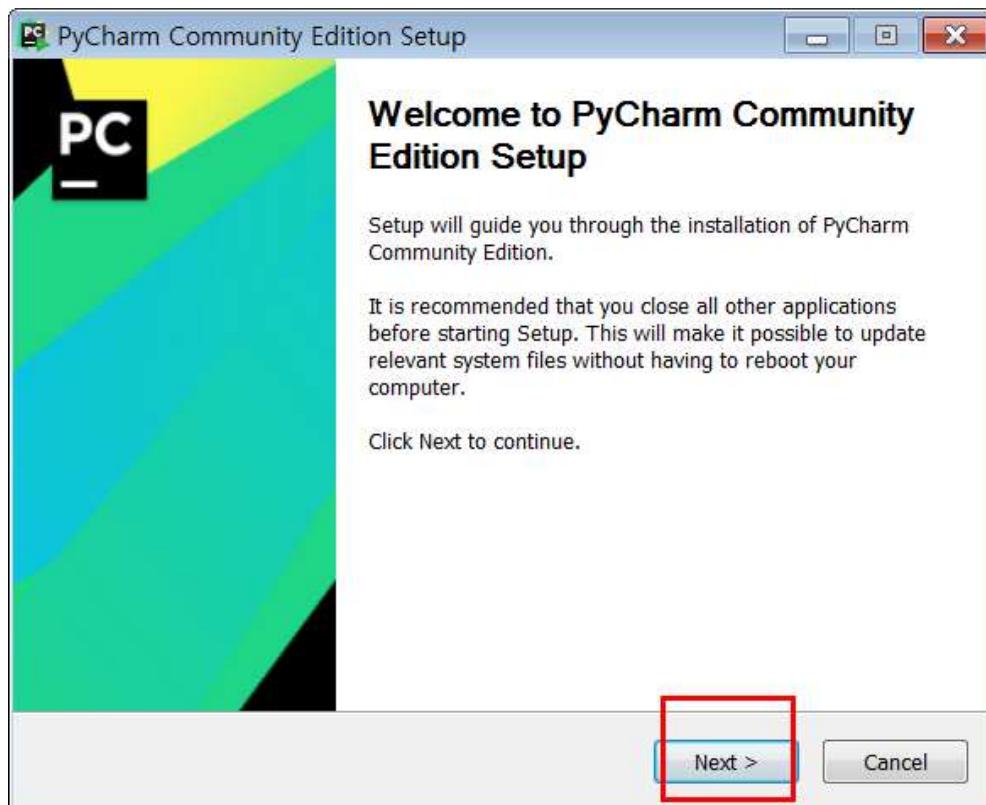
[DOWNLOAD](#)

Free, open-source

5) PyCharm 설치

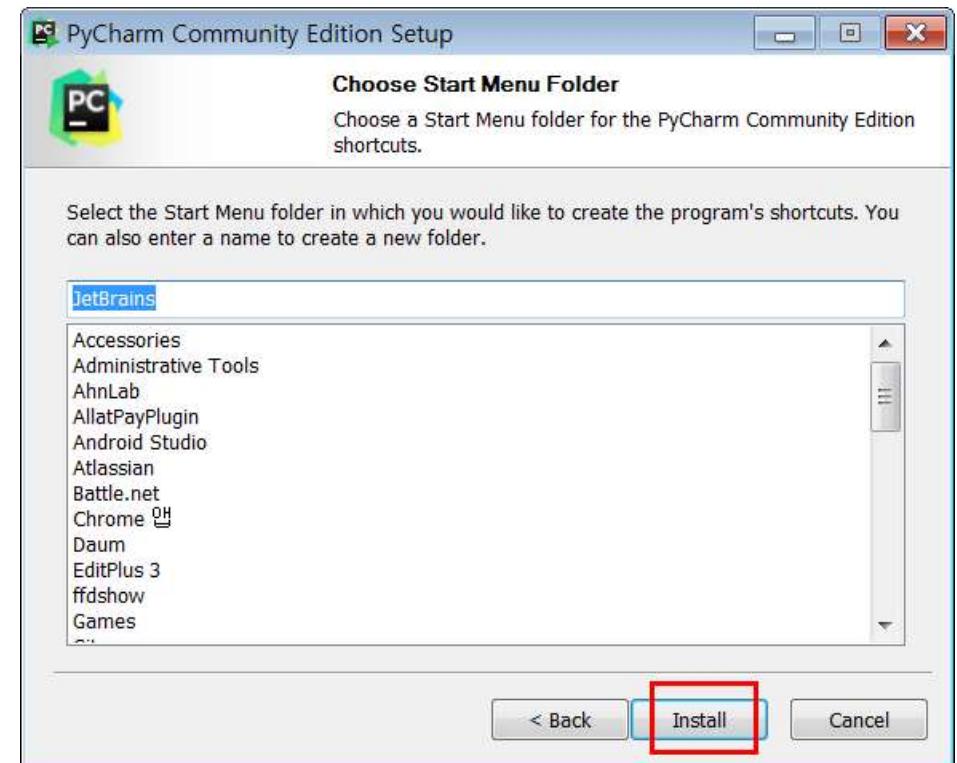
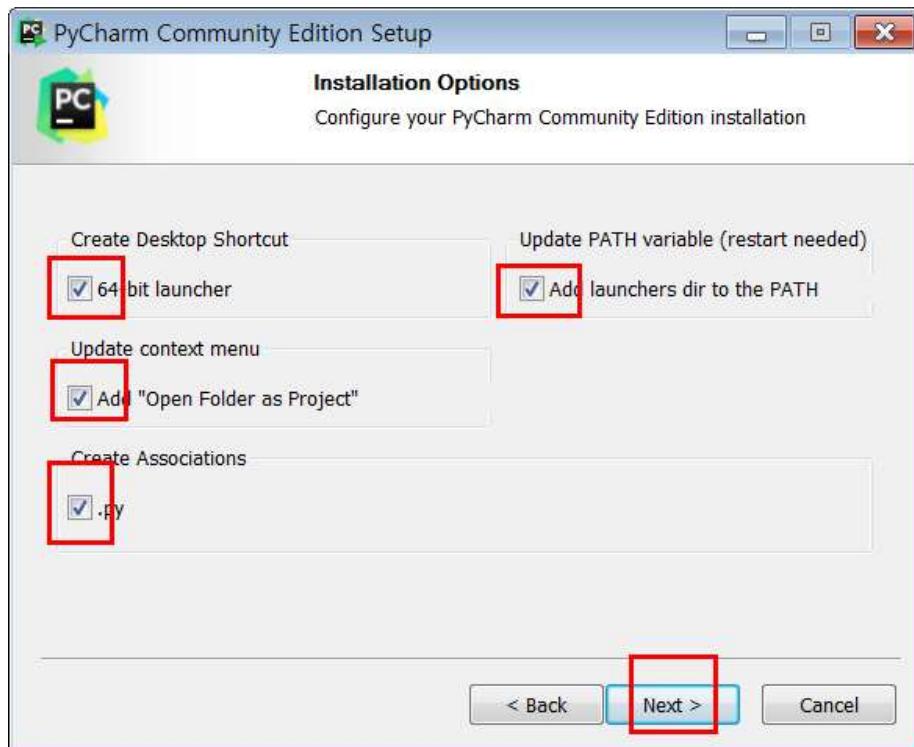
python

나. 프로그램 설치



5) PyCharm 설치

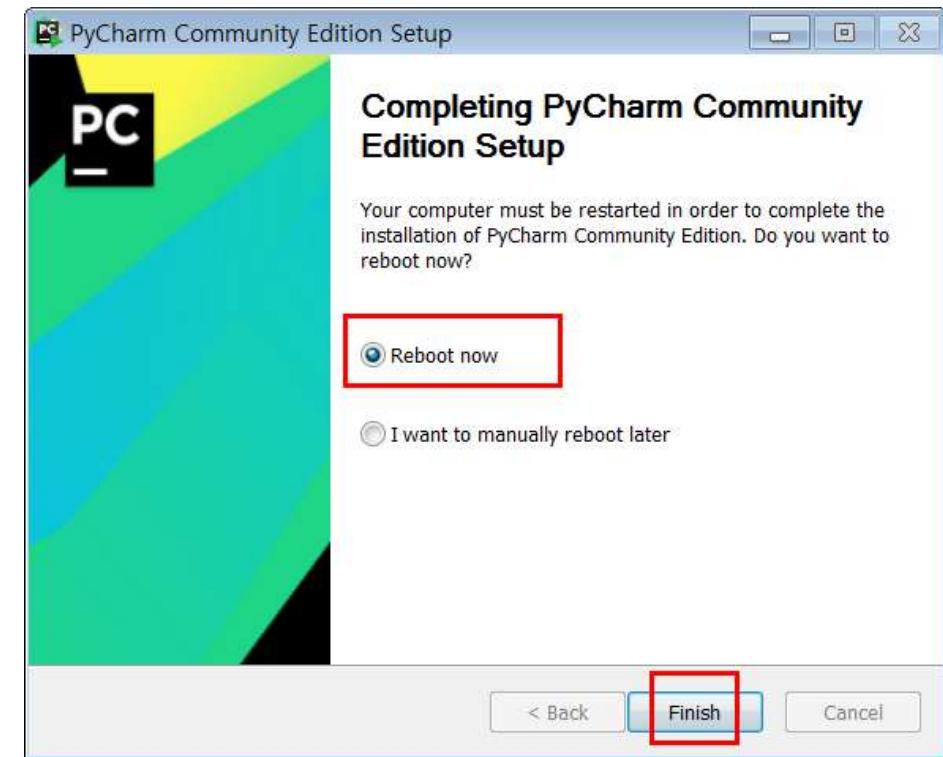
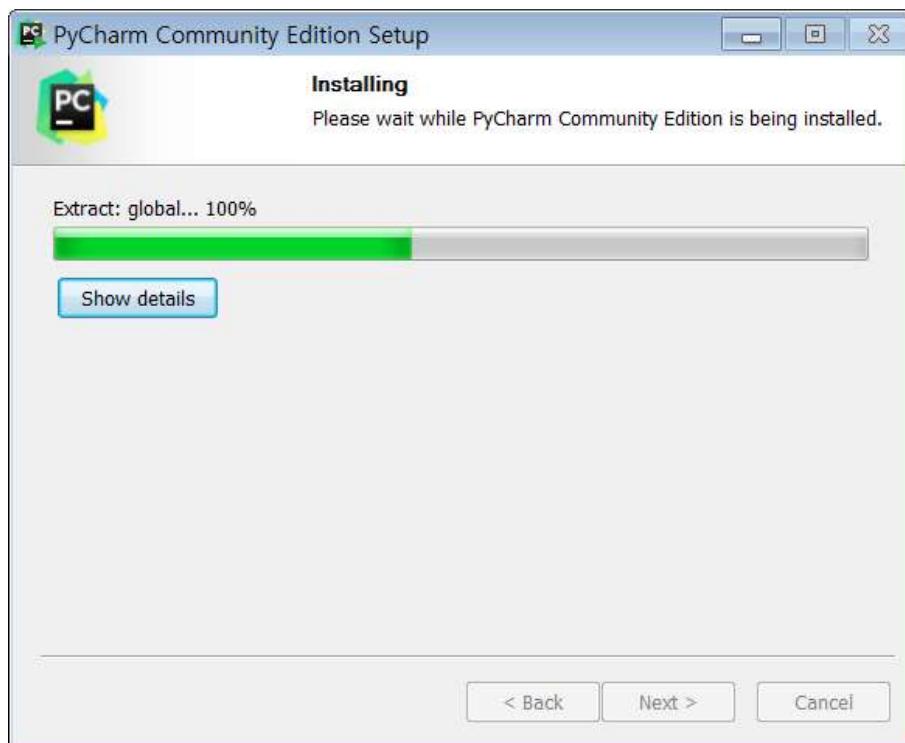
python



5) PyCharm 설치

python

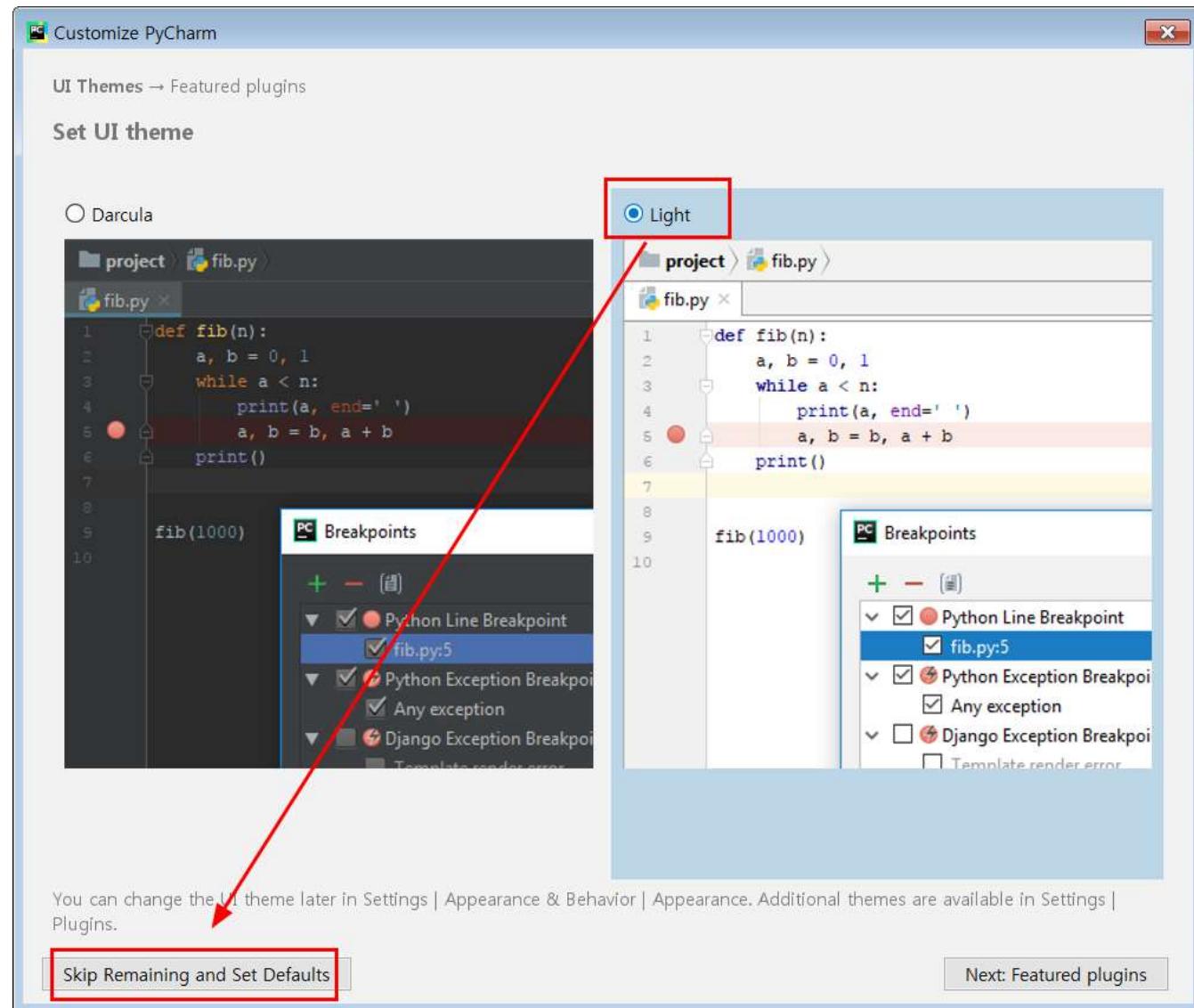
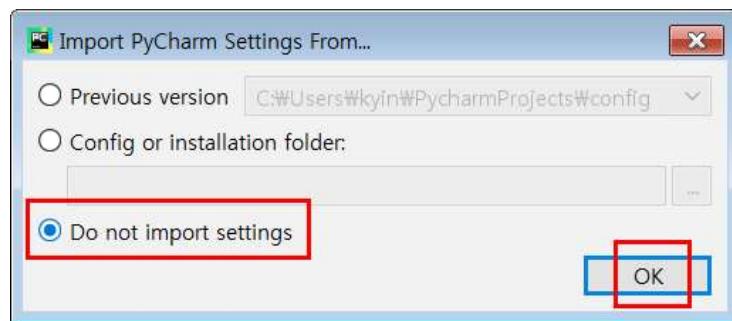
다. 프로그램 설치 완료



5) PyCharm 실행

python

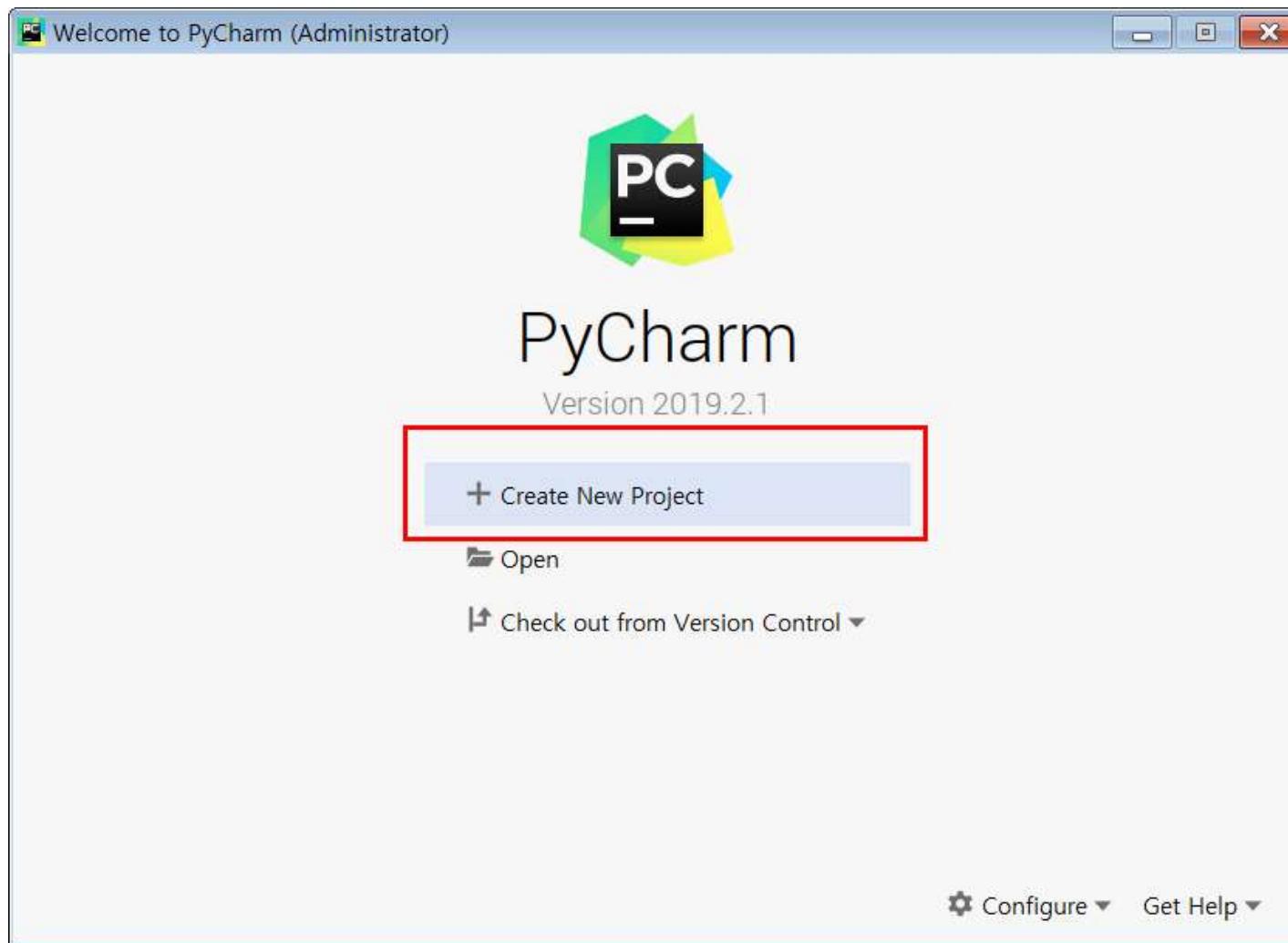
가. 프로그램 실행



5) PyCharm 실행

python

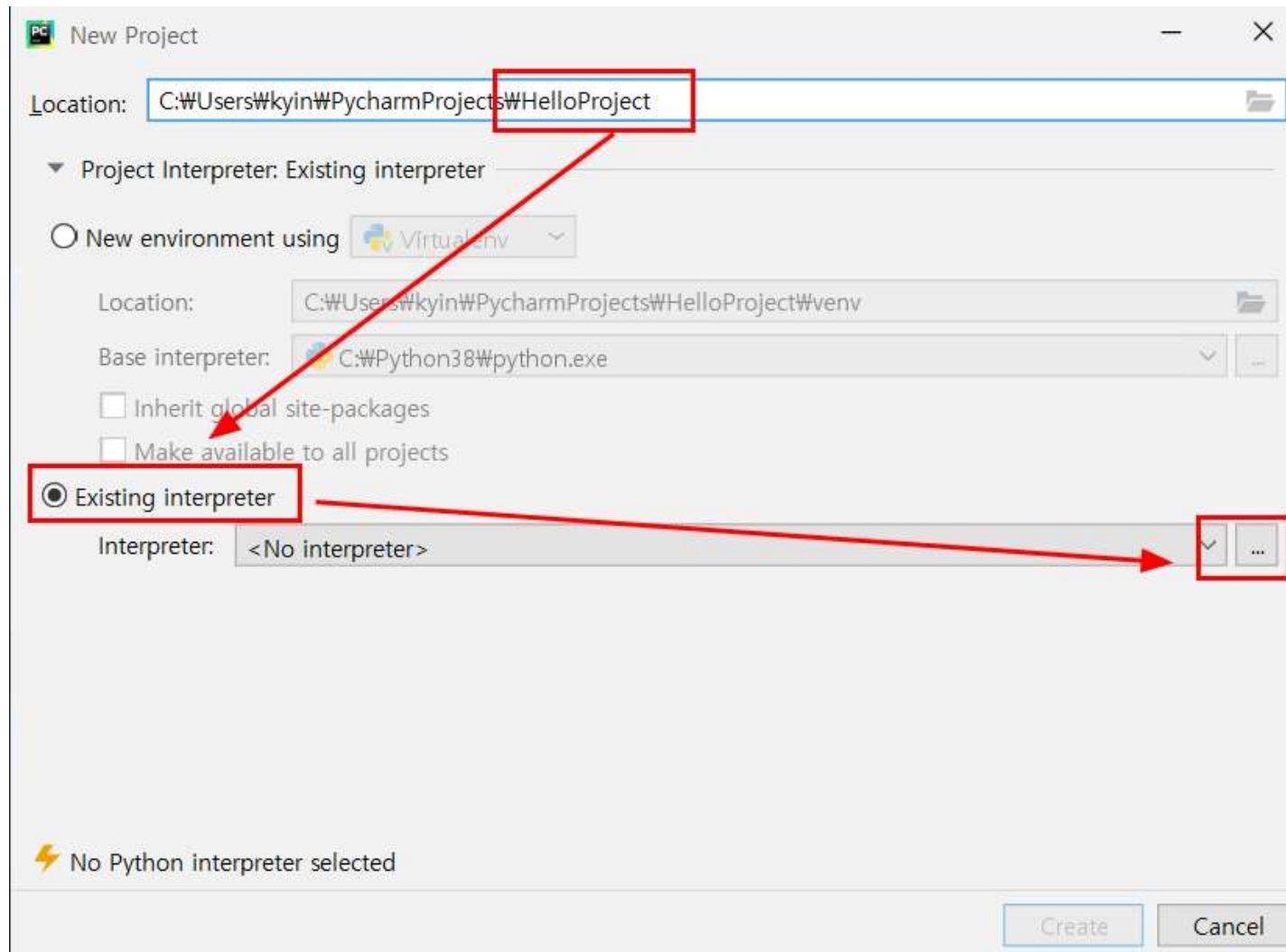
나. 프로젝트 생성



5) PyCharm 실행

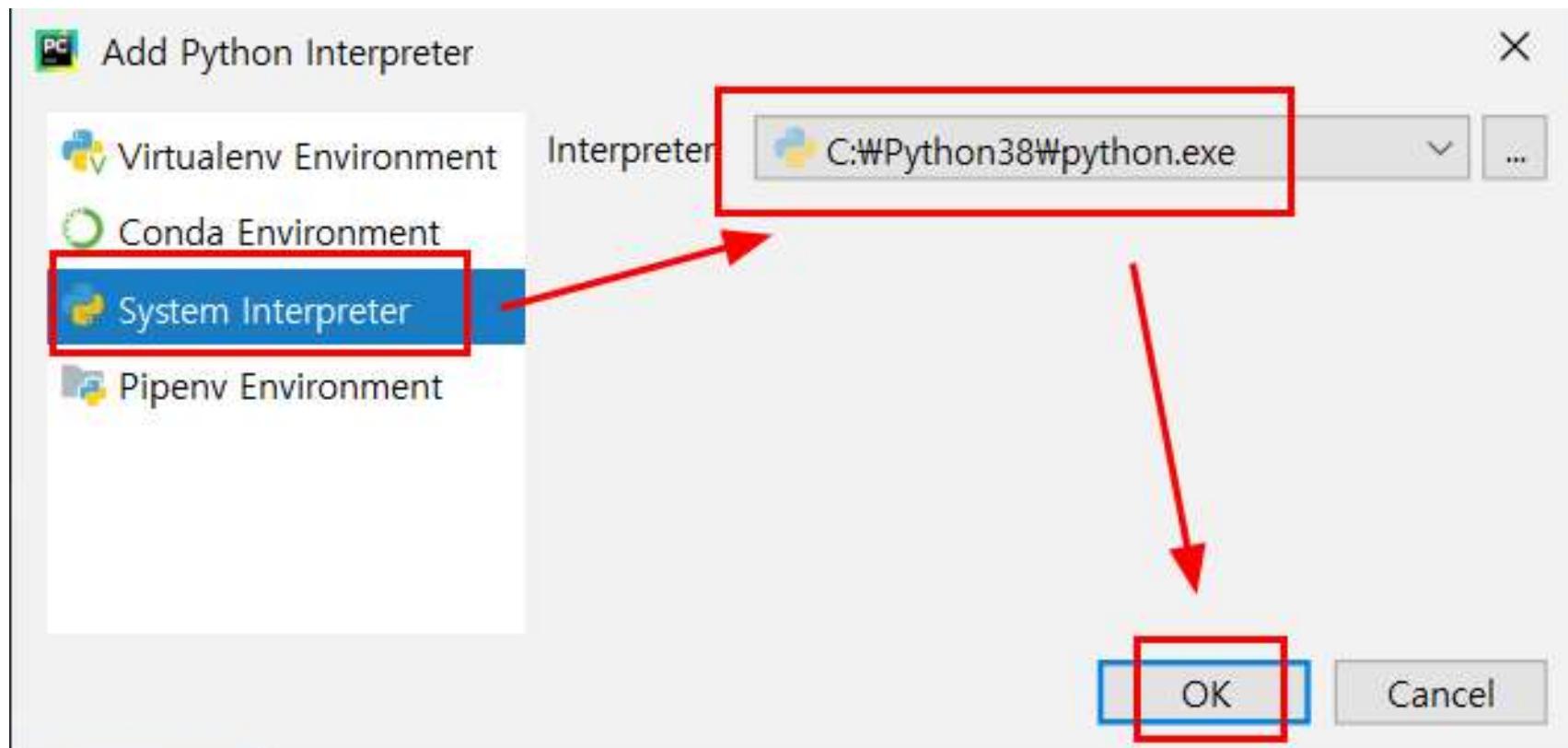
python

다. 프로젝트명 및 interpreter 지정



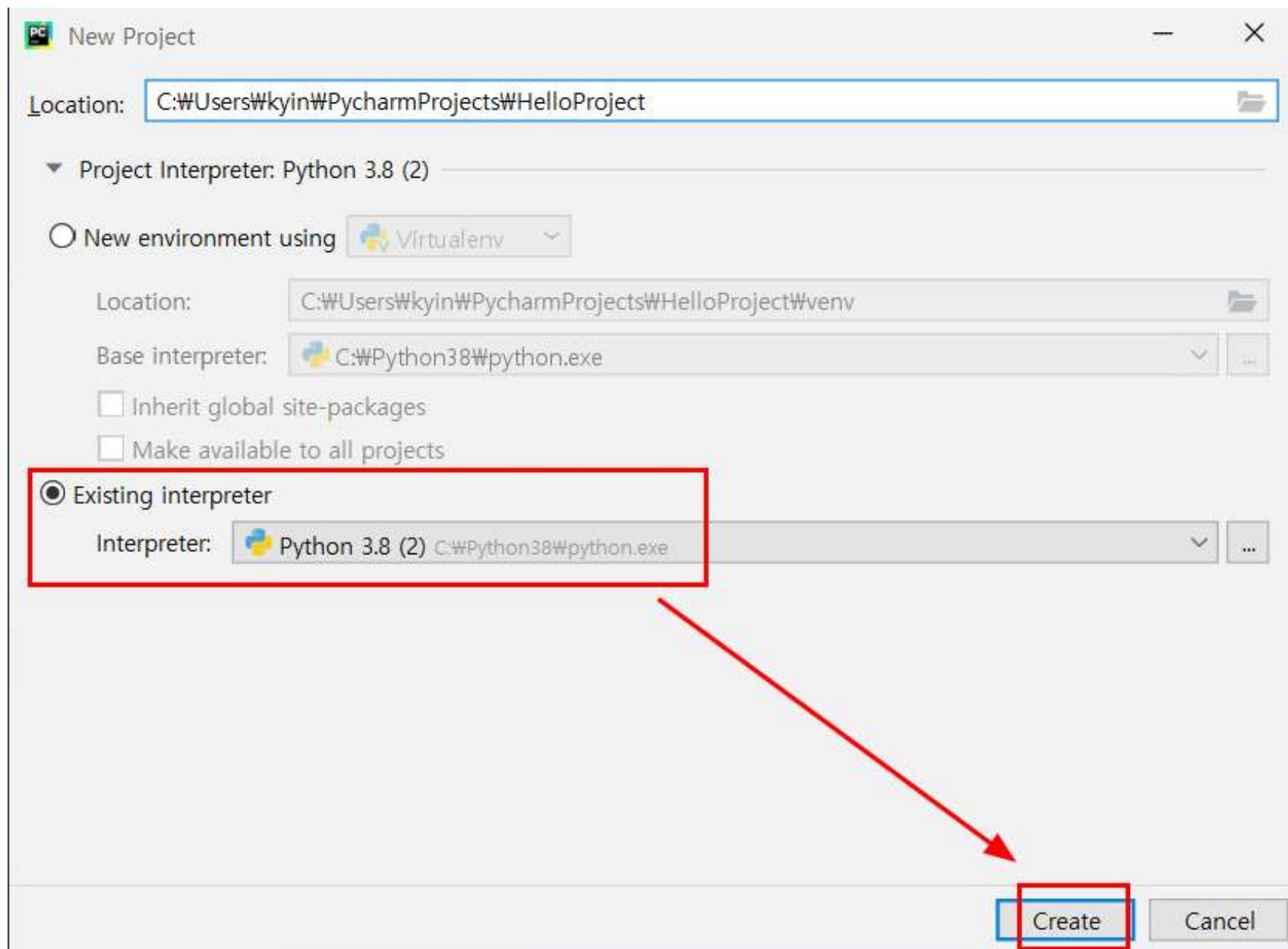
5) PyCharm 실행

python



5) PyCharm 실행

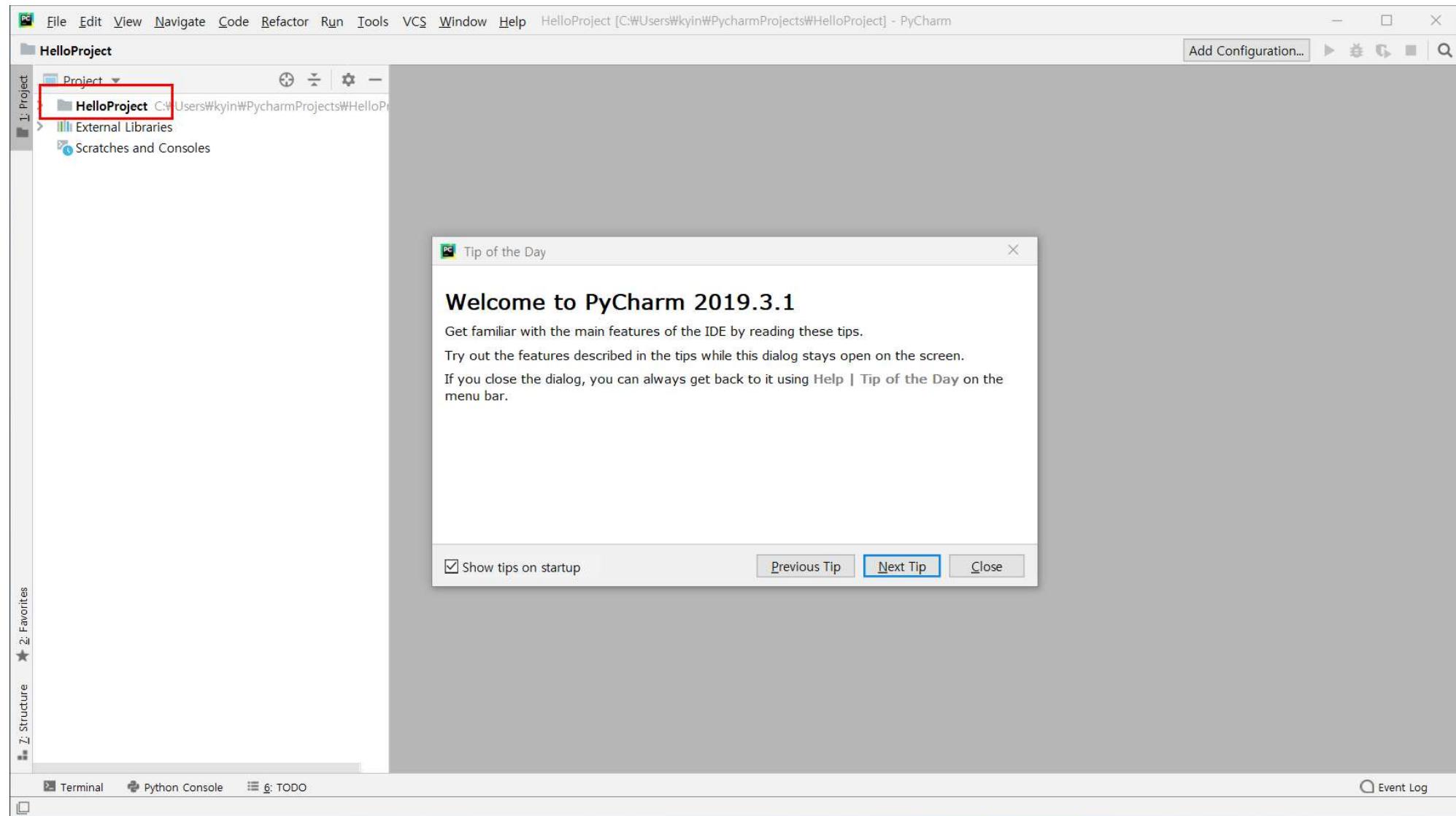
python



5) PyCharm 실행

python

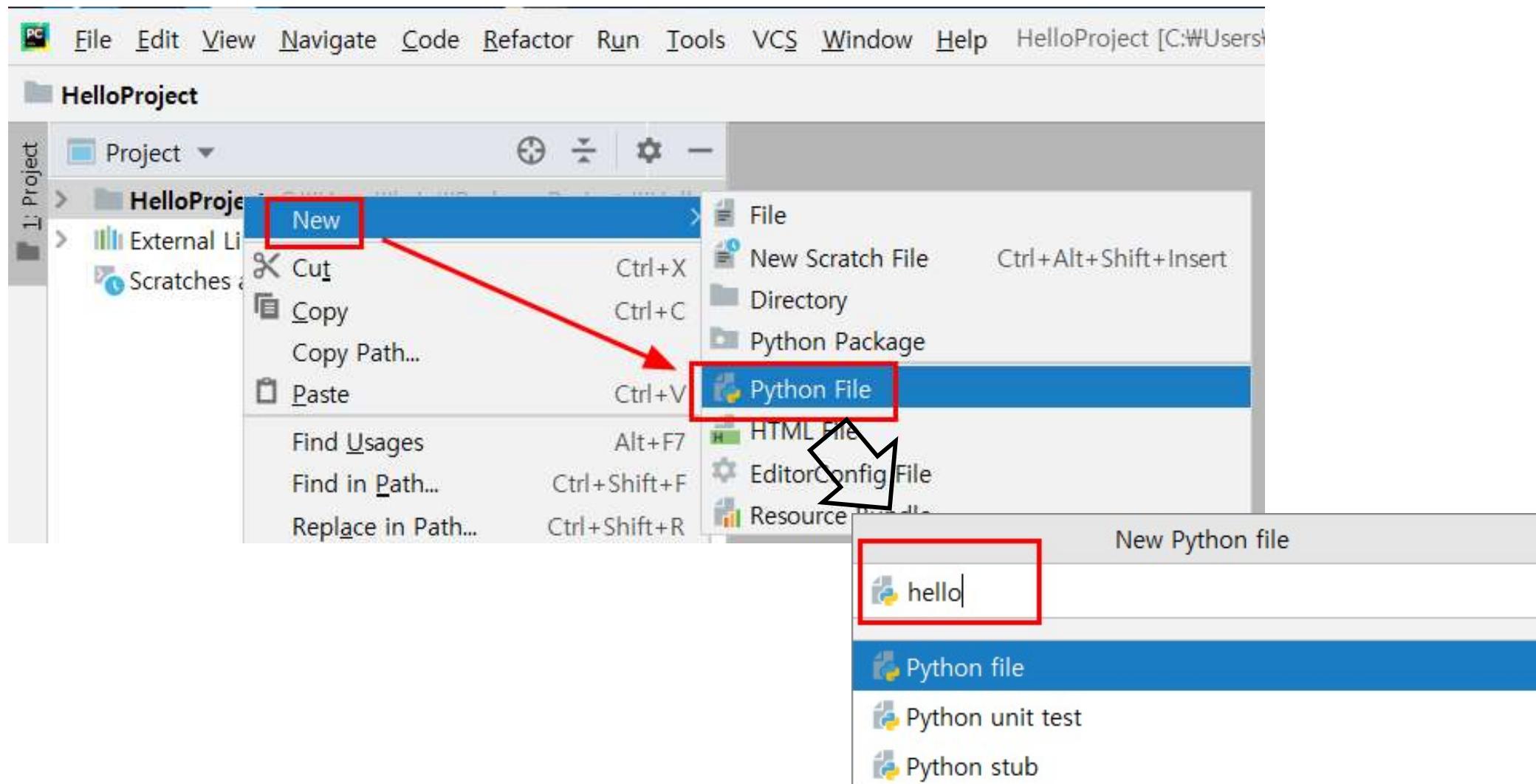
라. PyCharm 시작



5) PyCharm 실행

python

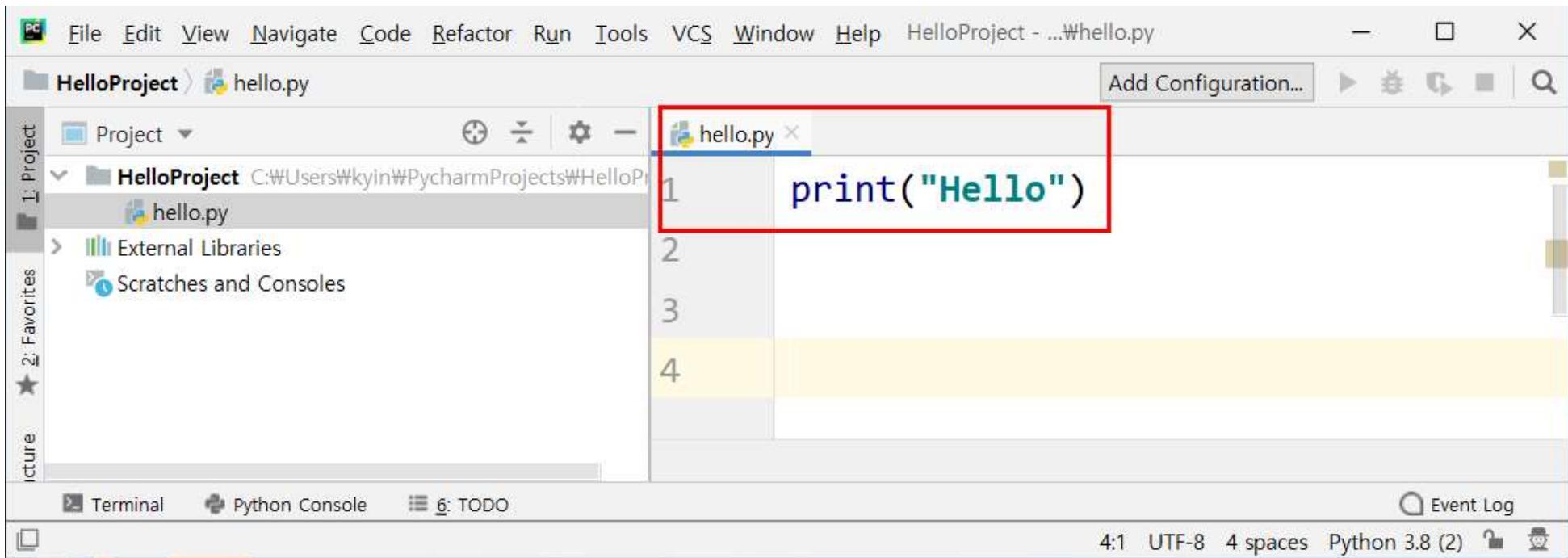
마. hello.py 파일 작성



5) PyCharm 실행

python

바. print("hello") 코드 작성



The screenshot shows the PyCharm IDE interface. The top menu bar includes File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, Help, and HelloProject - ...\\hello.py. The toolbar has icons for Add Configuration..., Run, Stop, and Refresh. The left sidebar shows a Project tree with a HelloProject folder containing a hello.py file, and sections for Favorites, External Libraries, and Scratches and Consoles. The main editor window displays the code:

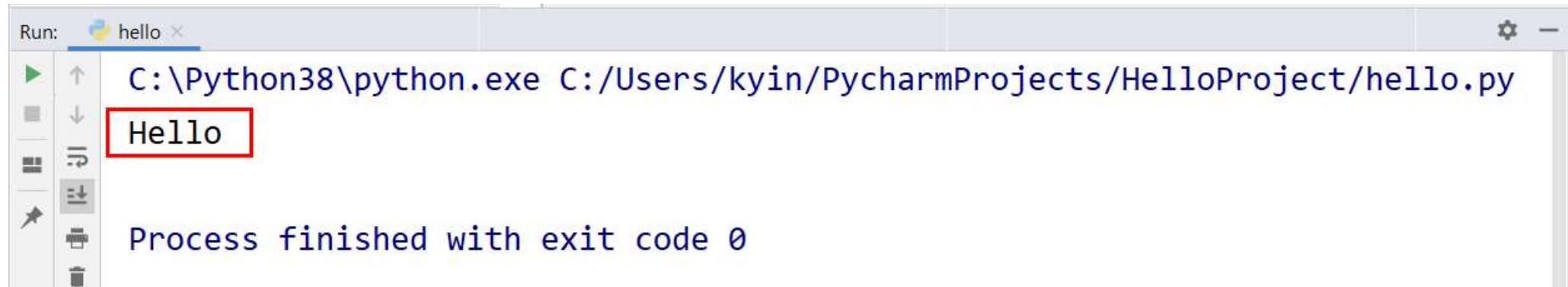
```
1 print("Hello")  
2  
3  
4
```

The line `print("Hello")` is highlighted with a red rectangle. The status bar at the bottom shows the file is 4:1, encoding is UTF-8, 4 spaces, Python 3.8 (2), and includes icons for Terminal, Python Console, TODO, and Event Log.

5) PyCharm 실행

python

사. 프로그램 실행 (단축키: ctrl + shift + F10)



The screenshot shows the PyCharm interface with the 'Run' tab selected. The title bar says 'Run: hello'. The run configuration dropdown shows 'C:/Python38/python.exe C:/Users/kyin/PycharmProjects>HelloProject/hello.py'. The output window displays the text 'Hello' in a red-bordered box and 'Process finished with exit code 0' below it.

2장. 자료형과 기본 문법



자료형(data type)

식별자(identifier)

변수 (variable)

데이터 타입 체크(type check)

표준 출력 및 포맷 설정(print)

표준 입력 (input)

1) 자료형 (Data Type)

python

가. 일반 자료형 (Scalar 타입)

- 정수 : 음수, 0, 양수로 구성된 숫자
- 실수 : 소수점을 가진 숫자
- 논리: True, False
- 함수: 가능한 코드를 표현
- None: 값이 없음 또는 null 의미

나. 집합 자료형

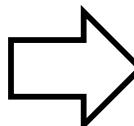
- 문자열(str) : 값 변경 불가 (immutable), “ ” 또는 ”” 또는 ””” ”””
- 리스트(list) : 중복 허용하고 값 변경 가능 (mutable), [] 표현
- 튜플(tuple) : 중복 허용하고 값 변경 불가 (immutable), () 표현
- 셋(set) : 중복 허용 불가, {} 표현, 반드시 immutable 값만 저장 가능
- 딕셔너리(dict) : key:value 형식 , {“key”:”value”} 표현

2) 기본 자료형 형태

python

일반 자료형

```
print("정수: ", 0)
print("정수(10진수): ", 10)
print("정수(2진수): ", 0b10)
print("정수(8진수): ", 0o10)
print("정수(16진수): ", 0x10)
print("정수: ", -10)
print("실수: ", 3.14)
print("실수(지수표기법): ", 3e+5)
print("논리: ", True)
print("논리: ", False)
print("None: ", None)
```

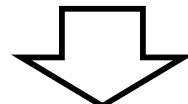


```
C:\Python38\python.exe C:/Us
정수:  0
정수(10진수):  10
정수(2진수):  2
정수(8진수):  8
정수(16진수):  16
정수:  -10
실수:  3.14
실수(지수표기법):  300000.0
논리:  True
논리:  False
None:  None
```

3) 집합 자료형 형태

python

```
# 집합 자료형  
print("문자열: ", "안녕하세요")  
print("문자열: ", '안녕하세요')  
print("문자열: ", """안녕하세요""")  
print("문자열: ", '''안녕하세요'''')  
print("리스트(list): ", [1, 2, 3, 4, 4, "홍길동"])  
print("셋(set): ", {1, 2, 3, 4, 4, "홍길동"})  
print("튜플(tuple): ", (1, 2, 3, 4, 4, "홍길동"))  
print("딕트(dict): ", {"name": "홍길동", "age": 20})
```



Run: sample01 dataType2_집합 ×

```
▶  C:\Users\kyin\PycharmProjects\01Day\venv\S  
▶  문자열: 안녕하세요  
▶  문자열: 안녕하세요  
▶  문자열: 안녕하세요  
▶  문자열: 안녕하세요  
▶  리스트(list): [1, 2, 3, 4, 4, '홍길동']  
▶  셋(set): {1, 2, 3, 4, '홍길동'}  
▶  튜플(tuple): (1, 2, 3, 4, 4, '홍길동')  
▶  딕트(dict): {'name': '홍길동', 'age': 20}
```

3) 집합 자료형 형태

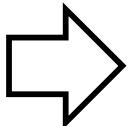
python

이스케이프 문자 (escape)

escape 문자	의미
'\n'	새로운 라인 (new line) 추가
'\t'	탭(tab) 선택한 기능
'\r'	리턴(return) 기능
'\b'	백스페이스(backspace) 기능
'\'	따옴표(single quotes) 기능
'\"'	쌍따옴표(double quotes) 기능
'\\'	백슬래쉬(backslash) 기능

이스케이프 문자

```
print("c:\\temp")
print("Hello\nworld")
print("Hellow\tworld")
print("\'")
print("\\")
```



Run: sample01 dataType3_이스케이프

▶	↑	c:\\temp
▶	↓	Hello
▶	⤵	world
▶	⤴	Hellow world
▶	🖨️	'
▶	⌫	"

이스케이프 문자 (escape)를 사용하지 않으려면 raw string 사용하면 된다.

4) 식별자 (identifier)

python

- 변수, 함수, 클래스, 모듈 또는 다른 개체를 식별하는데 사용하는 이름.
- 식별자는 영문자 및 _로 시작하고 숫자를 사용할 수 있다.
- 한글은 사용 가능하지만 권장 안함.
- @, \$, % 등의 특수문자 사용 불가.
- 대소문자 구별.
- 시스템 정의 식별자(키워드, 예약어)와 사용자 정의 식별자로 구분이 가능

○ 키워드 (keyword, 예약어)

False	await	else	import	pass
None	break	except	in	raise
True	class	finally	is	return
and	continue	for	lambda	try
as	def	from	nonlocal	while
assert	del	global	not	with
async	elif	if	or	yield

```
import keyword  
print('키워드 목록 :', keyword.kwlist)
```

5) 변수 (variable)

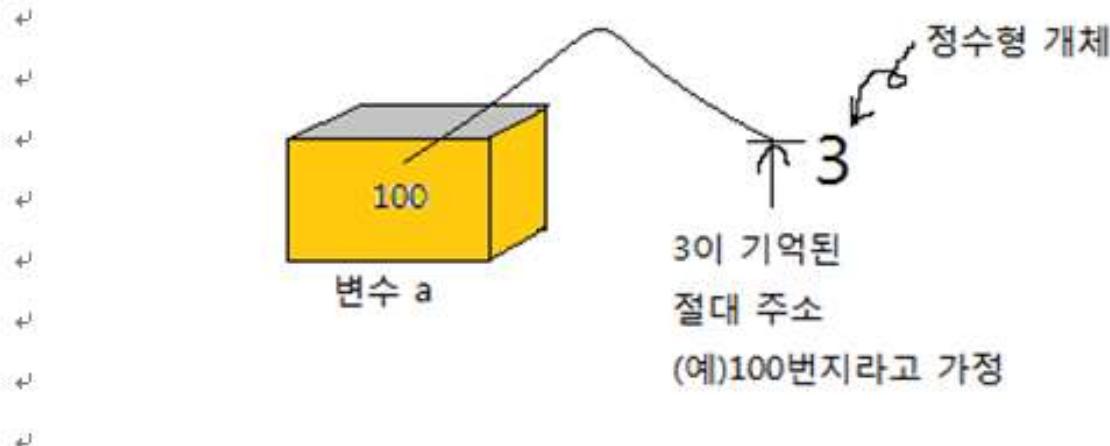
python

목적 및 특징:

데이터 저장.

모든 변수는 참조형 변수.

```
>>> a = 3
```



문법:

- 1) 변수명 = 값
- 2) 변수명=변수명2=변수명3=값
- 3) 변수명,변수명2,변수명3 = 값, 값2, 값3

5) 변수 (variable)

python

```
# 변수 사용
```

```
num = 4;
```

```
print(num, id(num))
```

```
num2 = 4
```

```
print(num2, id(num2))
```

```
name = "홍길동"
```

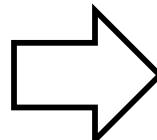
```
print(name, id(name))
```

```
name2 = "홍길동"
```

```
print(name2, id(name2))
```

```
print(num is num2)
```

```
print(name is name2)
```



Run: sample03_변수

▶	C:\Users\kyin\PyCh
↑	4 8791228379488
↓	4 8791228379488
⊸	홍길동 43266384
↶	홍길동 43266384
✖	True
✖	True

5) 변수 (variable)

python

```
# 동시에 변수 할당
```

```
a=b=c=1
```

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

```
# 동시에 여러 변수에 값 할당
```

```
name,age,address = "홍길동",20,"서울"
```

```
print(name,age,address)
```

```
(name,age,address) = ("홍길동",20,"서울")
```

```
print(name,age,address)
```

```
[name,age,address] = ["홍길동",20,"서울"]
```

```
print(name,age,address)
```



Run: sample03_변수2 ×

C:\Users\kyin\Py

1 1 1

홍길동 20 서울

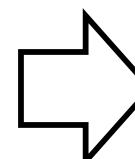
홍길동 20 서울

홍길동 20 서울

5) 변수 (variable)

python

```
# 집합형 데이터  
listValue = [10, 20, 30]  
tupleValue = (10, 20, 30)  
setValue = {10, 20, 30}  
dictValue = {'key': 100}  
strValue = "hello"  
  
listNestedValue = [10, 20, [100, 200]]  
listNestedValue2 = [10, 20, (100, 200)]  
dictNestedValue = {"key": [10, 20]}  
  
print(listValue)  
print(tupleValue)  
print(setValue)  
print(dictValue)  
print(strValue)  
print(listNestedValue)  
print(listNestedValue2)
```



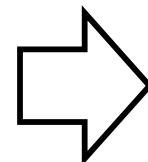
```
Run: sample03_변수3_집합형  
C:\Users\kyin\Pychar  
[10, 20, 30]  
(10, 20, 30)  
{10, 20, 30}  
{"key": 100}  
hello  
[10, 20, [100, 200]]  
[10, 20, (100, 200)]  
{"key": [10, 20]}
```

6) type 체크

python

```
# type 체크
```

```
a = 10  
b = 3.14  
c = True  
d = [10,20,30]  
e = (10,20,30)  
f = {10,20,30}  
g = {'key':100}  
h = "hello"
```



```
Run: sample04_타입체크  
C:\Users\kyin\PycharmProjects  
10 <class 'int'>  
3.14 <class 'float'>  
True <class 'bool'>  
[10, 20, 30] <class 'list'>  
(10, 20, 30) <class 'tuple'>  
{10, 20, 30} <class 'set'>  
{'key': 100} <class 'dict'>  
hello <class 'str'>
```

* tuple인 경우에는 값 한 개인 경우 주의할 것.

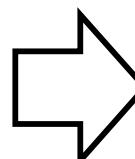
```
print(a,type(a))  
print(b,type(b))  
print(c,type(c))  
print(d,type(d))  
print(e,type(e))  
print(f,type(f))  
print(g,type(g))  
print(h,type(h))
```

```
print([10], type([10]))  
print({10}, type({10}))  
print((10), type((10)), type((10,)))  
[10] <class 'list'>  
{10} <class 'set'>  
10 <class 'int'><class 'tuple'>
```

7) 표준 출력 (print)

python

```
# 콘솔 출력  
help(print)  
  
print("Hello")  
print(100, 200) # 공백으로 자동 구분  
print(300)  
  
print(9, 8, sep=",")  
print(300)  
  
print(9, 8, sep=",", end=" ") # 9,8 300  
print(300)  
  
print(9, 8, sep=",", end=",") # 9,8,300  
print(300)  
  
print(1, 2, sep=",", end="\n\t")  
print(300)
```



Hello	
100 200	
300	
9,8	
300	
9,8 300	
9,8,300	
1,2	
300	

8) 포맷 출력

python

- help('FORMATTING')로 포맷팅 사용법 확인 가능

문법 1: 위치값 이용

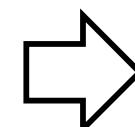
변수명 = “포맷 문자열 … {0} {1} …”.format(값1, 값2, …)

1. 위치값으로 접근

```
mesg = "이름: {}".format('홍길동')
print(mesg)
```

```
mesg = '이름: {}'.format('홍길동')
print(mesg)
```

```
mesg = '이름:{0}, 나이:{1}'.format('홍길동', 20)
print(mesg)
```



Run	Output
1	이름: 홍길동
2	이름: 홍길동
3	이름:홍길동, 나이:20

8) 포맷 출력

python

문법 2: name(key) 이용

변수명 = “포맷 문자열 … {key} {key2} …”.format(key=값1, key2=값2, …)

2. key 로 접근

```
mesg = '이름: {username}, 나이: {age}'.format(username='홍길동', age=20)
print(mesg)
```

혼합 가능

```
mesg = '이름: {0}, 나이: {age}'.format('홍길동', age=20)
print(mesg)
```



Run: sample06_포맷출력2_key

- ▶ ⏹ C:\Users\kyin\PycharmPro
- ➡ ⓘ 이름: 홍길동, 나이: 20
- ➡ ⓘ 이름: 홍길동, 나이: 20

8) 포맷 출력

python

문법 3: item 이용

변수명 = “포맷 문자열 … {0[0]} {0[1]} …”.format(리스트|튜플)

```
# 3. item으로 접근 ==> set 불가
coord = (3, 5) # 튜플
mesg = 'X: {0[0]}; Y: {0[1]}'.format(coord)
print(mesg)
```

```
coord2 = [3, 5] # List
mesg2 = 'X: {0[0]}; Y: {0[1]}'.format(coord2)
print(mesg2)
```



Run: sample06_포맷출력3_itemy ×

▶	↑	C:\Users\kyin\Pyc
◀	↓	X: 3; Y: 5
─	⟲	X: 3; Y: 5

8) 포맷 출력

python

문법 4: unpacking argument sequence

변수명 = “포맷 문자열 … {0} {1} …”.format(*’문자열|리스트’)

```
# unpacking argument sequence
mesg = '{0}:{1}:{2}'.format(*'홍길동')
print(mesg)
```

```
mesg = '{0}:{1}:{2}'.format(*['홍길동', '이순신', '강감찬'])
print(mesg)
```



홍:길:동

홍길동:이순신:강감찬

8) 포맷 출력

python

문법 5: unpacking argument sequence

변수명 = “포맷 문자열 … {key} {key2} …”.format(**dict타입)

```
# unpacking argument sequence
person = {'username': '홍길동', 'age': 20}
mesg = '이름: {username}, 나이: {age}'.format(**person)
print(mesg)
```



```
Run: sample06_포맷출력1_위치값 ×
C:\Users\kyin\PycharmPro
이름: 홍길동, 나이: 20
```

8) 포맷 출력

python

문법 6: 수치 데이터 표현

변수명 = “포맷 문자열 … {0:f} {0:d}{0:.4f} …”.format(정수|실수)

```
# 수치 데이터 표현  
mesg = '{0:f}'.format(123456789)  
print(mesg) #123456789.000000
```

```
mesg = '{0:f},{0:d}'.format(123456789)  
print(mesg) # 123456789.000000,123456789
```

```
mesg = '{0:.4f},{0:11d}'.format(123456789)  
print(mesg) #123456789.0000, 123456789
```

```
mesg = '{0:.3f},{0:.9f}'.format(123.4567)  
print(mesg) #123.457,123.456700000
```

```
mesg = '{0:,}'.format(123456789)  
print(mesg) # 123,456,789
```

```
mesg = '{0:+f},{0:-f}'.format(123456789)  
print(mesg) #+123456789.000000,123456789.000000
```

```
mesg = '{0:+f},{1:-f}'.format(123456789,-98765431)  
print(mesg) # +123456789.000000,-98765431.000000
```



```
Run: sample06_포맷출력4_수치  
C:\Users\kyin\PycharmProjects\01Day  
123456789.000000  
123456789.000000,123456789  
123456789.0000, 123456789  
123.457,123.456700000  
123,456,789  
+123456789.000000,123456789.000000  
+123456789.000000,-98765431.000000
```

8) 포맷 출력

python

문법 7: 정렬 (왼쪽, 가운데, 오른쪽)

변수명 = “포맷 문자열 … {0:>10} {0:<10}{0:^10} …”.format(‘hi’)

10자리 오른쪽 정렬

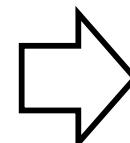
```
mesg = "{0:>10}".format("hi")
print(mesg)
```

10자리 왼쪽 정렬

```
mesg = "{0:<10}".format("hi")
print(mesg)
```

10자리 가운데 정렬

```
mesg = "{0:^10}".format("hi")
print(mesg)
```



```
Run: sample06_포맷출력5_정렬
C:\Users\kyin\F
          hi
          hi
          hi
```

9) 표준 입력

python

문법 7: 표준 입력

```
변수명 = input("이름 입력")
```

```
# 표준 입력
```

```
name = input("이름을 입력하시오\n")
age = int(input("나이를 입력하시오.\n"))
print("입력된 이름:", name)
print("입력된 나이:", age)
```



```
Run: sample07_표준입력 ×
▶ C:\Users\kyin\Pycharm
↑ 이름을 입력하시오
↓ 흥길동
इनपुट: 나이를 입력하시오.
→ 20
↙ 입력된 이름: 흥길동
↙ 입력된 나이: 20
```

3장. 연산자



산술 연산자

대입연산자 및 활용

비교 연산자

논리 연산자 및 활용

Bitwise 연산자

멤버쉽 연산자

1) 산술 및 대입 연산자

python

종 류	사용 예	
$+$	더하기	$c = a + b, c += a$
$-$	빼기	$c = a - b, c -= a$
$*$	곱하기	$c = a * b, c *= a$
$/$	나누기	$c = a / b, c /= a$
$\%$	나머지 반환	$c = a \% b, c \%= a$
$//$	몫을 반환	$c = a // b, c //= a$
$**$	지수 승	$c = a ** 3, c **= a$

1. 산술 연산자

```
a = 10  
b = 3  
  
print(a+b)  
print(a-b)  
print(a*b)  
print(a/b) #3.333333333333335  
print(a%b)  
print(a//b) # 3 , 소수점 버림  
print(a**b)
```



Run: sample08_연산자 x
C:\Users\kyin\Pycha
13
7
30
3.333333333333335
1
3
1000

1) 산술 및 대입 연산자

python

```
# 2. 대입 연산자
n = 10
n2 = 4
n3 = 10
n += n2
print(n,n2)
n -= n2
print(n,n2)
n *= n2
print(n,n2)
n /= n2
print(n,n2)
n // n2
print(n,n2)

n3 // n2
print(n3,n2)

n3 **= n2
print(n3,n2)
```



Run: sample08_연산자 ×

▶	↑	C:\Users\kyin'
◀	↓	14 4
─	⤒	10 4
─	⤓	40 4
✖	⤔	10.0 4
✖	⤖	2.0 4

2) 대입 연산자 활용

python

```
# 1) 동시에 변수 할당
```

```
a = b = c = 1
```

```
# 2) 복수 변수에 복수값 설정
```

```
v1, v2 = 10, 20
```

```
print(v1, v2)
```

```
v2, v1 = v1, v2
```

```
print(v1, v2)
```

```
v1, v2 = {'a': 100, 'b': 200}
```

```
print(v1, v2) # a b
```

```
m = {'a': 100, 'b': 200}
```

```
v1, v2 = m;
```

```
print(m[v1], m[v2]) # 100 200
```



Run: sample08_연산자2_대입연산자용

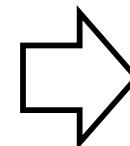
▶	↑	C:\Users\kyin\Py
■	↓	10 20
---	⤒	20 10
---	⤓	a b
➤	🖨️	100 200

2) 대입 연산자 활용

python

```
# 3) 변수와 값의 갯수가 불일치한 경우는 * 사용한다.  
#     남은 요소 전체를 리스트에 담는다.
```

```
v1, *v2 = [10, 20, 30]  
print(v1, v2) # 10 [20, 30]  
v1, *v2 = (10, 20, 30)  
print(v1, v2) # 10 [20, 30]  
v1, *v2 = {10, 20, 30}  
print(v1, v2) # 10 [20, 30]  
v1, *v2 = {'a':100, 'b':200, 'c':300}  
print(v1, v2) # a ['b', 'c']  
  
*v1, v2 = [100, 200, 300]  
print(v1, v2) # [100, 200] 300  
  
*v1, v2, v3 = [100, 200, 300]  
print(v1, v2, v3) # [100] 200 300
```



Run: sample08_연산자2_대입연산자.py

▶	↑	C:\Users\kyin\Py
■	↓	10 [20, 30]
☰	⤒	10 [20, 30]
⤓	⤓	10 [20, 30]
🖨️	🖨️	a ['b', 'c']
⬇	⬇	[100, 200] 300
⬇	⬇	[100] 200 300

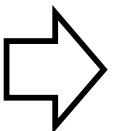
3) 비교 연산자

python

종 류	기 능
$A > B$	A가 B 보다 크다.
$A < B$	A가 B 보다 작다.
$A == B$	A와 B가 서로 같다.
$A <= B$	A가 B와 같거나 작다.
$A >= B$	A가 B와 같거나 크다.
$A != B$	A는 B와 같지 않다.

```
# 비교 연산자
```

```
k = 10
k2 = 5
print( k == k2 )
print( k != k2 )
print( k > k2 )
print( k >= k2 )
print( k < k2 )
print( k <= k2 )
```



Run: sample08_연산자3
C:\Users\
False
True
True
True
False
False

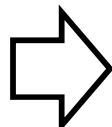
* is vs == 연산자
1) $a == b$: a 와 b 의 값을 비교한다.
2) $a is b$: a 와 b 의 identities를 비교한다.
($id(a) == id(b)$ 와 동일)

4) 논리 연산자

python

종 류	사용 예
x and y	논리 곱 if a and b
x or y	논리 합 if a or b
not x	논리 부정 if not(a and b)

```
# 논리 연산자의 논리식
print(True and True)
print(True and False)
print(False and True)
print(False and False)
print()
print(True or True)
print(True or False)
print(False or True)
print(False or False)
print()
print(not True)
print(not False)
```



Run: sample08_연산자4_논리
C:\Users\kyi:
True
False
False
False
True
True
True
False

False
True

5) 논리 연산자 활용

python

자바스크립트 언어와 동일하게 True 및 False 인 논리값이 아니어도 논리값으로 처리가 된다.

논리 연산자 활용

```
# bool 함수
print()
print(bool(0))
print(bool(''))
print(bool(None))
print(bool([]))
print(bool(()))
print(bool({}))
print()
print(bool(1))
print(bool('hello'))
print(bool(3.14))
print(bool([1,2,3]))
print(bool((1,2,3)))
print(bool({1,2,3}))
```

```
Run: sample08_연산자4
C:\Users\
▶ False
▶ True
```

일반 데이터의 논리값 적용

```
print("공백문자:", not "")           공백문자: True
print("0 문자:", not 0)              0 문자: True
print("None:", not None)            None: True
print("[]:", not [])                []: True
print("(()):", not ())              (): True
print("{}:", not {})                {}: True
print()
print("일반문자:", not "Hello")    일반문자: False
print("10 문자:", not 10)            10 문자: False
```

6) Bitwise 연산자

python

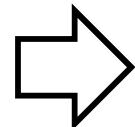
비트 단위의 연산을 하는데 사용된다.

```
# Bitwise 연산자

a = 8      # 0000 1000
b = 11     # 0000 1011

c = a & b    # AND    8
d = a | b    # OR     11
e = a ^ b    # XOR    3
f = a<<2    # shift   32

print("AND:", c, bin(c))
print("OR:", d, bin(d))
print("XOR:", e, bin(e))
print("shift:", f, bin(f))
```



Run: sample08_연산자5_ 비트와이즈연산자

▶	↑	C:\Users\kyin\Pychar
■	↓	AND: 8 0b1000
━	⤒	OR: 11 0b1011
⤓	⤔	XOR: 3 0b11
⤕	⤖	shift: 32 0b100000

7) 멤버쉽 연산자

python

문법 : 집합형 데이터안에 지정된 값이 포함되어 있는지 체크

변수명 = 값 in 집합형

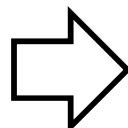
```
# 멤버쉽 연산자
```

```
n = [10, 9, 8]
result = 10 in n
print("10포함 여부: ", result)
```

```
n = (10, 9, 8)
result = 10 in n
print("10포함 여부: ", result)
```

```
n = {10, 9, 8}
result = 10 in n
print("10포함 여부: ", result)
```

```
n = {"name": "홍길동", "age": 20}
result = "name" in n
print("name 키포함 여부: ", result)
```



```
Run: sample08_연산자6_멤버쉽연산자 ×
C:\Users\kyin\PycharmPro
10포함 여부: True
10포함 여부: True
10포함 여부: True
name 키포함 여부: True
```

4장. 집합 자료형



문자열 자료형

리스트(list)

튜플(tuple)

셋(set)

딕트(dict)

집합 자료형 변환

1) 문자열 (str type)

python

문법 : 변수명 = “문자열값”

```
m = 'hello'  
m = "hello"  
m = '''hello'''  
m = """hello"""
```

Strings

Python 3.x Version ≥ 3.0

- **str**: a **unicode string**. The type of 'hello'
- **bytes**: a **byte string**. The type of b'hello'

- ‘ ’ 또는 “ ” 또는 “”” “”” 또는 “” “ ” 사용
- 생성된 문자열은 변경이 불가능하다. (immutable)
- 순서형(sequence)으로서 문자열 내 문자들은 순서를 갖는다.
- 첨자(index)를 사용하여 특정 문자 접근이 가능하다. (0부터 시작)
- 일부분의 문자열 추출이 가능한 slice기능이 제공된다.
- 다양한 형태의 함수가 제공된다.

- b'foo bar': results **bytes** in Python 3, **str** in Python 2
- u'foo bar': results **str** in Python 3, **unicode** in Python 2
- 'foo bar': results **str**
- r'foo bar': results so called raw string, where escaping special characters is not necessary, everything is taken verbatim as you typed

```
escaped = 'foo\\nbar' # foo\nbar  
raw      = r'foo\\nbar' # foo\nbar
```

1) 문자열 (str type)

python

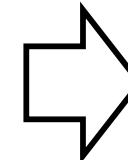
함수명	역 할
a.count(x)	문자열 a 중 x와 일치하는 문자의 갯수를 반환한다.
a.find(x)	문자열 a 중 문자 x가 처음으로 나온 위치를 반환. 없으면 -1을 반환.
a.rfind(x)	문자 x가 처음으로 나온 위치를 뒤에서부터 찾는다.
a.index(x)	문자열 a 중 문자 x가 처음으로 나온 위치를 반환. 없으면 에러 발생.
's'.join(a)	문자열 a 중 각각의 요소 문자 사이에 문자열 s를 삽입한다.
a.lower()	문자열 a를 모두 소문자로 바꾸어 준다.
a.upper()	문자열 a를 모두 대문자로 바꾸어 준다.
a.lstrip()	문자열 a의 왼쪽 공백을 모두 지운다.
a.rstrip()	문자열 a의 오른쪽 공백을 모두 지운다.
a.strip()	문자열 a의 양쪽 공백을 모두 지운다.
a.replace(s, r)	문자열 a의 s라는 문자열을 r이라는 문자열로 치환한다.
a.split('sep')	문자열 a를 구분자로 나누어 리스트값을 반환한다.
a.swapcase()	문자열 a의 대문자는 소문자로, 소문자는 대문자로 변경한다.
a.startswith('H')	H라는 문자로 시작되는지 여부를 True, False로 반환한다.
len(a)	문자열 a의 크기(길이)를 반환해 준다.

1) 문자열 (str type)

python

1. 문자열 함수

```
s = "sequence"
print("길이:", len(s))
print("포함횟수:", s.count('e'))
print()
print("검색위치1:", s.find('e'))
print("검색위치2:", s.find('e', 2))
print("검색위치3:", s.rfind('e'))
print()
print('첫 글자 유무 : ', s.startswith('s'), s.startswith('a'))
print("소문자로:", "HeLLo".lower())
print("대문자로:", "HeLLo".upper())
print("swap 문자로:", "HeLLo".swapcase())
print("    HeLLo    ".lstrip())
print(len("    HeLLo    ".lstrip()))
print("    HeLLo    ".rstrip())
print("    HeLLo    ".strip())
print(len("    HeLLo    ".strip()))
print()
print("문자열 변경:", "HeLHO".replace('H', 'A'))
print("구분자:", "aaa/bbb/ccc".split("/"))
print("join:", ",".join(["A", "B", "C"]))
```



Run: sample09_집합형1_문자열

```
C:\Users\kyin\PycharmProjects
길이: 8
포함횟수: 3

검색위치1: 1
검색위치2: 4
검색위치3: 7

첫 글자 유무 : True False
소문자로: hello
대문자로: HELLO
swap 문자로: hEllo
HeLLo
9
HeLLo
HeLLo
5

문자열 변경: AeLAO
구분자: ['aaa', 'bbb', 'ccc']
join: A,B,C
```

1) 문자열 (str type)

python

2. 문자열 slice

```
m = "대한민국"  
print(m[0])  
print(m[-1])  
print(m[0:3])  
print(m[1:])  
print(m[:2])  
print(m[-3:-1])  
print(m[0:3:2]) #마지막에 증가값을 주면 증가값 만큼 건너뛰며 출력된다  
print(m[:]) #전체를 출력한다  
print(m*2) # 반복출력
```



대
국
대한민
한민국
대한
한민
대민
대한민국
대한민국대한민국

2) 리스트 (list type)

python

문법 : 변수명 = [값, 값1, 값2, ...]

m = [10,20,30]

m = ["이순신", "홍길동", "유관순"]

m = [10,20,30,10,20]

m = ["이순신", 20, 3.14, True]

m = []

- 대괄호([]) 사용하여 서로 다른 데이터 저장 가능.
- 순서가 있고 중복 허용
- 데이터 변경 가능한 mutable한 특징 (Built-in mutable sequence)

2) 리스트 (list type)

python

```
# 2 . 리스트
```

```
m = [1,2,3,4]
m2 = []
print(m)
print(m2)

print(m*2) # 반복 출력
```

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

```
# 리스트 함수
```

```
# 1) 리스트 추가
```

```
m = [1,2,3]
m.append(10)
m.append([9,8])
m.append((100,200))
print("append:",m)
```

```
m.insert(0,100)
m.insert(0,[4,5,6])
print("insert:",m)
```

```
x = [1,2,3]
x.extend([10])
x.extend([9,8])
x.extend("XYZ")
x.extend((7,)) # tuple
print("extend:",x)
```

```
x2 = [1,2,3]
x2 += [4,5]
print("+로 추가:", x2)
print()
```

```
append: [1, 2, 3, 10, [9, 8], (100, 200)]
insert: [[4, 5, 6], 100, 1, 2, 3, 10, [9, 8], (100, 200)]
extend: [1, 2, 3, 10, 9, 8, 'X', 'Y', 'Z', 7]
+로 추가: [1, 2, 3, 4, 5]
```

```
# append와 extend 차이점
```

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

```
a.append([4,5,6]) # 리스트의 갯수 4개
```

```
a2 = [1,2,3]
```

```
a2.extend([4,5,6]) # 리스트의 갯수 6개
```

```
print(a , len(a))
```

```
print(a2 , len(a2))
```

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

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

2) 리스트 (list type)

python

2) 리스트 함수

```
print("리스트 길이:", len([1,2,3,4]))  
print("포함 갯수:", [10,2,3,2,9,2].count(2))  
x3 = [100,200,300]  
print("특정 위치:", x3.index(200))  
print("포함 여부1:", 9 in [9,8,7])  
print("포함 여부2:", 6 in [9,8,7])  
print()
```

리스트 길이: 4

포함 갯수: 3

특정 위치: 1

포함 여부1: True

포함 여부2: False

3) slice

```
kk = [1,2,3,4,5]  
print("특정 값:", kk[0]) # 1  
print("맨뒤 값:", kk[-1]) # 5  
print("slice:", kk[0:3]) # [1, 2, 3]  
print("slice:", kk[:6]) # [1, 2, 3, 4, 5]  
print("slice:", kk[3:]) # [4, 5]  
print("slice:", kk[:]) # [1, 2, 3, 4, 5]  
print("slice(배수):", kk[::-2]) # [1, 3, 5]  
print("slice(reverse):", kk[::-1]) # [5, 4, 3, 2, 1]  
print()
```



특정 값: 1

맨뒤 값: 5

slice: [1, 2, 3]

slice: [1, 2, 3, 4, 5]

slice: [4, 5]

slice: [1, 2, 3, 4, 5]

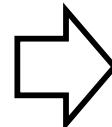
slice(배수): [1, 3, 5]

slice(reverse): [5, 4, 3, 2, 1]

2) 리스트 (list type)

python

```
# 4) 값 출력  
kk2 = [1,2,3]  
kk2[0] = 100 # 값 변경 가능  
print("값 변경1:" , kk2)  
kk3 =[1,2,3,[9,8,7]]  
kk3[3][0]=900  
print("값 변경2:" , kk3)  
print("특정 값 출력1:" , kk3[3][0])  
print("특정 값 출력2:" , kk3[3][2:])  
print("특정 값 출력3:" , kk3[3][:2])
```



```
# 5) 값 삭제  
kk4 = [1,2,3,4]  
kk4.pop()  
print("pop 함수",kk4)  
print()  
kk4.remove(2) # 값에 의한 삭제  
print(kk4)  
del kk4[0] # 위치에 의한 삭제  
print(kk4)  
print()
```

```
값 변경1: [100, 2, 3]  
값 변경2: [1, 2, 3, [900, 8, 7]]  
특정 값 출력1: 900  
특정 값 출력2: [7]  
특정 값 출력3: [900, 8]  
pop 함수 [1, 2, 3]  
[1, 3]  
[3]
```

2) 리스트 (list type)

python

```
# 6) 정렬
xxx = [88,2,5,3,9,7,10]
xxx.sort()
print("오름차순 정렬:", xxx)
xxx.sort(reverse=True)          #역순으로 정렬
print("내림차순 정렬:", xxx)
# ) 정렬2
yyy = ['123', '99', '43']
yyy.sort()
print("기본 문자열 정렬:", yyy)
yyy.sort(key=int)
print("문자형을 수치형으로 변환하여 정렬:", yyy) #문자열이지만 숫자형으로 정렬
```

오름차순 정렬: [2, 3, 5, 7, 9, 10, 88]
내림차순 정렬: [88, 10, 9, 7, 5, 3, 2]
기본 문자열 정렬: ['123', '43', '99']
문자형을 수치형으로 변환하여 정렬: ['43', '99', '123']

```
# 6) 정렬3 ==> 정렬 결과 저장
zzz = [3,1,2]
zzz2 = sorted(zzz)
print("정렬전:", zzz)
print("정렬후:", zzz2)
```

정렬전: [3, 1, 2]
정렬후: [1, 2, 3]

2) 리스트 (list type)

python

7) 참조형

```
x = [10, 20, 30]
x2 = x
print("변경전 x값:", x, id(x))
print("변경전 x2값:", x2, id(x2))
x[0]=9
print("변경후 x값:", x)
print("변경후 x2값:", x2)
print()
```

```
변경전 x값: [10, 20, 30] 2454789250632
변경전 x2값: [10, 20, 30] 2454789250632
변경후 x값: [9, 20, 30]
변경후 x2값: [9, 20, 30]
```

8) 복사방법 : copy(), list(), [:]

```
x = [9, 8, 7]
x2 = x.copy()
x3 = list(x)
x4 = x[:]
x[1]=199
print("원본 ", x)
print("복사본1 ", x2)
print("복사본2 ", x3)
print("복사본3 ", x4)
```

```
원본 [9, 199, 7]
복사본1 [9, 8, 7]
복사본2 [9, 8, 7]
복사본3 [9, 8, 7]
```

2) 리스트 (list type)

python

```
# 8) 필터  
#help(filter)  
x = [1,2,3,4,5]  
data = filter(lambda n:n%2==0 , x)  
print(list(data)) # [2, 4]  
  
x = ["홍","정조","순조","세종대왕","강감찬"]  
data = filter(lambda n: len(n)== 2 , x)  
print(list(data)) # ['정조', '순조']
```

3) 튜플 (tuple type)

python

- 소괄호(()) 사용하여 서로 다른 데이터 저장 가능.
- 순서가 있고 중복 허용, 따라서 리스트와 유사
- 데이터 변경 불가능한 immutable한 특징 (Built-in immutable sequence. 값 추가 및 수정, 삭제 불가) 따라서 자료변경을 위해 사용할 만한 메서드를 지원하지 않는다.
(append,insert,remove,pop,sort 지원 안됨)

3 . 튜플 (tuple)

```
m = (1,2,3,4)
m2 = ()
print(m)
print(m2)

print(m*2) # 반복 출력
```

(1, 2, 3, 4)

()

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

튜플 함수

```
# 1) 튜플 추가 및 삽입, 삭제불가, 정렬 불가
# 2) 튜플 함수
print("리스트 길이:" , len((1,2,3,4)))
print("포함 갯수:" , (10,2,3,2,9,2).count(2))
x3 = (100,200,300)
print("특정 위치:" , x3.index(200))
print("포함 여부1:" , 9 in (9,8,7))
print("포함 여부2:" , 6 in (9,8,7))
print()
```

리스트 길이: 4

포함 갯수: 3

특정 위치: 1

포함 여부1: True

포함 여부2: False

3) 튜플 (tuple type)

python

```
# 3) slice  
kk = (1,2,3,4,5)  
print("특정 값:", kk[0])  
print("slice:", kk[0:3])  
print("slice:", kk[:6])  
print("slice:", kk[3:])  
print("slice:", kk[:])  
print()
```

특정 값: 1
slice: (1, 2, 3)
slice: (1, 2, 3, 4, 5)
slice: (4, 5)
slice: (1, 2, 3, 4, 5)

```
# 4) 확장 출력  
kk2 = (1,2,3)  
print("특정 값 출력:", kk2[0], kk2[1], kk2[2] )  
#kk2[0] = 100 # 값 변경 불가  
print("값 변경1:", kk2)  
kk3 =(1,2,3,(9,8,7))  
print("값 변경2:", kk3)  
print("특정 값 출력1:", kk3[3][0])  
print("특정 값 출력2:", kk3[3][2:])  
print("특정 값 출력3:", kk3[3][:2])
```

특정 값 출력: 1 2 3
값 변경1: (1, 2, 3)
값 변경2: (1, 2, 3, (9, 8, 7))
특정 값 출력1: 9
특정 값 출력2: (7,)
특정 값 출력3: (9, 8)

3) 튜플 (tuple type)

python

```
# 5) list로 변환  
x = (10,20,30)  
x2 = list(x)  
print(x,x2)  
print()  
x3 = tuple(x2)  
print(x2,x3)  
print()
```

(10, 20, 30) [10, 20, 30]

[10, 20, 30] (10, 20, 30)

```
# 6) x/변수  
a = (10,20)  
print(a)  
print()  
x,y=(100,200)  
print(x,y)  
print()  
t = x,y  
print(t) #t는 튜플형 변수가 된다
```

(10, 20)

100 200

(100, 200)

4) 셋 (set type)

python

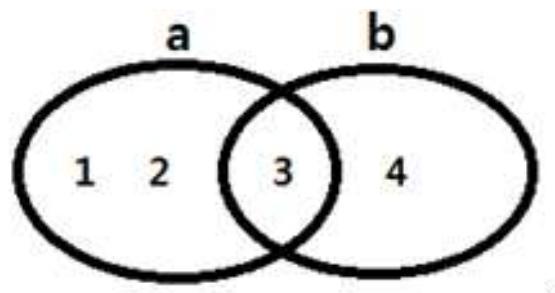
- 중괄호({}) 사용하여 서로 다른 데이터 저장 가능.
- 순서가 없고 중복 불가 (Build an unordered collection of unique elements.)
- 요소의 추가, 수정, 삭제를 위한 함수 지원
- set 함수 지원 (union(), intersection(), difference())
- 생성된 셋에 추가로 데이터 저장이 불가능한 Frozen Set이 제공된다.

4 . 셋 (set)

```
m = {1, 2, 3, 1}  
print(m)
```



```
C:\Python38\python.  
{1, 2, 3}
```



4) 셋 (set type)

python

```
# set 함수  
# 1) set 추가  
m = {1,2,3}  
m.add(10)  
m.add((9,8)) # tuple 만 가능  
print("add:",m)
```

2) set 변경 ==> 내부적으로 union으로 처리된다.

```
m2 = {1,2}  
m2.update({3,4})  
print("update1_set:" , m2)  
m2.update([5,6,7])  
print("update2_list:" , m2)  
m2.update((9,10))  
print("update3_tuple:" , m2)
```

3) set 삭제 ==>

```
m2 = {1,2,3,4}  
m2.discard(4) # If the element is not a member, do nothing.  
print(m2)  
m2.remove(3) # If the element is not a member, raise a KeyError.  
print(m2)
```

```
add: {1, 2, 3, 10, (9, 8)}  
update1_set: {1, 2, 3, 4}  
update2_list: {1, 2, 3, 4, 5, 6, 7}  
update3_tuple: {1, 2, 3, 4, 5, 6, 7, 9, 10}  
{1, 2, 3}  
{1, 2}
```

4) 셋 (set type)

python

```
# 4) set 풀수
print("set 길이:" , len({1,2,3,4,5}))
x3 = {100,200,300}
print("포함 여부1:" , 9 in {9,8,7})
print("포함 여부2:" , 6 in {9,8,7})
print()
x3.clear()
print(x3)
print()
```

```
set 길이: 5
포함 여부1: True
포함 여부2: False

set()
```

```
# 5) 혼별화
p = {100,200,300}
p2 = tuple(p)
p3 = list(p)
print("tuple로 :" , p2)
print("list로 :" , p3)

# 중복제거 제거
kkk = [1,3,4,3,1,4]
print("중복제거 전" , kkk )
kkk2 = set(kkk)
print("중복제거 후:" , kkk2 )
```

```
tuple로 : (200, 100, 300)
list로 : [200, 100, 300]
중복제거 전 [1, 3, 4, 3, 1, 4]
중복제거 후: {1, 3, 4}
```

4) 셋 (set type)

python

```
# 6) set 연산자 힙수
```

```
a = {1,2,3,1}
b = {3,4}
print("union:" , a.union(b))
print("intersection:" , a.intersection(b))
print("difference:" , a.difference(b))
```

```
union: {1, 2, 3, 4}
```

```
intersection: {3}
```

```
difference: {1, 2}
```

5) 딕트 (dict type)

python

- 중괄호({}) 내에 {'key':'value'} 쌍으로 데이터를 관리.
- 순서가 없기 때문에 전체 자료 출력시 순서 없이 불규칙하게 출력된다.
- key는 immutable 형태

```
# 5. 딕트 ( dict )
m = dict({'name':'홍길동', 'age':20})
m2 = dict(name='홍길동', age = 20)
m3 = dict()
print(m)
print(m2)
print(m3)
print()
```

```
{'name': '홍길동', 'age': 20}
{'name': '홍길동', 'age': 20}
{}
```

```
# dict 함수
# 1) dict 추가
coffee = {'espresso':'에스프레소', 'latte':'라떼'}
print(coffee)
coffee['lungo'] = '롱고'      #요소를 추가
print("요소추가:", coffee)

# 2) dict 변경 ==> 내부적으로 union으로 처리된다.
coffee['latte'] = '라떼2'
print("요소변경:", coffee)

# 3) dict 삭제 ==> discard & remove
del coffee['lungo']          #요소를 삭제
print("요소삭제:", coffee)
coffee.clear()                 #요소 전체 삭제
print("요소전체삭제:", coffee)
```

```
{'espresso': '에스프레소', 'latte': '라떼'}
요소추가: {'espresso': '에스프레소', 'latte': '라떼', 'lungo': '롱고'}
요소변경: {'espresso': '에스프레소', 'latte': '라떼2', 'lungo': '롱고'}
요소삭제: {'espresso': '에스프레소', 'latte': '라떼2'}
요소전체삭제: {}
```

5) 딕트 (dict type)

python

```
# dict 함수
coffee = {'espresso': '에스프레소', 'latte': '라떼'}
print(coffee)
print(len(coffee))          #길이 출력
print(coffee['espresso'])    #키를 이용해 값을 조회
print(coffee.get('latte'))   #key로 value 얻기
print(coffee.keys())         #key 목록 출력
print(coffee.values())       #value 목록 출력
print(coffee.items())        #key, value 출력

print()
print(list(coffee.keys()))   #key를 list type으로 출력
keys = list(coffee.keys())
print(keys[0])
print(list(coffee.values())) #value를 list type으로 출력

print()
print('latte' in coffee)     #in을 이용해 자료의 유무를 판단
print('cola' in coffee)      #없으면 False를 반환
```

```
{'espresso': '에스프레소', 'latte': '라떼'}
2
에스프레소
라떼
dict_keys(['espresso', 'latte'])
dict_values(['에스프레소', '라떼'])
dict_items([('espresso', '에스프레소'), ('latte', '라떼')])

['espresso', 'latte']
espresso
['에스프레소', '라떼']

True
False
{'espresso': '에스프레소', 'latte': '라떼', 'lungo': '롱고'}
{'espresso': '에스프레소', 'latte': '라떼'}
{}
```

6) 집합 자료형 형변환

python

```
# list -> tuple, set  
a = [10,20,30]  
a2 = tuple(a)  
a3 = set(a)  
print(a)  
print(a2)  
print(a3)
```

```
[10, 20, 30]  
(10, 20, 30)  
{10, 20, 30}
```

```
# set -> list, tuple  
a = {10,20,30}  
a2 = list(a)  
a3 = tuple(a)  
print(a)  
print(a2)  
print(a3)
```

```
{10, 20, 30}  
[10, 20, 30]  
(10, 20, 30)
```

```
# tuple -> list, set  
a = (10,20,30)  
a2 = list(a)  
a3 = set(a)  
print(a)  
print(a2)  
print(a3)
```

```
(10, 20, 30)  
[10, 20, 30]  
{10, 20, 30}
```

```
# dict -> list, set, tuple (키만 추출)  
a = {"name": "홍길동", "age": 100}  
a2 = list(a)  
a3 = set(a)  
a4 = tuple(a)  
print(a)  
print(a2)  
print(a3)  
print(a4)
```

```
  
'name': '홍길동', 'age': 100}  
['name', 'age']  
{'age', 'name'}  
('name', 'age')  
dict_keys(['name', 'age'])
```

5장. 제어문



단일 if문, if~else문, 다중 if문

while 문, for 문

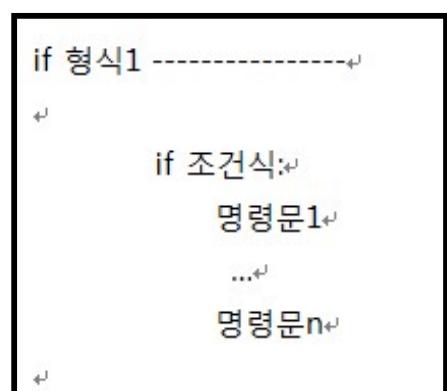
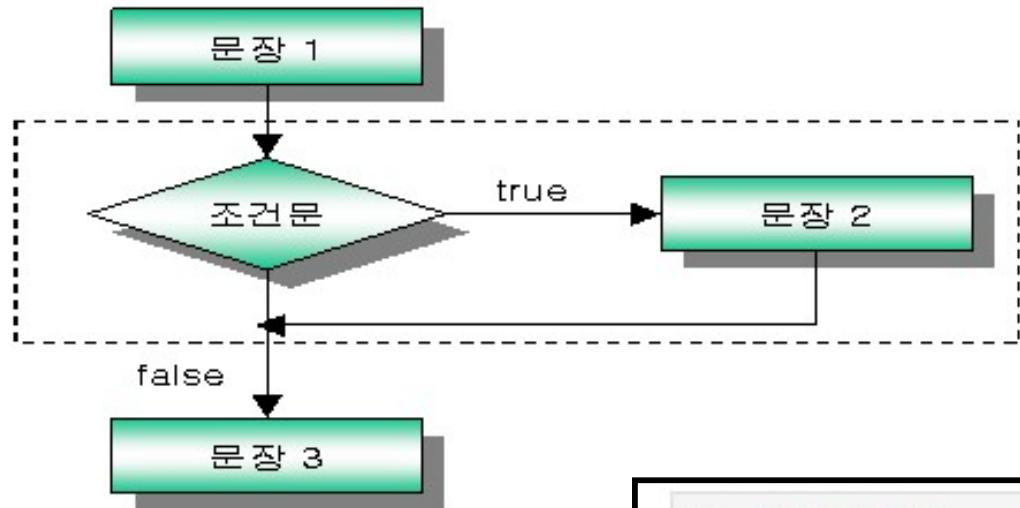
break, continue 문

range 함수

1) 단일 if문

python

https://docs.python.org/3/reference/compound_stmts.html



```
# 단일 if 문

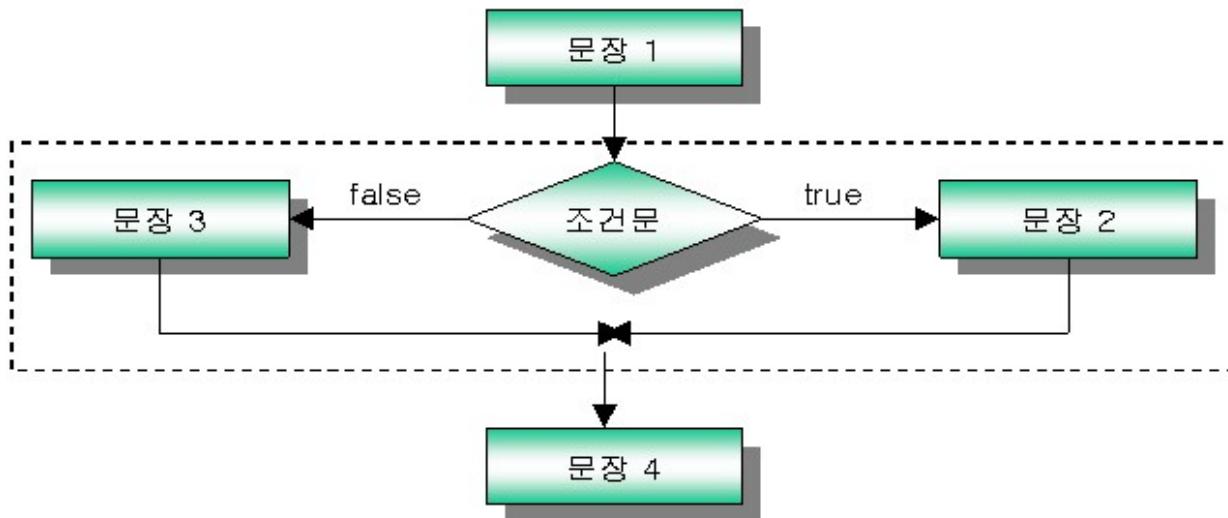
print("1")
if(3==4):
    print("2")
    print("3")
    print("4")
print("5")
```

1
5

This block shows a Python code example for a single if statement. It prints "1", then checks if 3 equals 4. Since it's false, nothing happens. Finally, it prints "5". The output is "1" on one line and "5" on the next.

2) if ~ else 문

python



```
# in 리스트 활용
pocket = ['paper', 'card', 'tel']
if 'tel' in pocket:
    print("tel")
else:
    print("None")
```

```
if 형식2
+
    if 조건식:
        명령문들1
    else:
        명령문들2
```

```
# if ~ else 문
n = int(input("점수입력"))
if(n%2==0):
    print("짝수")
else:
    print("홀수")
print("end")

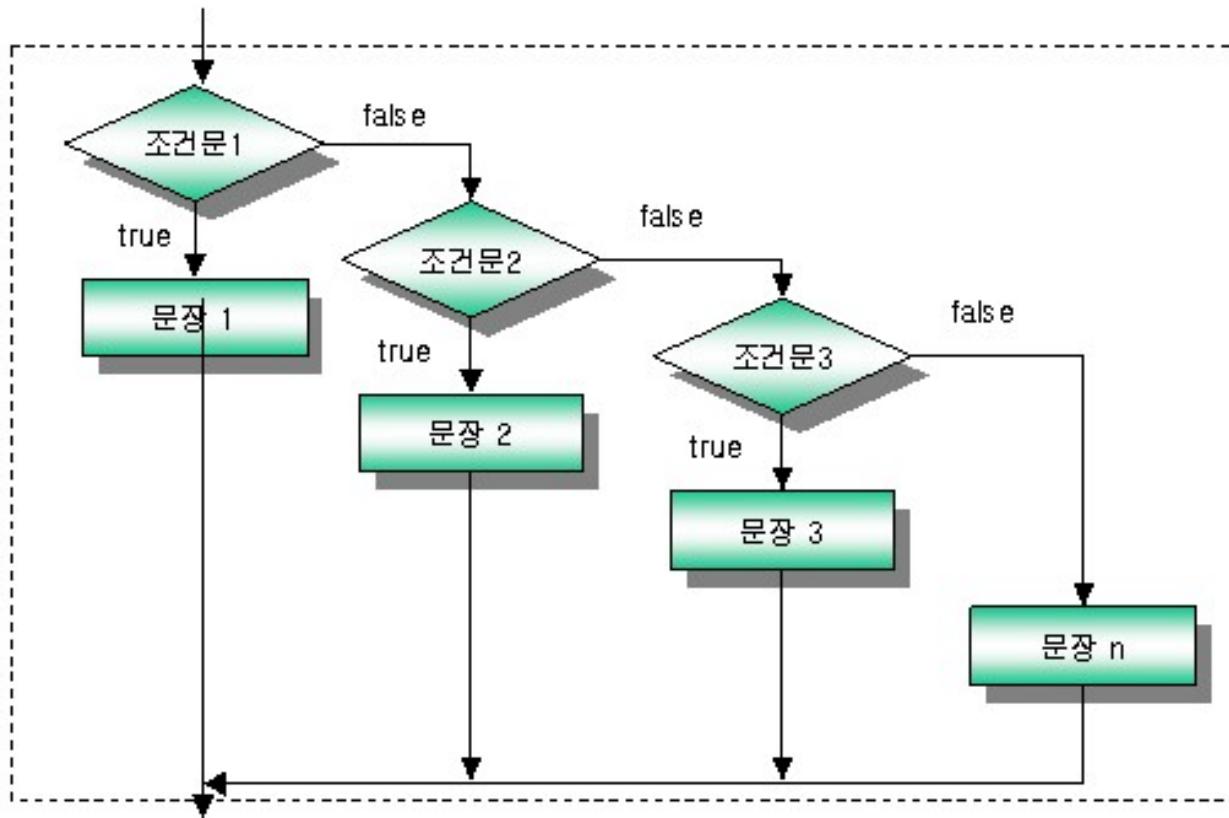
점수입력 4
짝수
end
```

```
# 중첩 if 문
jumsu = 85
if jumsu >= 90:
    print('우수')
else:
    if jumsu >= 70:
        print('보통')
    else:
        print('저조')
print('end2')

보통
end2
```

3) 다중 if ~ else 문

python



```
if 형식3  
-----  
if 조건식1:  
    명령문들1  
elif 조건식2:  
    명령문들2  
elif 조건식n:  
    명령문들n  
else:  
    명령문들
```

```
# 다중 if ~ else 문  
n = int(input("점수입력"))  
if(n>90):  
    print("A 학점")  
elif(n>80):  
    print("B 학점")  
elif(n>70):  
    print("C 학점")  
else:  
    print("F 학점")  
print("end")
```

```
점수입력 66  
F 학점  
end
```

4) 3항 연산자 (ternary operator)

python

문법 : 변수명 = 참 if 조건 else 거짓

<https://docs.python.org/3.7/reference/expressions.html#conditional-expressions>

6.12. Conditional expressions

```
conditional_expression ::= or_test ["if" or_test "else" expression]
expression           ::= conditional_expression | lambda_expr
expression_nocond    ::= or_test | lambda_expr_nocond
```

Conditional expressions (sometimes called a “ternary operator”) have the lowest priority of all Python operations.

The expression `x if C else y` first evaluates the condition, `C` rather than `x`. If `C` is true, `x` is evaluated and its value is returned; otherwise, `y` is evaluated and its value is returned.

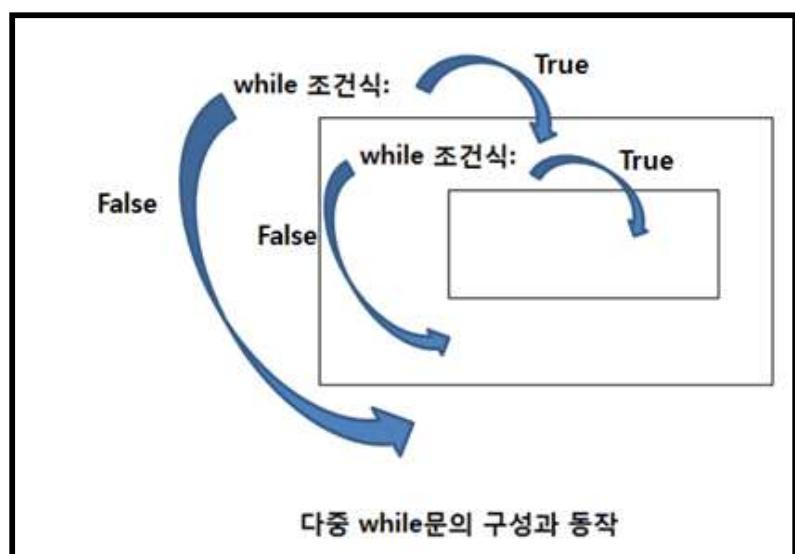
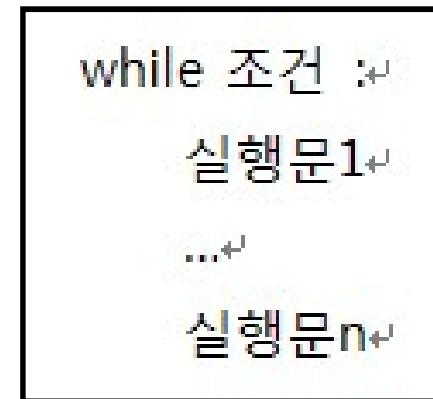
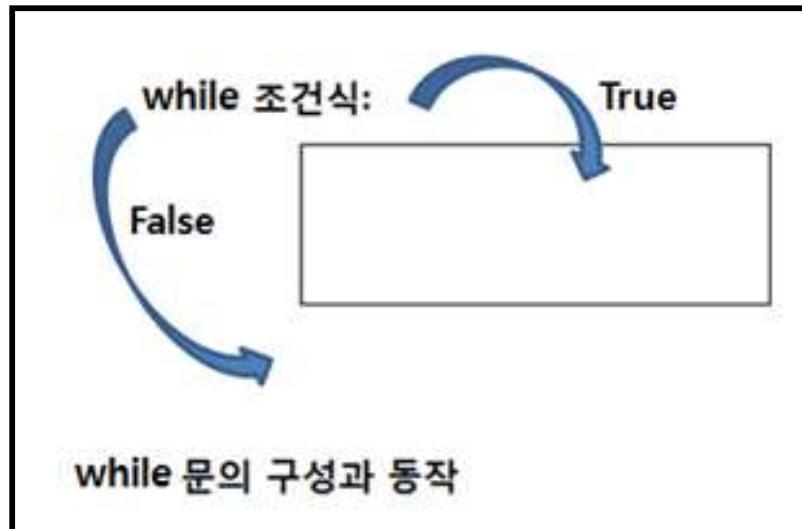
```
# 3항 연산자
# https://docs.python.org/3.7/reference/expressions.html#conditional-expressions
# 문법: 참 if 조건 else 거짓

a = 5
result = 100 if a==15 else 200
print(result)
```

200

5) while 반복문

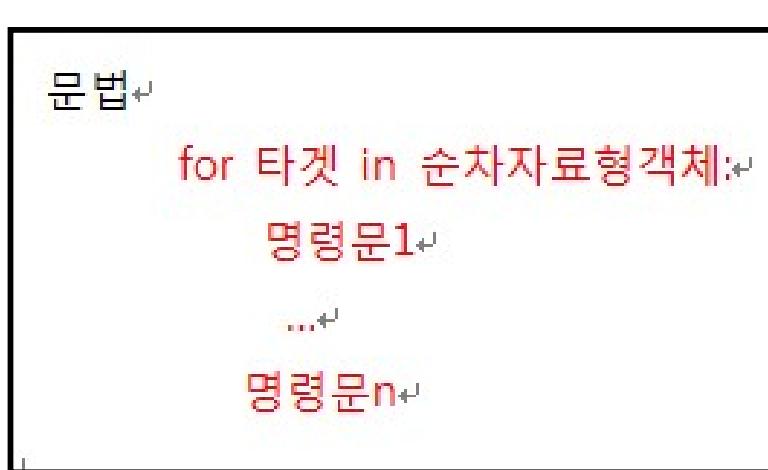
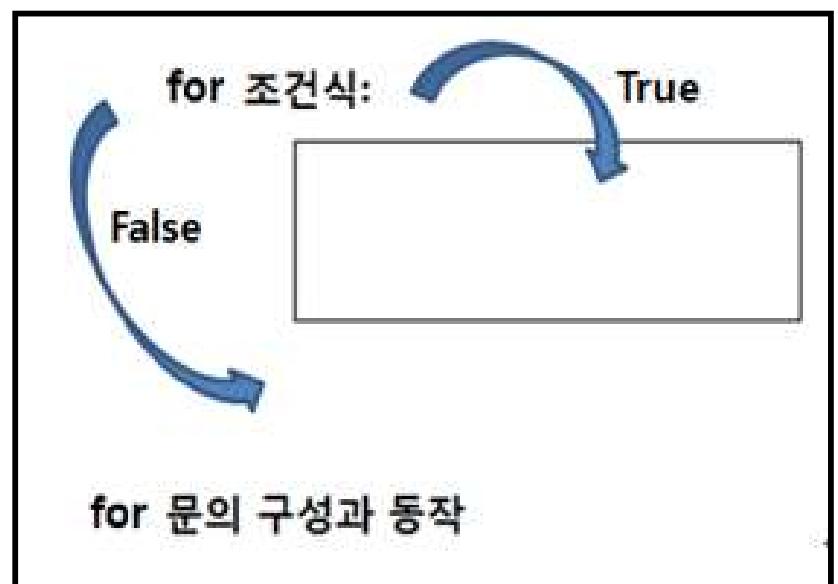
python



```
# while반복문  
  
a = 1  
while a <=5:  
    print(a, end=' ')  
    a += 1  
print("\nend")  
  
1 2 3 4 5  
end
```

6) for 반복문

python



```
# for 반복문

# 1) list 반복

for x in [10,20,30,40]:
    print(x)
print("end")
print()
for x in ['A','B','C']:
    print(x)
print("end")
```

```
# 2) set 반복 ==> 순서 없이 출력

for x in {10,20,30,40}:
    print(x)
print("end")

40
10
20
30
end
```

```
# 3) tuple 반복

for t in ('house', 'mouse', 'horse'):
    print(t, end=' ')
```

house mouse horse

6) for 반복문

python

```
# 4) dict 반복

soft = {'java':'웹용', 'python':'만능언어', 'oracle':'db처리'}
for i in soft.items():
    #print(i)
    print(i[0] + ', ' + i[1])
print()
for k in soft.keys():
    print(k, end=' ') #key 출력
print()
for v in soft.values():
    print(v, end=' ') #value 출력
```

java, 웹용

python, 만능언어

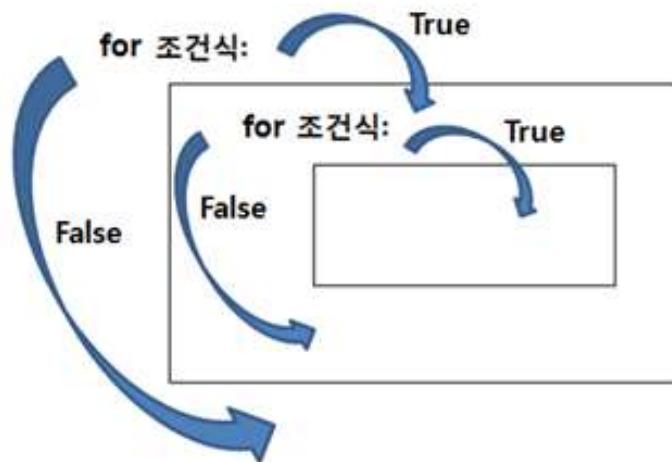
oracle, db처리

java python oracle

웹용 만능언어 db처리

7) 중첩 for 문

python



다중 for 문의 구성과 동작

```
# 5) 중첩 for 문
for n in [2, 3]:
    print('--{}단--'.format(n))
    for i in [1,2,3,4,5,6,7,8,9]:
        print('{0} * {1} = {2}'.format(n, i, n * i))
```

```
print()
li1 = [3, 4, 5]
li2 = [2, 3]

for a in li1:                      #3회 반복
    for b in li2:                  #2회 반복
        print(a * b, end = ' ')    #두 수를 곱한 값 출력
```

8) List comprehension

python

목적: 리스트로부터 가공한 새로운 리스트를 생성하여 반환.

문법1 : 변수명 = [표현식 for 항목 in 리스트]

```
# 1) 일반  
n = [10, 20, 30]  
for x in n:  
    print(x)  
  
# 2) List comprehension  
x2 = [x + 1 for x in n]  
print(x2)
```



10
20
30
[11, 21, 31]

8) List comprehension

python

문법2 : 변수명 = [표현식 for 항목 in 리스트 if 조건]

```
x2 = [x + 1 for x in [1, 2, 3, 4] if x % 2 == 0]  
print(x2)
```



[3, 5]

문법3 : 변수명 = [참 if 조건 else 거짓 for 항목 in 리스트]

```
x2 = [x + 1 if x % 2 == 0 else x-1 for x in [1, 2, 3, 4]]  
print(x2)
```



9) continue, break

python

반복문에 else문을 적어 줄 수도 있다. 이는 반복문이 종료된 시점에서 종료처리가 모든 반복을 다 수행한 후의 정상적인 종료인지, 아니면 break문에 의해 강제 종료된 것인지를 확인할 경우에 적어주면 효과적이다

```
a = 0
while a < 10:
    a += 1
    if a == 5: continue      #while문으로 수행 O/동
    if a == 7: break         #반복문 강제 종료
    print(a)
else:
    print("정상종료")
print("end")
```

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

for i in datas:
    if i == 3: continue
    if i == 4: break
    print(i, end=' ')
else:
    print("정상종료")
print("end")
```

1 2 end

10) range 함수

python

목적: 특정 숫자를 지정하면 그 범위 안에서 순차적인 숫자 그룹을 생성하는 함수.

문법 :

`x = range(n)` # 0부터 n-1까지의 요소를 가지는 range형 반환

`x = range(m,n)` # m부터 n-1까지의 요소를 가지는 range형 반환

`x = range(m,n,s)` # m부터 n-1까지의 요소 및 step를 가지는 range형 반환

1) python 3.X 방법

```
n = list(range(4))  
print(n)
```

```
n = set(range(4))  
print(n)
```

```
n = tuple(range(4))  
print(n)
```

[0, 1, 2, 3]
{0, 1, 2, 3}
(0, 1, 2, 3)

2) python 3.X 방법 - range(m, n)

```
n = list(range(1,6))  
print(n)
```

```
n = set(range(1,6))  
print(n)
```

```
n = tuple(range(1,6))  
print(n)
```

[1, 2, 3, 4, 5]
{1, 2, 3, 4, 5}
(1, 2, 3, 4, 5)

11) for + range 함수

python

```
# for 문 + range ()
print('\n\n1 ~ 10까지 합')
tot = 0

for i in range(1, 11):
    tot += i

print('합은 : ' + str(tot))
print('내장함수 sum() 사용:', sum(range(1, 11)))
```

1 ~ 10까지 합

합은 : 55

내장함수 sum() 사용: 55

...
map() 함수 + Lambda 이용한 반복처리
...

```
m = ["one", "two", "three"]
result = map(lambda x: x.upper(), m)
print(list(result))
```

6장. 함수



함수(function) 개요 및 특징

변수 scope

default 및 packing 파라미터

람다(lambda)함수

내장함수(built-in)

1) 함수 (function)

python

목적 및 특징:

- 여러 개의 실행문을 하나의 이름으로 묶은 실행단위이다. (반복제거 장점)
- 반드시 호출해야 수행되고, 수행 후에는 반드시 호출된 곳으로 돌아온다.
- 호출시 인자값 전달이 가능하고 반환시 리턴값을 받을 수 있다.
- 파이썬의 함수는 일급객체(first-class) 특성을 갖는다.
- 함수의 리턴값은 tuple을 사용하여 여러 개가 가능하고, 리턴값을 반환할 목적이 아닌 함수 수행을 끝마칠 목적으로 return 키워드를 사용할 수 있다.
- 함수 호출시 반드시 인자 개수는 일치시키거나 default 파라미터 및 packing/unpacking 을 사용할 수 있다.

문법 :

```
def 함수명(파라미터, …):  
    실행코드  
    return 반환값
```

함수 헤더(header)

함수 바디(body)
: 들여쓰기로 블럭효과

return 키워드 없으면 None 반환

2) 사용자 함수 (user-defined function) 유형

python

유형 1 : 파라미터 없고 리턴값 없는 형태

```
def fun1():
    print("fun1 함수 호출")
```

```
fun1()
```

유형 2 : 파라미터 있고 리턴값 없는 형태

```
def fun2(n, n2):
    print(n, n2)
```

```
fun2(10, 20) # 순서 매칭!
```

```
fun2(n2=10, n=20) # 변수명 매칭!
```

2) 사용자 함수 (user-defined function) 유형

python

유형 3 : 파라미터 없고 리턴값 있는 형태 (리턴값 한 개 및 여러 개)

```
| def fun3():
|     print("fun3 함수 호출")
|     return "Hello"
```

```
m = fun3()
print(m)
print(fun3())
```

```
| def fun4():
|     print("fun4 함수 호출")
|     return "Hello", 10
```

```
m, n = fun4()
print(m, n)
print(fun4())
```

유형 4 : 함수의 종료 목적으로 return 키워드 사용

```
| def even_or_odd(n):
|     if n % 2 == 0:
|         print("even")
|         return
|     print("odd")
```

2) 사용자 함수 (user-defined function) 유

형

python

유형 5 : 파라미터 있고 리턴값 있는 형태

```
def fun5(n,n2):
    return n+n2

m = fun5(10, 20)
print(m)
print(fun5(1, 3))
```

3) 일급 객체 (first-class)

python

특징:

- 함수를 변수에 저장할 수 있다.
- 함수를 함수 호출시 인자로 전달 할 수 있다.
- 함수를 함수의 리턴값으로 사용할 수 있다.

```
| def fun1():  
|     print("fun 호출")
```

```
| def fun2(f):  
|     print("fun2 호출")  
|     return f
```

```
f1 = fun1  
f1()  
f2 = fun2(fun1)  
f2()
```

4) 함수내의 변수 scope

python

특징:

- 자바스크립트의 var 변수 사용법과 비슷하다.
- 기본적으로 블록 scope가 아닌 함수 scope를 따른다.
(함수밖 변수를 global변수, 함수안 변수를 local변수라고 부른다.)
- 우선적으로 LEGB 규칙을 사용하여 변수를 참조한다.

```
# 1. if 및 for 블럭밖에서 사용 가능
if True:
    n = 10
    print(n)    # 10

for x in range(5):
    print("Hello")
    print(x)    # 4
```

```
# 2. 함수밖에서 사용 불가
def fun1():
    print("fun1 호출")
    m = 10

# print(m) # error 발생
```

4) 함수내의 변수 scope

python

```
# 3. Local 및 global 변수  
k = "global 변수"  
  
def fun2():  
    k2 = "local 변수"  
    print(k)      # 함수내에서 global 변수 접근 가능  
    print(k2)     # 함수내에서 local 변수 접근 가능  
  
fun2()  
print(k)          # 함수밖에서 global 변수 접근 가능
```

```
# 4) 중첩된 경우에는 Local  
player = '전국대표'          # global  
  
def FuncSoccer():  
  
    name = '홍길동'            # Local  
    player = '지역대표'        # Local  
    print(name, player)       # 홍길동 지역대표  
  
FuncSoccer()
```

4) 함수내의 변수 scope

python

L (Local) : 가장 가까운 함수 범위 참조

E (Enclosed) : 함수안의 함수에서 가장 가까운 함수가 아닌 두번째 이상의 함수
에서 가까운 함수범위 참조

G (Global) : 함수밖의 변수 또는 import 된 모듈

B (Built-in) : 파이썬에 내장되어 있는 함수 또는 속성

5) LEGB 규칙

```
n = 10          # global
n2 = 20         # global

def outer_fun():
    n = 100      # outer_fun()의 Local이면서 inner_fun()의 enclosed
    def inner_fun():
        n3 = 300
        print(n,n2,n3)
    inner_fun()  # 100 20 300
    n = 200      # outer_fun()의 Local이면서 inner_fun()의 enclosed
    inner_fun()  # 200 20 300

outer_fun()
```

4) 함수내의 변수 scope

python

* nonlocal 및 global 키워드

nonlocal 키워드 사용

1) nonlocal 변수 사용전 - 에러

```
def Kbs():
    a = 1          # Kbs 함수에서 유효한 지역 변수
    def Mbc():
        print('Mbc에서 출력:', a) #UnboundLocalError 에러
        a = 2      # Mbc 함수에서만 유효한 지역 변수
    Mbc()
Kbs()
```

2) global 키워드

```
a = 1
def Kbs():
    a = 2
    def Mbc():
        global a # 1 출력
        #nonlocal a # 2 출력
        print('Mbc에서 출력:', a)
        a = 3
    Mbc()
Kbs()
```

1) nonlocal 변수 사용후 - 에러 해결1

```
def Kbs():
    a = 1          # Kbs 함수에서 유효한 지역 변수
    def Mbc():
        a = 2
        print('Mbc에서 출력:', a)
    Mbc()
Kbs()
```

Mbc에서 출력: 2

1) nonlocal 변수 사용후 - 에러 해결2

```
def Kbs():
    a = 1          # Kbs 함수에서 유효한 지역 변수
    def Mbc():
        #a는 Mbc 함수의 부모함수인, Kbs 함수의 a가 됨
        nonlocal a
        print('Mbc에서 출력:', a)
        a = 2
    Mbc()
Kbs()
```

Mbc에서 출력: 1

5) 함수 파라미터 - default 파라미터

python

문법 :

```
def 함수명(파라미터, 파라미터=default값):  
    pass
```

```
# parameter default  
  
def aaa(n,n2):  
    print(n,n2)  
  
aaa(10,20) # 순서로 매칭  
aaa(n2=10,n=20) # 변수명 매칭
```

```
#aaa(10) #error
```

```
10 20  
20 10
```

```
# 1) default  
  
def aaa(n,n2=100):  
    print(n,n2)  
  
aaa(10,20)  
aaa(10)
```

```
10 20  
10 100
```

6) 함수 파라미터 - packing 1

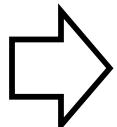
python

문법 : 인자의 개수가 일치하지 않는 경우 tuple로 전달된다.

```
def 함수명(*파라미터):  
    pass
```

함수명(값1, 값2, 값3)

```
def func1(*lunch):  
    print(lunch)  
    for i in lunch:  
        print('음식:' + i)  
  
func1('비빔밥', '김밥', '볶음밥')
```



('비빔밥', '김밥', '볶음밥')
음식:비빔밥
음식:김밥
음식:볶음밥

6) 함수 파라미터 - packing 2

python

문법 : dict 인 경우에는 ** 사용한다.

```
def 함수명(** 파라미터):  
    pass
```

함수명 (값1, key=값, key2=값)

```
# dict 자료는 - ** 사용  
def func3(**z):  
    print(z) # {'name': '홍길동', 'age': 20}  
  
func3(name='홍길동', age=20)  
# func3({'name': '홍길동', 'age': 20}) #error
```

6) 함수 파라미터- packing 3

python

문법 : *와 ** 혼합 사용할 때는 반드시 순서를 준수한다.

```
def 함수명(변수명, *변수명, **변수명):
```

```
    pass
```

```
함수명 ( 값1, 값2, 값3, key=값, key2=값 )
```

*와 ** 혼합은 반드시 순서대로 지정

```
def func4(x,y,*z,**k):  
    print(x,y,z,k)
```

```
func4(9, 8, 7, 6, name="홍길동", age=20 )
```



```
9 8 (7, 6) {'name': '홍길동', 'age': 20}
```

7) 함수 파라미터 - unpacking

python

문법 : 인자값이 분해되어 함수 파라미터로 전달된다.

```
def 함수명(n,n2,n3):  
    pass
```

```
함수명( *[값1,값2,값3] )  
함수명( *{값1,값2,값3} )  
함수명( *(값1,값2,값3) )
```

```
def fun1(x, y, z):  
    print(x, y, z)  
  
#fun1([10, 20, 30]) #error  
fun1(*[10, 20, 30])
```

```
def fun2(x, y, *z):  
    print(x, y, z) # 10 20 (30, 40)  
  
fun2(*[10, 20, 30, 40])
```

8) 람다(lambda) 함수

python

특징 :

- def 함수의 다른 표현방법으로서, 단일 표현식만 적용이 가능하다.
- 사용자 정의 함수 유형 개수만큼 람다 함수 유형이 제공된다.
- <https://docs.python.org/3/tutorial/controlflow.html#define-functions>
- 익명 함수 또는 축약함수라고도 부른다.

문법:

변수명 = lambda param, …: 표현식

8) 람다(lambda) 함수

python

유형 1 : 파라미터 없고 리턴값 없는 형태

```
def fun1():
    print("fun1 함수 호출")      → fun1 = lambda: print("fun1 함수 호출")
fun1()                                fun1()
```

유형 2 : 파라미터 있고 리턴값 없는 형태

```
def fun2(n, n2):
    print(n, n2)      → fun2 = lambda n,n2: print(n, n2)
fun2(10, 20) # 순서 매칭!           fun2(10,20)
fun2(n2=10, n=20) # 변수명 매칭!
```

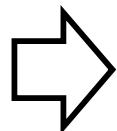
8) 람다(lambda) 함수

python

유형 3 : 파라미터 없고 리턴값 있는 형태

```
def fun3():
    return "Hello"

m = fun3()
print(m)
print(fun3())
```

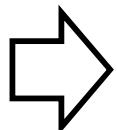


```
fun3 = lambda: "Hello"
print(fun3())
```

유형 4 : 파라미터 있고 리턴값 있는 형태

```
def fun4(n, n2):
    return n+n2

m = fun4(10, 20)
print(m)
print(fun4(1, 3))
```



```
fun4 = lambda n, n2: n+n2
print(fun4(1, 3))
```

8) 람다(lambda) 함수

python

유형 5 : default 파라미터 지정

```
def fun5(n, n2=100):  
    return n+n2  
print(fun5(10))
```



```
fun5 = lambda n, n2=100: n+n2  
print(fun5(10))
```

유형 6 : packing 연산자

```
def fun6(n, *n2, **n3):  
    print(n, n2, n3)  
fun6(10, 20, 30, k=9, k2=9)
```



```
fun6 = lambda n, *n2, **n3: print(n, n2, n3)  
fun6(10, 20, 30, k=9, k2=9)
```

9) 내장 함수 (built-in)

python

함수명	기능	함수명	기능
sum(x)	총합을 구하는 함수	round(x)	반올림
max(x)	최대값을 구하는 함수	all(x)	모두 참이면 참
min(x)	최소값을 구하는 함수	any(x)	하나라도 참이면 참
int(x)	정수로 변경하는 함수	zip(x,y)	쌍을 지어 tuple로 반환
float(x)	실수로 변경하는 함수	divmod(a,b)	a을b로 나눈 몫/나머지 반환
bool(x)	불린으로 변경하는 함수	input()	표준 입력 함수
str(x)	문자열로 변경하는 함수	open("파일명")	파일 입/출력 함수
list(x)	리스트로 변경하는 함수	tuple(x)	튜플로 변경하는 함수
set(x)	셋으로 변경하는 함수	dict(x)	딕트로 변경하는 함수
enumerate(x)	리스트에서 idx와 값을 반환	filter(lambda)	리스트에서 조건과 일치하는 항목 반환
dir(x)	객체가 가지고 있는 변수나 함수리스트 반환	len(x)	값의 길이를 반환하는 함수

7장. 클래스



클래스 및 객체생성

변수, 메서드, 생성자

property 속성

(다중)상속 및 super() 사용법

오버라이딩 (overriding)

private 속성

1) 클래스 개요

python

개념 및 특징:

- 현실세계의 객체(Object)를 OOP기반으로 표현한 형태이다.
- 객체의 구성요소인 속성과 동작은 변수와 메서드로 표현한다.
- 클래스를 사용하기 위해서는 반드시 객체생성이 필요하다.
- 클래스의 이름은 반드시 첫 글자는 대문자 형태의 camelCase로 지정한다.

문법:

class 클래스명:

 변수명

 메서드

변수명 = 클래스명() # 객체생성 (인스턴스화)

2) 클래스 정의 및 생성

python

```
# 1) 클래스 정의 및 생성
```

```
class Student:  
    pass
```

```
stu = Student()  
print("stu 생성:", stu, dir(stu))
```



```
stu 생성: <__main__.Student object at 0x00000203E2CFD4C0>  
['__class__', '__delattr__', '__dict__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__',
```

3) 메서드 (인스턴스 메서드)

python

개념 및 특징:

- 클래스의 구성요소로서 기능(동작)처리를 담당하는 함수를 의미한다.
- 반드시 self로 지정된 파라미터를 설정해야 된다.
(self 에는 자동으로 생성된 인스턴스가 전달된다.)
- 메서드는 반드시 객체 생성후에 ‘변수명.메서드’ 형식으로 사용할 수 있다.
- 메서드명은 소문자로 지정하고 _(언더바) 사용 가능하다.

문법:

class 클래스명:

```
def 메서드명(self):  
    pass
```

변수명 = 클래스명()

변수명.메서드명() # 메서드 사용 방법

3) 메서드 (인스턴스 메서드)

python

1) 메서드 : 클래스내의 함수 의미로서 반드시 self 파라미터를 갖는다.

```
class Student:
```

```
    def get_name(self):  
        return "홍길동"
```

```
stu = Student()
```

```
print(stu.get_name())
```



C:\Python38\python.exe C:

홍길동

4) 생성자 (constructor)

python

개념 및 특징:

- 클래스를 객체 생성할 때 사용하는 표현식이다.
- 자동으로 `__init__` 메서드가 호출되고, 이 메서드에서 변수 초기화 작업처리.

문법:

`class 클래스명:`

```
def __init__(self):  
    pass
```

`변수명 = 클래스명() # 생성자 (constructor)`

4) 생성자 (constructor)

python

1) 생성자 : 변수 초기화 작업, `__new__(cls) ==> __init__(self)` 호출됨.

```
class Student:
```

```
    def __init__(self):  
        print("init 호출, 변수 초기화 작업")
```

```
    def get_name(self):  
        return "홍길동"
```

```
stu = Student()  
print(stu.get_name())
```



```
C:\Python38\python.exe C:/Users  
init 호출, 변수 초기화 작업  
홍길동
```

5) 변수 (인스턴스 변수)

python

개념 및 특징:

- 클래스의 구성요소로서 객체의 속성값을 저장하는 용도로 사용된다.
- `__init__` 메서드내에서 ‘`self.변수명`’ 형식으로 변수 초기화를 할 수 있다.
- 다른 메서드에서도 ‘`self.변수명`’ 형식으로 사용할 수 있다.
- 객체생성시 매번 새롭게 생성되는 변수이다.

문법:

`class 클래스명:`

```
def __init__(self, 파라미터):  
    self.변수명 = 파라미터      # 인스턴스 변수
```

`변수명 = 클래스명()`

`변수명.메서드명()`

5) 변수 (인스턴스 변수)

python

```
# 1) 인스턴스 변수 : 클래스내의 구성요소로서 객체의 속성정보를 저장한다.  
# 반드시 self. 변수명 형식을 따른다.  
class Student:  
  
    def __init__(self, name):  
        self.name = name  
  
    def get_name(self):  
        return self.name  
  
stu = Student("홍길동")  
print("첫 번째 학생이름: ", stu.get_name())  
stu2 = Student("이순신")  
print("두 번째 학생이름: ", stu2.get_name())
```



```
C:\Python38\python.exe C:/  
첫 번째 학생이름: 홍길동  
두 번째 학생이름: 이순신
```

5) 변수 (인스턴스 변수)

python

```
# 1) 인스턴스 변수 : 클래스내의 구성요소로서 객체의 속성정보를 저장한다.  
# 반드시 self.변수명 형식을 따른다.
```

```
class Student:
```

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

초기화 작업

```
    def get_name(self):  
        return self.name
```

조회 작업

```
    def set_name(self, name):  
        self.name = name
```

변경 작업

```
stu = Student("홍길동")  
print("원래이름: ", stu.get_name())  
stu.set_name("홍두깨")  
print("변경된 이름: ", stu.get_name())
```



```
C:\Python38\python.exe C:/
```

원래이름: 홍길동

변경된 이름: 홍두깨

6) 변수 (클래스 변수)

python

개념 및 특징:

- 여러 인스턴스가 공유할 목적으로 사용된다.
- ‘클래스명.변수명’ 형식으로 사용할 수 있다.
- 단 한번 생성되는 변수이다.

문법:

class 클래스명:

 변수명 = 값 # 클래스 변수 선언

 def 메서드(self):

 클래스명.변수명 = 값 # 클래스 변수 사용

변수명1 = 클래스명()

변수명2 = 클래스명()

6) 변수 (클래스 변수)

python

```
class Student:  
  
    count = 0 # 클래스 변수  
    def __init__(self):  
        Student.count = Student.count + 1  
  
    def get_count(self):  
        return Student.count  
  
stu = Student()  
print(stu.get_count(), Student.count)  
stu2 = Student()  
print(stu2.get_count(), Student.count)
```



C:\Python38\|

1 1

2 2

7) 메서드 (클래스 메서드)

python

개념 및 특징:

- 클래스 변수를 참조 또는 객체 생성 없이 메서드 사용 목적으로 사용된다.
- 반드시 cls로 지정된 파라미터를 설정해야 된다.
(cls 에는 자동으로 클래스가 전달된다.)
- ‘클래스명.메서드’ 형식으로 사용할 수 있다.
- @classmethod 데코레이터를 사용하여 클래스 메서드임을 설정한다.

문법:

```
class 클래스명:  
    변수명 = 값  
    @classmethod  
    def 클래스메서드명(cls):  
        cls.변수명 = 값    # 클래스 변수 사용
```

```
변수명1 = 클래스명()
```

7) 메서드 (클래스 메서드)

python

```
class Student:  
  
    count = 0 # 클래스 변수  
    def __init__(self):  
        Student.count += 1  
  
    @classmethod  
    def get_count(cls): # 클래스 메서드  
        return cls.count  
  
stu = Student()  
print(Student.count, Student.get_count())  
stu2 = Student()  
print(Student.count, Student.get_count())
```



C:\Python38\|

1 1

2 2

8) property 속성

python

목적:

- 클래스의 get 과 set 메서드 기능을 하나의 변수(프라퍼티)로 사용 가능.

문법:

class 클래스명:

```
def get_xxx(self):
```

```
    pass
```

```
def set_xxx(self):
```

```
    pass
```

```
xxx = property( get_xxx, set_xxx ) # xxx 프라퍼티 선언
```

8) property 속성

python

```
class Student:

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

    def get_name(self):
        return self.name

    def set_name(self, name):
        self.name = name

    # property 속성 설정 ( 주의: 마지막 메서드 선언 맨 마지막에 )
    username = property(get_name, set_name)

stu = Student("홍길동", 20)
print("학생정보1: ", stu.username, stu.age)
stu.username = "이순신" # 수정
print("학생정보1: ", stu.username, stu.age)
```

9) 상속 (inheritance)

python

개요 및 특징:

- 같은 종류의 클래스에서 공통적인 속성 및 기능을 추출하여 상위 클래스로 작성하고 이것을 하위 클래스에서 상속받아서 구현하는 것을 의미한다.
- 상위 클래스(부모 클래스)의 모든 변수와 메서드를 하위 클래스(자식 클래스)에서 선언없이 사용이 가능하다. (재사용 향상)
- __변수, __메서드 명으로 지정하면 상속이 안된다.
- 자식 클래스에서 부모 클래스를 직접 참조하기 위해서 super() 를 사용한다.
- 부모 클래스의 메서드를 자식 클래스에서 재정의(overriding) 할 수 있다.
- 다중 상속이 지원된다.

문법:

```
class 클래스명(부모 클래스명):  
    pass
```

9) 상속 (inheritance)

python

```
class Pet:  
  
    def __init__(self, name, age):  
        self.name = name  
        self.age = age  
  
    def cry(self, action):  
        return action  
  
    def sleep(self, action):  
        return action  
  
    def eat(self, action):  
        return action  
  
    def pet_info(self):  
        return self.name + "\t" + str(self.age)
```

```
pet1 = Dog("바둑이", 20, "암컷")  
print(pet1.dog_info())  
print(pet1.cry("멍멍"))  
print(pet1.eat("쩝쩝"))  
print(pet1.sleep("새근새근"))  
  
pet2 = Cat("야옹이", 3)  
print(pet2.cat_info())  
print(pet2.cry("야옹"))  
print(pet2.eat("냠냠"))  
print(pet2.sleep("쿨쿨"))
```

```
class Dog(Pet):  
  
    def __init__(self, name, age, gender):  
        self.name = name  
        self.age = age  
        self.gender = gender  
  
    def dog_info(self):  
        return self.name + "\t" + str(self.age) + "\t" + self.gender
```

```
class Cat(Pet):  
  
    def __init__(self, name, age):  
        self.name = name  
        self.age = age  
  
    def wash(self, action):  
        return action  
  
    def cat_info(self):  
        return self.name + "\t" + str(self.age)
```

10) super()

python

용도:

- 자식 클래스에서 명시적으로 부모 클래스를 참조할 때 사용한다.

문법:

super().변수

super().메서드

```
class Pet:
```

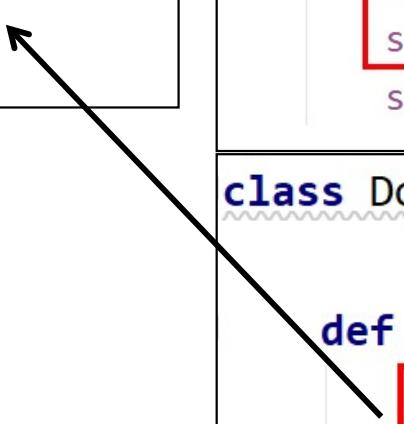
```
    def __init__(self, name, age):  
        self.name = name  
        self.age = age
```

```
class Dog(Pet):
```

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

```
class Dog(Pet):
```

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



11) 오버라이딩(overriding)

python

개요 및 목적:

- 자식 클래스에서 상속받은 부모의 메서드를 재정의 하는것을 의미한다.
- 동일한 메서드명으로 사용하기 위한 목적이다. (재사용성 증가)

```
class Pet:  
  
    def __init__(self, name, age):  
        self.name = name  
        self.age = age  
  
    def cry(self, action):  
        return action  
  
    def sleep(self, action):  
        return action  
  
    def eat(self, action):  
        return action  
  
    def pet_info(self):  
        return self.name + "\t" + str(self.age)
```

```
class Dog(Pet):  
  
    def __init__(self, name, age, gender):  
        super().__init__(name, age)  
        self.gender = gender  
  
    # 재정의  
    def pet_info(self):  
        return self.name + "\t" + str(self.age) + "\t" + self.gender
```

12) 다중상속 (multi inheritance)

python

```
# 다중 상속

class Animal:
    def eat(self):
        print("eat")

class Person:
    def study(self):
        print("study")

class Man(Person, Animal):
    pass

m = Man()
m.eat()
m.study()
```

12) private 속성

python

개요:

- 부모 클래스의 요소중에서 자식 클래스가 상속받지 못하도록 설정하는 방법.
- __속성 (두 개의 under bar) 이용하면 상속이 안된다.

문법:

self.__변수명 = 값

def __메서드명(self):

 pass

```
class SuperClass:  
    __gold = "금" # SuperClass class 영역 안에서만 호출할 수 있다.  
    sliver = "은"  
  
    def __init__(self):  
        self.__bronze = "동"  
  
    def __private_method(self): # SuperClass class 영역 안에  
        return 'private_method'  
  
    def public_info(self):  
        print(SuperClass.__gold, SuperClass.sliver, self.__bronze, self.__private_method())  
  
class ChildClass(SuperClass):  
    def public_info(self):  
        print(super().sliver, self.sliver) # 출력 가능  
        #print(super().__gold) #출력 불가능
```

8장. 모듈 및 예외처리



모듈 개요

import문 활용한 모듈 접근

예외처리 개요

try ~ except ~ finally 문

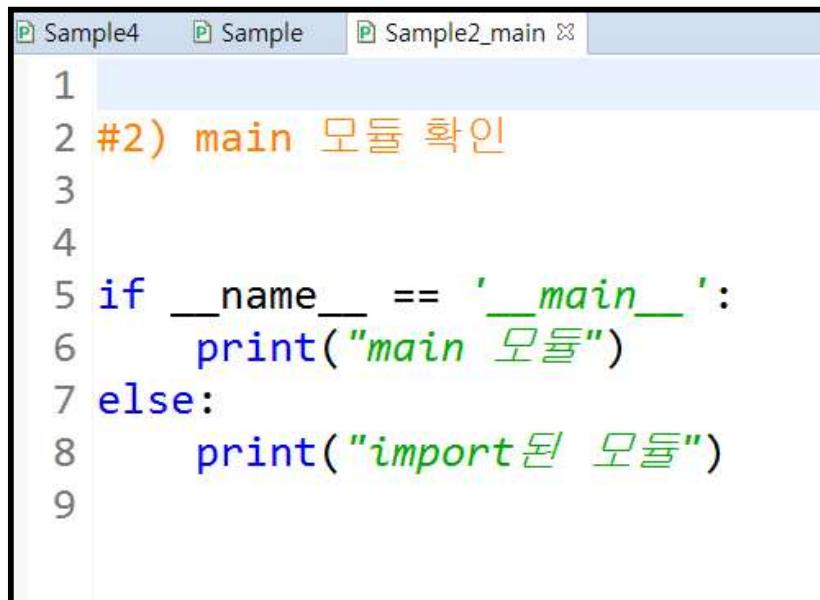
raise문 및 사용자 정의 예외 클래스

1) 모듈

python

정의:

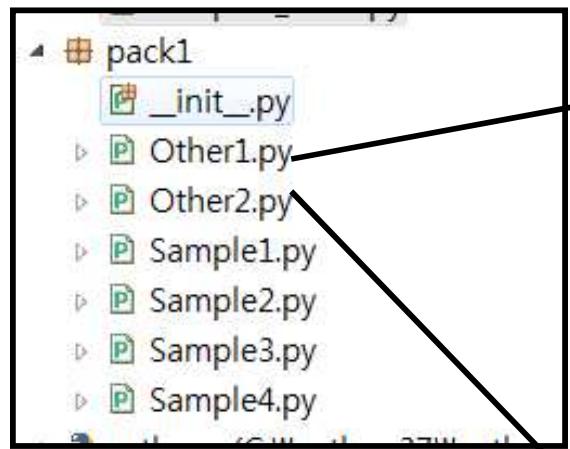
- 하나의 물리적인 파이썬의 파일을 의미한다. (예 > hello.py)
- 서로 관련된 모듈들의 그룹을 ‘패키지(package)’라고 부른다.
- 모듈간의 접근은 반드시 import 문을 사용해야 된다.
- 시작점 역할의 모듈(메인모듈)에는 다음 코드를 설정하고 사용하는 것이 관례이다.



```
Sample4 Sample Sample2_main
1
2 #2) main 모듈 확인
3
4
5 if __name__ == '__main__':
6     print("main 모듈")
7 else:
8     print("import 된 모듈")
9
```

2) 사용자 정의 모듈

python



```
3 # Other1.py
4
5 aaa1 = 100
6
7 def bbb1():
8     print("Other1.bbb1()")
9
10 class Student:
11     def __init__(self):
12         print("Student 생성자")
```

```
4 # Other2.py
5
6 aaa2 = 100
7
8 def bbb2():
9     print("Other2.bbb2()")
```

3) 사용자 정의 모듈

python

1) 경로지정방법 : import 패키지명.모듈명

```
4 # 1) 경로지정방법 : import 패키지명.모듈명
5
6 import pack1.Other1
7 import pack1.Other2
8
9 if __name__ == '__main__':
10     print("멤버 확인:", dir(pack1.Other1))
11     print("멤버 확인:", dir(pack1.Other2))
12     print()
13     print("경로 및 파일 확인:", pack1.Other1.__file__)
14     print("모듈명 확인:", pack1.Other1.__name__)
```

The screenshot shows a Windows command-line interface (CMD) window titled 'Console'. The title bar also displays the path 'C:\eclipse37\workspace\moduleTest\pack1\Other1.py'. The window contains the following text:

```
<terminated> Sample1.py [C:\eclipse37\workspace\moduleTest\pack1\Other1.py]
멤버 확인: ['Student', '__builtins__', '__cached__', '__doc__', '__file__', '__loader__', '__name__', '__package__', '__spec__']
멤버 확인: ['__builtins__', '__cached__', '__doc__', '__file__', '__loader__', '__name__', '__package__', '__spec__']
경로 및 파일 확인: C:\eclipse37\workspace\moduleTest\pack1\Other1.py
```

3) 사용자 정의 모듈

python

2) 경로지정방법 : from 패키지명 import (모듈명,모듈명2)

```
# 2) 경로지정방법 : from 패키지명 import (모듈명,모듈명2)

# from pack1 import Other1
# from pack1 import Other2

from pack1 import (Other1,Other2)

if __name__ == '__main__':
    # 멤버 접근
    print(">",Other1.aaa1)
    Other1.bbb1()
    stu = Other1.Student()

    print(">>",Other2.aaa2)
    Other2.bbb2()
```

3) 사용자 정의 모듈

python

3) 경로지정방법 : from 패키지명.모듈명 import 멤버, 멤버2

```
# 3) 경로지정방법 : from 패키지명.모듈명 import 멤버...
from pack1.Other1 import aaa1,bbb1,Student
from pack1.Other2 import *

if __name__ == '__main__':
    # 멤버 접근
    print(">",aaa1)
    bbb1()
    stu = Student()

    print(">>",aaa2)
    bbb2()
```

4) 시스템 정의 모듈

python

* 시스템 정의 모듈 예

```
# 시스템 module 예

import sys
print(sys.path)
#sys.exit() #프로그램의 종료

import math
print(math.pi)

print('\n작업 경로 관련 정보 얻기')
import os
print(os.getcwd())      #현재 작업 경로를 반환
print(os.listdir('./'))  #현재 작업 경로 내의 파일 목록

print('\n난수 출력하기')
import random
print(random.random())
print(random.randrange(1, 10))  # 100/ 포함 안됨
print(random.randint(0,10))    # 100/ 포함
```

5) 예외처리 (exception handling) 개요

python

개요:

- 예외는 프로그램 실행 중에 발생되는 의도하지 않은 사건을 의미한다. 필요시 사용자가 직접 명시적으로 예외를 발생시킬 수도 있다.
- 프로그램 실행 중에 예외가 발생되면 비정상 종료된다.
- 비정상 종료되는 프로그램을 정상 종료로 처리하는 작업이 ‘예외처리’이다.
- 파이썬에서는 예외처리를 할 수 있는 예외클래스를 제공한다. 필요시 사용자가 직접 예외클래스를 생성하여 사용할 수도 있다.

문법:

```
try:  
    실행코드;  
  
except [발생에러[ as 에러메시지변수]]:  
    예외처리 코드;
```

```
try:  
    일반 비즈니스 로직 ...  
  
except [발생에러[ as 에러메시지변수]]:  
    에러 발생시 처리할 내용  
  
finally:  
    에러와 상관없이 반드시 수행할 문을 적음
```

5) 예외처리 (exception handling) 개요

python

```
# 예외 처리

print("1")
n = 10
n2 = n/2
print("결과값:",n2 , int(n2))
print("정상종료")
```

```
1
결과값: 5.0
정상종료
```



```
# 예외 발생되면 비정상 종료된다.
print("1")
n = 10
n2 = n/0
print("결과값:",n2 , int(n2))
print("정상종료")
```

```
1
```

```
-----
ZeroDivisionError                                     Traceback (most recent call last)
<ipython-input-4-66ce36292882> in <module>
      2 print("1")
      3 n = 10
----> 4 n2 = n/0
      5 print("결과값:",n2 , int(n2))
      6 print("정상종료")
```

```
ZeroDivisionError: division by zero
```

6) 예외 클래스 계층구조

python

- 예외 처리 참고

<https://docs.python.org/3/library/exceptions.html?highlight=exception%20hierarchy>

Exception hierarchy

The class hierarchy for built-in exceptions is:

```
BaseException
+-- SystemExit
+-- KeyboardInterrupt
+-- GeneratorExit
+-- Exception
    +-- StopIteration
    +-- StopAsyncIteration
    +-- ArithmeticError
        +-- FloatingPointError
        +-- OverflowError
        +-- ZeroDivisionError
    +-- AssertionError
    +-- AttributeError
    +-- BufferError
    +-- EOFError
    +-- ImportError
        +-- ModuleNotFoundError
    +-- LookupError
        +-- IndexError
        +-- KeyError
```

7) try ~ except 문

python

```
# 예외처리  
  
# 예외 발생되면 비정상 종료된다.  
print("1")  
try:  
    n = 10  
    n2 = n/0  
    print("결과값:",n2 , int(n2))  
except:  
    print("예외 발생")  
print("정상종료")
```

1
예외 발생
정상종료

```
# 예외 발생되면 비정상 종료된다.  
print("1")  
try:  
    n = 10  
    n2 = n/0  
    print("결과값:",n2 , int(n2))  
except ZeroDivisionError as e:  
    print("예외 발생",e)  
print("정상종료")
```

1
예외 발생 division by zero
정상종료

8) finally 문

python

* finally 문

```
# finally 문
print("1")
try:
    n = 10
    n2 = n/0
    print("결과값:",n2 , int(n2))
except ZeroDivisionError as e:
    print("예외 발생",e)
finally:
    print("반드시 수행")
print("정상종료")
```

1
결과값: 10.0 10
반드시 수행
정상종료

9) 다중 except 문

python

* 다중 except 문

```
# 다중 except 문
print("1")
try:
    n = 10
    n2 = n/1
    print("결과값:",n2 , int(n2))

    m = [1,2]
    print(m[7])
except ZeroDivisionError as e:
    print("예외 발생1",e)
except IndexError as e:
    print("예외 발생2",e)
finally:
    print("반드시 수행")
print("정상종료")
```

```
1
결과값: 10.0 10
예외 발생2 list index out of range
반드시 수행
정상종료
```

```
# 다중 except 문
print("1")
try:
    n = 10
    n2 = n/1
    print("결과값:",n2 , int(n2))

    m = [1,2]
    print(m[7])
except ZeroDivisionError as e:
    print("예외 발생1",e)
except IndexError as e:
    print("예외 발생2",e)
except Exception as e:
    print('이외 에러 발생 : ', e)
```

```
finally:
    print("반드시 수행")
print("정상종료")
```

10) 명시적 예외 발생

python

* 명시적 예외 발생

```
# 명시적 예외 발생
```

```
# 1) 명시적 예외 발생
```

```
import random

print("프로그램 시작")
n = random.randrange(1,3)
print("랜덤값:",n)
if(n==2):
    raise Exception;
print("프로그램 종료")
```

프로그램 시작

랜덤값: 2

Exception
ast)

<ipython-inpu
o print

```
1 import random
2
3 def randomValue():
4     n = random.randrange(1,3)
5     print("랜덤값:", n)
6     if(n==2):
7         raise Exception
8
9 print(" 프로그램 시작")
10 randomValue()
11 print(" 프로그램 정상종료")
```

```
# 2) 명시적 예외 발생 - 함수 이용
```

```
import random
```

```
def randomValue():
    n = random.randrange(1,3)
    print("랜덤값:",n)
    if(n==2):
        raise Exception("랜덤값 2:error");
```

```
print("프로그램 시작")
```

```
try:
```

```
    randomValue()
except Exception as e:
    print("예외메시지:",e)
print("프로그램 종료")
```

프로그램 시작

랜덤값: 2

예외메시지: 랜덤값 2:error

프로그램 종료

11) 사용자 정의 예외 클래스

python

```
# 3) 사용자 정의 예외 클래스
```

```
class UserException(Exception):  
  
    def __init__(self,mesg):  
        self.mesg =mesg
```

```
import random
```

```
def randomValue():  
    n = random.randrange(1,3)  
    print("랜덤값:",n)  
    if(n==2):  
        raise UserException("랜덤값 2:error");
```

```
print("프로그램 시작")
```

```
try:  
    randomValue()
```

```
except Exception as e:  
    print("예외메시지:",e)  
print("프로그램 종료")
```

프로그램 시작
랜덤값: 2
예외메시지: 랜덤값 2:error
프로그램 종료

9장. 파일 입출력 및 정규 표현식



파일 입출력 개요

쓰기 모드 및 읽기 모드

with 문

정규 표현식

1) 파일 입출력 (File I/O)

python

1) 표준 입출력

```
# 1) 표준 입출력
num = input("이름을 입력하시오")
print(num)
```

이름을 입력하시오 흥길동
흥길동

help(open)

Character Meaning

Character	Meaning
'r'	open for reading (default)
'w'	open for writing, truncating the file first
'x'	create a new file and open it for writing
'a'	open for writing, appending to the end of the file if it exists
'b'	binary mode
't'	text mode (default)
'+'	open a disk file for updating (reading and writing)
'U'	universal newline mode (deprecated)

2) 파일 입출력

문법: f = open(파일명, 파일처리모드)
f.close()

처리 방법	설명
r	읽기 - 파일의 자료를 읽기 할 때 사용
w	쓰기 - 파일로 자료를 쓰기 할 때 사용
a	추가 - 파일의 뒷부분에 자료를 추가 시킬 때 사용

2) 파일 입출력 - 쓰기모드

python

* 파일 입출력 - 쓰기모드 (w , a)

```
# 2) 파일 입출력 - 쓰기 (덮어쓰기)

#help(open)

f = open("c:\\새파일.txt", "w")
for i in range(10):
    data = str(i)+"안녕2...\\n"
    f.write(data)
f.close()
```

```
# 3) 파일 입출력 - 쓰기 (append)

f = open("c:\\새파일.txt", "a")
for i in range(10):
    data = str(i)+"Hello2...\\n"
    f.write(data)
f.close()
```

3) 파일 입출력 – 읽기 모드

python

* 파일 입출력 – 읽기모드(r)

```
# 1) 파일 입출력 - 읽기( readline() 함수 이용)
```

```
# 첫 줄 읽기  
f = open("c:\\새파일.txt", 'r')  
line = f.readline()  
print(line)  
f.close()
```

0안녕2...

```
# 모두 읽기 1 - readline() 함수 이용
```

```
f = open("c:\\새파일.txt", 'r')  
while True:  
    line = f.readline()  
    if not line: break  
    print(line)  
f.close()
```

0안녕2...

1안녕2...

```
# 모두 읽기 2 - readlines() 함수 이용
```

```
# 데이터를 모두 읽어서 List로 반환  
f = open("c:\\새파일.txt", 'r')  
lines = f.readlines()  
for line in lines:  
    print(line)  
f.close()
```

0안녕2...

1안녕2...

```
# 모두 읽기 3 - read() 함수 이용
```

```
# 데이터를 모두 읽어서 str로 반환  
f = open("c:\\새파일.txt", 'r')  
data = f.read()  
print(data)  
f.close()
```

0안녕2...

1안녕2...

2안녕2...

3안녕2...

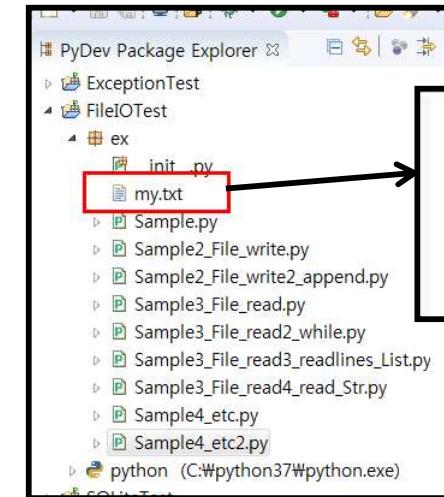
4) 파일 입출력 고려사항

python

1) 현재 모듈의 실행경로 알기

```
import os  
fdirectory = os.getcwd()      #현재 모듈의 경로 얻기  
print(fdirectory)
```

c:\python\Anaconda3\envs\test\pythonStudy



2) 예외처리 및 인코딩 처리

```
import os  
try:  
    f = open(os.getcwd() + r'\my.txt', 'r', encoding='utf-8')  
    print(f.read())  
    f.close()  
except e:  
    print("error", e)
```

hello
world
안녕하세요

```
print('파일의 일부분만 읽기')  
f = open(os.getcwd() + r'\ftest.txt')  
for i in range(3):      #3회 반복  
    line = f.readline();  
    print(line, end='');  
f.close()
```

5) with문

python

특징: with문을 사용하면 파일 처리시 자동으로 close() 처리되기 때문에 손쉽게 파일처리가 가능하다.

문법:

with open(파일명, 모드) as f:
 f.read | f.write

```
f = open("c:\\새파일.txt", 'w')
for i in range(10):
    data = str(i)+"안녕2...\\n"
    f.write(data)
f.close()
```

```
import os
try:
    f = open( os.getcwd() + r'\my2.txt' , 'r' , encoding='utf-8')
    print(f.read())
    f.close()
except FileNotFoundError as e:
    print("error",e)

error [Errno 2] No such file or directory: 'c:\\python\\Anaconda3\\honStudy\\my2.txt'
```

```
with open("c:\\새파일2.txt", 'w') as f:
    for i in range(4):
        data = "안녕...\\n"
        f.write(data)
```

```
import os
try:
    with open( os.getcwd() + r'\my2.txt' , 'r' , encoding='utf-8') as f:
        print(f.read())
except FileNotFoundError as e:
    print("error",e)

error [Errno 2] No such file or directory: 'c:\\python\\Anaconda3\\envs\\honStudy\\my2.txt'
```

6) 정규 표현식 (regular expression)

python

목적:

다음과 같은 기본 메타 문자를 이용하여 패턴과 일치하는 문자열을 찾거나 확인하는 방법이다.

. ^ \$ * + ? { } [] () \W |

주요함수:

```
import re

help(re.compile) # compile(pattern, flags=0)
help(re.match) # Try to apply the pattern at the start of the string, returning
               # a Match object, or None if no match was found.
help(re.fullmatch) # Try to apply the pattern to all of the string, returning
                   # a Match object, or None if no match was found.
help(re.search) # Scan through string looking for a match to the pattern, returning
                # a Match object, or None if no match was found.
help(re.findall) # Return a list of all non-overlapping matches in the string.
help(re.finditer) # Return an iterator over all non-overlapping matches in the
```

6) 정규 표현식 (regular expression)

python

문법 1: [abc] 는 a, b, c 중에서 한 개의 문자와 매치된다.

```
import re

# 1) 문자 클래스: [abc] ==> a, b, c 중에서 한 개의 문자와 매치
result = re.match('[abc]', "abcx")
result = re.match('[abc]', "bcx")
result = re.match('[abc]', "cx")
result = re.match('[abc]', "xabc") # None
result = re.fullmatch('[abc]', "a")
result = re.fullmatch('[abc]', "b")
result = re.fullmatch('[abc]', "c")
result = re.fullmatch('[abc]', "abc") # None
result = re.search('[abc]', "abcx")
result = re.search('[abc]', "bcx")
result = re.search('[abc]', "cx")
result = re.search('[abc]', "xabc")
result = re.findall('[abc]', "xabcabc") # ['a', 'b', 'c', 'a', 'b', 'c']
result = re.finditer('[abc]', "xabcabc") # <callable_iterator object at 0x0000000000000000>
```

6) 정규 표현식 (regular expression)

python

문법 2: [-]는 범위와 매치된다.

[a-z] : a-z 사이의 범위의 문자와 매치된다.

[A-Z] : A-Z 사이의 범위의 문자와 매치된다.

[0-9] : 0-9 사이의 범위의 문자와 매치된다.

[a-zA-Z0-9]: 영문자 및 숫자와 매치된다.

```
import re
result = re.match('[a-d]', "xdbc") # None
result = re.search('[a-d]', "xdbc")
result = re.match('[A-Z]', "Abc")
result = re.match('[0-9]', "9Abc")
result = re.match('[a-zA-Z]', "xAbs")
result = re.match('[a-zA-Z0-9]', "99xAbs")
result = re.findall('[a-zA-Z]', "99xAbs") # ['x', 'A', 'b', 'c']
```

6) 정규 표현식 (regular expression)

python

문법 3: [^] 는 [-] 표현식의 부정과 매치된다.

```
import re

result = re.match('[^a-d]', "Adbc")
result = re.search('[^a-d]', "adbc") # None
result = re.match('[^a-d]', "9dbc")
result = re.match('[^a-d]', "\uacfcdbc")
result = re.findall('[^a-zA-Z]', "99xAbc") # ['x', 'A', 'b', 'c']
```

6) 정규 표현식 (regular expression)

python

문법 4: 간추린 표현식

‘\d’ : [0-9] 와 동일

‘\D’ : [^0-9] 와 동일

‘\s’ : 공백문자와 일치

‘\S’ : 공백문자가 아닌 문자와 일치

‘\w’ : [a-zA-Z0-9] 와 동일

‘\W’ : [^a-zA-Z0-9] 와 동일

```
import re

result = re.match('\d', "9Abc")
result = re.match('\D', "Abc")
result = re.match('\s', ' 9Abc')
result = re.search('\s', '9 Abc')
result = re.match('\S', '9Abc')
result = re.match('\w', '홍9Abc')
result = re.match('\W', '9Abc') # None
```

6) 정규 표현식 (regular expression)

python

문법 5: . (dot)

모든 문자 하나와 매치된다. (\n 제외)

```
import re

result = re.match('a.b', 'axb')
result = re.match('a.b', 'a0b')
result = re.match('a.b', 'aHongb')
result = re.match('a.b', 'axxb') # None
```

문법 6: 문자?

?앞의 문자가 없거나 하나의 문자와 매치된다. (0 또는 1)

```
import re

result = re.match('ca?b', 'cb')
result = re.match('ca?b', 'cab')
result = re.match('ca?b', 'caab') # None
result = re.match('ca?b', 'caaab') # None
```

6) 정규 표현식 (regular expression)

python

문법 7: 문자+

+앞의 문자가 1 번 이상 반복되는 문자와 매치된다

```
import re
result = re.match('ca+b', 'cb') # None
result = re.match('ca+b', 'cab')
result = re.match('ca+b', 'caaaab')
result = re.match('ca+b', 'cacb') # None
```

문법 8: *

*앞의 문자가 0 번 이상 반복되는 문자와 매치된다.

```
import re
result = re.match('ca*b', 'cb')
result = re.match('ca*b', 'cab')
result = re.match('ca*b', 'caaaab')
```

6) 정규 표현식 (regular expression)

python

- 문법 9: 문자{m} - m 번 반복하는 문자와 매치된다 .
문자{m,n} - m ~ n번 반복하는 문자와 매치된다 .
문자{m,} - m ~ 무한대 반복하는 문자와 매치된다 .
문자{,n} - 0 ~ n번 반복하는 문자와 매치된다 .

```
import re

result = re.match('ca{3}b', 'cb') # None
result = re.match('ca{3}b', 'cab') # None
result = re.match('ca{3}b', 'caaab')

result = re.match('ca{2,}b', 'cb') # None
result = re.match('ca{2,}b', 'cab') # None
result = re.match('ca{2,}b', 'caab')
result = re.match('ca{2,}b', 'caaab')

result = re.match('ca{,2}b', 'cb')
result = re.match('ca{,2}b', 'cab')
result = re.match('ca{,2}b', 'caab')
result = re.match('ca{,2}b', 'caaab') # None
```

6) 정규 표현식 (regular expression)

python

```
result = re.match('ca{1,2}b', 'cb') # None
result = re.match('ca{1,2}b', 'cab')
result = re.match('ca{1,2}b', 'caab')
result = re.match('ca{1,2}b', 'caaab') # None
```

문법 10: [a-z] +

+앞의 [a-z] 범위문자가 1 번 이상 반복되는 문자와 매치된다

```
import re
result = re.match('[a-z]+', '') #None
result = re.match('[a-z]+', 'a')
result = re.match('[a-z]+', 'abc')
result = re.match('[a-z]+', '9abc') #None
result = re.search('[a-z]+', '9abc')
```

6) 정규 표현식 (regular expression)

python

문법 10: | - 또는 의미

^ - 문자열의 맨 처음과 일치함을 의미.

\$ - 문자열의 맨 끝과 일치함을 의미.

(ABC)+ - 그룹문자인 ABC가 1번 이상 반복

```
import re

result = re.match('Crow|Servo', 'CrowHello')
result = re.match('Crow|Servo', 'ServoHello')
result = re.search('Crow|Servo', 'XXSServoHello')

result = re.match('^Life', 'Life is XXXX')
result = re.match('^Life', 'My Life is XXXX') # None

result = re.search('t$', 'Life is to short')
result = re.search('t$', 'My Life is to short')

result = re.search('(ABC)+', "ABCABCABC ok?")
result = re.search('(ABC)+', "AB-2 ABC ok?")
```

10장. 데이터베이스 연동



MariaDB 다운로드 및 설치

HeidiSQL 툴 실행

MariaDB 드라이버 설정 (mysqlclient)

SQL 및 CRUD 코드 작성

1) Maria DB

python

가. 프로그램 다운로드

<https://downloads.mariadb.org/mariadb/10.4.11/>

MariaDB 10.4.11 Stable 2019-12-11

[View all releases](#)

[Release Notes](#)

[Changelog](#)

Affordable, enterprise class product support, professional services, and training for your MariaDB database is available from the MariaDB Foundation's release sponsor, MariaDB Corporation. To learn more about them and their services for MariaDB, visit their [website](#), or email MariaDB Corporation at sales@mariadb.com.

File Name	Package Type	OS / CPU	Size	Meta
Galera 26.4.3 source and packages		Source		
For best results with RPM and DEB packages, use the Repository Configuration Tool .				
mariadb-10.4.11.tar.gz	source tar.gz file	Source	78.4 MB	Checksum Instructions
mariadb-10.4.11-winx64-debugsymbols.zip	ZIP file	Windows x86_64	103.5 MB	Checksum Instructions
mariadb-10.4.11-winx64.msi	MSI Package	Windows x86_64	56.6 MB	Checksum Instructions
mariadb-10.4.11-winx64.zip	ZIP file	Windows x86_64	64.3 MB	Checksum Instructions

Operating System

- DEB Package
- Generic Linux
- RPM Package
- Source Code
- Windows

Package Type

- MacOS pkg
- DEB Package

1) Maria DB

python

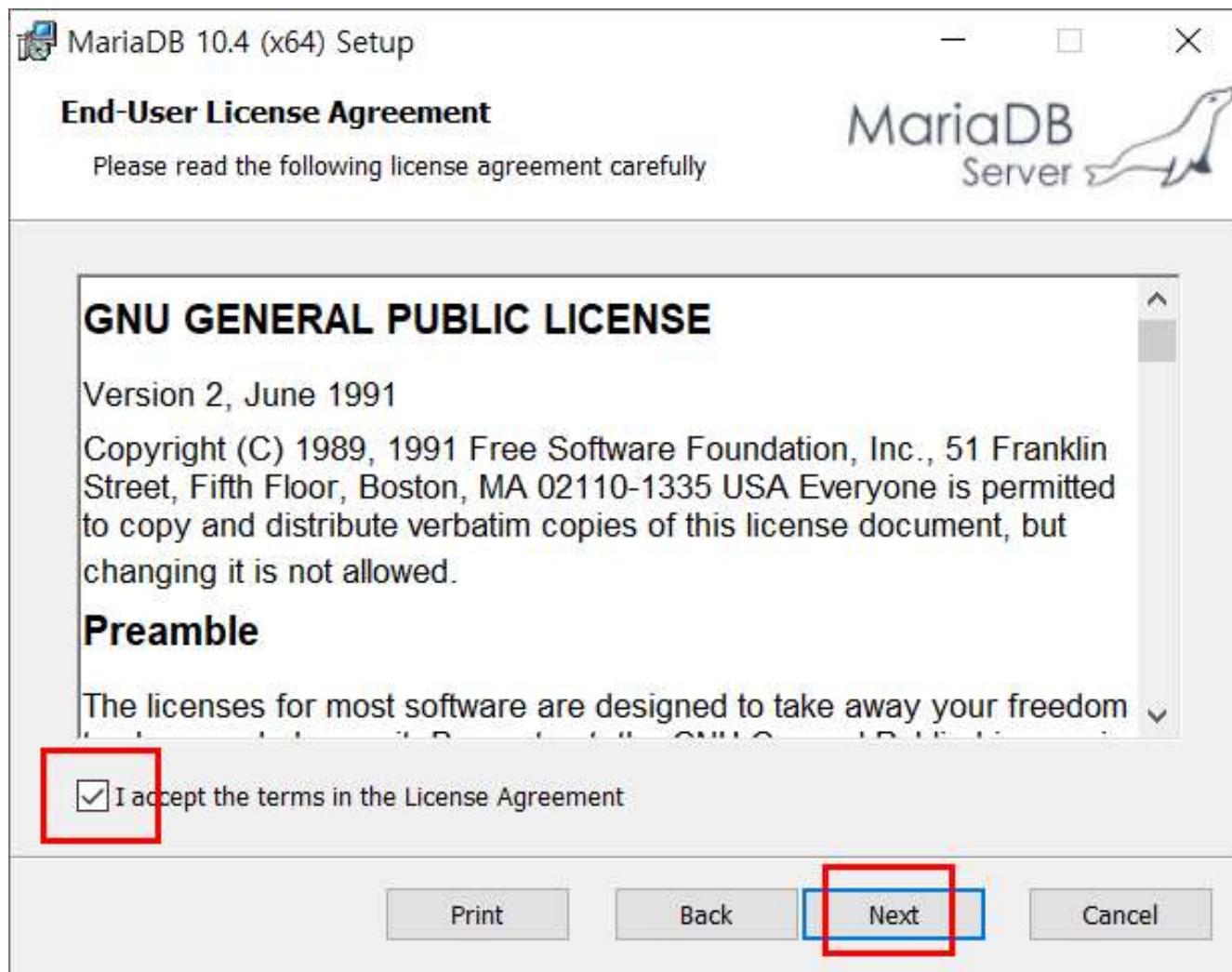
나. 프로그램 설치



1) Maria DB

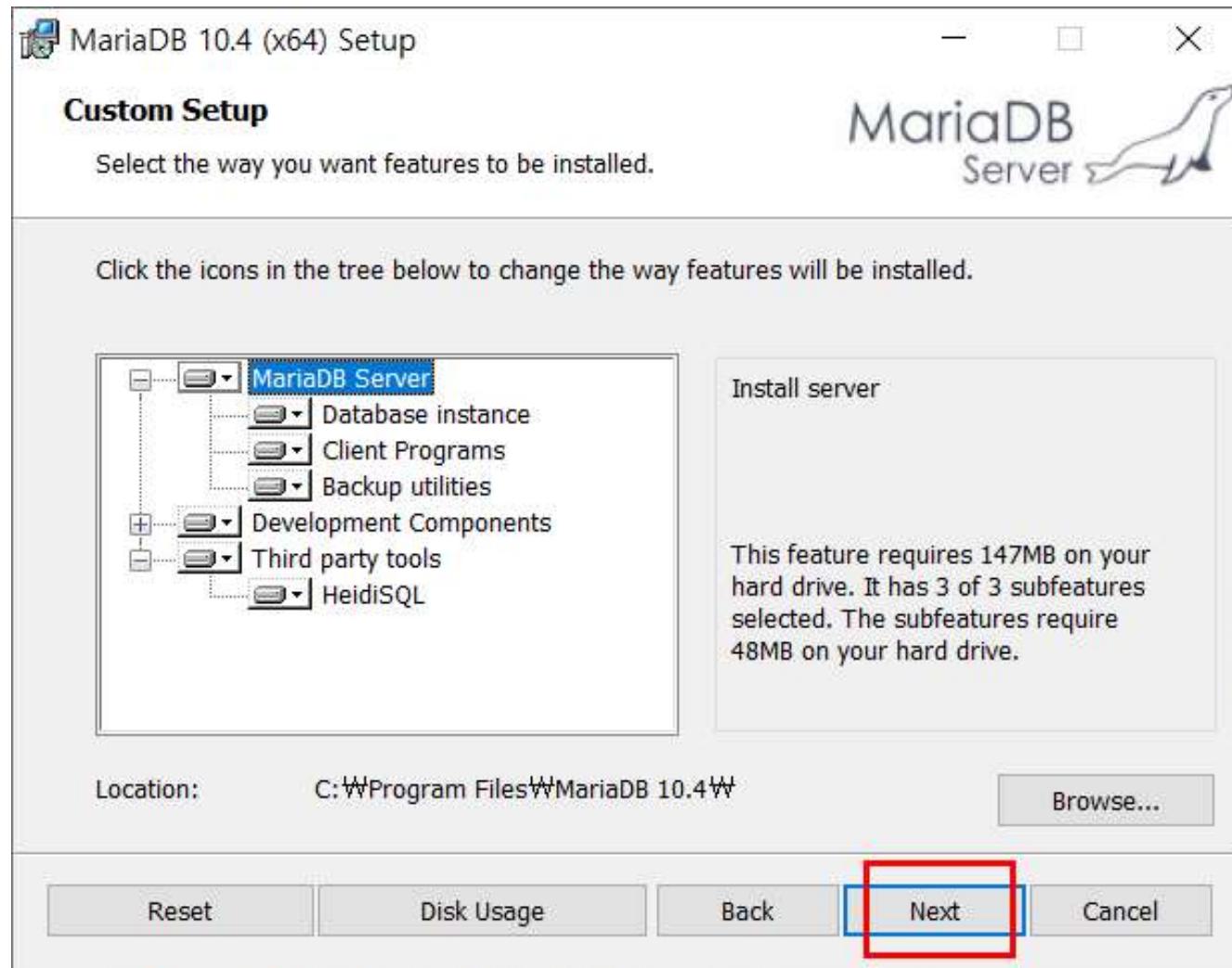
python

다. 라이선스 동의



1) Maria DB

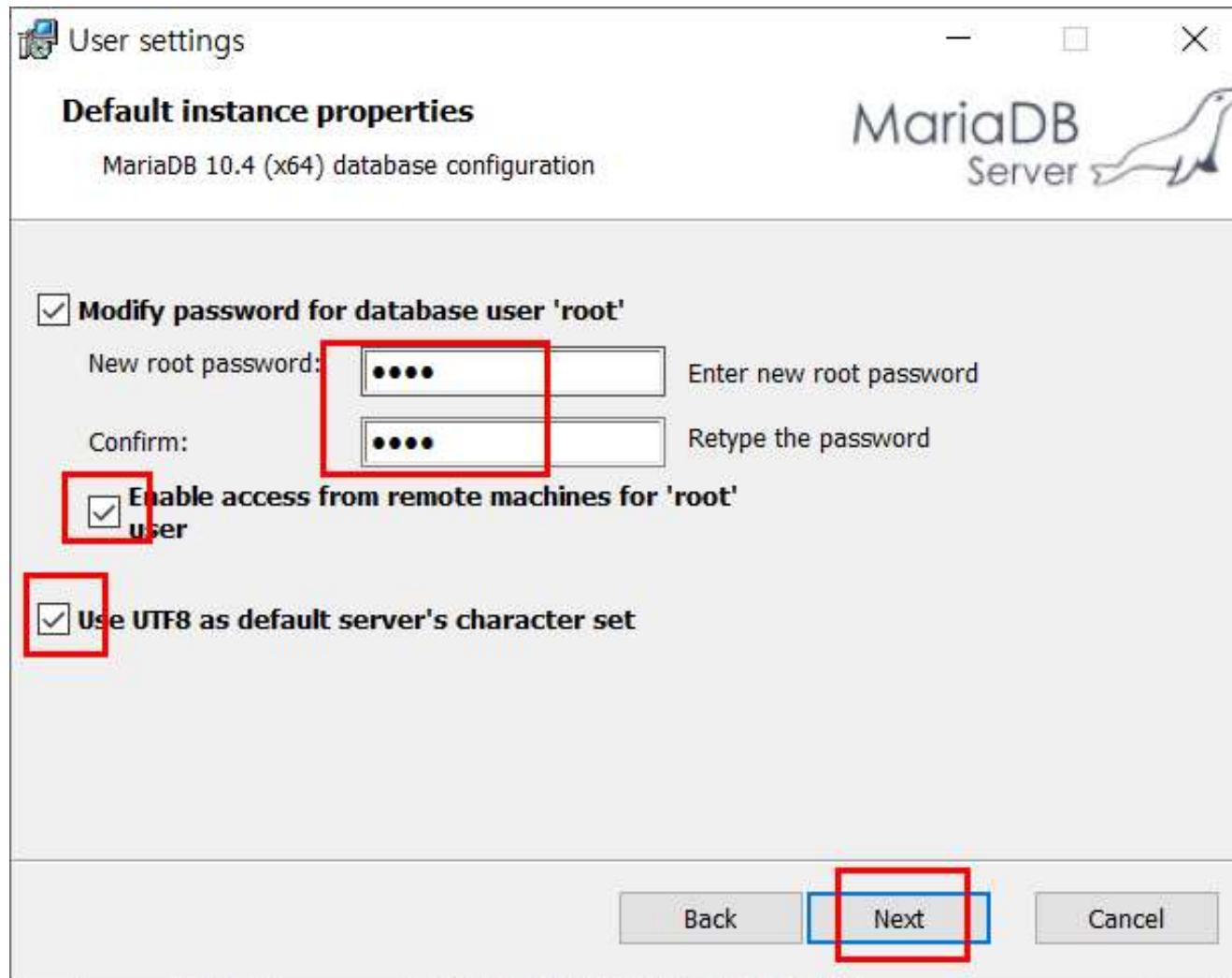
python



1) Maria DB

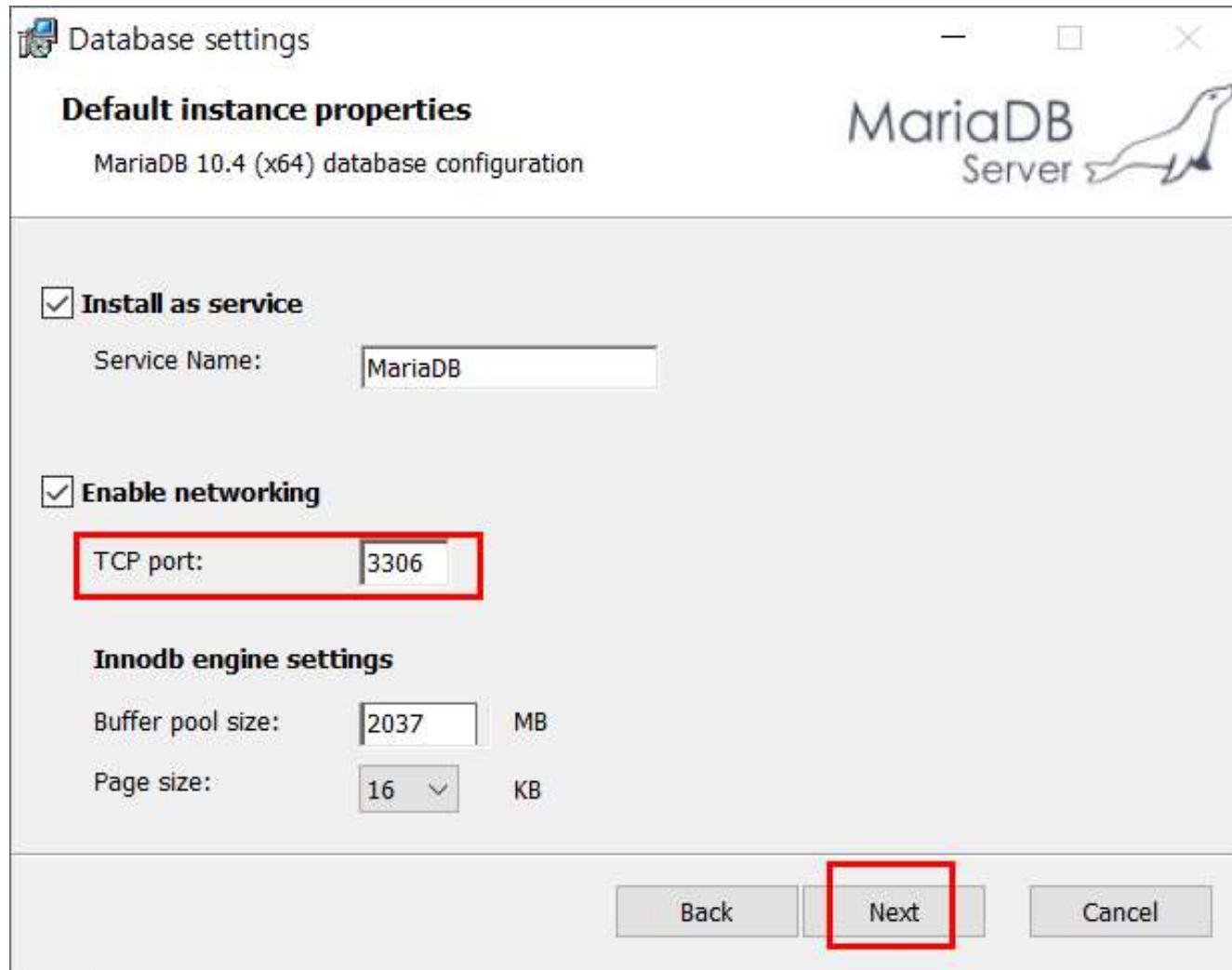
python

라. root 관리자 비밀번호 설정 (1234 로 설정)



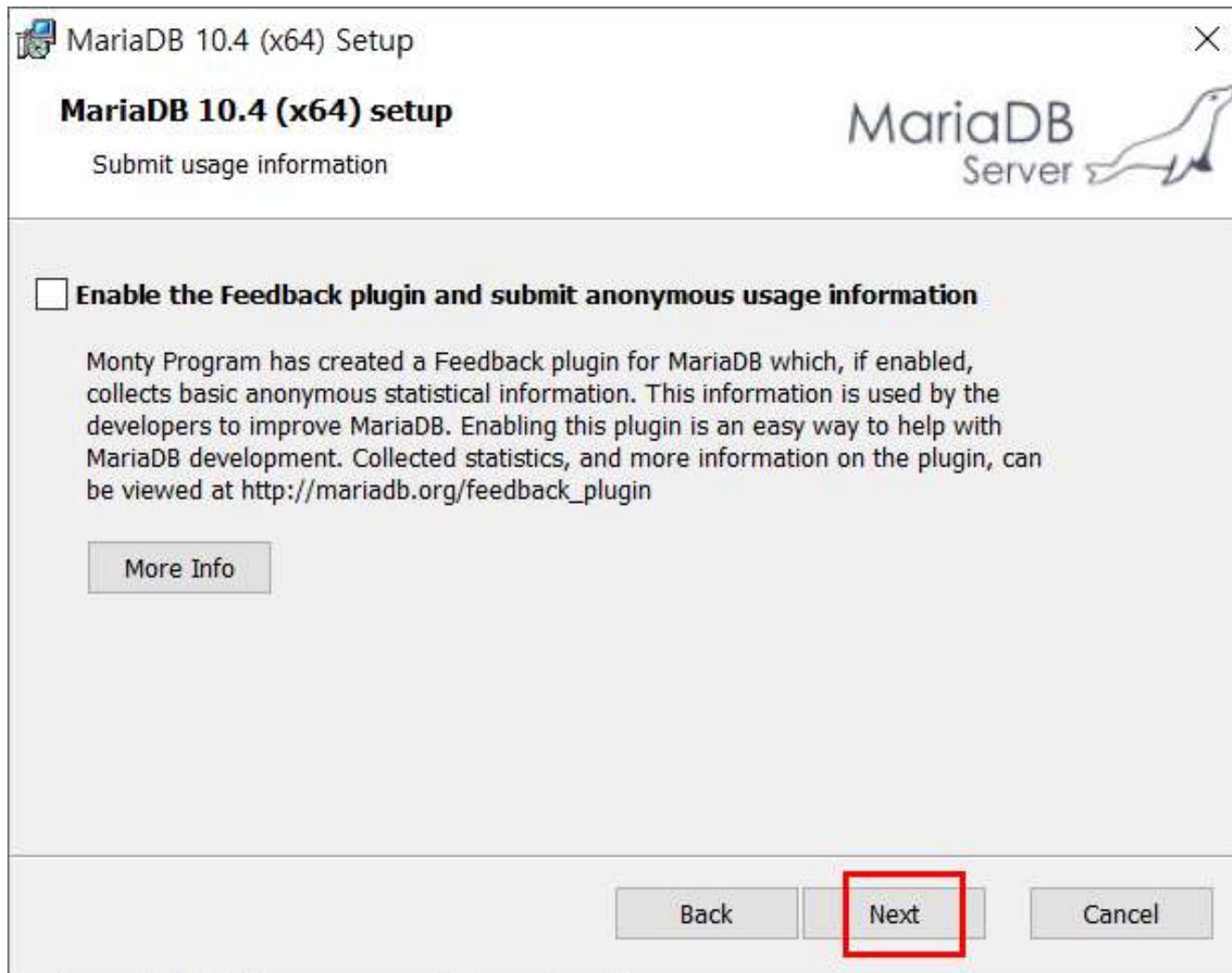
1) Maria DB

python



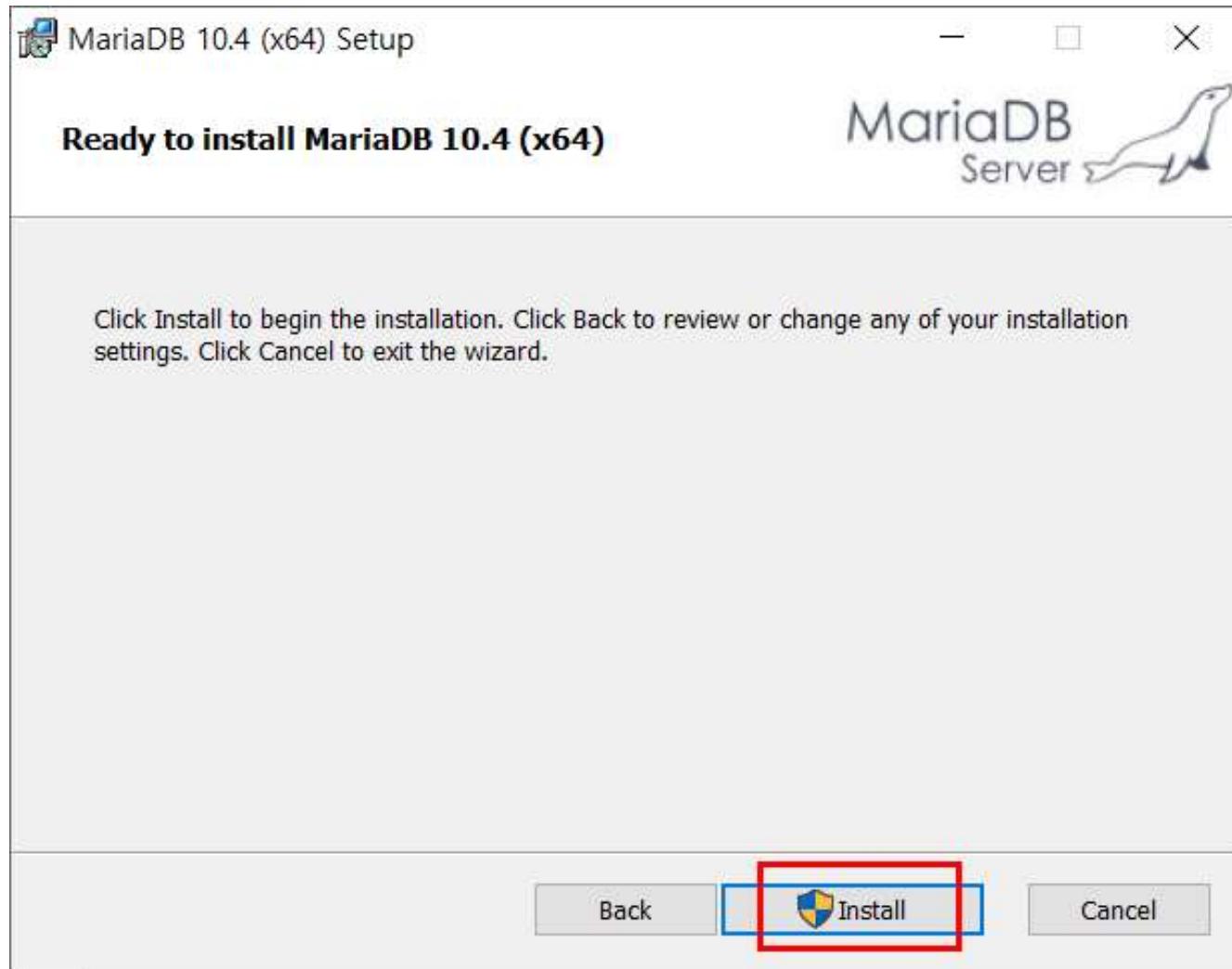
1) Maria DB

python



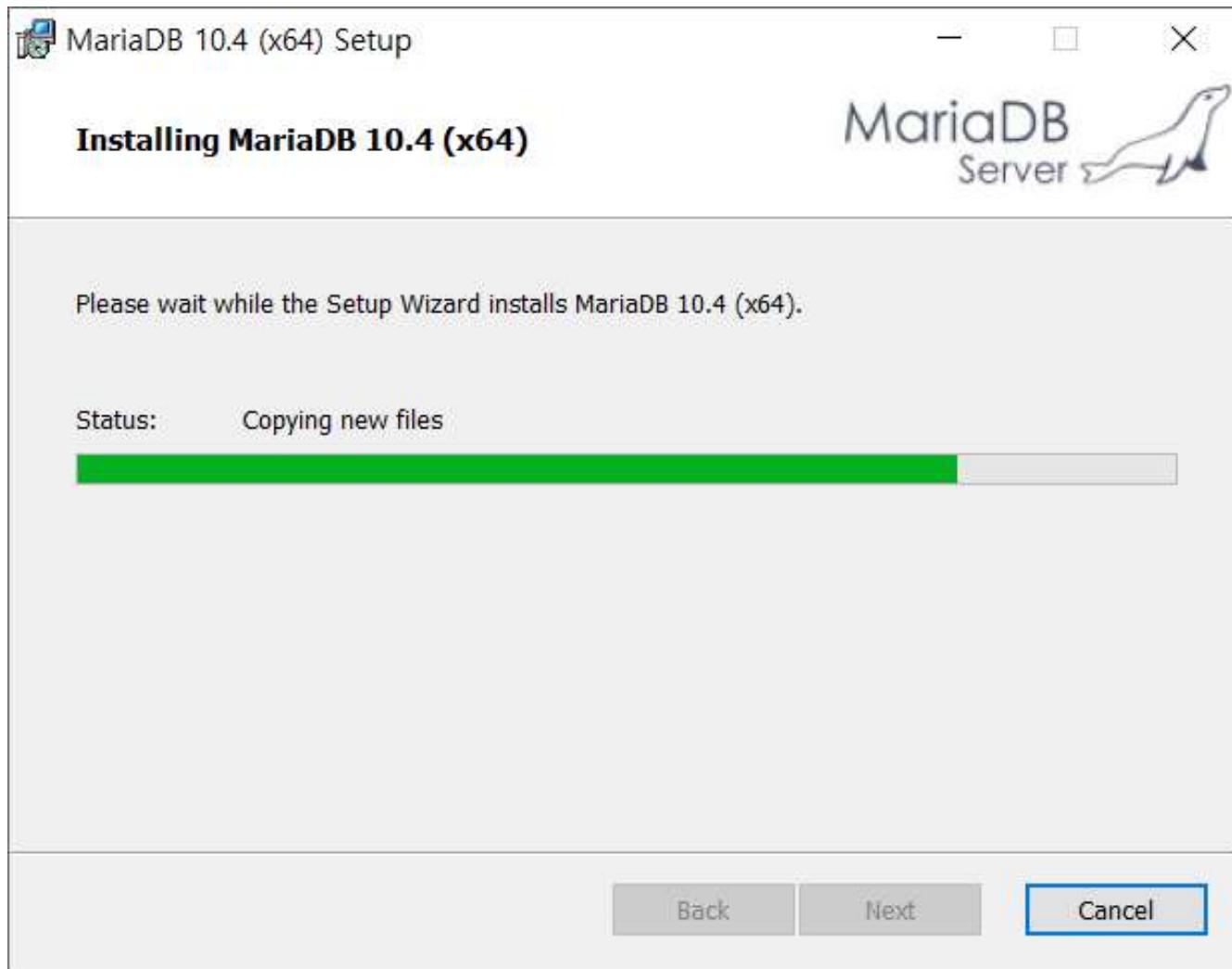
1) Maria DB

python



1) Maria DB

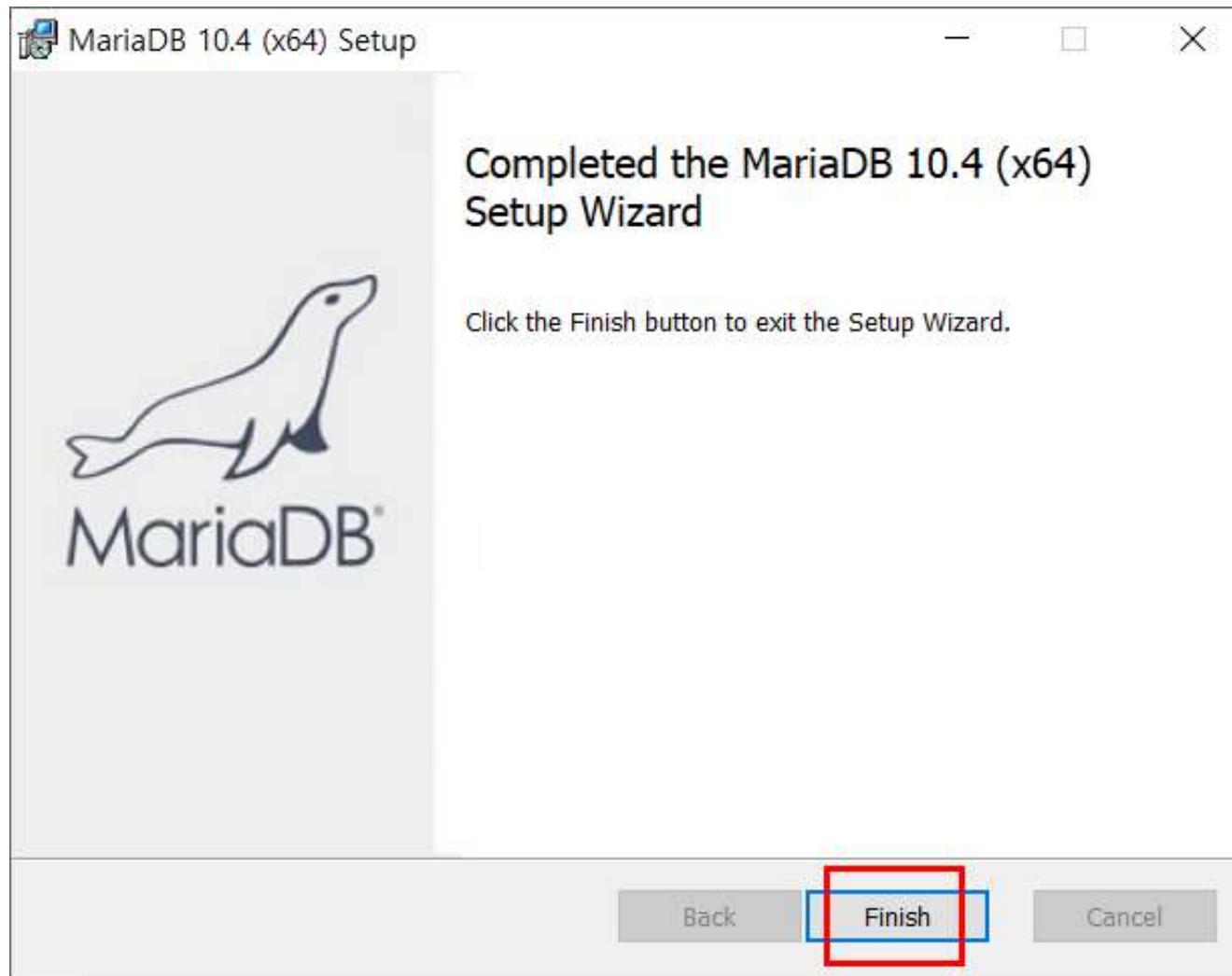
python



1) Maria DB

python

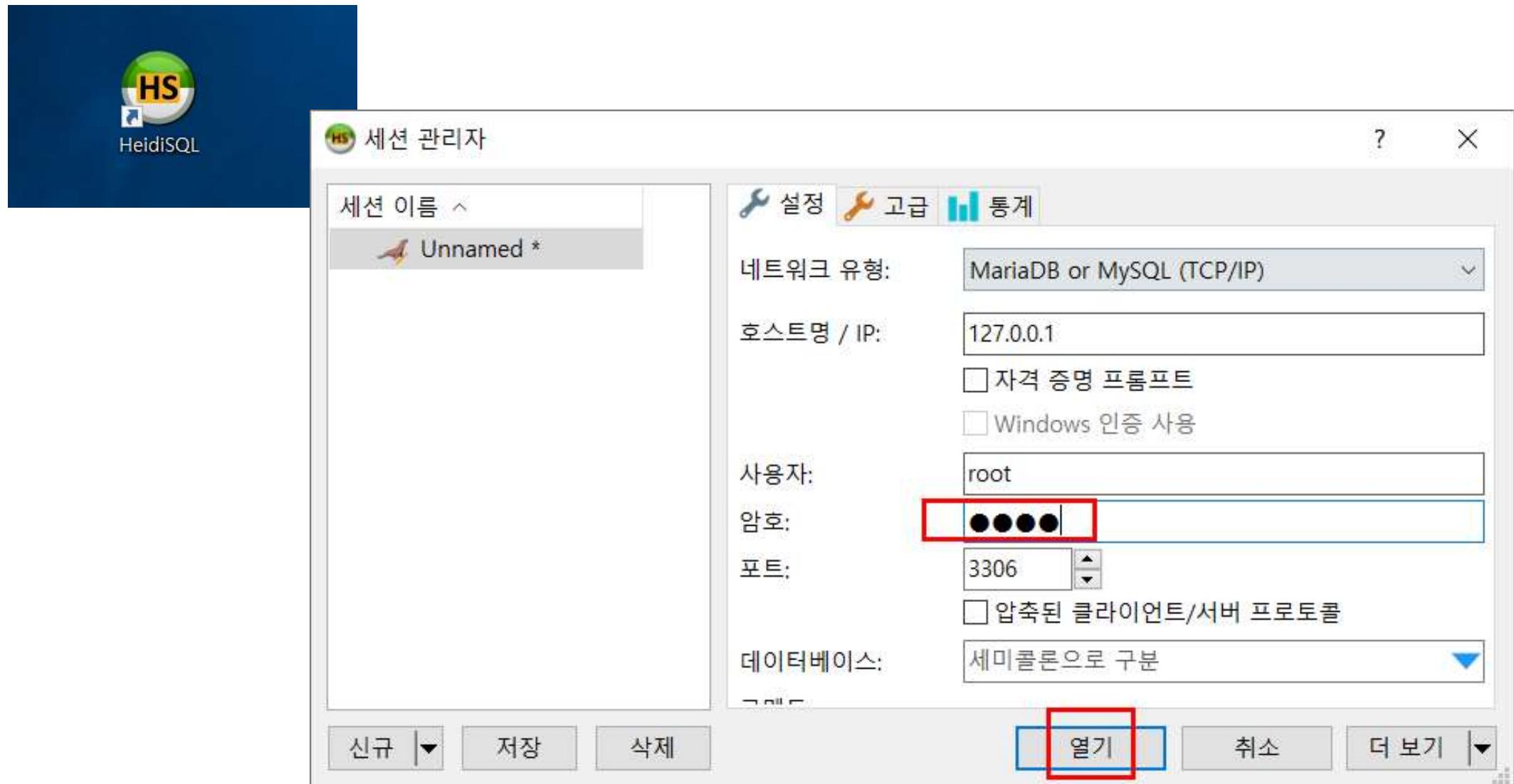
마. 프로그램 설치 완료



2) HeidiSQL 툴

python

가. HeidiSQL 툴 실행



2) HeidiSQL 툴

python

나. 빌트인 Database

The screenshot shows the HeidiSQL interface version 10.2.0.5599. On the left, the database tree shows four built-in databases: information_schema, mysql, performance_schema, and test, which are highlighted with a red box. The main pane displays a table of database statistics:

데이터베이스	크기	항목	마지...	테이블	뷰	함수
information_schema	192.0 KiB	77	2020...	77	0	0
mysql	0 B	52		52	0	0
performance_schema	0 B	52		0	0	0
test	0 B	0		0	0	0

At the bottom, the SQL query window contains the following code:

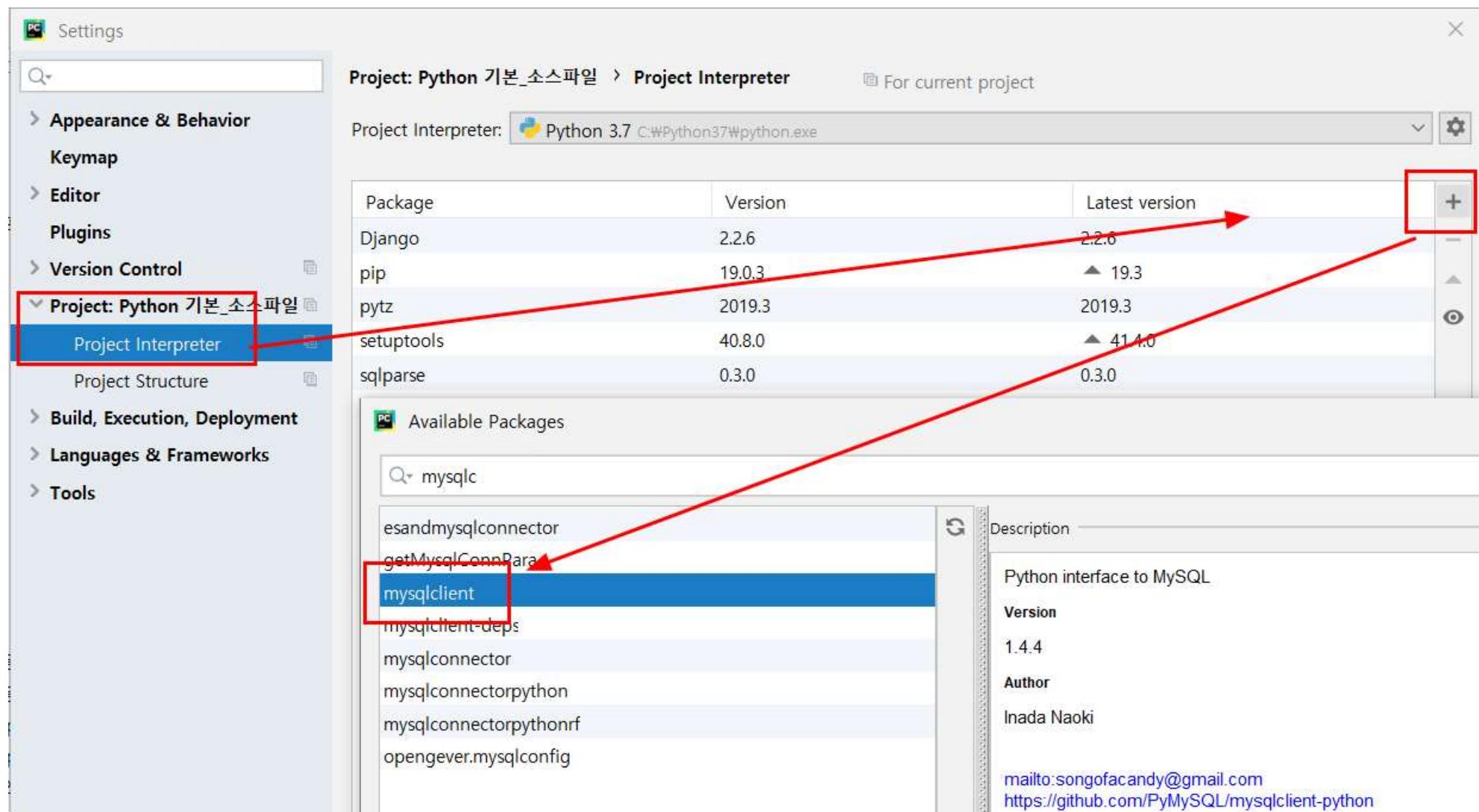
```
29 SELECT *, EVENT_SCHEMA AS `Db`, EVENT_NAME AS `Name` FROM information_schema.EVENTS WHERE `EVENT_SCHEMA`='performance_schema';
30 USE `test`;
31 SHOW COLLATION;
32 SHOW VARIABLES LIKE 'collation_server';
```

Information at the bottom of the interface includes: 연결됨: 00:01 MariaDB 10.4.11, 가동 시간: 00:06 h, 서버 시간: 오 유 휴.

3) Maria DB 연동 위한 드라이버 설치

python

* Mariadb 드라이버 설정



4) Maria DB 연동

python

* Mariadb 접속 확인

```
import MySQLdb

conn = MySQLdb.connect(host = '127.0.0.1', \
                       user = 'root', password='1234', database='testdb')
print(conn)
#<_mysql.connection open to '127.0.0.1' at 00000000027F33C8>
conn.close()
```

* Select 문

```
#sangdata 테이블 자료 읽기
sql = "select code, sang, su, dan from sangdata"
cursor.execute(sql)

#출력 실습 1)
for data in cursor.fetchall():
    #print(data)
    print('%s %s %s %s'%data)

#출력 실습 2)
print()
cursor.execute(sql)
for r in cursor:
    #print(r)
    print(r[0], r[1], r[2], r[3])

#출력 실습 3)
print()
cursor.execute(sql)
#필드의 자료가 변수와 일대일 매칭
for (code, sang, su, dan) in cursor:
    print(code, sang, su, dan)
```

4) Maria DB 연동

python

* insert 문

```
#-----  
#insert, update, delete에 대한 연습을 위한 코드를 입력할 영역  
sql = "insert into sangdata(code,sang,su,dan) values(10, '상품1', 2, 2000)"  
cursor.execute(sql)  
conn.commit()  
  
#Sqlite와 달리 원격 DB에서는 %s 서식을 사용해야 한다.  
sql = "insert into sangdata values(%s, %s, %s, %s)"  
sql_data = ('11', '상품2', 5, 1000) #tuple type으로 %s와 매핑되게 한다.  
cursor.execute(sql, sql_data)  
conn.commit()
```

* update/delete 문

```
#insert, update, delete에 대한 연습을 위한 코드를 입력할 영역  
sql = "update sangdata set sang=%s,su=%s,dan=%s where code=%s"  
sql_data = ('파이썬', 7, 7000, '11')  
cursor.execute(sql, sql_data)  
conn.commit()  
  
code = 10  
sql = "delete from sangdata where code=%s"  
cursor.execute(sql, (code,)) #%s에 전달 값은 tuple type 이어야 한다.  
conn.commit()
```



감사합니다.