



Python 이란

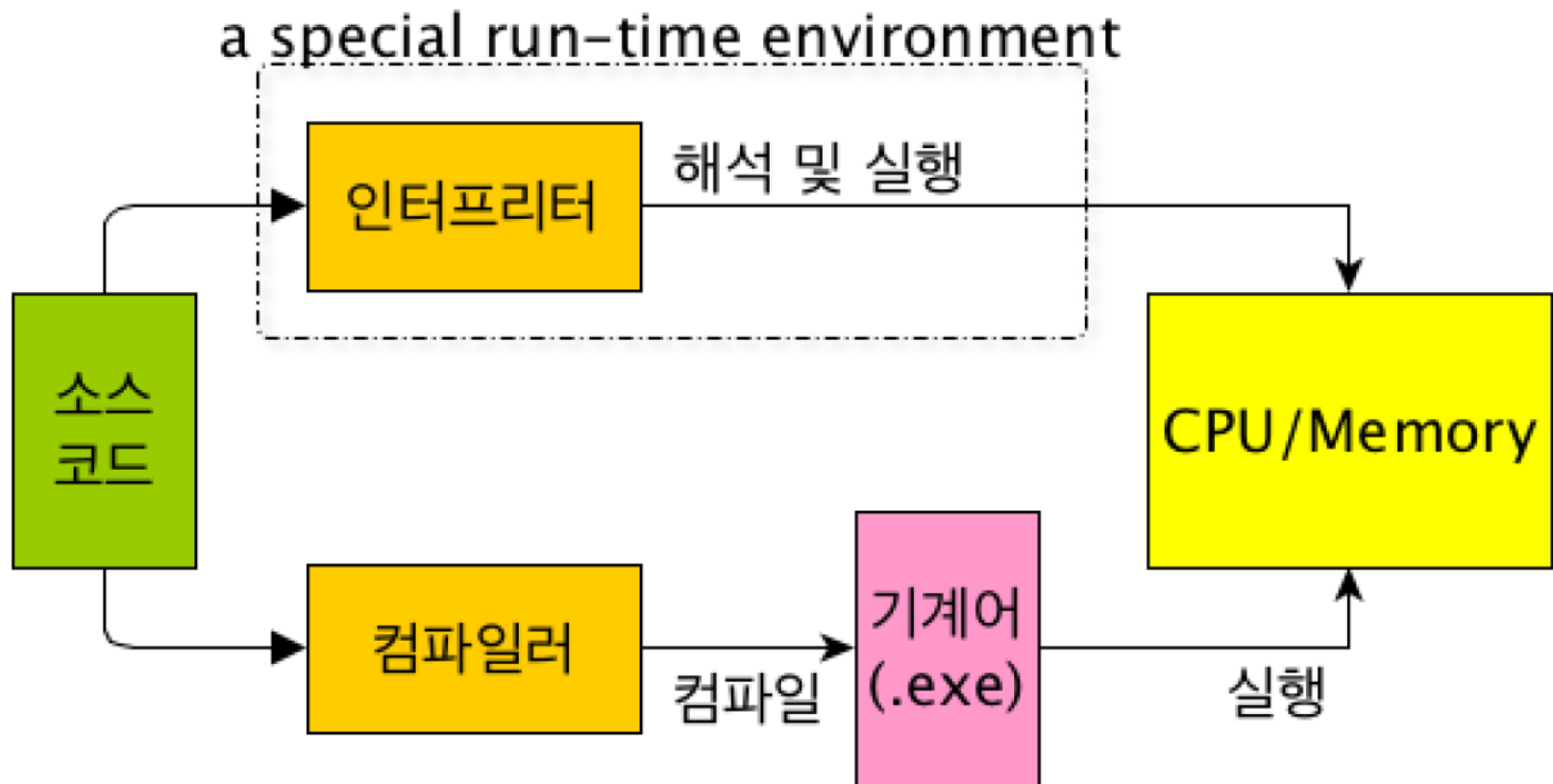
스크립트 언어의 이해

- 스크립트 언어란 무엇인가?

- 소스코드 = 스크립트
- 연기가자가 스크립트를 보고 연기 = 컴퓨터가 스크립트를 읽어 수행
- 해석과 수행을 함
- 라인 단위로 해석하여 수행함

스크립트 언어의 이해

- 컴파일 언어와 스크립트 언어와의 비교



스크립트 언어의 이해

- **대표적인 스크립트 언어**

JavaScript

ActionScript

Perl

PHP

Python

Lua

Ruby

- 브라우저에서 웹페이지를 동적으로 만들어주는 Javascript
- Actionscript : flash 개발 시 사용
- Perl : 요즘에 많이 사용 x
- PHP : 동적인 웹 페이지 구현 시 많이 사용
- Lua, Ruby → 최근에 개발된 스크립트 언어

프로그래밍 언어, 파이썬

- 파이썬 (Python)

- 1991년 귀도 반 로섬 (Guido van Rossum)이 개발
- 초보자가 쉽게 배울 수 있는 프로그래밍 언어



- 파이썬의 장점

- 비전공자도 쉽게 배울 수 있음
- 다양한 분야에서 활용할 수 있음
- 대부분의 운영체제에서 동일하게 사용됨
- Guido가 생각했던 Python 문법적 특징은 들여쓰기를 철저히 지키도록 언어를 설계했다는 점이다.
- 이는 코드의 가독성을 현격히 높여준다.
- C 언어에서처럼 {} 등의 괄호를 넣지 않기 때문에 프로그램을 좀더 깔끔하게 만들어준다.
- 파이썬 코드는 재사용하기가 쉽다.
- 코드의 분석이 쉽기 때문에 다른 사람이 작성한 코드를 받아서 작업하는 사람들이 훨씬 더 작업을 편하게 해준다.

- 파이썬의 단점

- C언어에 비해 일반적으로 10~350배 느림
- 최근에는 컴퓨터 성능이 좋아져 연산이 많이 필요한 프로그램이 아니라면 차이 크게 느낄 수 없음



- 파이썬의 필요성

- 가장 중요한 대답: "생산성이 높기 때문"
- 먼저 개발하라! 그리고 나서 성능을 개선하라.

- 파이썬의 특징

- 대화 기능의 인터프리터 언어

대화 기능 → 마치 컴퓨터와 개발자가 대화하는 듯한 느낌

- 동적인 데이터 타입 결정 지원
- 플랫폼 독립적 언어

- 파이썬의 특징

- 플랫폼 독립적 언어
- 개발 기간 단축에 초점을 둔 언어
- 간단하고 쉬운 문법
- 고수준의 내장 객체 자료형 제공

- 파이썬의 특징

- 메모리 자동 관리

- 쉬운 유지 보수

- 많은 수의 라이브러리 제공

코딩 시 필요한 모듈 끌어와 사용 가능

- 짧아지는 코드

C언어 또는 Java → 100라인, 파이썬 → 5~10라인

- 높은 확장성

직접 만든 모듈도 다른 사람에게 제공 가능

- 파이썬의 특징

- 메모리 자동 관리

- 쉬운 유지 보수

- 많은 수의 라이브러리 제공

코딩 시 필요한 모듈 끌어와 사용 가능

- 짧아지는 코드

C언어 또는 Java → 100라인, 파이썬 → 5~10라인

- 높은 확장성

직접 만든 모듈도 다른 사람에게 제공 가능

- 파이썬 활용처

- 시스템 유틸리티
- 라즈베리파이 기반의 IoT 디바이스
- GUI - wxpython, tkinter
- 게임 프로그래밍
 - 파이썬 게임엔진: PyOpenGL PySDL PyGame Kivy PyOgre Panda3D Cocos2D PySoy
- 웹 프로그래밍
 - Django 프레임워크
- 수치 프로그래밍
 - numpy 및 pandas 모듈
 - networks 모듈
- 데이터베이스 프로그래밍
- 기타

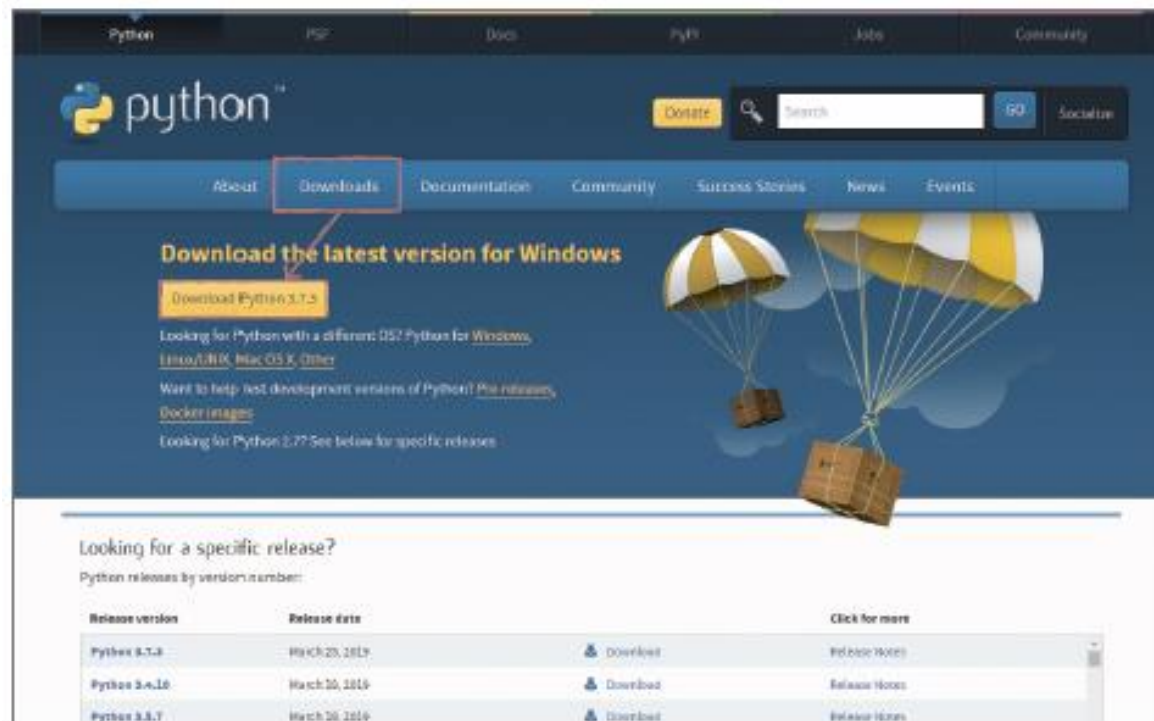
파이썬 개발환경 설정

파이썬 설치하기

- 파이썬 설치 프로그램 다운로드

1) 파이썬 공식 홈페이지 (<http://www.python.org>) 접속 – [Downloads]

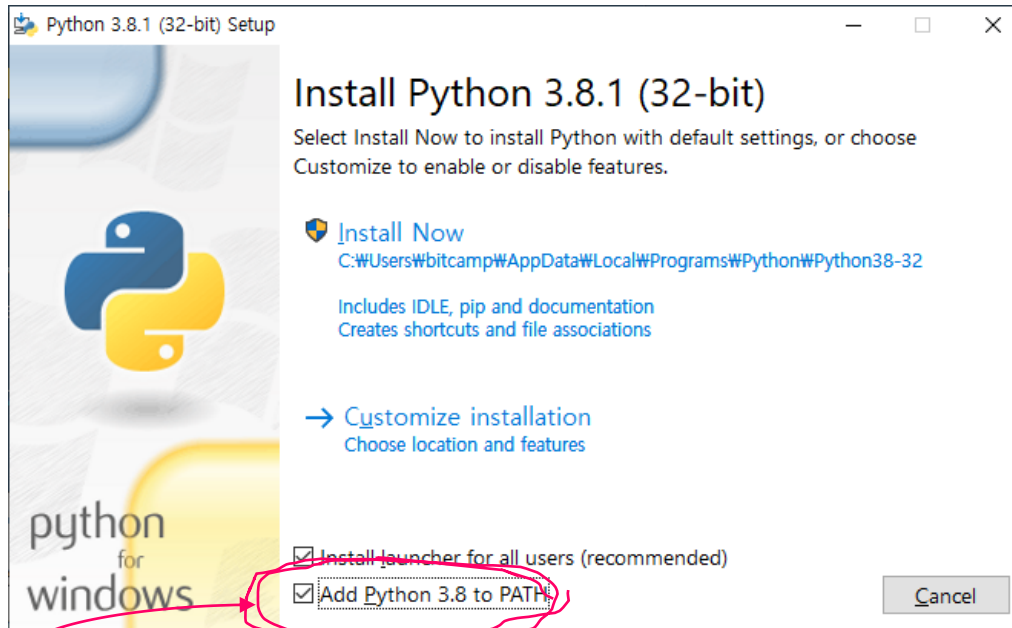
2) [Download Python 3.8.1] 클릭



파이썬 설치하기

- 파이썬 설치하기

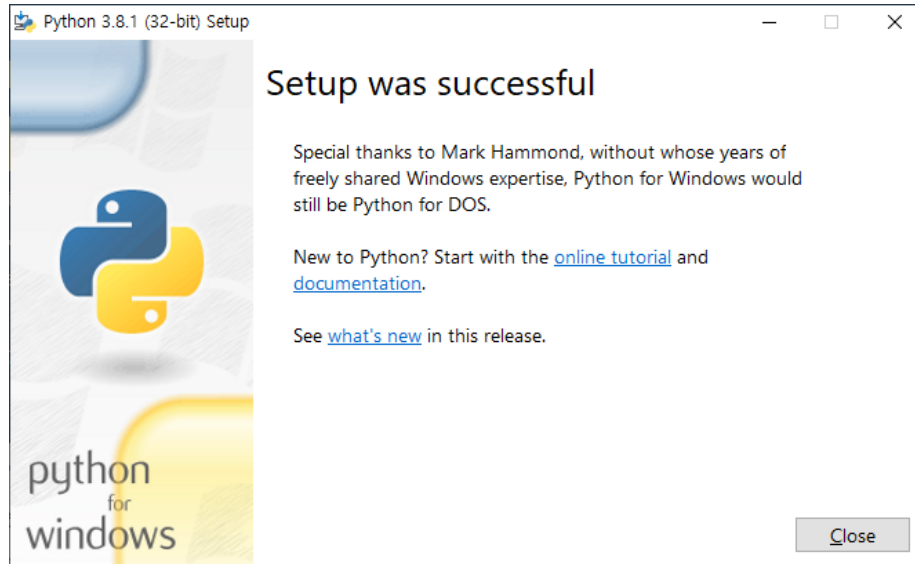
- 1) 설치 프로그램 실행하여 아래 화면에서 [Add Python 3.8 to PATH] 체크
- 2) [Install Now] 클릭



여기를 체크하지 않고 설치하면 파이썬이 실행되지 않으므로 꼭 확인합니다.

파이썬 설치하기

3) 설치 완료 화면이 나타나면 [Close] 클릭



4) 윈도우 [시작] 메뉴에서 [Python 3.7] 프로그램 확인

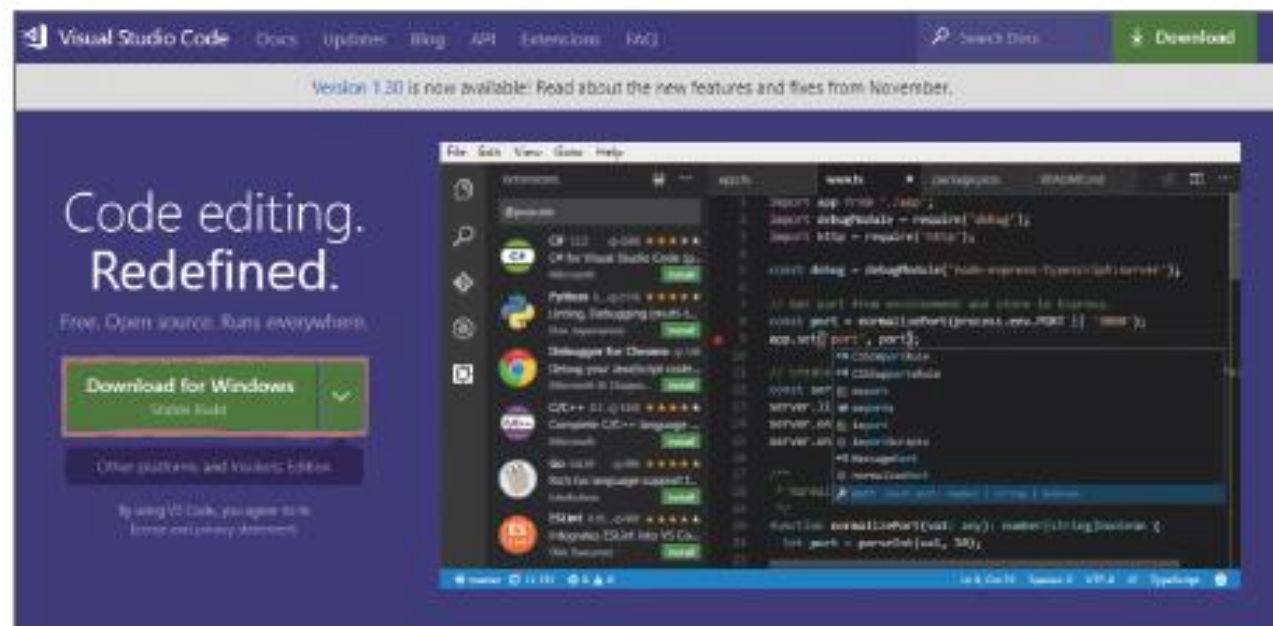


텍스트 에디터 사용하기 : 비주얼 스튜디오 코드

- 비주얼 스튜디오 코드 다운로드해 설치하기

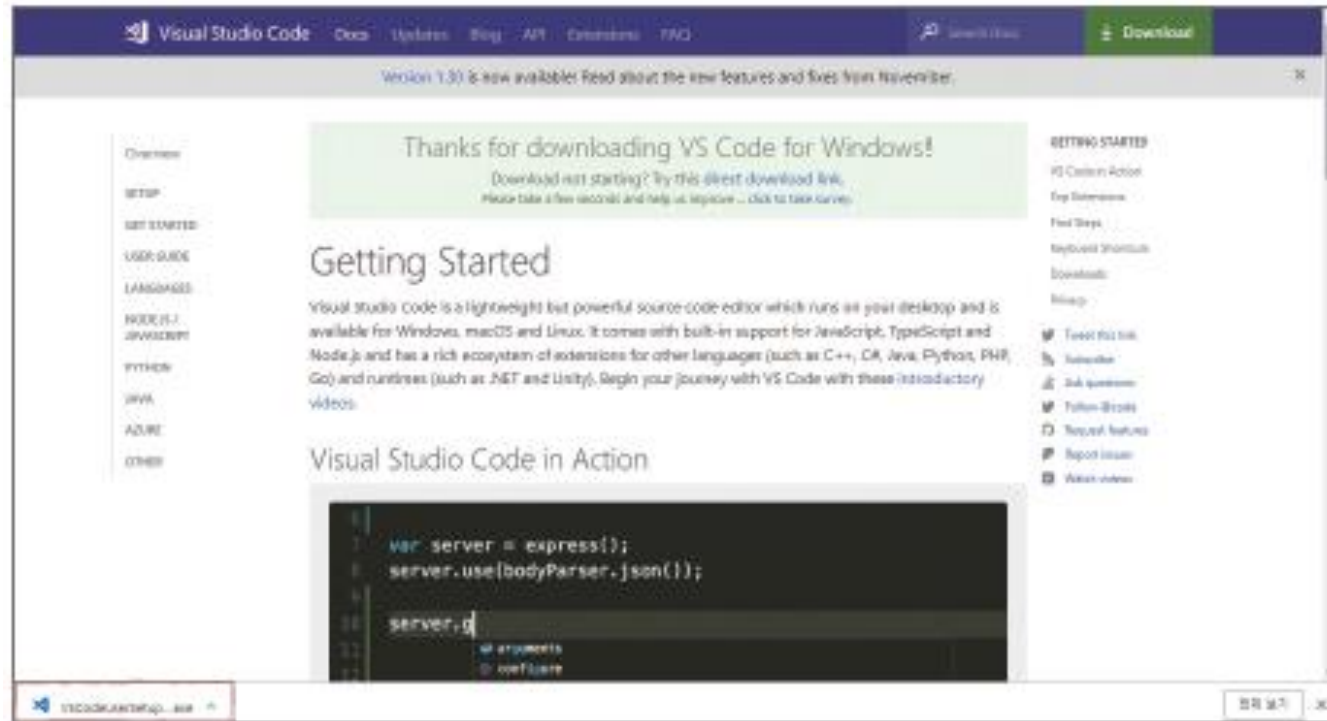
- 홈페이지 (<https://code.visualstudio.com>) 접속

1) [Downloads for Windows] 클릭하여 설치 파일 다운로드



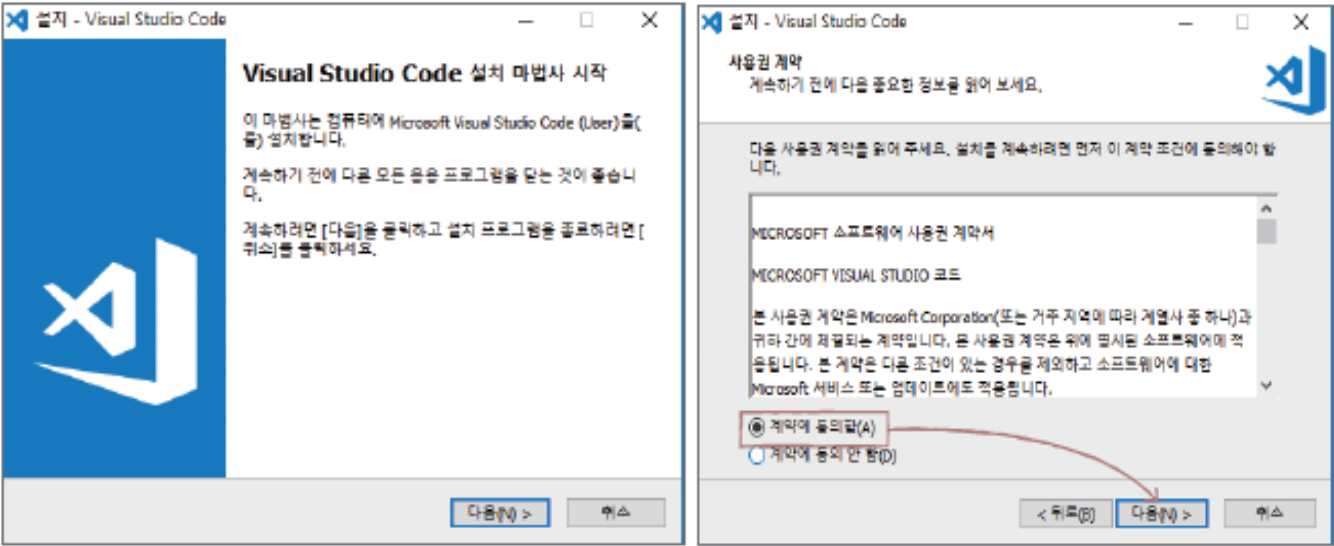
텍스트 에디터 사용하기 : 비주얼 스튜디오 코드

2) 다운로드한 설치 프로그램을 실행



텍스트 에디터 사용하기 : 비주얼 스튜디오 코드

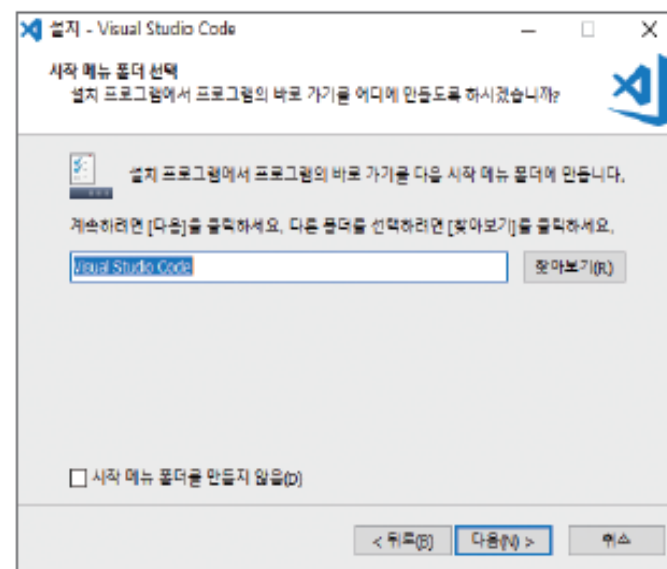
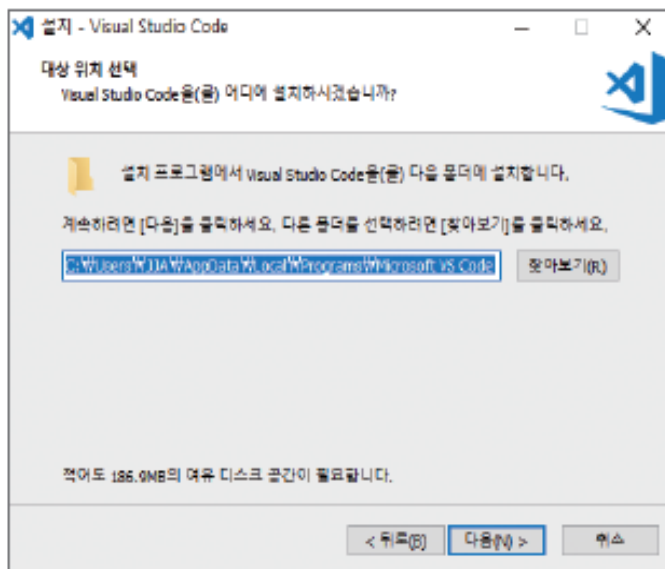
- 3) 설치 마법사 시작 화면에서 [다음] 클릭
- 4) [계약에 동의함] 선택 후 [다음] 클릭



텍스트 에디터 사용하기 : 비주얼 스튜디오 코드

5) 설치 폴더 지정 후 [다음] 클릭

6) 시작 메뉴 폴더 이름 지정 후 [다음] 클릭

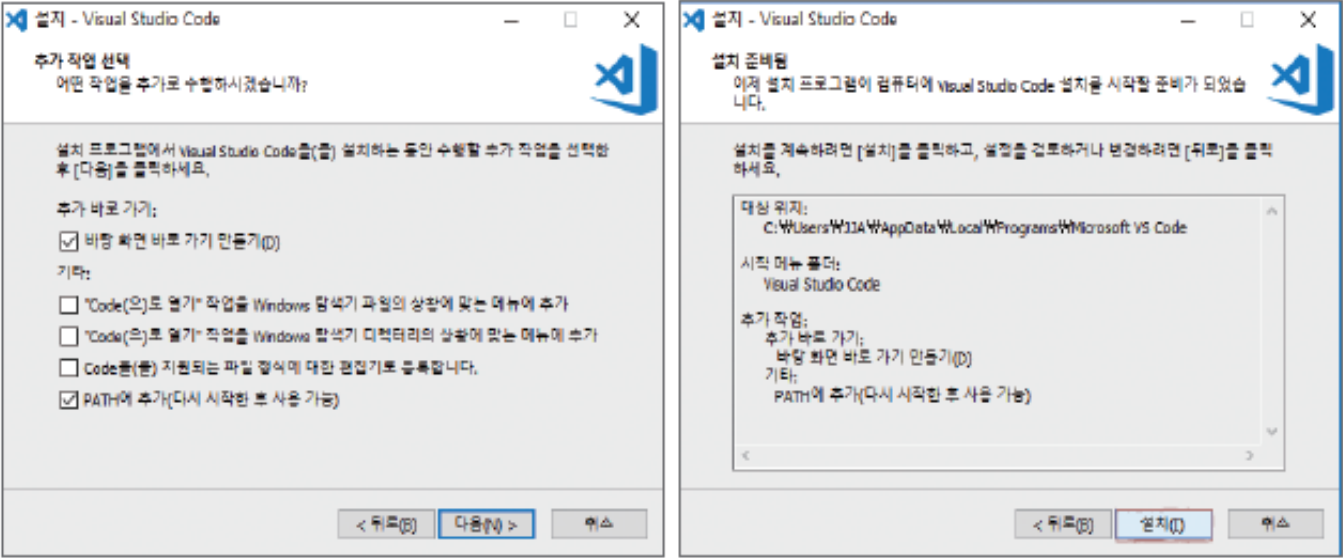


텍스트 에디터 사용하기 : 비주얼 스튜디오 코드

7) [바탕 화면 바로 가기 만들기] 체크 후 [다음] 클릭

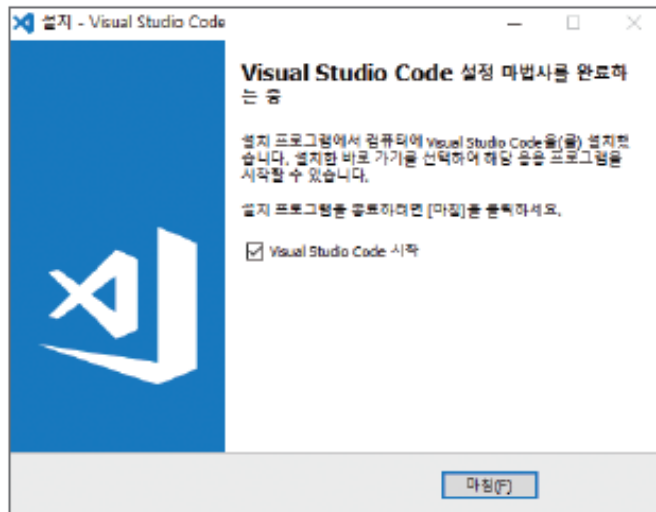
나머지 부분도 작업 시 유용하므로 모두 체크

8) 대상 위치, 시작 메뉴 폴더, 추가 설정 항목 확인 후 [설치] 클릭



텍스트 에디터 사용하기 : 비주얼 스튜디오 코드

9) [마침] 클릭. 설치 완료!



텍스트 에디터 사용하기 : 비주얼 스튜디오 코드

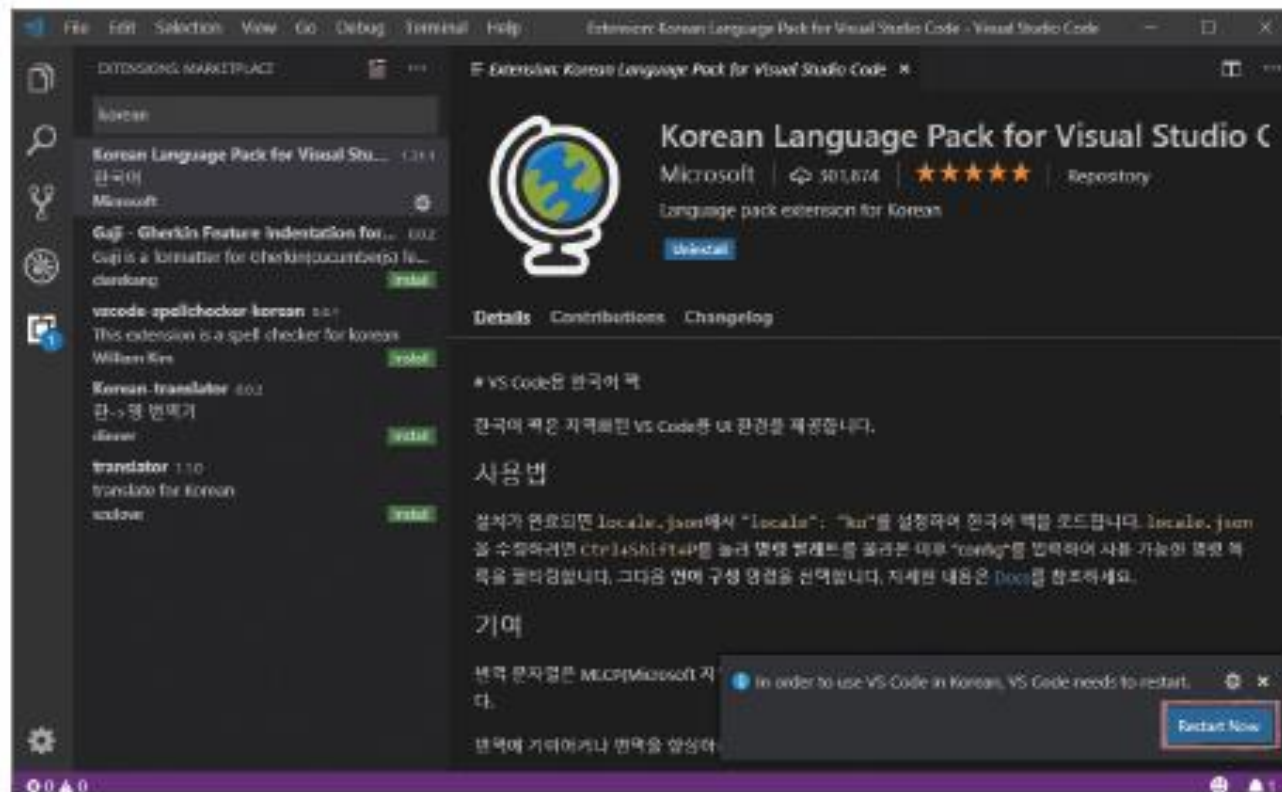
• 비주얼 스튜디오 코드 한글 언어 팩 설치

- 1) 도구 바에서 [확장] 클릭
- 2) 검색 창에 [korean] 입력
- 3) [Korean Language Pack for Visual Studio Code]의 [Install] 클릭



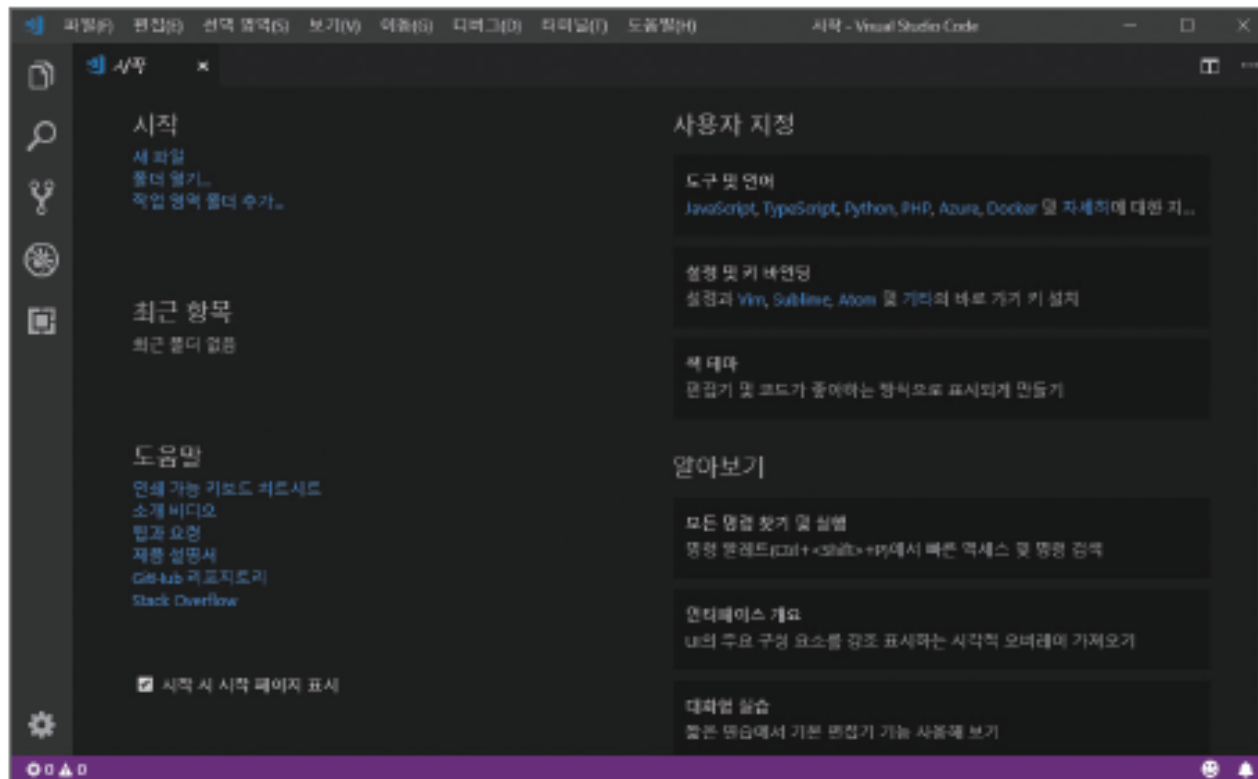
텍스트 에디터 사용하기 : 비주얼 스튜디오 코드

4) 설치 완료 후 [Restart Now] 버튼 클릭하여 재시작



텍스트 에디터 사용하기 : 비주얼 스튜디오 코드

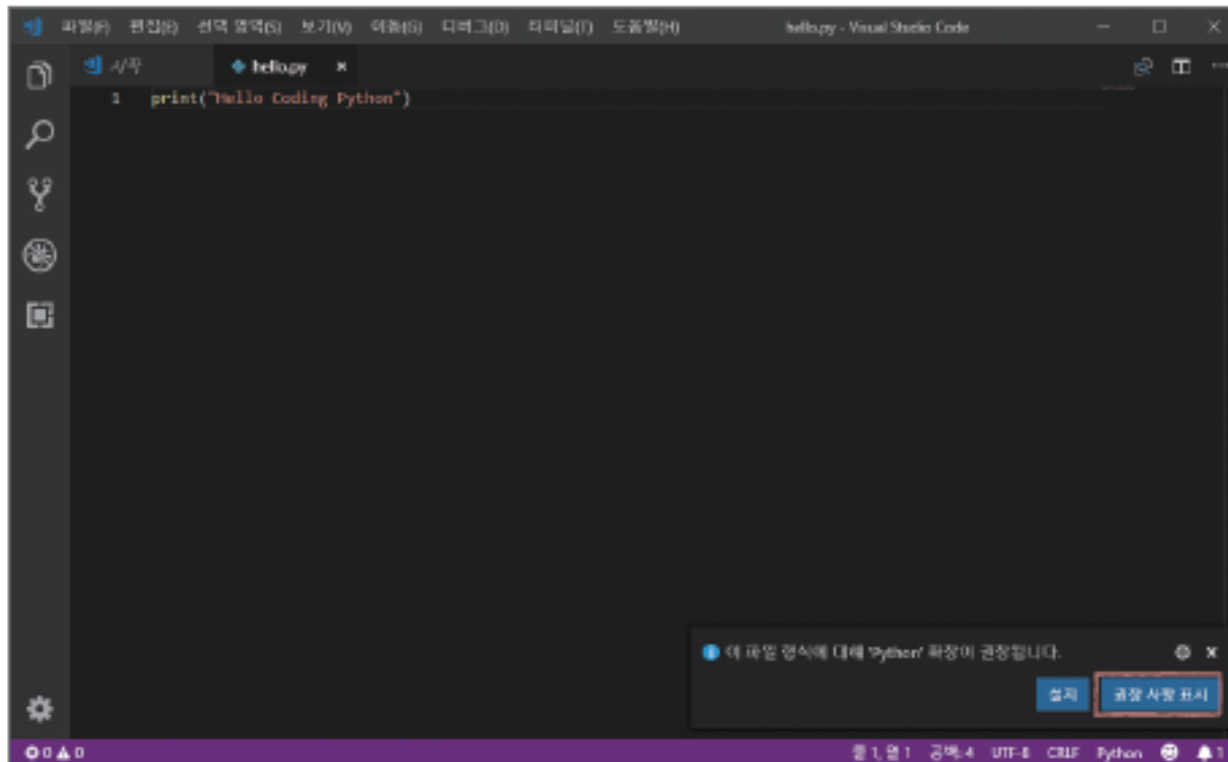
5) 메뉴가 한글로 바뀌었음을 확인



텍스트 에디터 사용하기 : 비주얼 스튜디오 코드

- 비주얼 스튜디오 코드에서 코드 작성하고 실행하기

1) 시작 화면에서 [파일] - [새 파일] 메뉴 선택 (단축키 [Ctrl] + [N])



텍스트 에디터 사용하기 : 비주얼 스튜디오 코드

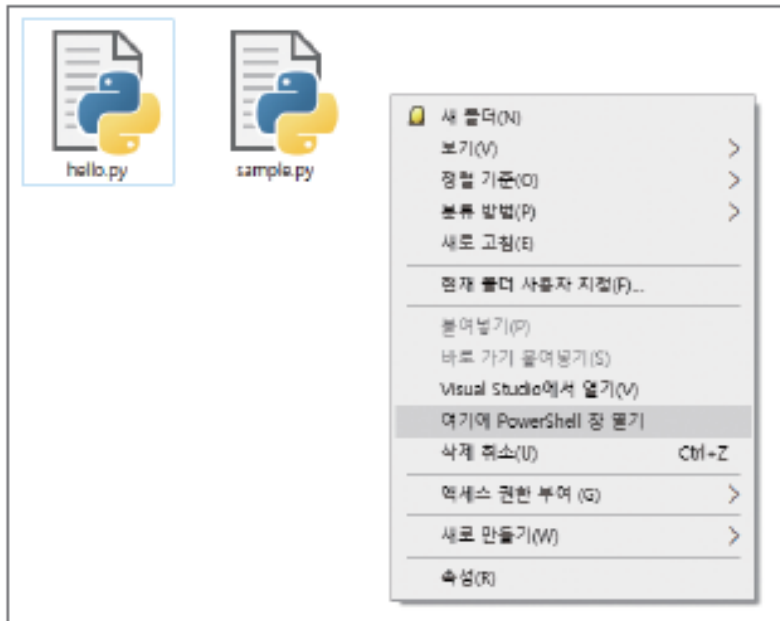
2) [확장] 메뉴에서 [Python] 클릭 - [설치] - [다시 로드]



텍스트 에디터 사용하기 : 비주얼 스튜디오 코드

3) 탐색기에서 코드 파일 저장한 폴더로 이동

4) [Shift] 누른 상태로 빈 곳을 마우스 오른쪽 클릭 – [여기에 명령 창 열기]



텍스트 에디터 사용하기 : 비주얼 스튜디오 코드

5) 해당 폴더에서 명령 프롬프트가 실행

6) [python hello.py] 입력 후 [Enter] 클릭하면

```
> python hello.py 
```

7) Hello Coding Python 출력



```
Windows PowerShell
PS C:\Users\bitcamp\Documents> python test.py
안녕
파이썬
출력 중...
다시 파이썬 출력 중...
PS C:\Users\bitcamp\Documents>
```

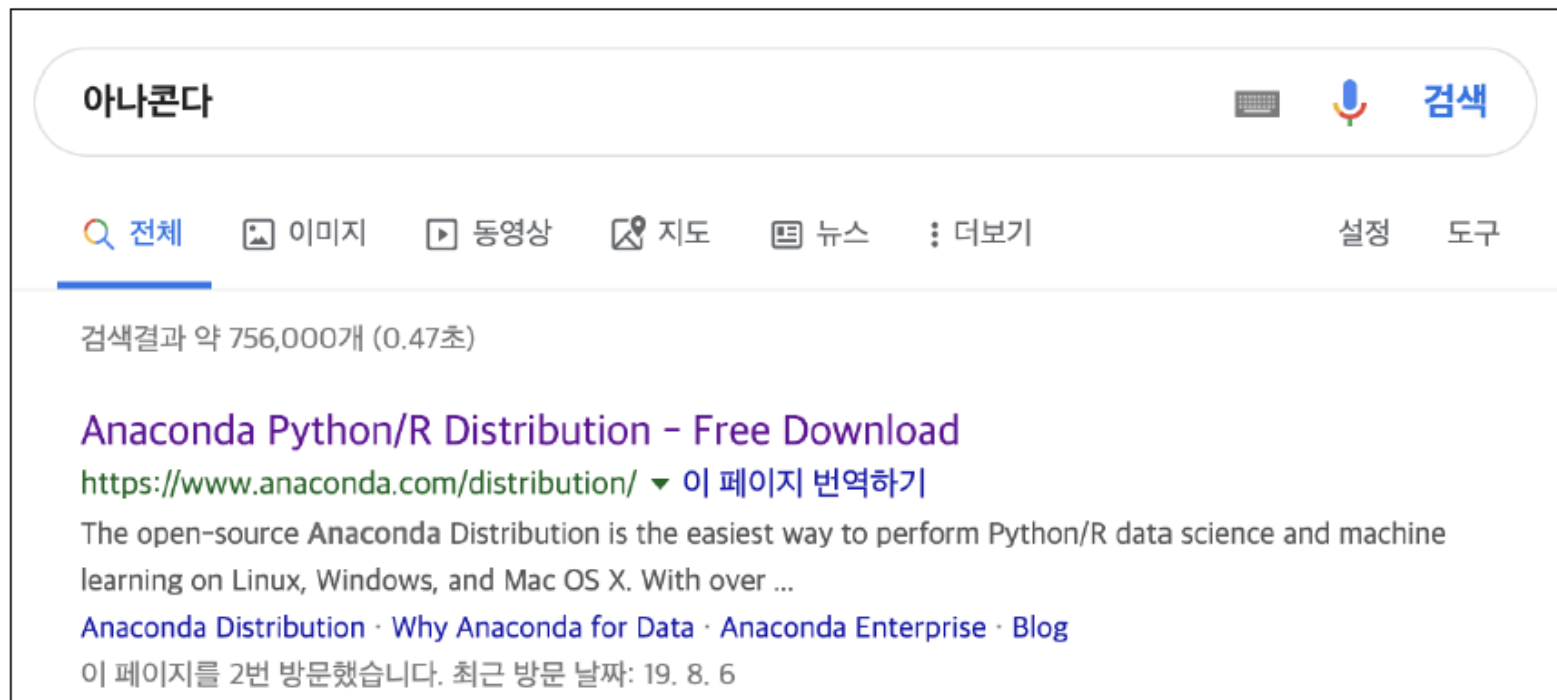


- 수학, 과학 분야에서 필요한 중요한 라이브러리들(NumPy, Pandas...)을 함께 설치
- 필요에 따라 환경을 여러 개 만들어 각자 다른 환경에서 프로그래밍할 수 있음
- 최근 4차 산업혁명 관련해 IT분야에서 파이썬을 많이 활용
 - ➡ 아나콘다(Anaconda)가 거의 정석처럼 사용되고 있는 추세

1

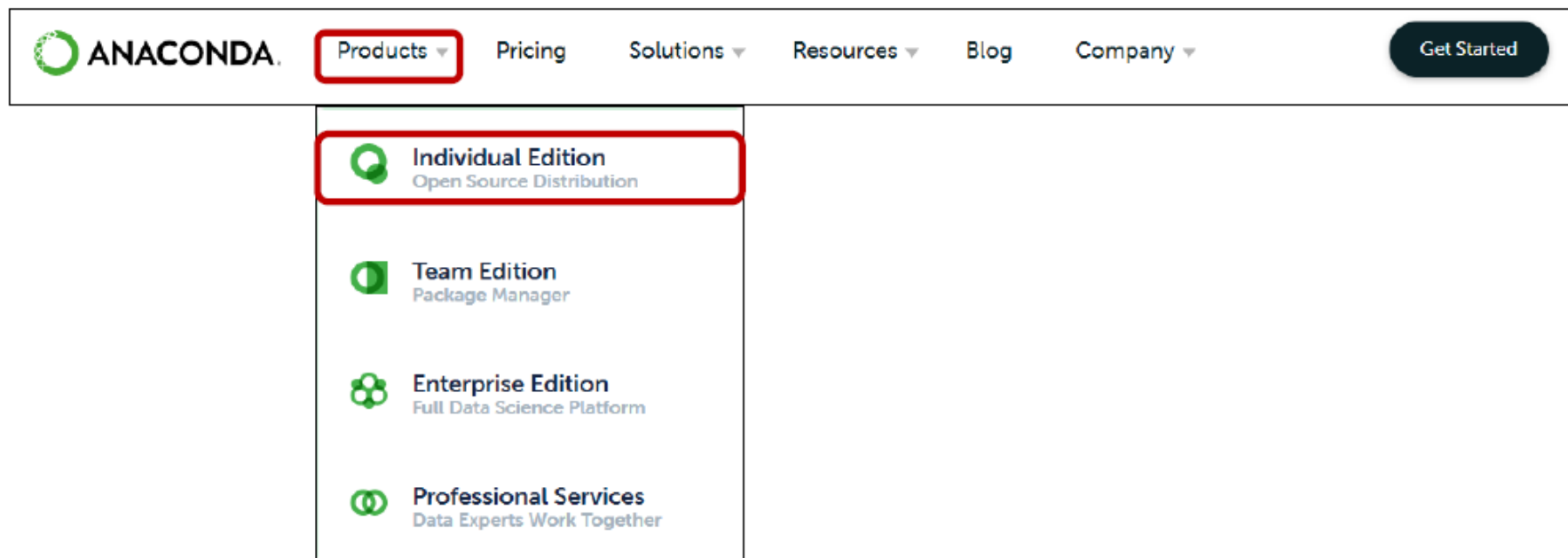
아나콘다 사이트 접속

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



2

우측 상단의 Products - Individual Edition
메뉴 클릭

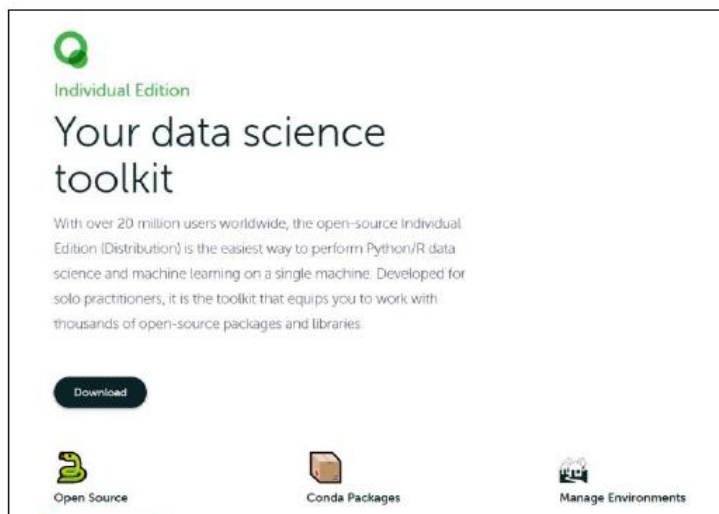


텍스트 에디터 사용하기 : 아나콘다 주피터노트북

3

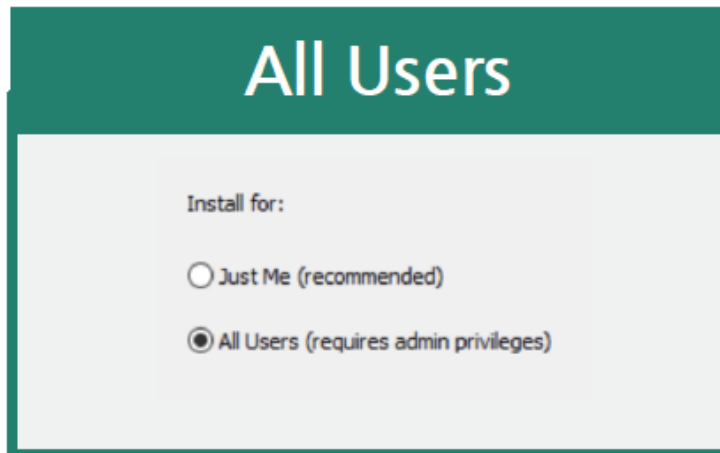
다운로드 메뉴 클릭

- 윈도우, 맥, 리눅스 버전 제공



Anaconda Installers		
Windows 	MacOS 	Linux 
Python 3.7 64-Bit Graphical Installer (466 MB) 32-Bit Graphical Installer (423 MB)	Python 3.7 64-Bit Graphical Installer (442 MB) 64-Bit Command Line Installer (430 MB)	Python 3.7 64-Bit (x86) Installer (522 MB) 64-Bit (Power8 and Power9) Installer (276 MB)
Python 2.7 64-Bit Graphical Installer (413 MB) 32-Bit Graphical Installer (356 MB)	Python 2.7 64-Bit Graphical Installer (537 MB) 64-Bit Command Line Installer (409 MB)	Python 2.7 64-Bit (x86) Installer (477 MB) 64-Bit (Power8 and Power9) Installer (295 MB)

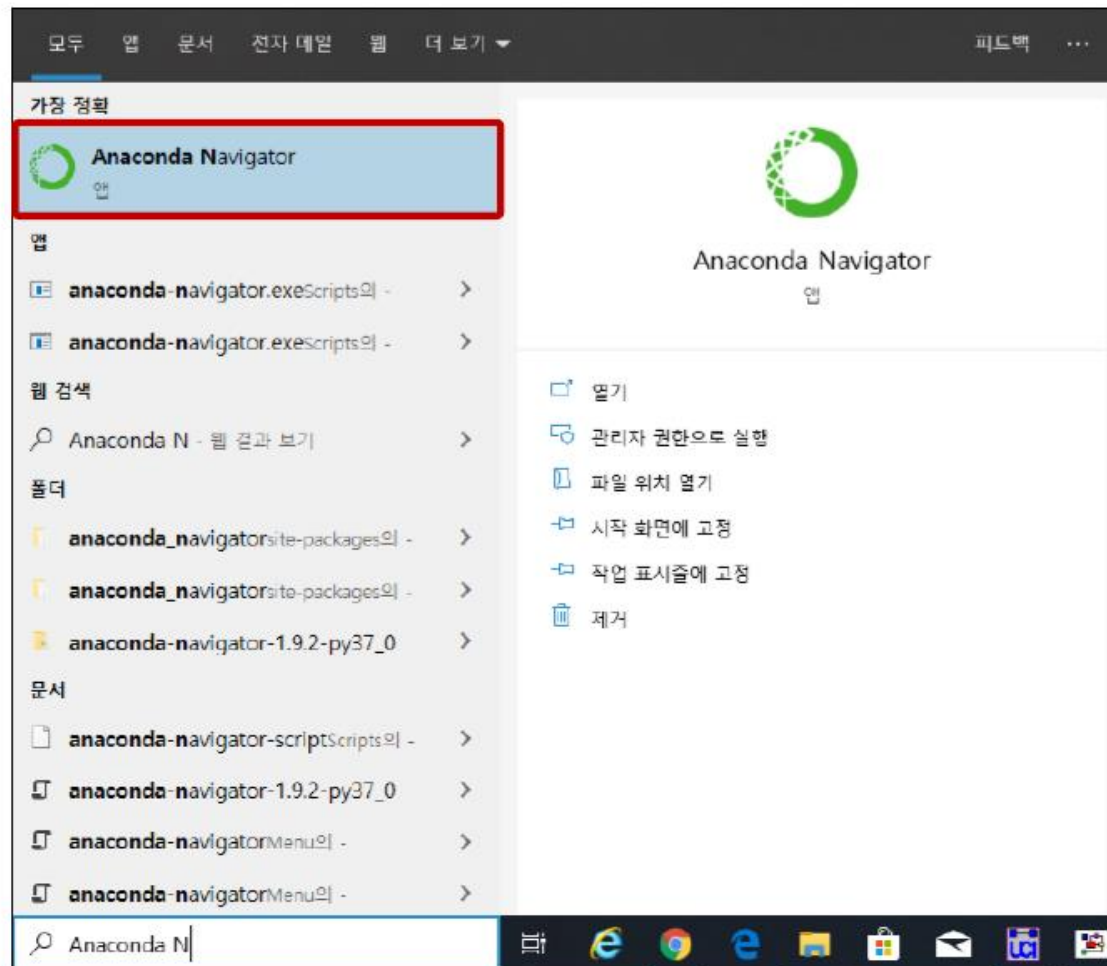
④ 추천 설치 옵션



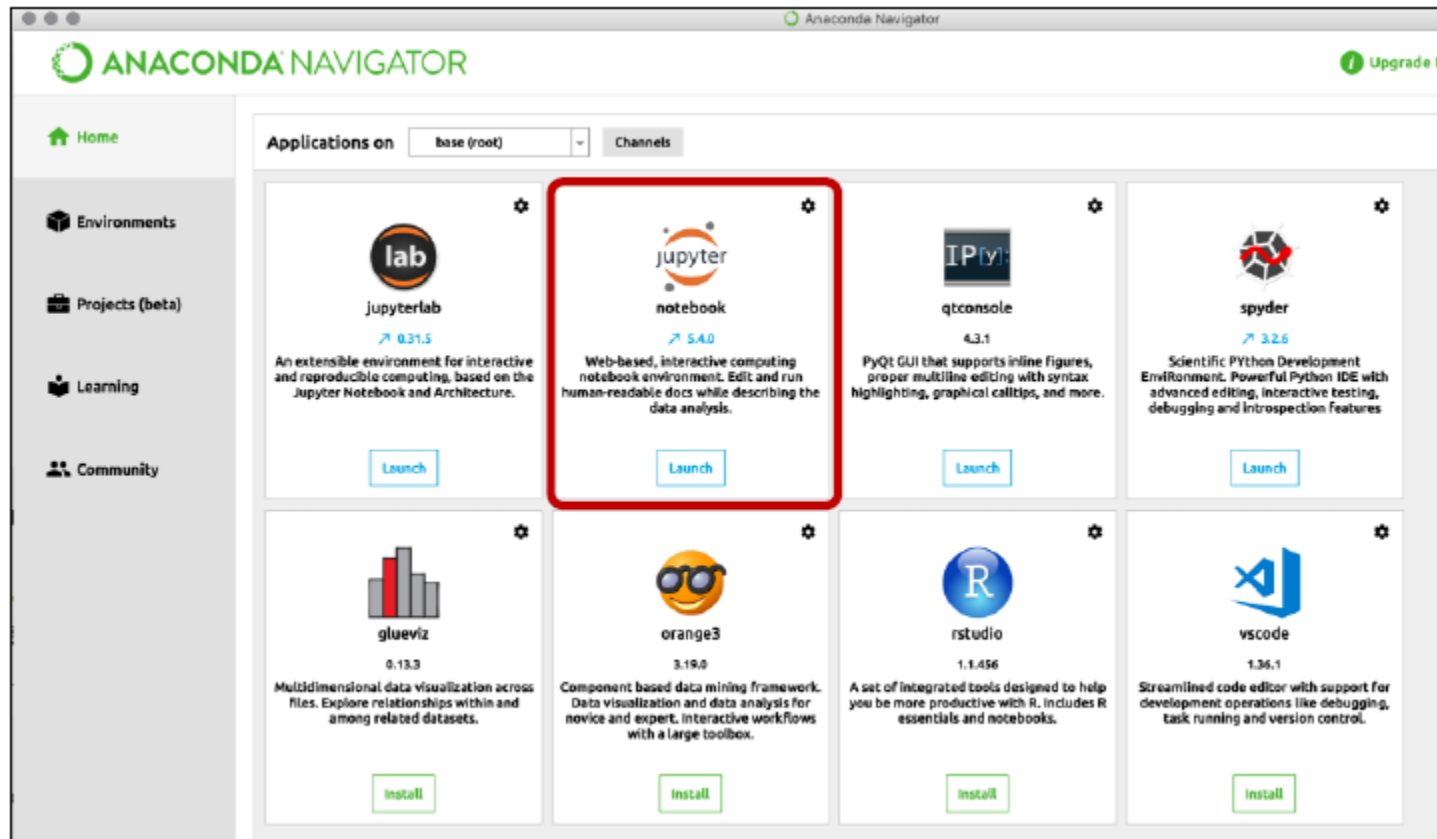
주의! 설치 경로에 한글이 있으면
제대로 설치되지 않을 수 있음

텍스트 에디터 사용하기 : 아나콘다 주피터노트북

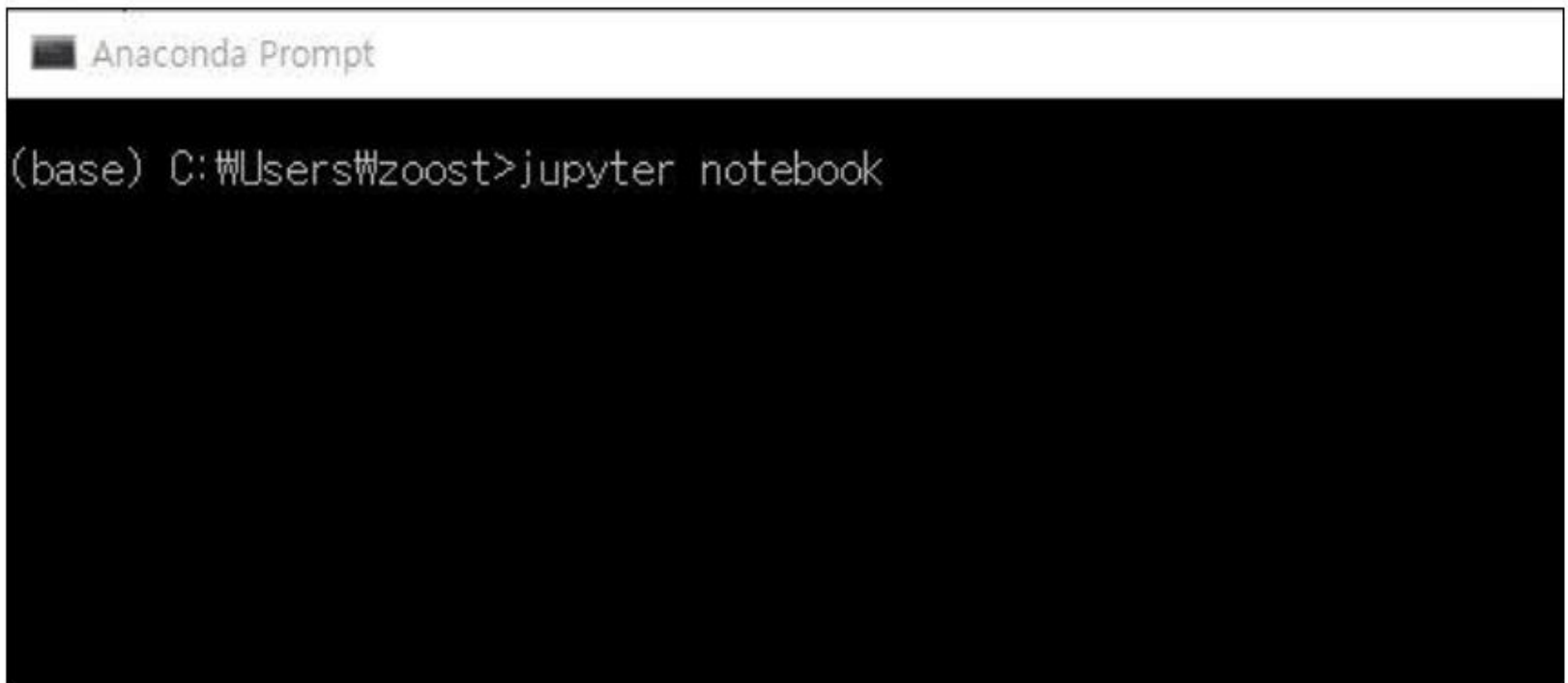
① 아나콘다 네비게이터(Anaconda Navigator)



텍스트 에디터 사용하기 : 아나콘다 주피터노트북

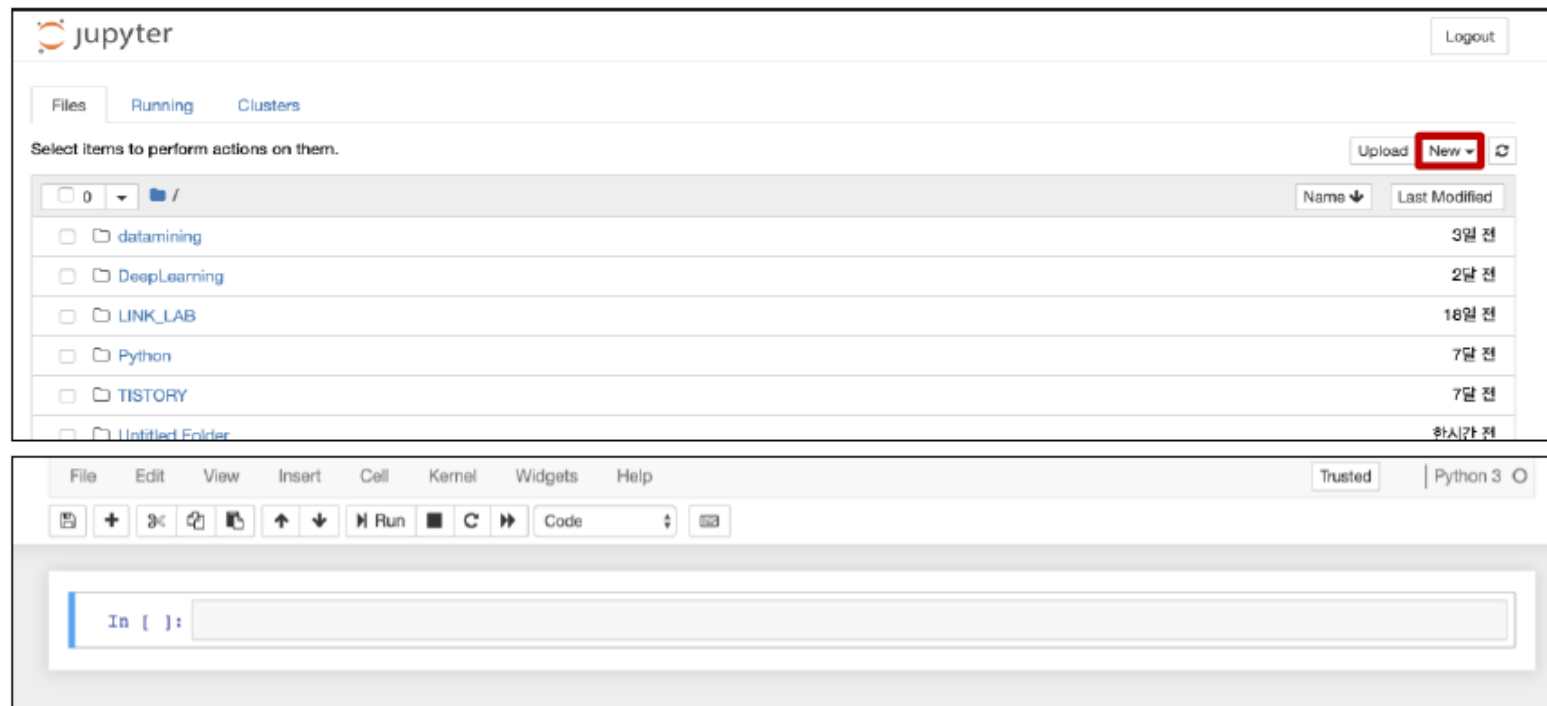


② 콘솔 창에서 실행



```
Anaconda Prompt
(base) C:\Users\zoost>jupyter notebook
```

③ 우측 상단 New → Python 3 클릭





파이썬 살펴보기

목차

- 표현식과 문장
- 키워드
- 식별자
- 주석
- 연산자와 자료형
- 출력 : `print()`

키워드

- 키워드 (keyword) : 예약어

- 특별한 의미가 부여된 단어
- 파이썬에서 이미 특정 의미로 사용하기로 예약해 놓은 것
- 프로그래밍 언어에서 이름 정할 때 똑같이 사용할 수 없음

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

- 대소문자 구별
- 예약어를 변수에 활용 → 에러는 없으나 고유 기능은 사라짐

키워드

- 아래 코드로 특정 단어가 파이썬 키워드인지 확인 가능

```
>>> import keyword  
>>> print(keyword.kwlist)
```

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

키워드

- 파이썬의 예약어 종류를 알아보는 방법은?
 - keyword 모듈을 불러옴(import)
 - keyword 모듈이 지원하는 kwlist를 출력(print)
 - 이 때 print도 예약어이므로 다른 식별자로 사용하면 안됨
 - len → 특별한 모듈 추가 없이 사용할 수 있는 내장 함수
 - keyword 모듈에 있는 kwlist에 몇 개의 단어가 있는지 알아봄
 - 파이썬의 버전에 따라 예약어 종류 달라짐
 - 앞서 알아본 len이 내장 함수(Built-in Function)
 - import keyword에서 keyword가 모듈
 - 모듈을 불러올 때는 import(내장 함수) 사용
 - 현재 프로그램에 사용된 예약어 → import, print
 - 파이썬을 설치하면 수많은 모듈이 설치됨
 - import keyword → 그 중 keyword 모듈을 사용하겠다
 - 모듈 안에 있는 함수는 모듈을 import 해야 사용 가능

내장함수

- 별도의 모듈(Module)의 추가 없이 기본적으로 제공되는 함수들
 - 참고사이트
 - 내장(Built-in) 함수: <https://docs.python.org/3.8/library/functions.html>
 - 대표적인 내장 함수
 - abs, max, min, pow, chr, str, range, type, ...
 - abs(x) : 수치형 자료 x에 대해 x의 절대값을 반환하는 함수
- **내장 함수는 모듈 추가 없이 활용 가능**

내장함수

- 별도의 모듈(Module)의 추가 없이 기본적으로 제공되는 함수들
 - `abs` → 수치형 자료를 절대값으로 반환하는 내장 함수
 - `max` → 주어진 자료 중 최소값을 반환하는 내장 함수
 - `pow(a, b)` → `a`의 `b`승 값을 반환
 - `str(a)` → `a`를 출력(`print`)하면 어떻게 나타나는지를 반환
 - `range([start,]stop[,step])`
 - 수치형 자료형으로 `start`, `stop`, `step` 등을 입력 받아 해당 범위에 해당하는 정수를 리스트로 반환하는 함수
 - `type(a)` → `a`의 자료형을 반환

내장함수

- max

시퀀스 자료형(문자열, 리스트, 튜플)을 입력 받아 그 자료가 지닌 원소 중 최대값을 반환하는 함수

```
print max(1,2)
print max([1, 2, 3])
print max("python")
```

```
2
3
y
```

- `print max(1, 2)` → 1과 2 중 더 큰 2를 반환
- `print max([1, 2, 3])` → 리스트의 원소 중 제일 큰 3을 반환
- `print max("python")` → 문자열 중 아스키 코드 값이 가장 큰 문자 반환
- `max` → 주어진 자료 중 최소값을 반환하는 내장 함수

내장함수

- min

시퀀스 자료형(문자열, 리스트, 튜플)을 입력 받아 그 자료가 지닌 원소 중 최소값을 반환하는 함수

```
print min(1,2)
print min([1, 2, 3])
print min("python")
```

```
1
1
h
```

- `print min(1, 2)` → 1과 2 중 더 작은 1을 반환
- `print min("python")` → 문자열 중 아스키 코드 값이 가장 작은 문자 반환

내장함수

- str(a)

- 정수 형태의 ASCII코드 값을 입력으로 받아 그에 해당하는 문자를 반환하는 함수
- 인수 i의 범위: 0부터 255까지

```
print chr(97)  
print chr(65)  
print chr(48)
```

```
a  
A  
0
```

- print chr(97) → 아스키 코드값이 97인 문자 → a
- print chr(65) → 아스키 코드값이 65인 문자 → A
- print chr(48) → 아스키 코드값이 48인 문자 → 0
- 이 때 0은 숫자 0이 아니라 문자로써의 0

내장함수

- `str(object)`

임의의 객체 `object`에 대해 해당 객체를 표현하는 문자열을 반환하는 함수

```
print str(3)  
print str([1, 2])
```

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

- `str(3)` → 3이라는 객체를 출력했을 때 나타나는 결과 → 3
- `str([1,2])` → [1, 2]를 출력했을 때 나타나는 결과 → [1, 2]
- `str` → 해당 객체를 표현하는 문자열로 반환해주는 함수

내장함수

- `range([start,]stop[,step])`

수치형 자료형으로 `start`, `stop`, `step` 등을 입력 받아 해당 범위에 해당하는 정수를 리스트로 반환하는 함수

- 인수가 하나(`stop`)인 경우

- 0부터 `stop-1`까지의 정수 리스트를 반환한다.

- 인수가 두 개(`start`, `stop`)인 경우

- `start`부터 `stop-1`까지의 정수 리스트를 반환한다

- 인수가 세 개(`start`, `stop`, `step`)인 경우

- `start`부터 `stop-1`까지의 정수를 반환하되 각 정수 사이의 거리가 `step`인 것들만 반환한다.

내장함수

```
print range(10)
print range(3, 10)
print range(3, 10, 2)
```

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

- `range(10)` → 0부터 10-1까지의 정수 리스트를 반환
- `range(3, 10)` → 3부터 10-1까지의 정수 리스트를 반환
- `range(3, 10, 2)` → 3부터 10-1까지 2개씩 건너 뛴 정수 리스트를 반환
- `range(3, 10, 2)` → [3, 5, 7, 9]
- `range` 함수는 각각 원소의 차이가 `step`값만큼 나도록 할 수 있음
- `range([start], stop, [step])` → `start`와 `step`은 생략 가능
- `range(10)` → `stop`만 10으로 기술한 것
- `range(10)` → 10-1까지의 정수 리스트를 반환
- `start`가 생략되었을 경우 0부터 시작
- `range(a, b, c)` → `a`부터 `b-1`까지 `c`씩 차이 나게 정수 리스트를 반환

내장함수

- `type(a)` → `a`의 자료형을 반환

```
print type(-1)
print type('abc')
print type([1, 2, 3])
```

```
<type 'int'>
<type 'str'>
<type 'list'>
```

- `print type(-1)` → `-1`의 자료형을 반환 → `int`(정수형)
- `print type('abc')` → `'abc'`의 자료형을 반환 → `str`(문자형)
- `print type([1, 2, 3])` → `[1, 2, 3]`의 자료형을 반환 → `list`(리스트)
- `type` 함수를 사용하여 객체의 자료형을 알 수 있음

식별자

- 식별자 (identifier)

- 프로그래밍 언어에서 이름 붙일 때 사용하는 단어
- 변수 또는 함수 이름 등으로 사용
- 키워드 사용 불가
- 특수문자는 언더바(_)만 허용

- 숫자로 시작 불가

- 공백 포함 불가

- 알파벳 사용이 관례

- 의미 있는 단어로 할 것

- 대소문자 구별함

- 예약어, 내장함수, 모듈 이름을 변수명으로 만드는 일이 없도록 할 것

사용 가능한 단어	사용 불가능한 단어
alpha	break → 키워드라서 안 됩니다.
alpha10	
_alpha	273alpha → 숫자로 시작해서 안 됩니다.
AlpHa	
ALPHA	has space → 공백을 포함해서 안 됩니다.

식별자

- 스네이크 케이스와 캐멀 케이스

```
itemlist      loginstatus      characterhp      rotateangle
```

- 공백이 없어 이해하기 어려움
 - 스네이크 케이스 (snake case) : 언더바(_)를 기호 중간에 붙이기
 - 캐멀 케이스 (camel case) : 단어들의 첫 글자를 대문자로 만들기

식별자에 공백이 없는 경우	단어 사이에 _ 기호를 붙인 경우 (스네이크 케이스)	단어 첫 글자를 대문자로 만든 경우 (캐멀 케이스)
itemlist loginstatus characterhp rotateangle	item_list login_status character_hp rotate_angle	ItemList LoginStatus CharacterHp RotateAngle

- 파이썬에서는 스네이크 및 캐멀 케이스 둘 모두 사용

식별자

- 식별자 구분하기

- 캐멀 케이스에서는 첫 번째 글자를 소문자로 적지 않음

캐멀 케이스 유형 1 : `PrintHello` → 파이썬에서 사용합니다.

캐멀 케이스 유형 2 : `printHello` → 파이썬에서 사용하지 않습니다.

`print`

`input`

`list`

`str`

`map`

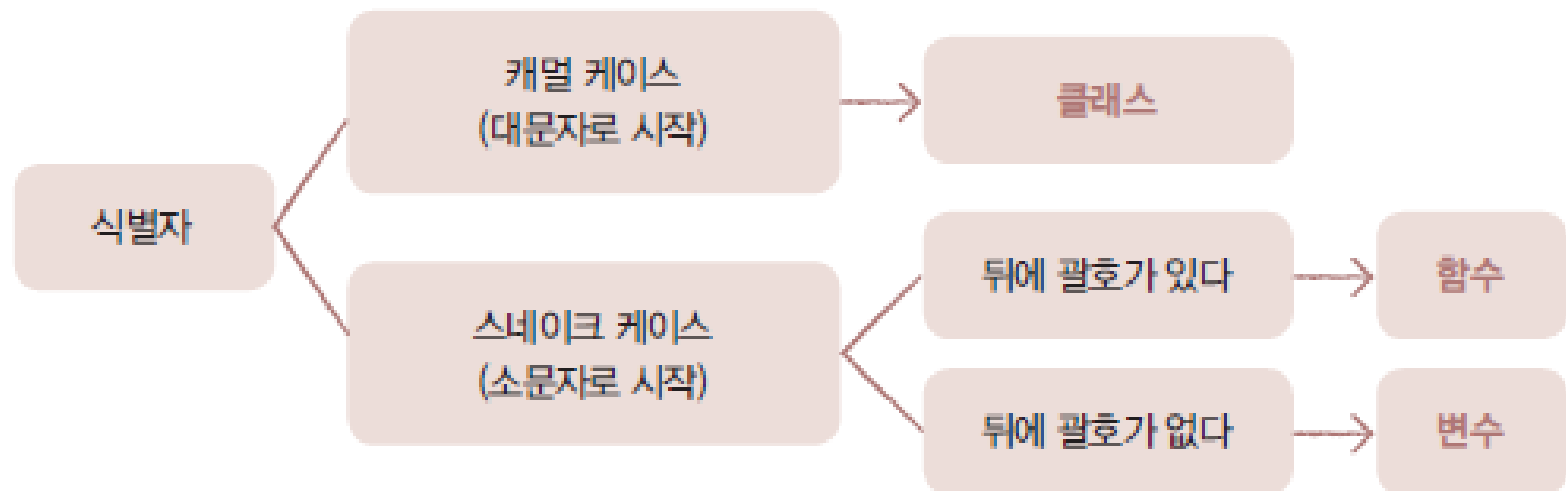
`filter`

`Animal`

`Customer`

식별자

- 캐멀 케이스로 작성되었으면 클래스
- 스네이크 케이스로 작성되어 있으면 함수 또는 변수
- 뒤에 괄호 붙으면 함수
- 뒤에 괄호 없으면 변수



식별자

- 변수명 만들 때 조심할 점
 - 변수명 만들 때 예약어, 내장 함수, 모듈 이름으로 만들지 않도록 주의
 - 예약어, 내장 함수, 모듈 이름은 이미 활용되고 있기 때문

```
str = 'abc'  
print str(12345)
```

```
-----  
-----  
TypeError Traceback (most recent call last)  
<ipython-input-2-ad06f2f248af> in <module>()  
1 str = 'abc'  
2  
---- 3 print str(12345)  
TypeError: 'str' object is not callable
```

- 만약 str을 다른 것으로 정의하게 되면?
 - 본래 str 함수가 가지고 있던 기능을 잃어버림

변수의 생성 및 사용

- 파이썬에서 변수가 생성되는 시점은 해당 변수에 임의의 값이 할당될 때이다.

```
a = 1  
print a
```

1

- 파이썬과 다른 언어의 큰 차이점 → 변수 생성 시 타입(type)을 적지 않음
- 변수에 값이 할당될 때 변수의 타입이 정해짐
- type 함수를 통해 a의 타입을 확인해보면 int(정수)로 나타남
- 이 때 a의 타입은 값이 할당이 되었을 때 int로 정해진 것
- 즉, 변수의 타입을 처음부터 정할 수 없음
- 이를 파이썬의 동적 특성, 동적 변수 할당이라 함

변수의 생성 및 사용

- 변수의 삭제 : del이라는 예약어 사용

```
b = 2
print b
del b
print b
```

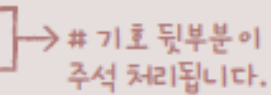
```
2
-----
----
-----
NameError Traceback(most recent call last)
<ipython-input-24-026dc2933af4> in <module>()
3
4 del b
---- 5 print b
NameError : name 'b' is not defined
```

- del → 변수를 삭제 할 때 사용
- b에 2를 할당한 후 print b → 2가 정상적으로 출력됨
- del b → b라는 변수를 삭제
- 그 후 print b → 에러 발생

주석

- 주석 (comment)
 - 프로그램 진행에 영향 주지 않는 코드
 - 프로그램 설명 위해 사용
 - # 기호를 주석으로 처리하고자 하는 부분 앞에 붙임

```
>>> # 간단히 출력하는 예입니다.  
>>> print("Hello! Python Programming...") # 문자열을 출력합니다.  
Hello! Python Programming...
```



기호 뒷부분이
주석 처리됩니다.

연산자와 자료

- 연산자

- 스스로 값이 되는 것이 아닌 값과 값 사이에 무언가 기능 적용할 때 사용

```
>>> 1 + 1
2
>>> 10 - 10
0
```

- 리터럴 (literal)

- 자료 = 어떠한 값 자체

```
1
10
"Hello"
```

출력 : print()

- print() 함수

- 출력 기능
- 출력하고 싶은 것들을 괄호 안에 나열

```
print(출력1, 출력2, ...)
```

- 하나만 출력하기

```
>>> print("Hello! Python Programming...")
Hello! Python Programming...
>>> print(52)
52
>>> print(273)
273
```

Mission 1 | type() 함수를 활용하여 데이터 타입 확인

```
a = 1
```

```
b = '1'
```

```
c = 1.0
```

```
d = [1]
```

Mission 2

Hello, World 출력

```
a = 'He'
```

```
b = 'llo'
```

```
c = ', World'
```

1. 컴파일 언어와 스크립트 언어의 이해



컴파일 언어	<ul style="list-style-type: none">• C언어, 자바(JAVA)와 같이 코드를 한번에 기계어로 번역하여 실행하는 언어• 메모리가 많이 필요하지만 실행 시간이 빠른 장점이 있음
스크립트 언어	<ul style="list-style-type: none">• 소스코드를 한 줄 한 줄 읽어 바로 실행하는 인터프리터 언어• 플랫폼에 독립적이고 메모리가 적게 필요한 장점이 있음

2. 파이썬 언어의 이해와 특징



파이썬 언어의 이해	<ul style="list-style-type: none">• 인공지능, IoT, 빅데이터 등 최신 IT 분야에 많이 활용되는 언어• 대표적인 스크립트 언어 중 하나• 컴파일 과정이 따로 없는 인터프리터 언어
파이썬 언어의 특징	<ul style="list-style-type: none">• 대화형으로 한 줄씩 프로그래밍이 가능하기 때문에, 메모리를 적게 차지하며 코드 작성이 간결하고 쉬운 장점이 있음• 동적인 데이터 타입을 지원• 리스트와 같은 고수준의 자료형을 제공함• 플랫폼 독립적 언어로 어느 운영체제에서도 동일하게 수행됨• 확장성이 높고 유지보수가 편리함

3. 파이썬 코딩 환경 구축 및 실행하기



파이썬 코딩 환경 구축하기	<ul style="list-style-type: none">• 홈페이지에서 직접 설치• 아나콘다(Anaconda)라는 플랫폼을 설치해 필요한 라이브러리들을 함께 설치할 수 있음
파이썬 실행하기	<ul style="list-style-type: none">• 단순히 텍스트 에디터에서 코드를 작성해 실행할 수 있음• Jupyter Notebook을 활용• 실시간으로 코드 결과를 확인해 보며 프로그래밍을 할 수 있음