

Python 소개

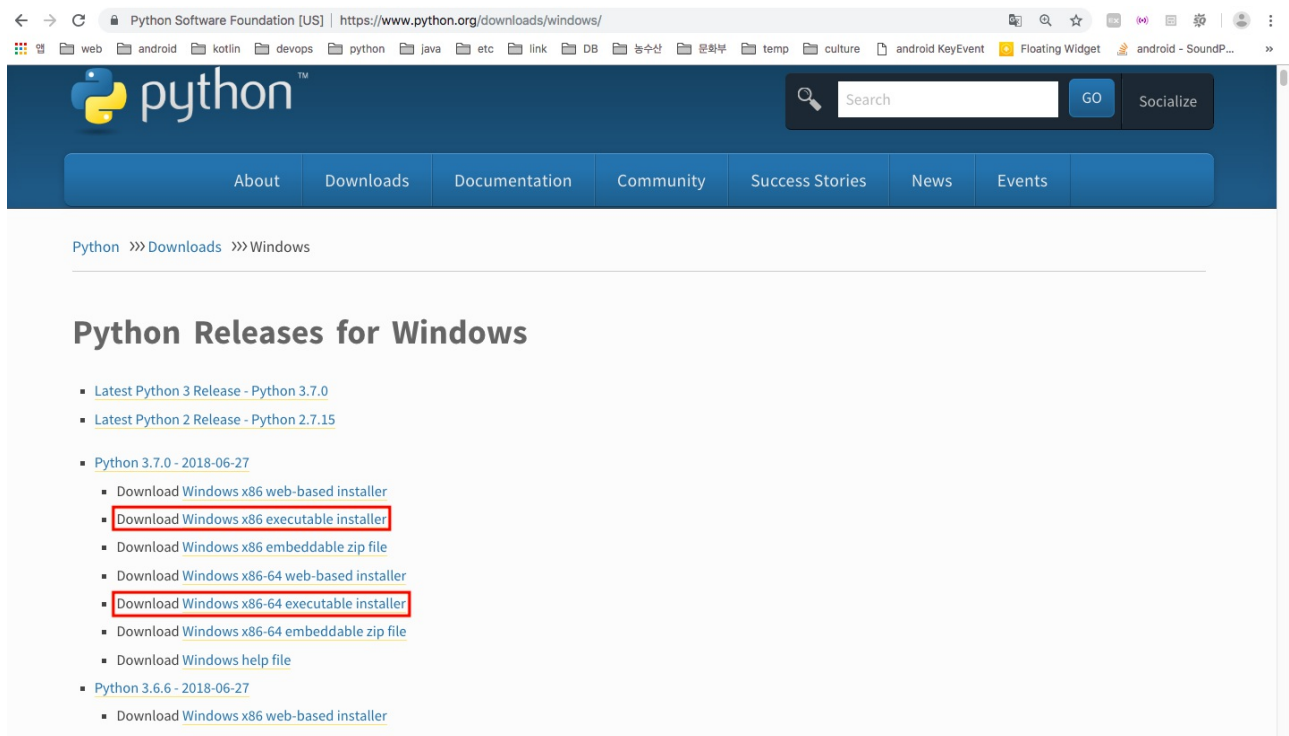
키워드 : [Python \(https://docs.python.org/ko/3/reference/index.html\)](https://docs.python.org/ko/3/reference/index.html),
[pip \(https://ko.wikipedia.org/wiki/Pip \(패키지 관리자\)\)](https://ko.wikipedia.org/wiki/Pip_(패키지_관리자)),
[Python 가상환경 \(https://docs.python.org/ko/3/tutorial/venv.html\)](https://docs.python.org/ko/3/tutorial/venv.html)

여기까지 잘 오셨다면 드디어 우리의 마지막 챕터인 파이썬 공부를 진행할 준비가 되었습니다! 새로운 챕터를 시작했으니, 새로운 마음가짐으로 공부를 계속 진행해주시면 좋겠습니다. :) 내용이 조금 긴 점 양해 부탁드립니다!

Python 설치

Windows 운영체제

우선, 파이썬 코딩을 진행하기 위해서 파이썬을 다운로드해야겠죠? [여기 \(https://www.python.org/downloads/\)](https://www.python.org/downloads/)에 접속해서 파이썬을 OS에 맞게 다운로드해줍니다. 우리는 가장 최신 버전을 사용할 것이기 때문에 **3.7.0** 버전을 다운로드해줍니다.

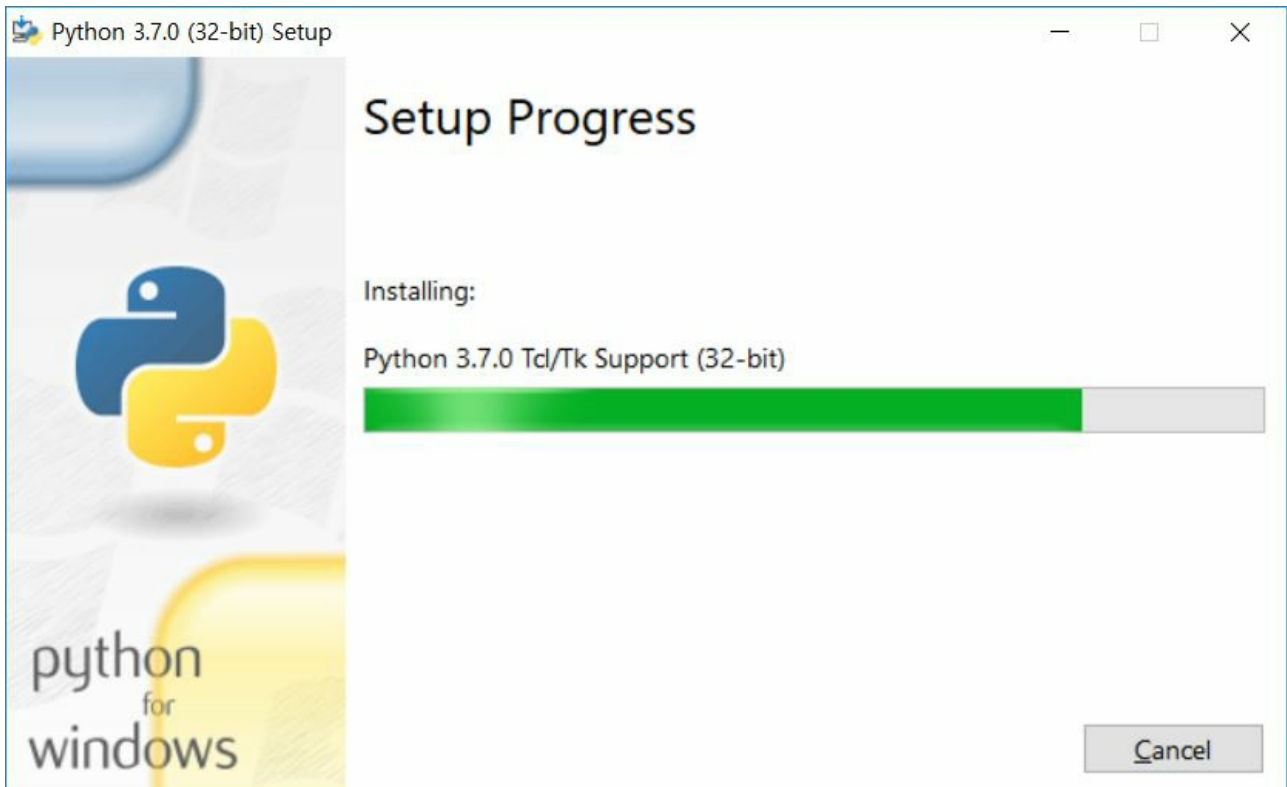


파이썬 설치시에 파이썬 설치 버전은 윈도우 32 비트를 사용하시는 경우에는 x86 executable installer를, 윈도우 64 비트 머신을 사용하시는 경우에는 x86-64 executable installer를 설치합니다.

파이썬을 다운로드하면, 파이썬을 비롯한 pip, IDLE 등 여러 가지가 함께 다운로드됩니다. 우리는 앞으로 pip(Python Package Index)를 이용해 각종 패키지를 관리하게 된다는 점, 참고로 기억해주세요!



윈도우 OS를 사용하시는 분은 설치 마법사가 나타나면, Add Python 3.7 to Path를 체크하고, Install Now를 클릭해줍니다.





설치가 진행되고, 잠시 기다리면 설치가 완료됩니다.

```
선택 관리자: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\WINDOWS\system32> python --version
Python 3.7.0
PS C:\WINDOWS\system32> python
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:06:47) [MSC v.1914 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> exit()
PS C:\WINDOWS\system32>
```

파이썬이 정상적으로 설치되었는지 확인하기 위해서 명령 프롬프트를 실행하고, 아래 명령어를 입력합니다. python 3.7.0 버전이 나타나며 파이썬이 실행되면 정상 설치된 것입니다. exit()을 입력하여 빠져나옵니다.

python

Mac 운영체제

맥 OS를 사용하시는 분은 파이썬 설치 전에 선행해주셔야 할 것이 있습니다. (Xcode가 설치되어 있으시다면, 이 부분은 건너뛰고 바로 파이썬을 설치해주시면 됩니다.)

```
moonseongjae — moonseongjae@munseongjaeui-MacBook-Air — ~ — -zsh...
Last login: Thu Aug 30 18:30:07 on ttys002
→ ~ git:(master) * xcode-select --install
```

터미널을 열고 아래 명령어를 실행합니다.

```
xcode-select --install
```



위와 같이 설치 창이 나오면 설치를 눌러 설치를 진행합니다. 설치가 완료되면, 라이선스 동의를 클릭하고 완료합니다. (설치 시간이 일정 시간 이상 소요될 수 있습니다.)

```
moonseongjae — moonseongjae@munseongjaeui-MacBook-Air — ~ — -zsh — 8...
Last login: Thu Aug 30 20:56:16 on ttys002
[→ ~ git:(master) * gcc --version]
Configured with: --prefix=/Library/Developer/CommandLineTools/usr --with-gxx-inc
include-dir=/usr/include/c++/4.2.1
Apple LLVM version 9.1.0 (clang-902.0.39.2)
Target: x86_64-apple-darwin17.2.0
Thread model: posix
InstalledDir: /Library/Developer/CommandLineTools/usr/bin
→ ~ git:(master) *
```

터미널에서 아래 명령어를 입력해줍니다. Installed Dir이라는 설치 경로가 정상적으로 나오면 잘 설치가 된 것입니다. 아래 내용을 보시고 파이썬 설치를 계속 진행해주세요.

```
gcc --v
```

맥 OS는 기본적으로 파이썬 2.x 버전이 설치되어 있습니다. 우리는 파이썬 3.7.0 버전을 사용할 것이기 때문에 [여기](https://www.python.org/downloads/) (<https://www.python.org/downloads/>)에 가서 파이썬 3.7.0을 다운로드해주세요. 맥 운영체제의 경우 pkg 파일로 다운로드가 됩니다.

간혹, 스토어가 아닌 곳에서 pkg 파일을 다운로드하고 설치 시 오류가 발생할 수가 있는데요, 이를 해결하기 위해서 시스템 환경 설정 -> 보안 및 개인 정보 -> 일반 -> 다운로드한 응용 프로그램 허용을 설정합니다. 만약, 다운로드한 응용 프로그램 허용부분이 Mac App Store로 되어 있을 경우에 Mac App Store 및 확인된 개발자로 변경해주시면 됩니다.

```
moonseongjae — moonseongjae@munseongjaeui-MacBook-Air — ~ — -zsh — 8...
Last login: Thu Aug 30 17:27:55 on ttys002
➔ ~ git:(master) > python3
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 26 2018, 23:26:24)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
[>>> exit()
➔ ~ git:(master) * █
```

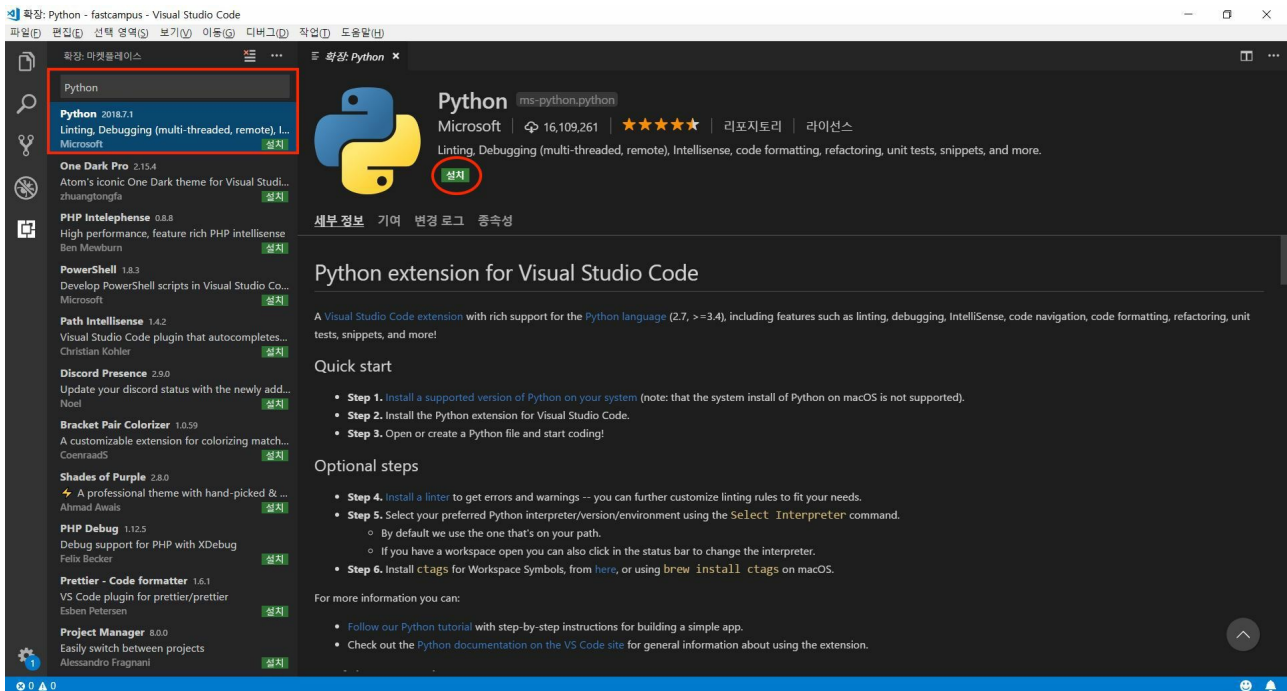
파이썬이 정상적으로 설치되었는지 확인하기 위해서, 터미널을 열고 아래 명령어를 입력합니다. python 3.7.0 버전이 나타나며 파이썬이 실행되면 정상 설치된 것입니다. exit()을 입력하여 빠져나옵니다.

```
python3
```

파이썬 설치뿐만 아니라, 우리가 앞으로 파이썬 개발을 위해서 VS code에서 선행되어야 하는 작업이 있습니다. 아래 내용을 계속해서 진행해주세요!

VS Code 확장(공통)

위 단계까지 정상적으로 이루어지고 나면, VS Code로 파이썬 개발환경을 구성하기 위해서 Python 확장을 설치해야 합니다.



"마켓플레이스에서 확장 검색"란에 Python을 입력하고, 설치를 진행합니다. 설치가 완료되면, "다시 로드"를 클릭해줍니다.

Python 가상 환경

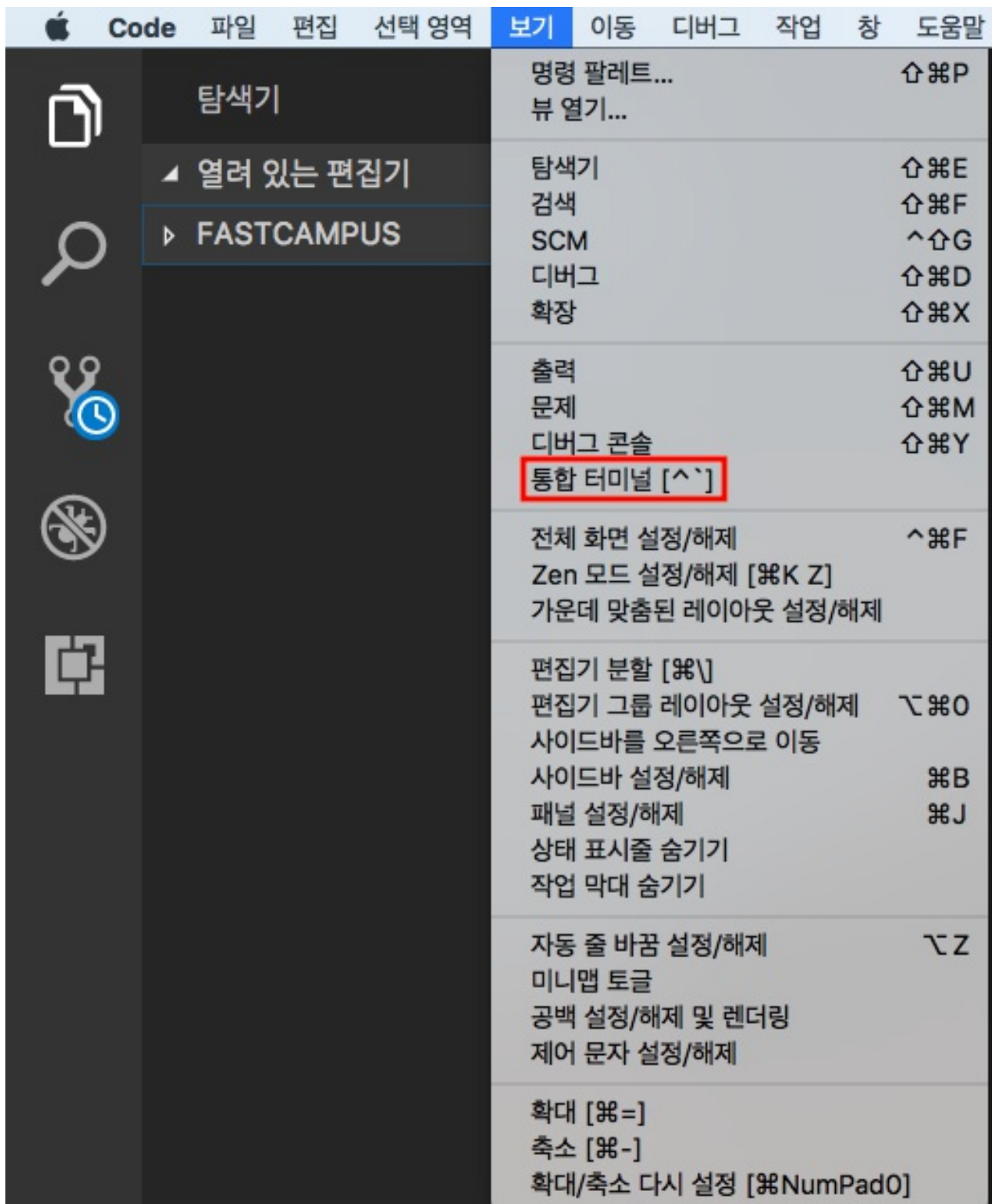
파이썬이 정상적으로 설치되었다면, 이제 파이썬 가상환경을 구성해야합니다. 우리가 파이썬 언어를 이용해서 개발을 진행할 때, 여러 가지 기존의 개발되어 있는 라이브러리를 활용해서 개발을 진행하게 됩니다.

대부분의 라이브러리는 개발이 계속 진행되기 때문에, 버전이 업그레이드 되게 됩니다. 사용하게 되는 라이브러리의 종류와 버전은 어떤 프로그램을 개발하느냐에 따라 달라질 수 있습니다. 예를 들어, 서로 다른 프로젝트에 A라는 공통적으로 사용하는 라이브러리가 있다고 가정하겠습니다.

개발을 진행하던 도중, 필요에 의해 "A" 라이브러리를 업그레이드했습니다. 업그레이드를 진행했을 때, 갑자기 하나의 프로젝트에서 버전 충돌이 발생해서 프로그램이 먹통이 되는 현상을 마주하게 되었다고 가정해보겠습니다.

이러한 경우, 하나의 프로젝트는 "A" 라이브러리를 사용할 수 없게 됩니다. 그렇기 때문에, 가상 환경을 구성하여 서로 독립적인 개발 환경을 구성하고, 다른 버전의 라이브러리를 사용하는 것입니다.

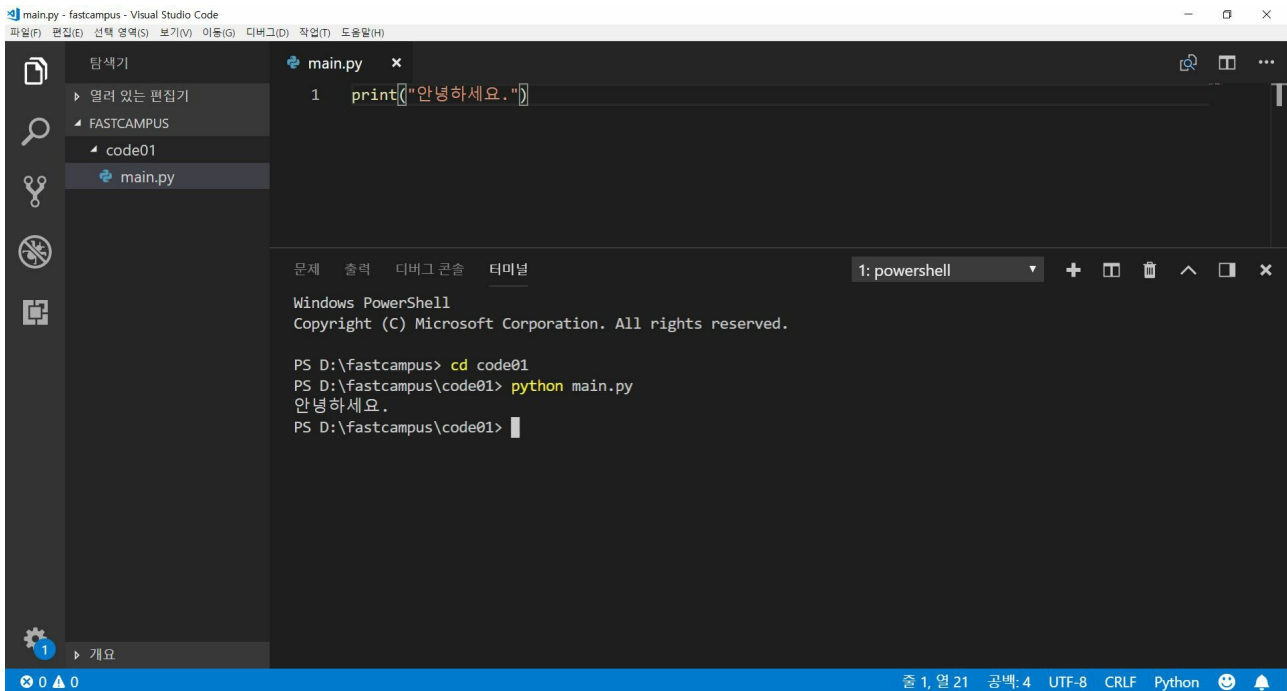
가상 환경을 구성하기 전에, 기존의 웹 개발 기초 과정에서 사용했던 폴더가 아닌, 새로운 파이썬 개발을 위한 프로젝트 폴더를 원하는 위치(경로)에 만들어주세요!



VS Code를 실행하고, 새로 생성한 프로젝트 폴더를 엽니다. 우리는 VS Code의 통합 터미널을 이용해 많은 작업을 진행하게 될 것입니다. 터미널이 보이지 않을 경우 위 사진처럼, 상단 메뉴에서 보기 -> 통합 터미널을 실행합니다.

- 터미널 실행 단축키

윈도우: `ctr + ~`, 맥: `cmd + ~`



우선 파이썬 코드를 실행할 준비가 되었는지 확인하기 위해서, 테스트 코드를 작성해보도록 하겠습니다. 테스트 폴더로 code01 폴더를 만들고, main.py 파일을 만들어 줍니다. main.py 파일 안에 아래 코드를 입력합니다. (파이썬 파일은 기본 확장자가 "*.py"입니다.)

```
print("안녕하세요")
```

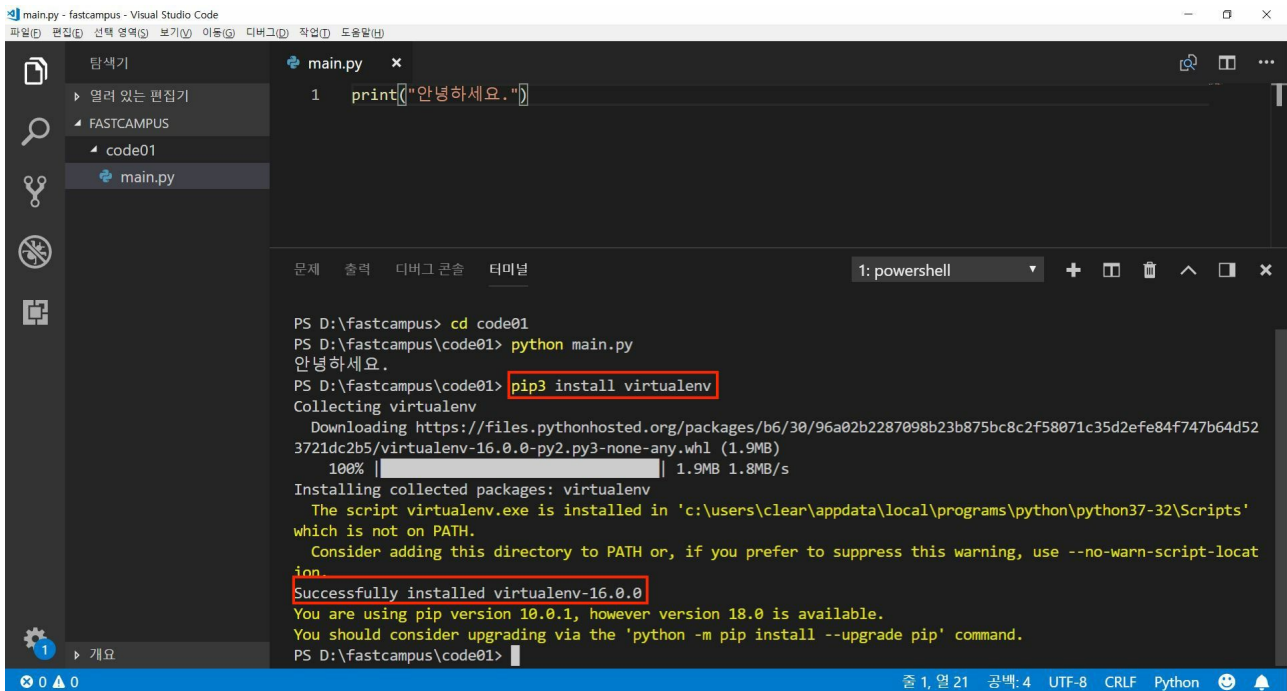
이제 터미널에 마우스 커서를 가져가 명령어를 입력할 수 있게되면, cd code01 명령어를 터미널에 입력합니다. 윈도우의 경우엔 python main.py를, 맥의 경우엔 python3 main.py를 입력하면 파이썬 코드를 실행할 수 있습니다. 터미널에 "안녕하세요"라고 잘 나오면, 성공입니다!

가상환경 설치 및 실행

우선, 파이썬 패키지 관리자인 pip가 최신 버전인지 확인하기 위해 아래 명령어를 터미널에 입력합니다.

```
pip3 install --upgrade pip
```

이제 본격적으로 프로젝트 가상 환경을 구성해보도록 하겠습니다.



```
main.py - fastcampus - Visual Studio Code
파일(F) 편집(E) 선택 영역(S) 보기(V) 이동(G) 디버그(D) 작업(T) 도움말(H)

탐색기
  열려 있는 편집기
    FASTCAMPUS
      code01
        main.py

main.py
1 print("안녕하세요.")

문제 출력 디버그 콘솔 터미널
1: powershell

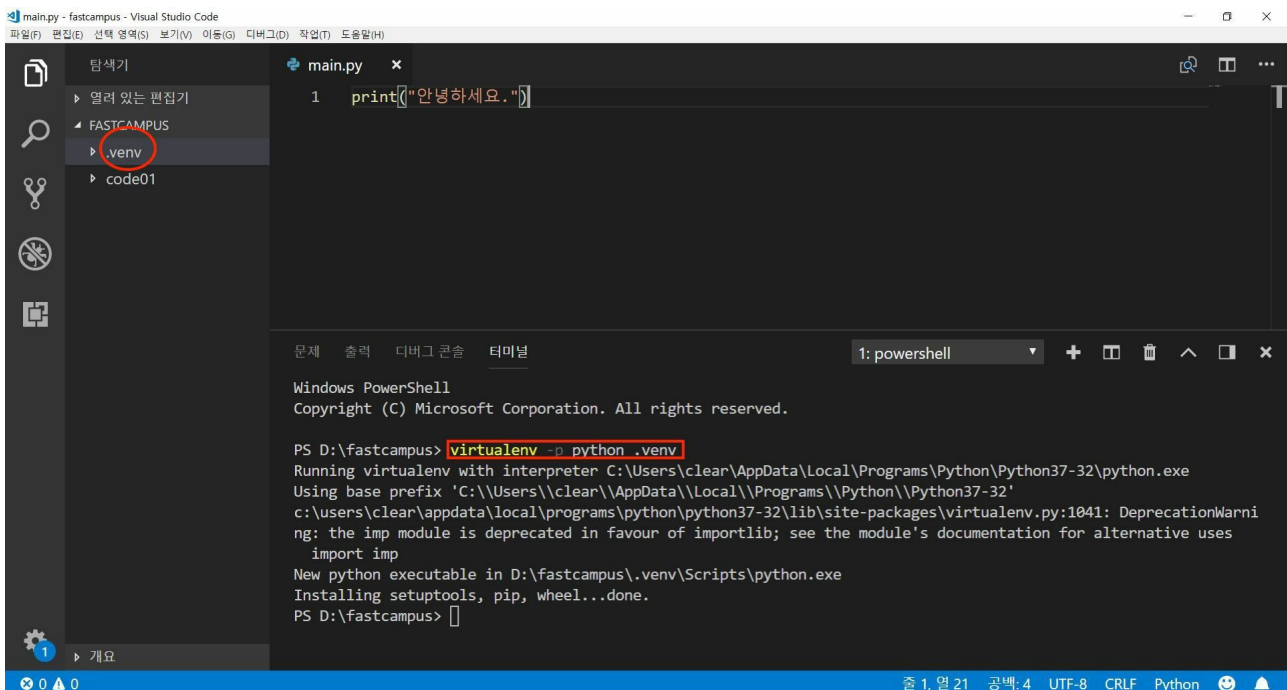
PS D:\fastcampus> cd code01
PS D:\fastcampus\code01> python main.py
안녕하세요.
PS D:\fastcampus\code01> pip3 install virtualenv
Collecting virtualenv
  Downloading https://files.pythonhosted.org/packages/b6/30/96a02b2287098b23b875bc8c2f58071c35d2efe84f747b64d523721dc2b5/virtualenv-16.0.0-py2.py3-none-any.whl (1.9MB)
    100% |#####| 1.9MB 1.8MB/s
Installing collected packages: virtualenv
  The script virtualenv.exe is installed in 'c:\users\clear\appdata\local\programs\python\python37-32\Scripts'
  which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location
  Successfully installed virtualenv-16.0.0
  You are using pip version 10.0.1, however version 18.0 is available.
  You should consider upgrading via the 'python -m pip install --upgrade pip' command.
PS D:\fastcampus\code01>
```

터미널에 아래 명령어를 입력합니다. Successfully installed virtualenv-version이 나오면 정상적으로 설치된 것입니다.

```
pip3 install virtualenv
```

이제 가상 환경을 만들고, 실행 시켜보도록 하겠습니다.

Windows 가상 환경



```
main.py - fastcampus - Visual Studio Code
파일(F) 편집(E) 선택 영역(S) 보기(V) 이동(G) 디버그(D) 작업(T) 도움말(H)

탐색기
  열려 있는 편집기
    FASTCAMPUS
      .venv
      code01

main.py
1 print("안녕하세요.")

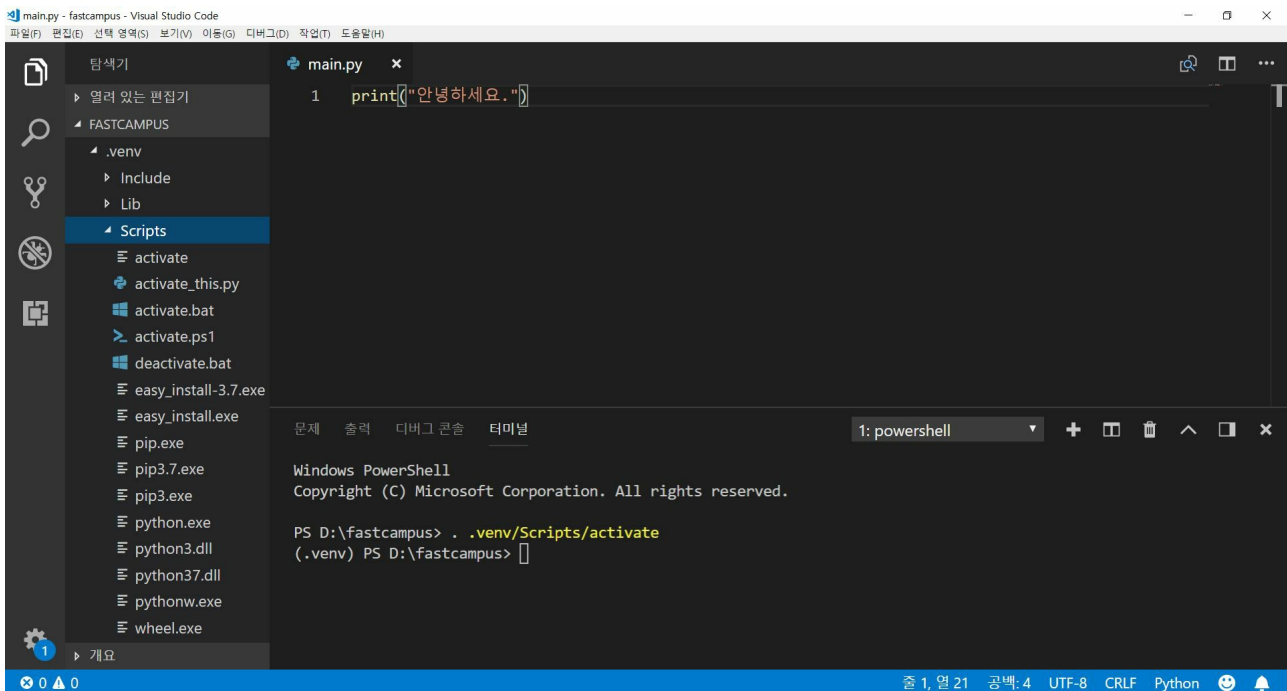
문제 출력 디버그 콘솔 터미널
1: powershell

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS D:\fastcampus> virtualenv -p python .venv
Running virtualenv with interpreter C:\Users\clear\AppData\Local\Programs\Python\Python37-32\python.exe
Using base prefix 'C:\Users\clear\AppData\Local\Programs\Python\Python37-32'
c:\users\clear\appdata\local\programs\python\python37-32\lib\site-packages\virtualenv.py:1041: DeprecationWarning: the imp module is deprecated in favour of importlib; see the module's documentation for alternative uses
  import imp
New python executable in D:\fastcampus\.venv\Scripts\python.exe
Installing setuptools, pip, wheel...done.
PS D:\fastcampus>
```

터미널에 아래 명령어를 실행합니다. 아래 명령어를 실행하면, .venv라는 폴더가 생성되게 됩니다. 여기서 말하는 .venv는 가상환경 프로젝트를 말합니다. 통상적으로, .venv, .env 등의 이름으로 작성하게 되고, 변경해서 사용해도 괜찮습니다. 잠시 기다리면 설치가 완료됩니다.

```
virtualenv -p python .venv
```



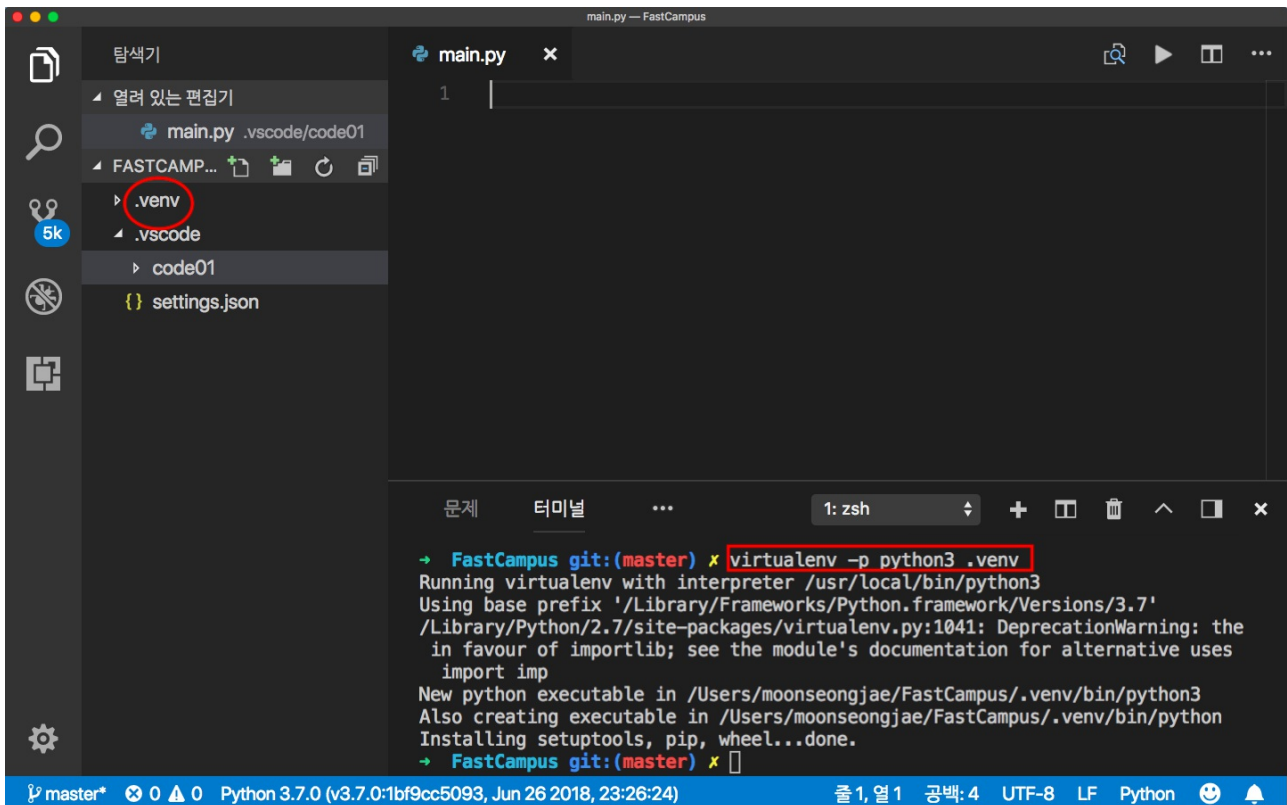
이제 가상 환경을 실행 시키기 위해서 아래 명령어를 입력합니다. 명령어 가장 앞에있는 "."도 입력해주셔야 합니다!!
윈도우의 경우 가상 환경 폴더명/Scripts/activate 위치에 가상 환경을 실행시킬 수 있는 파일이 있습니다. 터미널 라인 처음에 (.venv)가 생기면 정상 실행된 것입니다.

```
. .venv/Scripts/activate
```

- 간혹, 환경 변수 문제로 인해서 위 명령어로 가상 환경을 실행할 경우 '.'은(는) 내부 또는 외부 명령, 실행할 수 있는 프로그램, 또는 배치 파일이 아닙니다. 라고 나타나는 경우가 있습니다. 그러한 경우엔 .을 제외하고 아래와 같이 명령어를 입력해주세요.

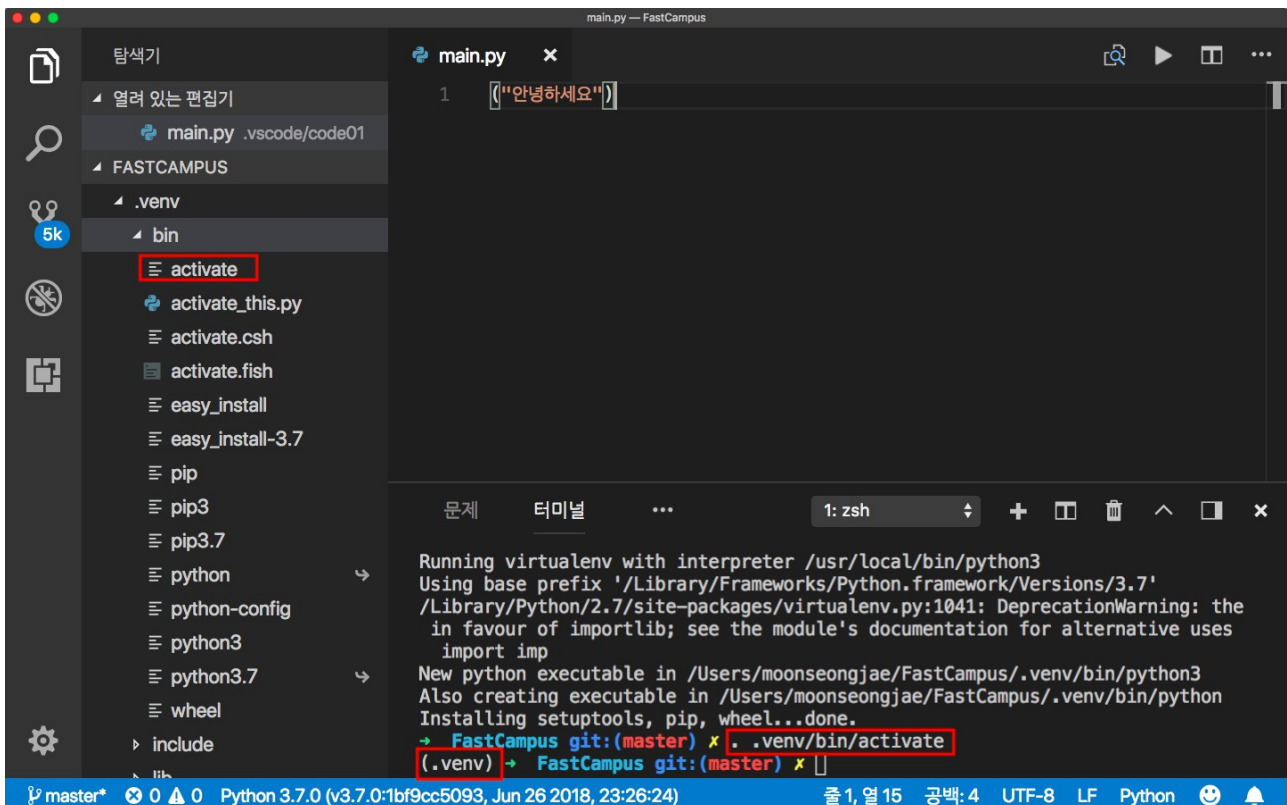
```
.venv/Scripts/activate
```

Mac 가상 환경



터미널에 아래 명령어를 실행합니다. 아래 명령어를 실행하면, .venv라는 폴더가 생성되게 됩니다. 여기서 말하는 .venv는 가상환경 프로젝트를 말합니다. 통상적으로, .venv, .env 등의 이름으로 작성하게 되고, 변경해서 사용해도 괜찮습니다. 잠시 기다리면 설치가 완료됩니다.

```
virtualenv -p python3 .venv
```

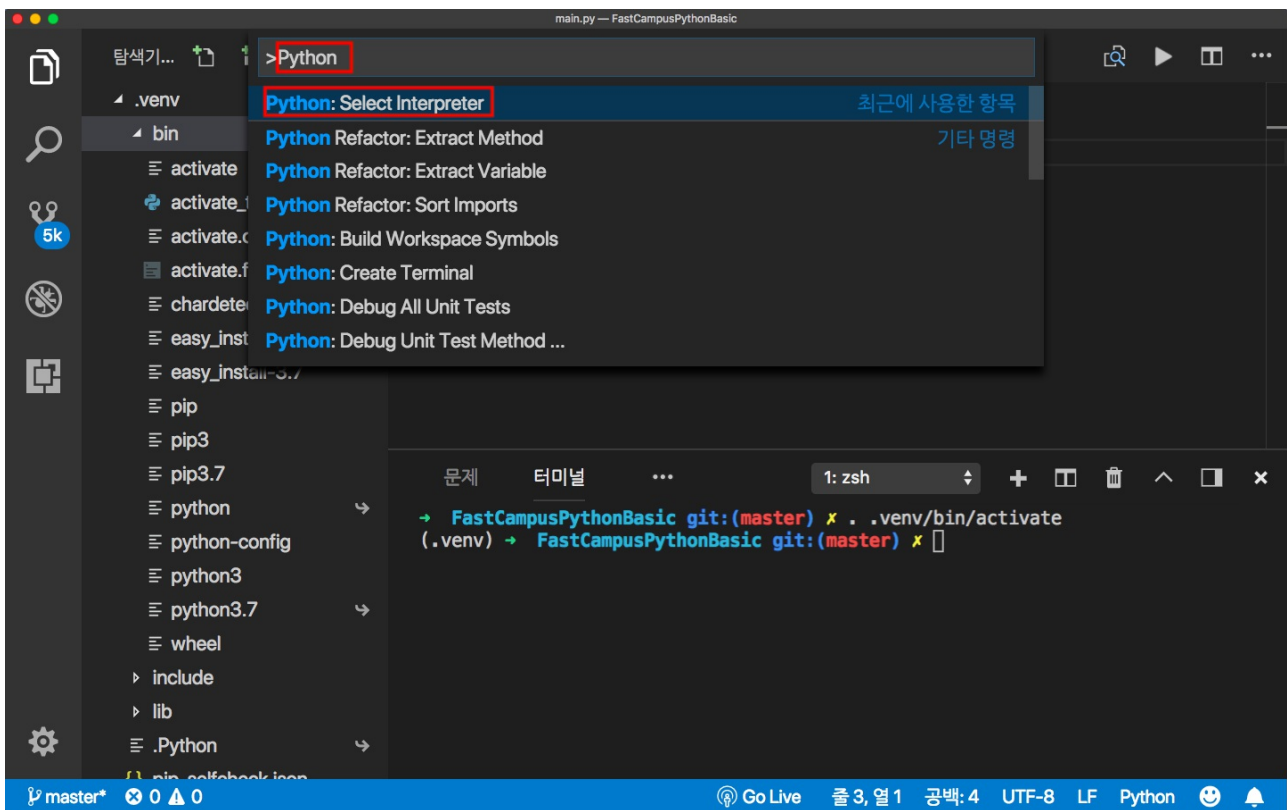


이제 가상 환경을 실행 시키기 위해서 아래 명령어를 입력합니다. 명령어 가장 앞에있는 "."도 입력해주셔야 합니다!!
윈도우의 경우 가상 환경 폴더명/bin/activate 위치에 가상 환경을 실행시킬 수 있는 파일이 있습니다. 터미널 라인
처음에 (.venv)가 생기면 정상 실행된 것입니다.

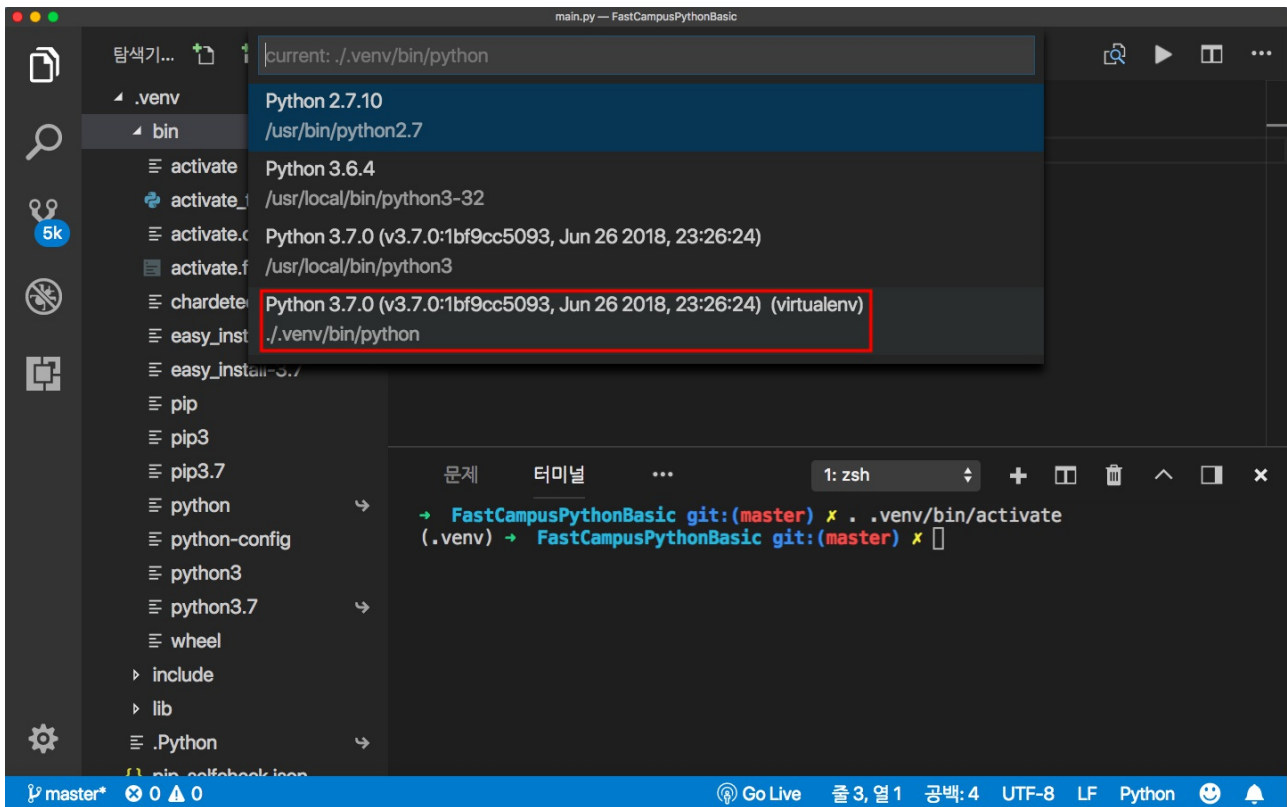
```
. .venv/bin/activate
```

VS Code 인터프리터(공통)

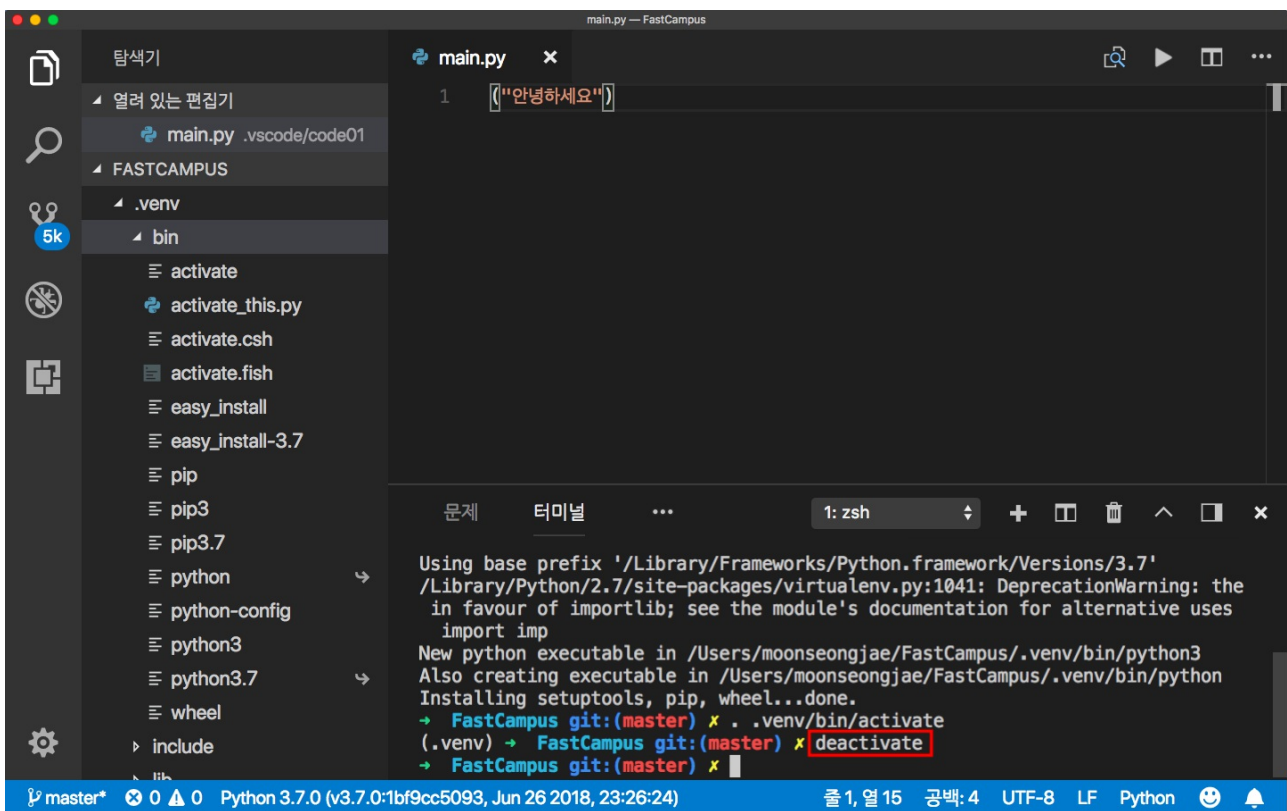
이제 VS Code의 인터프리터를 선택해야 합니다. 강의에서 말씀드렸듯이, 파이썬도 자바스크립트와 같이 인터프리터
모드로 동작하여 코드 실행 시에 한 줄 한 줄 코드를 해석하게 됩니다. VS Code 상에서 어떤 인터프리터(코드 해석기)를
사용할지 설정하는 작업이 필요한데요, 아래 내용을 보고 설정해주세요!



먼저, VS Code의 명령 팔레트를 실행합니다. (단축키, 윈도우: ctrl + shift + p, 맥: cmd + shift + p) 명령 팔레트가 실행되면, 검색 창에 Python을 입력합니다. 하위 선택 메뉴 중에서 Python: Select Interpreter를 선택합니다.



경우에 따라서 여러 가지 파이썬 버전 정보가 나올 수 있는데요, 파이썬 버전 3.7.0 중에서 가상 환경이 실행된, virtualenv가 붙은 인터프리터를 선택합니다.



(주의) 하나의 프로젝트에 가상 환경을 activate 명령어로 실행시키고, 다른 프로젝트로 가서 가상 환경을 실행할 경우엔, 기존의 가상 환경에서 deactivate 명령어를 입력하여 가상 환경을 먼저 종료시키고 실행시켜야 합니다!!

고생하셨습니다! 여기까지 잘 따라오셨다면, 이제 파이썬을 본격적으로 시작할 준비가 되었습니다. 생각보다 환경 설정을 하는 부분이 많이 까다롭죠..? 비단 파이썬뿐만 아니라, 다른 많은 프로그래밍 언어가 처음 환경 설정을 하는 데에 시간이 많이 소요됩니다.

또한, 내용도 생소한 부분들이 많이 등장하기 때문에, 검색 엔진을 통한 검색과 공식 문서를 보며 진행해야 하는 경우가 대부분입니다. 우리가 처음 강의를 시작할 때, 그림을 그리기 위한 스케치북을 고르는 작업이 이 VS Code를 설치하는 것이었다면, 물감을 비롯한 여러 가지 기타 도구를 고르는 작업이 환경 설정 부분이라고 이해해주시면 쉬울 것 같습니다. :)

Python 기본 문법

다양한 프로그래밍 언어는 각자 다른 키워드와 문법 체계를 갖습니다. 우리가 지난 챕터에서 공부했던 자바스크립트 문법과는 상당 부분 다른 형식으로 코드를 작성해야 합니다. 하지만, 우리는 이미 자바스크립트를 공부했기 때문에, 처음 보는 키워드도 많이 생소하진 않을 겁니다. :)

코드 블록

자바스크립트를 비롯한 많은 프로그래밍 언어는 특정 코드 구문의 시작과 끝을 나타내기 위해 코드 블록이라고 불리는 중괄호({})로 실행 코드를 감싸는 형태로 작성합니다. 하지만, 파이썬에서는 코드 블록을 나타내기 위해 콜론(:)과 탭(tab) 혹은, 스페이스바(Space bar) 네 번을 사용합니다. 또한, 파이썬에서는 코드 라인의 끝을 알리기 위해 세미 콜론(;)을 사용하지 않습니다.

코드 블록의 시작을 나타내기 위해 콜론(:)을 작성하고, 코드 블록이 시작하는 줄은 탭이나 스페이스바 네 번으로 구분 짓습니다. 권장하는 표현은 스페이스바 네 번이지만, 우리는 처음 익히기 위해 탭을 사용해도 무방합니다.

아래 코드는 sum이라는 함수를 선언하는 두 언어 간의 차이를 나타냅니다.

- 자바스크립트 코드

```
function sum(x, y){  
  return x + y;  
}
```

- 파이썬 코드

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

기존에 다른 언어에 대한 지식이 있는 상태에서 처음 이 문법을 접하면, 조금 헷갈릴 수 있습니다. 하지만, 조금만 코드를 작성하다 보면, 금방 익숙해지실 수 있습니다. :)

기본 자료형

파이썬에서 변수를 선언할 때는 아무런 키워드도 사용하지 않습니다. 또한, 자바스크립트에서 처럼 변수에 할당하는 값에 따라 해당 변수의 타입이 결정됩니다.(타입 추론)

자료형	의미	예
int	정수	a = 100 a = 0xFF(16진수) a = 0o56(8진수)
float	소숫점을 포함한 수	a = 10.01
bool	참, 거짓을 표현하는 불 값	a = True a = False
None	Null 값	a = None
"문자열" 혹은, '문자열'	문자열 리터럴	a = "문자열" b = '문자열'

#	한 줄 주석 (샵 기호를 쓰고 한 칸 띄우는 것을 권장)	# 파이썬
""" 문자열 """	다중 라인 주석	"""파이썬은 정말이지 멋진 언어입니다."""

새롭게 등장한 None, #, '''문자열''' 등에 대해서만 이해해주시면 나머지 내용은 어렵지 않게 이해하실 수 있을 거라 생각합니다. (참과 거짓을 나타내는 bool 자료형의 경우 True, False로 첫 번째 알파벳이 대문자란 점도 기억해주세요!)

연산자

다음은 파이썬의 연산자와 관련된 내용입니다. 대부분의 연산자는 이미 잘 알고 있는 연산자일 것입니다!

연산자	의미	예
+, -, *, /	더하기, 빼기, 곱하기, 나누기	a = (1 + 2 - 3 * 4) / 5, 결과: -2
**, //, %	제곱, 소숫점은 버리고 몫만 취함, 나머지	a = 2 ** 3 // 3 % 4, 결과: 2
==, !=, >, <, <=, >=	등호, 같지 않음, 부등호	if a == 1:
*=, +=, -=, /=, %=, //=	피연산자와 연산 결과를 할당	a = 2 a *= 2 print(a), 결과: 4
is, is not	같은 Object를 가리키는지 확인	a = 1 b = a print(a is b), 결과: True
and, or, not	모두 참이면 참, 둘 중 하나만 참이어도 참, 참과 거짓 반전	ret = a == 1 and "a = 1" or "a != 1" print(ret), 결과: a = 1
in, not in	피연산자가 컬렉션에 있는지 확인	a = [1,2,3] b = 1 in a print(b), 결과: True

위 표의 내용 중 print, 컬렉션 등의 몇 가지 내용은 아직 이해가 되지 않을 수도 있습니다.

하나하나 간단히 살펴보도록 하겠습니다. 우선, print() 함수의 경우 파이썬에서 기본적으로 제공해주는 함수이며, 이름에서 느껴지듯이, 터미널에 값을 출력하는 함수입니다.

컬렉션은 우리가 자바스크립트를 공부할 때 살펴봤던, 배열과 같은 역할을 하는 자료형입니다. 파이썬에서 제공하는 컬렉션 자료형은 크게 네 가지가 있습니다. 더 자세한 설명은 파이썬 컬렉션 파트에서 정리해드리도록 하겠습니다.

and, or, not 연산자의 경우 자바스크립트에서 공부했던 논리 연산자와 동일하다고 이해하시면 됩니다. 순서대로 각각 &&, ||, !과 대응됩니다.

지금 당장은 전부 이해가 되지 않거나, 내용이 적응되지 않더라도, 코드를 조금만 작성하다 보면 빠르게 익숙해지실 수 있습니다!

조건문

파이썬에서 제공하는 논리적 조건을 처리하기 위한 구문은 if~elif~else 구문이 있습니다. 또한, 타 언어에서처럼 switch~case 구문이 지원되지 않습니다.

아래 코드는 파이썬 조건문을 사용하는 간단한 예시를 보여줍니다.

```
# 조건문 기본형
if 조건:
# 조건에 맞는 실행문
elif 다른 조건:
# 위 조건과 다른 조건 실행
elif 다른 조건:
    pass # 아무것도 실행하지 않고 조건문 종료
else:
# 위 조건에 모두 해당하지 않으면 실행
```

기본 조건문 형식은 자바스크립트 코드와 크게 다른 점이 없습니다. 단, `else if` 구문은, `elif` 구문으로 대체됩니다. 위에서 설명한 코드 블록의 내용대로 코드 블록 구분을 콜론(:)으로 하는 것을 확인할 수 있습니다.

또한, 파이썬에서는 `pass`라는 키워드를 지원합니다. `pass` 키워드는 특정 조건을 만났을 때, 아무것도 실행하지 않고 조건문을 빠져나와 다음 코드를 실행시키는 키워드입니다.

아래 코드는 멤버십 연산자인 `in`과 `not in` 키워드를 이용해 컬렉션 자료형 중 배열과 같은 역할을 하는 리스트(List) 자료형에 특정 값이 있는지 확인하는 코드입니다.

```
# 리스트 선언 및 1, 2, 3 할당
a = [1, 2, 3]

if 1 in a:
    print("a 리스트에 1이 들어있습니다.") # a 리스트에 1이 들어있습니다.
elif not (1 in a):
    print("a 리스트에 1이 들어있지 않습니다.")
```

반복문

다음은 반복문입니다. 코드를 작성할 때 반복문의 사용은 굉장히 중요한데요, 파이썬 코드를 작성할 때도 예외는 아닙니다. 반복문은 전체적인 코드의 실행 시간과 직결되는 부분이 많으며, 필요에 따라 다양한 연출이 가능합니다. `while`문과 `for`문은 특정 조건을 만족할 때, 구문을 계속해서 반복하게 할 때 사용하게 됩니다.

이때, `break`문과 `continue`문은 자바스크립트와 마찬가지로 조건문과 함께 사용되어 각각 반복문을 바로 종료하거나, 코드를 더 실행하지 않고 다시 반복문 처음으로 돌아가게 하는 키워드입니다.

아래 코드는 `while`문을 이용해 1부터 10까지 수 중에서 홀 수의 합을 구하는 예시입니다.

```
sum = 0 # 합을 저장할 변수 선언
i = 0 # 반복문의 조건을 조절할 변수 선언
# i가 10보다 작을 경우 반복
while i < 10:
    i += 1 # 반복을 하는 동안 i의 값을 1씩 증가(파이썬은 증감 연산자를 제공하지 않습니다!)
    if i % 2 == 0: # i를 2로 나눴을 때, 나머지가 0이라면 짝수
        continue # 반복문 처음으로 돌아가서 조건 검사
    sum += i # 합을 나타낼 변수에 i 할당

print(sum) # 25
```

다음은, `for`문입니다. `for`문은 보통 멤버십 연산자 `in`과 함께 사용되는데요, `in` 연산자는 컬렉션 자료형의 값을 하나씩 반환해줍니다.

```
a = [1, 2, 3]
# a 리스트의 값을 하나씩 받아와서 i에 할당합니다.
for i in a:
    # 순서대로 받아온 i 값을 출력합니다.
    print(i) # 1, 2, 3
```

파이썬에서 for문을 사용할 때, 유용한 함수인 range() 함수가 존재하는데요, 위처럼 순서대로 하나씩 받아오는 것이 아니라, 인덱스 형태로 값에 접근할 수도 있습니다.

range() 함수는 입력받은 조건에 맞는 숫자 객체를 만들어 값을 반환합니다. 기본적으로 초기값, 최댓값, 증감 값으로 구성되며, 초기값이 주어지지 않을 경우 인덱스는 0부터 시작합니다.

주의할 점은, range() 함수의 반복 요소는 range() 함수에 선언한 최댓값의 n-1까지만 반환합니다. 아래 표와 같이 사용법은 총 세 가지입니다.

예	의미	반환값
range(3)	최댓값	0, 1, 2
range(3,6)	초기값, 최댓값	3, 4, 5
range(2,11,2)	초기값, 최댓값, 증감값	2, 4, 6, 8, 10

파이썬에는 입력값의 길이를 계산해서 반환하는 len() 함수가 존재합니다. 길이를 계산할 때 자주 사용되므로, 함께 기억해두시면 좋습니다.

아래 코드는 for문을 이용한 간단한 문자열 출력 예시입니다.

```
# 문자열을 담은 리스트 선언
luv = ["I", "do", "love", "you!"]
# 길이 4
length = len(luv)
# 4번 반복 (0, 1, 2, 3)
for i in range(length):
    # 인덱스로 접근
    print(luv[i], end = ' ') # I do love you!
```

위 코드에서 print 함수의 인자의 end = ' '를 작성한 이유는, print 함수는 기본적으로 자동 줄 바꿈이 존재합니다. end = ' '를 작성해주면 줄 바꿈이 일어나지 않고 이전 출력 값 바로 옆에서 값이 나오게 됩니다. 또한, 공백을 통해 값을 한 칸씩 띄운 결과입니다.

파이썬 언어로 할 수 있는 것은 굉장히 많습니다. 하지만, 언제나 그렇듯, 새로운 개념들과 용어가 많이 등장하기 때문에, 상당 부분 어렵고 힘든 부분이 있습니다. 하지만, 공부를 진행하며 그 매력에 금세 빠져들 수 있으실 거라고 믿습니다! :) 긴 글 읽어주시느라 고생하셨습니다. 감사합니다. :)