

---

# Lecture 1

파이썬 프로그래밍 개요 및 개발환경 설정

---

# 프로그래밍이 무엇인가?

- 프로그램이란 문제를 해결하거나, 목표를 달성하기 위한 순차적인 명령이다
  - 명령(Instruction)이란 컴퓨터가 수행할 수 있는 일을 의미한다
  - 명령들을 결합하여 보다 추상적인 새로운 명령을 정의할 수 있다

# 왜 프로그래밍을 배우는 것이 좋을까?

- “어떻게 프로그래밍을 해야 컴퓨터가 내가 하고 싶은 일을 해낼 수 있을까”를 항상 고민해야 된다
  - 어떤 명령을 어느 순서로 써야 목표를 달성할 수 있는지 계획 (알고리즘)
  - 작업을 어떻게 더 효율적으로 할 수 있는지 고민
  - 복잡하고 어려운 문제를 여러 문제로 나눠서 푸는 방법
  - 작성한 코드를 실행할 때 결과가 생각했던 것과 다를 때 문제점을 확인하고 고쳐나가는 과정
- 문제 해결 능력을 키울 수 있다!

# 왜 파이썬을 배우는 것이 좋을까?



- Python은 프로그래밍 언어 중 쉽게 배울 수 있는 편이고, 아주 유용하다
  - 요즘 AI 및 기계학습이 뜨는 데 대부분의 인공지능 프로그램이 Python 기반이다
  - AI 외에 다른 데이터 과학 관련 분야에서도 많이 쓰인다 (통계, 데이터 분석 및 시각화 등)



# 왜 파이썬을 배우는 것이 좋을까?



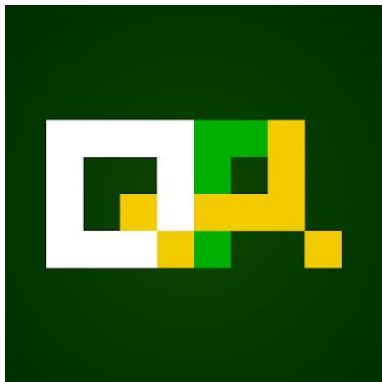
- Python은 프로그래밍 언어 중 쉽게 배울 수 있는 편이고, 아주 유용하다
  - Django, Flask 등 Python 기반 프레임워크가 웹개발에서도 많이 쓰인다



# 왜 파이썬을 배우는 것이 좋을까?



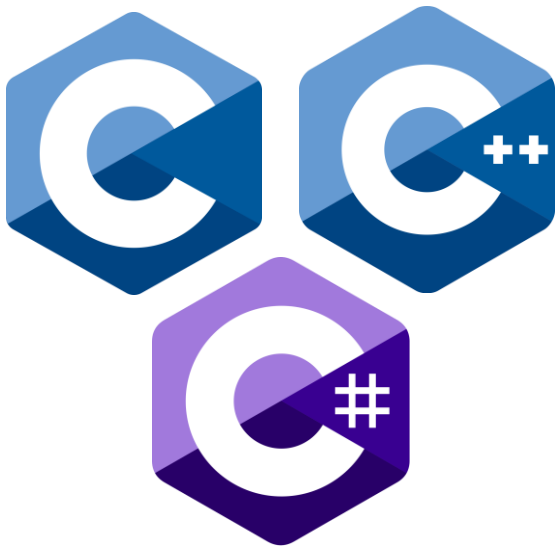
- Python은 프로그래밍 언어 중 쉽게 배울 수 있는 편이고, 아주 유용하다
  - 스마트폰 환경에서도 사용할 수 있다
  - 많은 게임들(예. 문명4)의 개발에 Python이 사용되고 있다



# 왜 파이썬을 배우는 것이 좋을까?

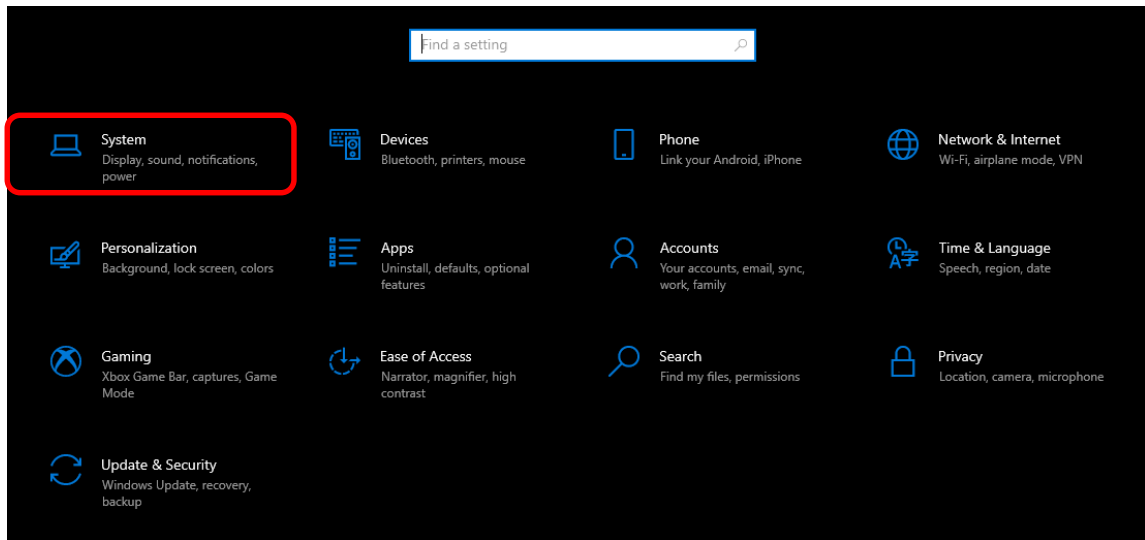
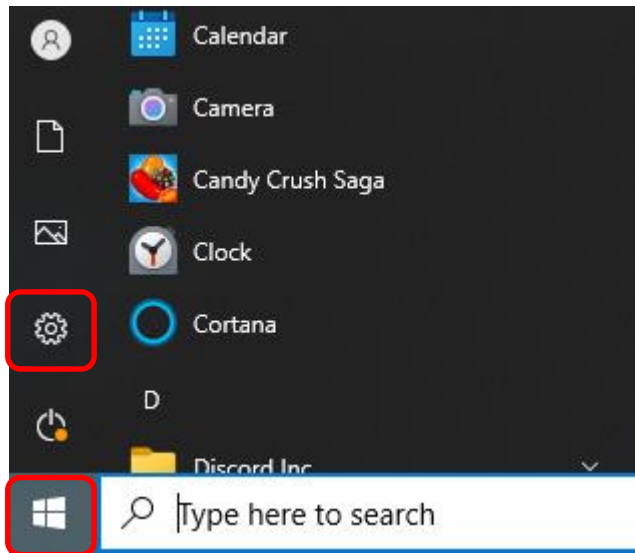


- Python은 프로그래밍 언어 중 쉽게 배울 수 있는 편이고, 아주 유용하다
  - 하나의 프로그래밍 언어를 배워두면, 다른 프로그래밍 언어를 더 쉽게 배울 수 있다



# 개발환경 설정

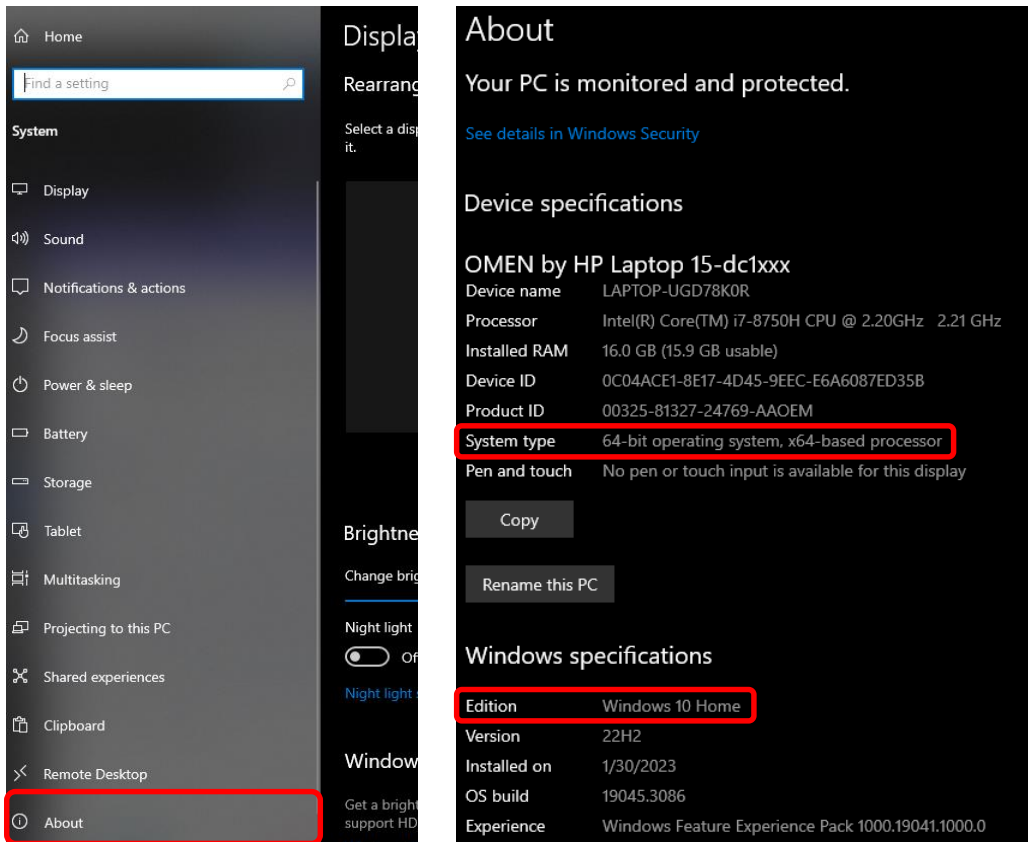
- 윈도우 사양 확인하기:
  - 시작 버튼 > 설정 > 시스템 > 정보





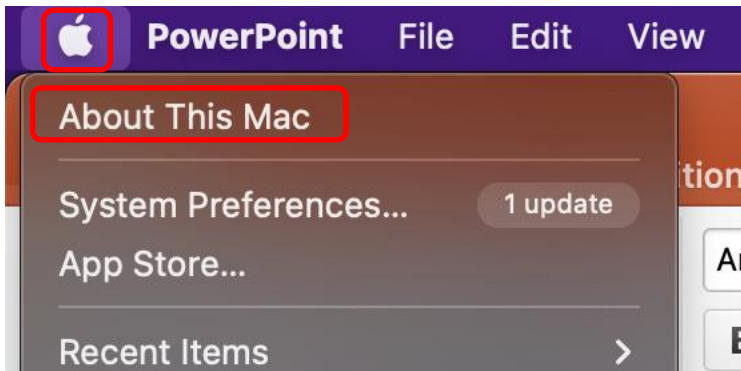
# 개발환경 설정

- 윈도우 사양 확인하기:
  - 시작 버튼 > 설정 > 시스템 > 정보



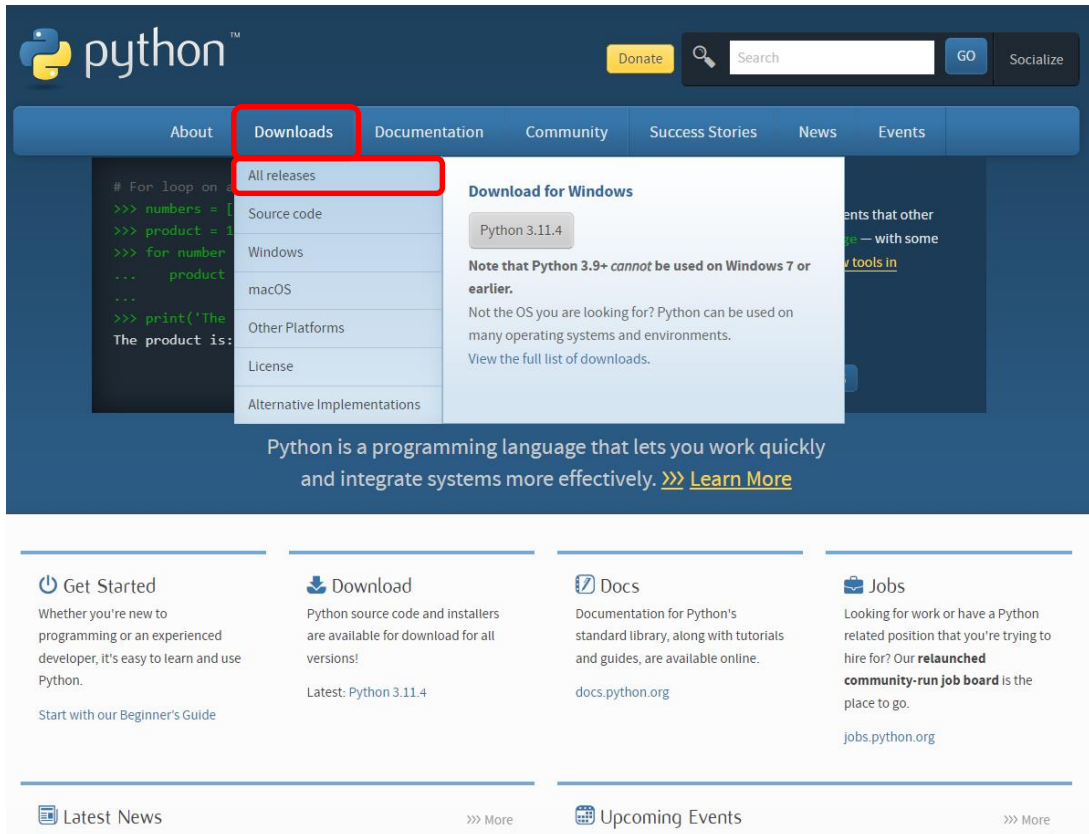
# 개발환경 설정

- 맥북 사양 확인하기:
  - 애플 로고 > 이 Mac에 관하여



# 개발환경 설정

- 파이썬 설치
  - [www.python.org](http://www.python.org)
- Downloads -> All releases



# 개발환경 설정

- 3.10.11을 다운로드

## Active Python Releases

For more information visit the [Python Developer's Guide](#).

Python version	Maintenance status	First released	End of support	Release schedule
3.12	prerelease	2023-10-02 (planned)	2028-10	PEP 693
3.11	bugfix	2022-10-24	2027-10	PEP 664
3.10	security	2021-10-04	2026-10	PEP 619
3.9	security	2020-10-05	2025-10	PEP 596
3.8	security	2019-10-14	2024-10	PEP 569

## Looking for a specific release?

Python releases by version number:

Release version	Release date		Click for more
<a href="#">Python 3.10.12</a>	June 6, 2023	<a href="#">Download</a>	<a href="#">Release Notes</a>
<a href="#">Python 3.11.4</a>	June 6, 2023	<a href="#">Download</a>	<a href="#">Release Notes</a>
<a href="#">Python 3.7.17</a>	June 6, 2023	<a href="#">Download</a>	<a href="#">Release Notes</a>
<a href="#">Python 3.8.17</a>	June 6, 2023	<a href="#">Download</a>	<a href="#">Release Notes</a>
<a href="#">Python 3.9.17</a>	June 6, 2023	<a href="#">Download</a>	<a href="#">Release Notes</a>
<b><a href="#">Python 3.10.11</a></b>	April 5, 2023	<a href="#">Download</a>	<a href="#">Release Notes</a>
<a href="#">Python 3.11.3</a>	April 5, 2023	<a href="#">Download</a>	<a href="#">Release Notes</a>
<a href="#">Python 3.10.10</a>	Feb. 8, 2023	<a href="#">Download</a>	<a href="#">Release Notes</a>

[View older releases](#)

# 개발환경 설정

## Files

Version	Operating System	Description	MD5 Sum	File Size	GPG	Sigstore
<a href="#">Gzipped source tarball</a>	Source release		7e25e2f158b1259e271a45a249cb24bb	26085141	<a href="#">SIG</a>	<a href="#">.sigstore</a>
<a href="#">XZ compressed source tarball</a>	Source release		1bf8481a683e0881e14d52e0f23633a6	19640792	<a href="#">SIG</a>	<a href="#">.sigstore</a>
<a href="#">macOS 64-bit universal2 installer</a>	macOS	for macOS 10.9 and later	f5f791f8e8bfb829f23860ab08712005	41017419	<a href="#">SIG</a>	<a href="#">.sigstore</a>
<a href="#">Windows embeddable package (32-bit)</a>	Windows		fee70dae06c25c60cbe825d6a1bfda57	7650388	<a href="#">SIG</a>	<a href="#">.sigstore</a>
<a href="#">Windows embeddable package (64-bit)</a>	Windows		f1c0538b060e03cbb697ab3581cb73bc	8629277	<a href="#">SIG</a>	<a href="#">.sigstore</a>
<a href="#">Windows help file</a>	Windows		52ff1d6ab5f300679889d3a93a8d50bb	9403229	<a href="#">SIG</a>	<a href="#">.sigstore</a>
<a href="#">Windows installer (32 -bit)</a>	Windows		83a67e1c4f6f1472bf75dd9681491bf1	27865760	<a href="#">SIG</a>	<a href="#">.sigstore</a>
<a href="#">Windows installer (64-bit)</a>	Windows	Recommended	a55e9c1e6421c84a4bd8b4be41492f51	29037240	<a href="#">SIG</a>	<a href="#">.sigstore</a>

# 개발환경 설정

Python 3.10.11 (64-bit) Setup

## Install Python 3.10.11 (64-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.



Install Now

C:\Users\james\AppData\Local\Programs\Python\Python310

Includes IDLE, pip and documentation  
Creates shortcuts and file associations



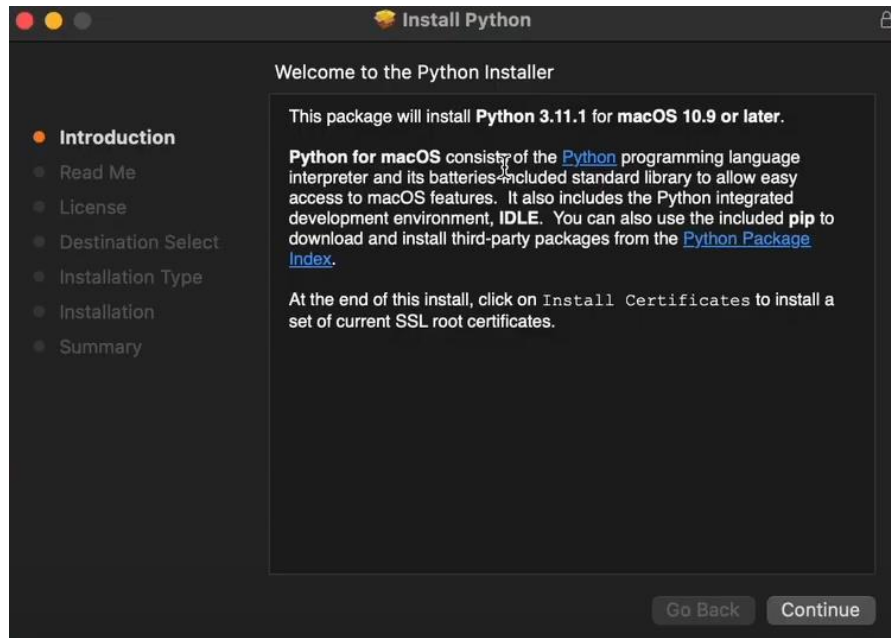
Customize installation

Choose location and features

☒ Use admin privileges when installing py.exe

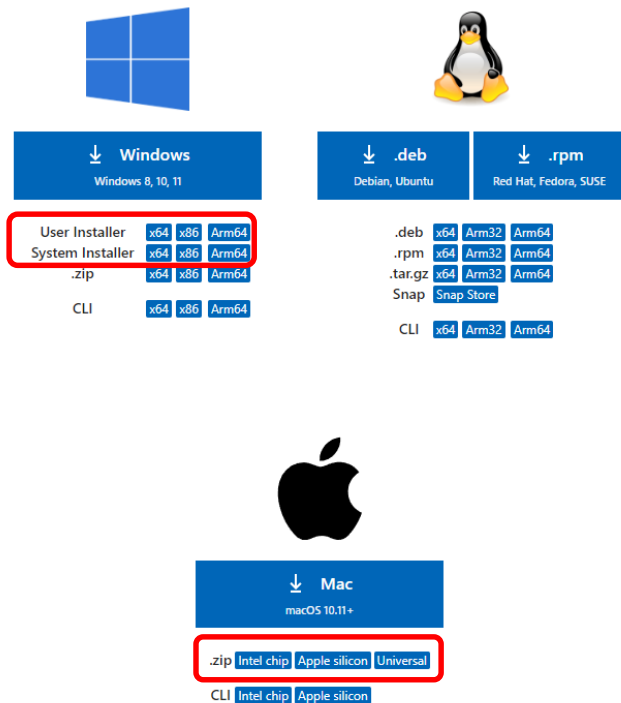
☒ Add python.exe to PATH

Cancel



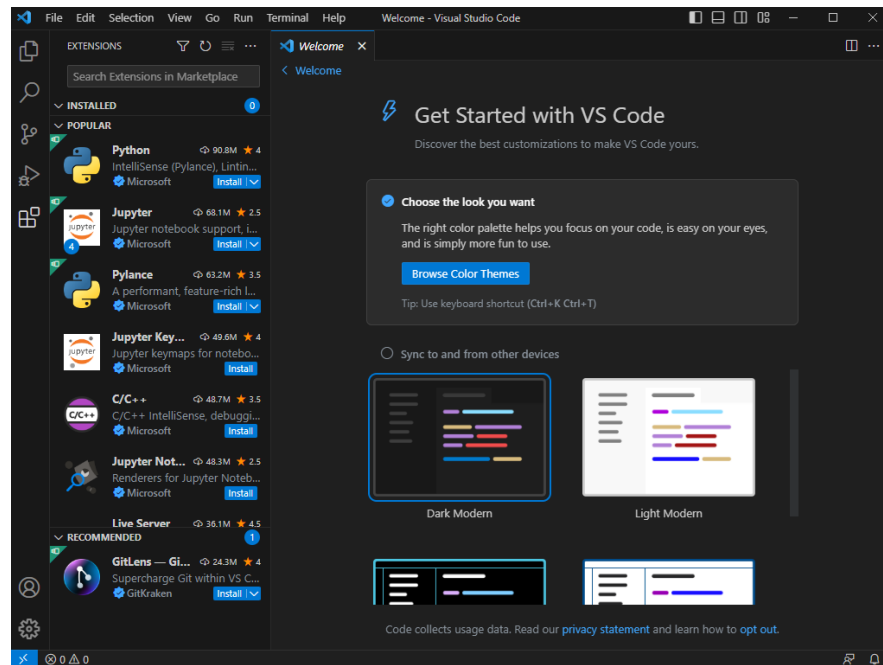
# 개발환경 설정

- Visual Studio Code 설치
  - <https://code.visualstudio.com/download>
  - 윈도우 32 비트 OS를 사용하고 있다면 x86, 64 비트는 x64
  - macOS는 프로세서에 따라 설치



# 개발환경 설정

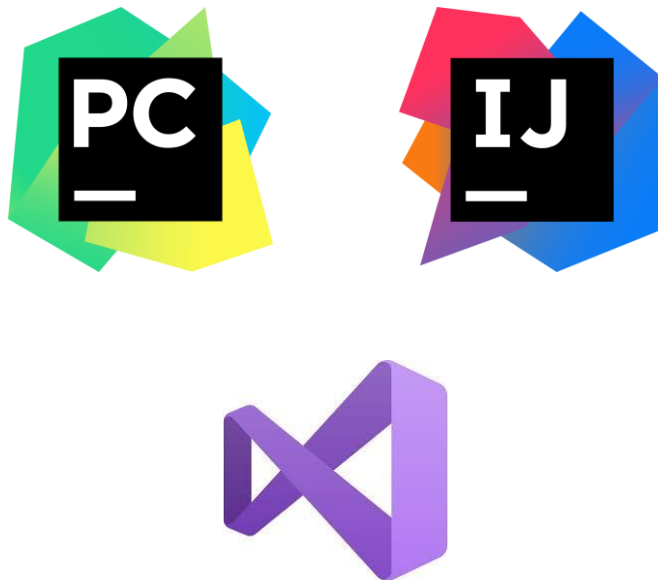
- Visual Studio Code 설치
  - Visual Studio Code는 요즘 핫한 텍스트 에디터다
  - Visual Studio Code로 코드를 작성하고 실행할 수 있다





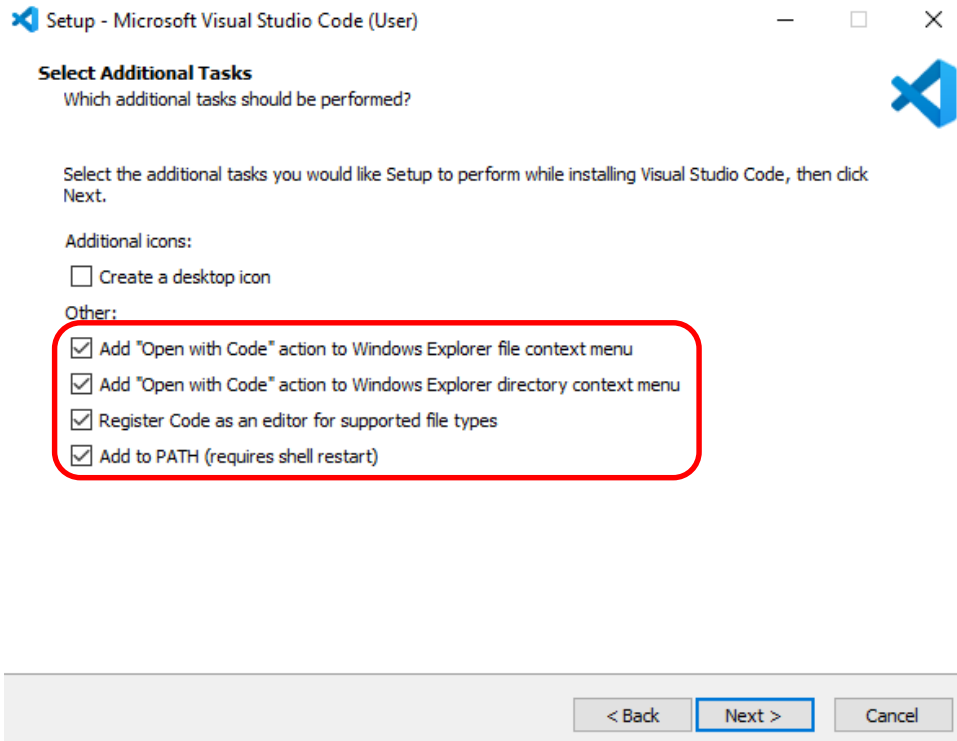
# 개발환경 설정

- Visual Studio Code 설치
  - 확장 프로그램을 설치하면 IDE (Integrated Development Environment – 통합 개발 환경)와 가깝다고 볼 수 있다
  - IDE는 개발자가 더 효율적으로 프로그래밍을 할 수 있게끔 많은 개발 도구를 한 프로그램으로 모은 것



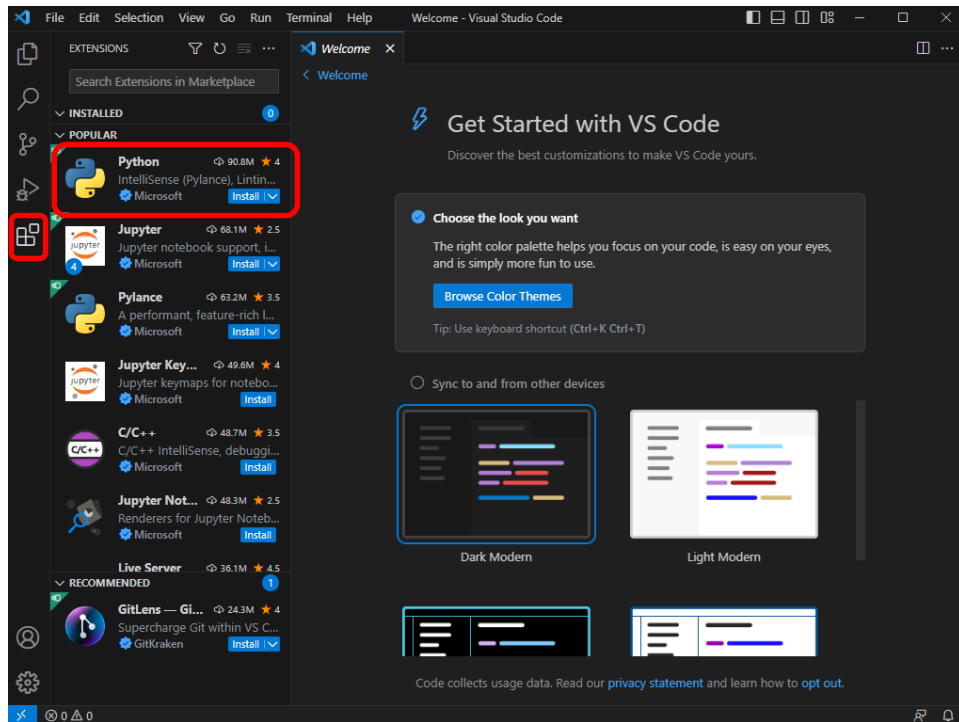
# 개발환경 설정

- Visual Studio Code 설치
  - <https://code.visualstudio.com/download>
  - 윈도우 32 비트 OS를 사용하고 있다면 x86, 64 비트는 x64
  - macOS는 프로세서에 따라 설치



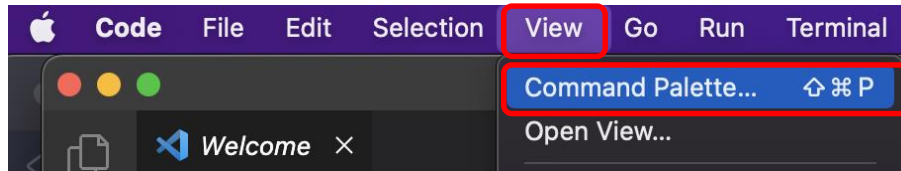
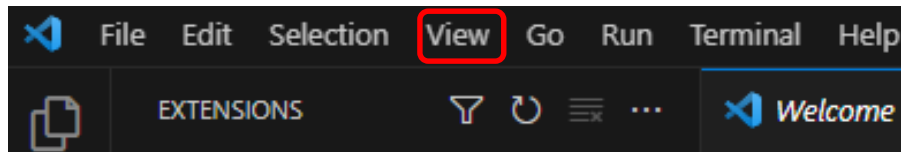
# 개발환경 설정

- Visual Studio Code 설치
  - 파이썬 확장 설치



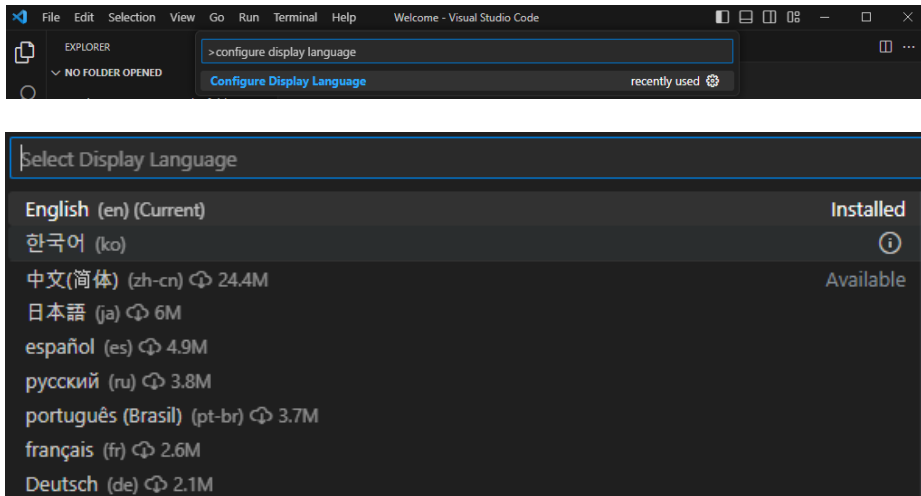
# 개발환경 설정

- Visual Studio Code 설정
  - Command Palette (명령 팔레트)로 Visual Studio Code를 설정할 수 있다
    - 보기 > 명령 팔레트 (View > Command Palette)
    - Ctrl+Shift+P (윈도우) / Command+Shift+P (맥)



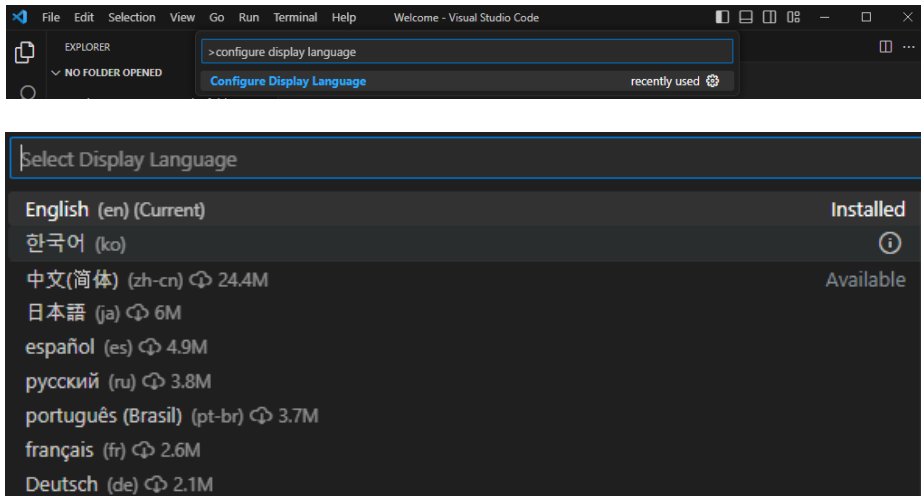
# 개발환경 설정

- Visual Studio Code 설정
  - 명령 팔레트를 열고 “Configure Display Language” 입력한 후 “한국어”를 눌러 언어 팩을 다운로드



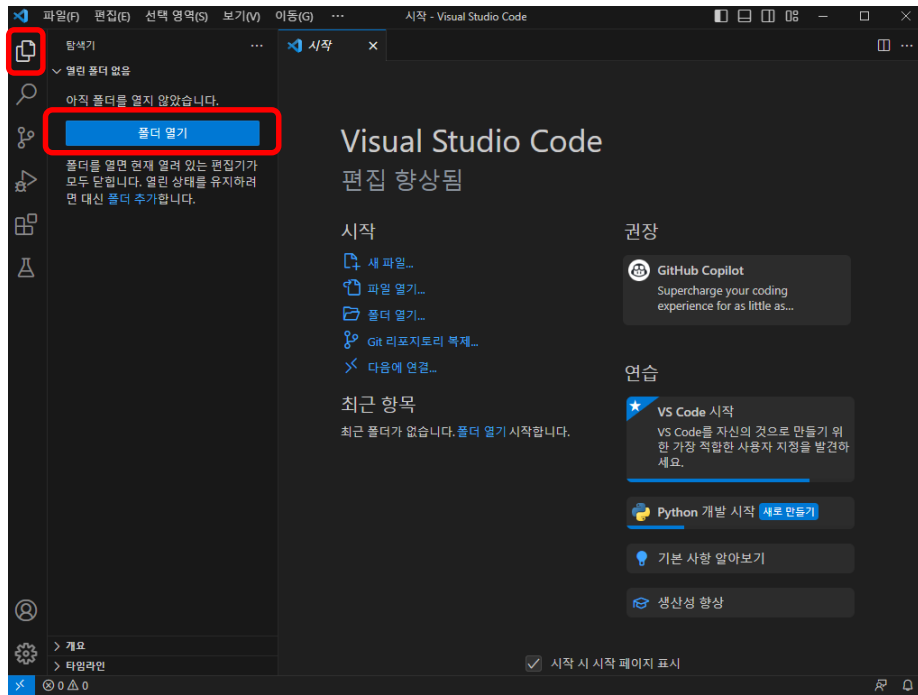
# 개발환경 설정

- Visual Studio Code 설정
  - 다운로드 후 다시 “Configure Display Language” 입력한 후 “한국어”를 눌러 언어 바꿀 수 있다



# 개발환경 설정

- Visual Studio Code 설정
  - 코딩 작업을 할 폴더를 열기



# 개발환경 설정

- Visual Studio Code 설정
  - 가상환경(virtual environment) 만들기
  - 가상환경이란?
    - 다른 코딩 환경과 독립된 코딩 환경

가상환경 1

- 파이썬 3.10

가상환경 2

- 파이썬 3.10



# 개발환경 설정

- Visual Studio Code 설정
  - 가상환경을 왜 사용하는 것이 좋은가?
    - 소프트웨어 버전 간 충돌을 방지하기 위한 관리에 유용하다
    - 코드를 실행하기 위해 필요한 환경을 쉽게 관리하고 재현할 수 있다

가상환경 1

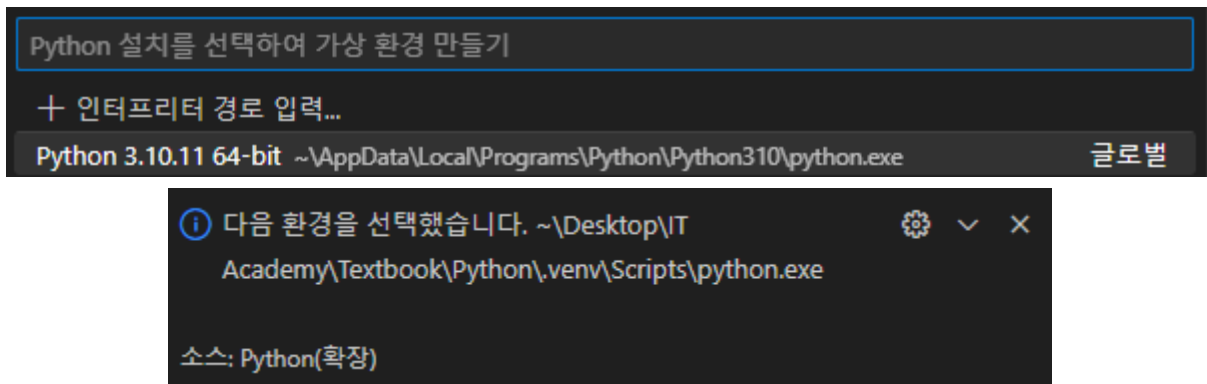
- 파이썬 3.10
- 소프트웨어 A 버전 1
- 소프트웨어 B

가상환경 2

- 파이썬 3.10
- 소프트웨어 A 버전 2
- 소프트웨어 C

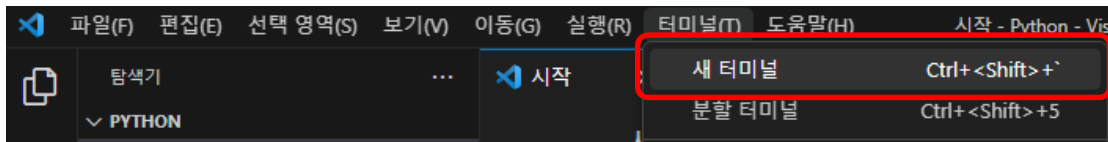
# 개발환경 설정

- 가상환경(virtual environment) 만들기
  - 명령 팔레트에서 “Python: Create Environment” / “Python: 환경 만들기” > “Venv”
  - 설치한 파이썬 버전 클릭



# 개발환경 설정

- Visual Studio Code 설정
  - 새 터미널 열기
  - 윈도우면 PowerShell이 켜지고 에러가 뜰 수 있음

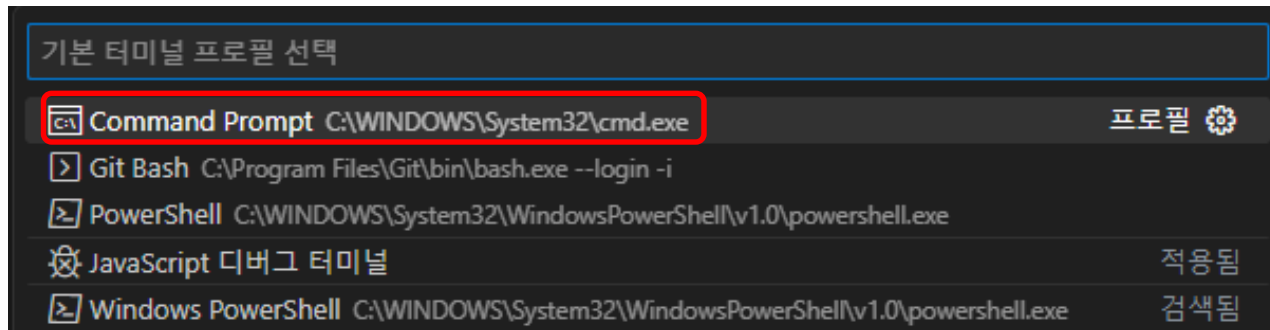
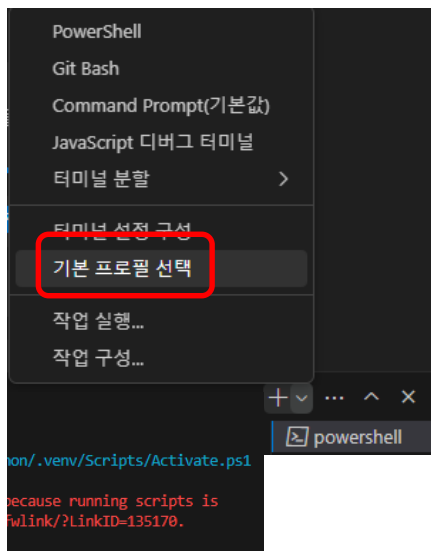


```
문제    출력    디버그 콘솔    터미널

PS C:\Users\james\Desktop\IT Academy\Textbook\Python> & "c:/Users/james/Desktop/IT Academy/Textbook/Python/.venv/Scripts/Activate.ps1"
& : File C:\Users\james\Desktop\IT Academy\Textbook\Python\.venv\Scripts\Activate.ps1 cannot be loaded because running scripts is disabled on this system. For more information, see about_Execution_Policies at https://go.microsoft.com/fwlink/?LinkID=135170.
At line:1 char:3
+ & "c:/Users/james/Desktop/IT Academy/Textbook/Python/.venv/Scripts/Ac ...
+ ~~~~~
+ CategoryInfo          : SecurityError: (:) [], PSSecurityException
+ FullyQualifiedErrorId : UnauthorizedAccess
PS C:\Users\james\Desktop\IT Academy\Textbook\Python> |
```

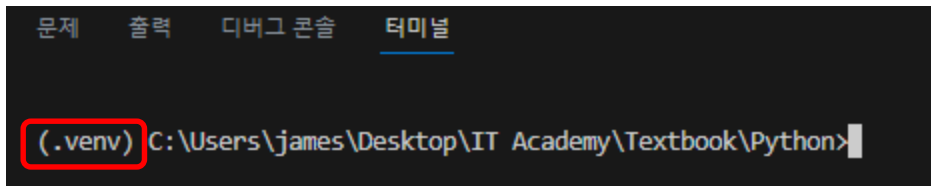
# 개발환경 설정

- Visual Studio Code 설정
  - 디폴트 터미널을 명령 프롬프트 (윈도우), bash/zsh (맥)으로 수정



# 개발환경 설정

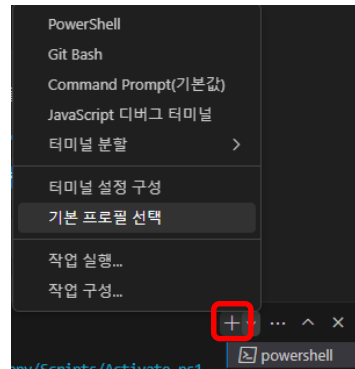
- Visual Studio Code 설정
  - 설정이 잘 됐다면 터미널 창 좌단에 (.venv)라 떴야 된다



The image shows a screenshot of the Visual Studio Code interface, specifically the terminal window. The terminal has tabs at the top labeled '문제', '출력', '디버그 콘솔', and '터미널', with '터미널' being the active tab. The terminal prompt shows the current directory as 'C:\Users\james\Desktop\IT Academy\Textbook\Python>'. The prompt is preceded by the environment variable '(.venv)', which is highlighted with a red rectangular box, indicating that the virtual environment is successfully activated.

# 개발환경 설정

- Visual Studio Code 설정
  - (.venv) 표시가 없으면
    - 명령 팔레트 > “Python: Select Interpreter” > “.venv” > 새 터미널 창 클릭하거나
    - 터미널에서 “.venv\Scripts\activate.bat” (윈도우) / “source .venv/bin/activate” (맥) 입력
      - 이 명령어는 Code에서 열려 있는 폴더가 “.venv” 폴더의 상위 폴더일 때 입력을 해야 된다
      - 상위 폴더 말고 다른 폴더가 열려 있다면...?



# 기초 터미널 명령어 이해하기

- dir (윈도우) / ls (맥) (dir은 directory, ls는 list의 약자다)
  - 현재 보고 있는 폴더 안에 있는 모든 것을 보여줘!
  - “.”은 항상 현재 보고 있는 폴더를 의미한다
  - “..”은 항상 상위폴더를 의미한다

```
Directory of C:\Users\james\Desktop\IT Academy\Textbook\Python
07/13/2023  07:29 PM    <DIR>          .
07/13/2023  07:29 PM    <DIR>          ..
07/12/2023  09:06 PM    <DIR>          .venv
07/13/2023  07:29 PM                0 hello_world.py
                        1 File(s)                0 bytes
```

# 기초 터미널 명령어 이해하기

- cd (change directory의 약자)
  - 보고 싶은 폴더로 가줘!
  - “cd .venv” 하면 컴퓨터가 먼저 현재 폴더에서 .venv라는 하위폴더가 있는지 확인한다
  - 있으면 .venv 하위 폴더로 들어가서 현재 보고 있는 폴더를 .venv로 바꾼다

```
(.venv) C:\Users\james\Desktop\IT Academy\Textbook\Python>cd .venv

(.venv) C:\Users\james\Desktop\IT Academy\Textbook\Python\.venv>dir
Volume in drive C is Windows
Volume Serial Number is FE14-2FD8

Directory of C:\Users\james\Desktop\IT Academy\Textbook\Python\.venv

07/12/2023  09:06 PM    <DIR>          .
07/12/2023  09:06 PM    <DIR>          ..
07/12/2023  09:06 PM                1 .gitignore
07/12/2023  09:06 PM    <DIR>          Include
07/12/2023  09:06 PM    <DIR>          Lib
07/12/2023  09:06 PM                120 pyenv.cfg
07/12/2023  09:06 PM    <DIR>          Scripts
                   2 File(s)                121 bytes
                   5 Dir(s)  8,252,366,848 bytes free
```



# 기초 터미널 명령어 이해하기

- cd (change directory의 약자)
  - “cd .venv”는 상대 경로(relative path)를 사용한다
    - 현재 보고 있는 폴더 기준으로 다른 폴더(이 경우에는 .venv)의 경로
    - “cd” 이후에 쓴 경로의 첫 캐릭터가 슬래시가 아니라 상대 경로로 이해한다
  - 절대 경로(absolute path)도 사용할 수 있다

```
(.venv) C:\Users\james\Desktop\IT Academy\Textbook\Python>cd C:\Users\james\Desktop\IT Academy\Textbook\Python\.venv
```

```
(.venv) C:\Users\james\Desktop\IT Academy\Textbook\Python\.venv>|
```

# 기초 터미널 명령어 이해하기

- cd (change directory의 약자)
  - 현재 .venv 폴더를 보고 있는데, 다시 원래 보고 있던 폴더 (.venv의 상위폴더)를 보고 싶어!
  - 어떤 명령을 입력하면 될까?

# 기초 터미널 명령어 이해하기

- cd (change directory의 약자)
  - 현재 .venv 폴더를 보고 있는데, 다시 원래 보고 있던 폴더 (.venv의 상위폴더)를 보고 싶어!
  - 어떤 명령을 입력하면 될까?
    - 절대 경로: “cd C:\foldername1\foldername2\foldername3\.....”
    - 상대 경로: “cd ..”

# 기초 터미널 명령어 이해하기

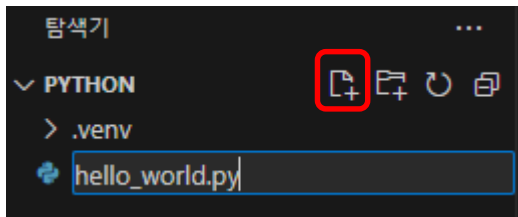
- `cls` (윈도우) / `clear` (맥)
  - 터미널 창에 나와 있는 모든 결과값을 없앤다

# 개발환경 설정

- 가상환경으로 들어가려면 터미널에서 “.venv\Scripts\activate.bat” (윈도우) / “source .venv/bin/activate” (맥) 입력
  - 현재 폴더 안에 있는 .venv 폴더 안에 있는 Scripts(윈도우) / bin(맥) 하위 폴더 안에 있는 activate 파일을 실행해줘!
  - .venv의 상위 폴더가 아닌 다른 폴더를 보고 있을 때 가상환경을 키려면?
- 가상환경을 끄고 싶을 때 “deactivate” 입력

# Hello, Python!

- 개발환경 설정 드디어 끝!
- 파이썬 파일 만들기
  - Visual Studio Code 탐색기에 “새 파일” 버튼 누르고 파일 이름 입력



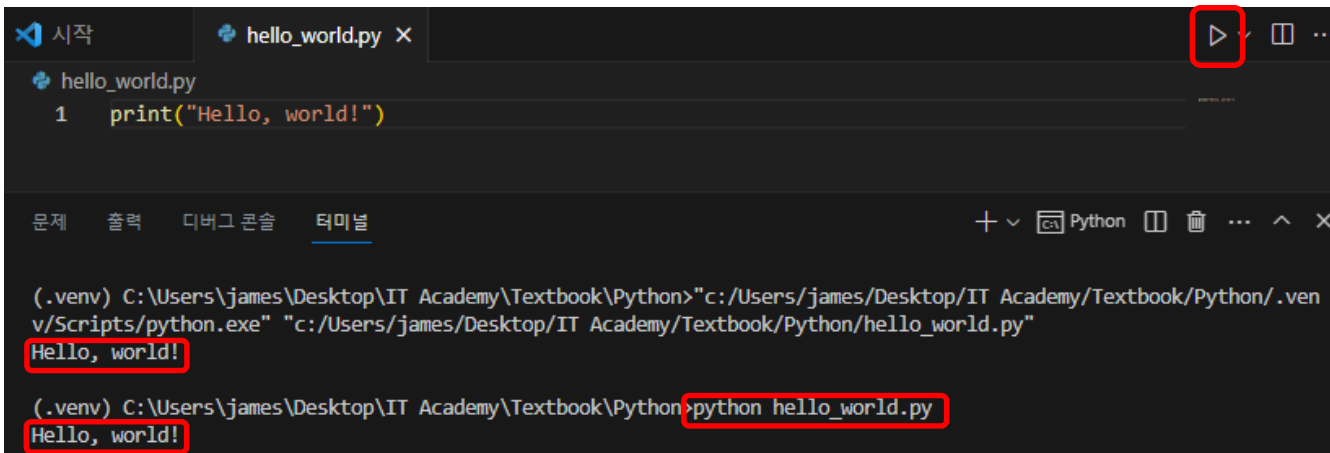
# Hello, Python!

- 배울 첫 명령: print()
- 터미널에 print() 안에 들어가 있는 것을 출력
- 명령에 괄호가 있으면 함수라고 부른다
  - 수학 시간에 배우는 함수와 비슷하다
    - $y = f(x) \Rightarrow x$ 를 입력하면  $y$ 가 나온다
  - 프로그래밍에서 보이는 함수들의 입력값을 인수, 출력값을 반환값이라 부른다

```
1 print("Hello, world!")
```

# Hello, Python!

- 파이썬 코드를 실행하는 방법:
  - 우측 상단에 있는 실행 버튼 클릭
  - 터미널에서 “python hello\_world.py” 입력
    - 여기서 “hello\_world.py”가 상대 경로이고, python 명령어는 python 코드를 실행해주는 명령이다



The screenshot shows the Visual Studio Code interface. At the top, there are two tabs: '시작' (Start) and 'hello\_world.py X'. The 'hello\_world.py' tab is active, showing a single line of code: `1 print("Hello, world!")`. In the top right corner of the editor, a red box highlights the 'Run' button (a play icon). Below the editor, the 'TERMINAL' tab is selected. It shows the command prompt output: `(.venv) C:\Users\james\Desktop\IT Academy\Textbook\Python>"c:/Users/james/Desktop/IT Academy/Textbook/Python/.venv/Scripts/python.exe" "c:/Users/james/Desktop/IT Academy/Textbook/Python/hello_world.py"`. The output `Hello, world!` is shown on the next line and is highlighted with a red box. Below this, the command `(.venv) C:\Users\james\Desktop\IT Academy\Textbook\Python>python hello_world.py` is entered, and the output `Hello, world!` is again shown and highlighted with a red box.




# Hello, Python!

- 줄이 여러 개면 순차적으로 하나씩 실행이 된다

```
1 print("Hello, world 1!")  
2 print("Hello, world 2!")  
3 print("Hello, world 3!")  
4 print("Hello, world 4!")
```

실행



```
Hello, world 1!  
Hello, world 2!  
Hello, world 3!  
Hello, world 4!
```

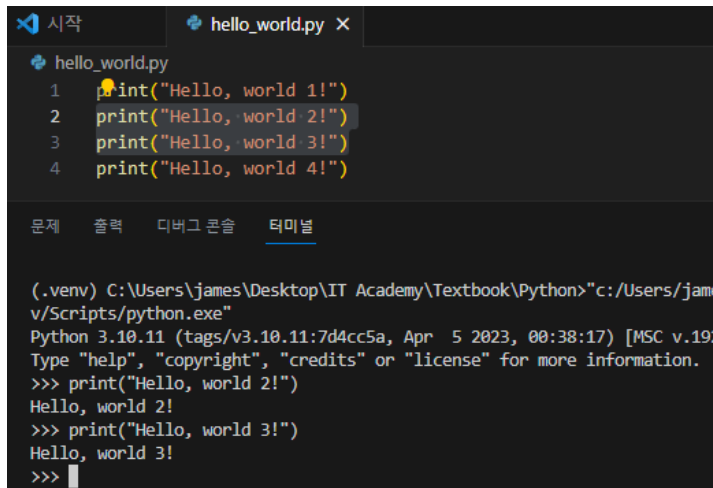
# Hello, Python!

- 터미널에서 “python”만 입력하면 파이썬 인터프리터가 켜진다
  - 코드를 한줄씩 입력하면서 코드의 결과값을 쉽게 확인할 수 있는 프로그램이 켜진다
  - 파이썬 인터프리터를 끄려면 `exit()` 입력

```
(.venv) C:\Users\james\Desktop\IT Academy\Textbook\Python>python
Python 3.10.11 (tags/v3.10.11:7d4cc5a, Apr  5 2023, 00:38:17) [MSC v.19
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello, world!")
Hello, world!
>>> █
```

# Hello, Python!

- Visual Studio Code로 선택한 줄을 파이썬 인터프리터로 돌리는 기능도 있다
  - 실행하고 싶은 줄을 선택한 후 Shift+Enter 입력하거나 우측 클릭 > “Python 실행” > “파이썬 터미널에서 선택/줄 실행”



The screenshot shows the Visual Studio Code interface. The top editor pane displays a file named `hello_world.py` with the following code:

```
1 print("Hello, world 1!")
2 print("Hello, world 2!")
3 print("Hello, world 3!")
4 print("Hello, world 4!")
```

The bottom pane shows the integrated terminal with the following output:

```
(.venv) C:\Users\james\Desktop\IT Academy\Textbook\Python>"c:/Users/jame
v/Scripts/python.exe"
Python 3.10.11 (tags/v3.10.11:7d4cc5a, Apr  5 2023, 00:38:17) [MSC v.192
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello, world 2!")
Hello, world 2!
>>> print("Hello, world 3!")
Hello, world 3!
>>> 
```

# 디버깅 (Debugging)

- 버그란 (bug)?
  - 프로그램에서 잘못된 부분
- 디버깅이란?
  - 버그를 찾고 고치는 작업
- 코딩하면서 수많은 생각치 못한 오류가 생긴다
  - Syntax error: 파이썬이 프로그램을 이해 못해서 실행이 아예 안 되는 오류
  - Runtime error: 프로그램 실행 중에 에러 메시지가 뜨고 프로그램이 종료되는 오류
  - Semantic error: 프로그램 실행 중 에러 메시지는 없지만 예상치 못한 결과가 나오는 오류

# 디버깅 (Debugging)

```
hello_world.py
1  print("Hello, world 1!")
2  print("Hello, world 2!")
3  prin("Hello, world 3!")
4  print("Hello, world 4!")
```

어느 줄을 실행했을 때 오류가  
뜨는지 보여주는 Traceback

```
(.venv) C:\Users\james\Desktop\IT Academy\Textbook\Python>python hello_world.py
Hello, world 1!
Hello, world 2!
Traceback (most recent call last):
  File "C:\Users\james\Desktop\IT Academy\Textbook\Python\hello_world.py", line 3, in <module>
    prin("Hello, world 3!")
NameError: name 'prin' is not defined. Did you mean: 'print'?
```

어떤 종류의  
오류인지

왜 이 오류가  
났는지에 대한  
추가적인 정보

# Hello, Python! 2

- input() 함수
  - 괄호 안에 있는 것을 터미널에 출력한 후 사용자의 입력을 기다린다

```
1 input("Tell me something!")
```

사용자가 Hello, Python!을 입력



Tell me something! ■

Tell me something! Hello, Python!

# Hello, Python! 2

- input() 명령
  - 사용자가 입력한 것을 반환을 한다
  - print() 함수와 같이 쓰면 이 사실을 확인할 수 있다

1 print(input("Tell me something! "))

사용자가 Hello, Python!을 입력



Tell me something! ■

Tell me something! Hello, Python!  
Hello, Python!

# Hello, Python! 2

- 왜 이런 결과가 나올까?
  - 프로그램이 각 줄을 왼쪽부터 오른쪽으로 읽고, 먼저 나오는 함수를 실행하려고 한다
  - 인수나 피연산자는 값이어야 한다
  - print() 함수 안에 있는 것을 출력하고 싶은데 print 함수에 들어가는 인수가 함수라 함수의 반환값을 받을 때까지 기다린다

```
1 print(input("Tell me something! "))
```

```
Tell me something! Hello, Python!  
Hello, Python!
```



# Hello, Python! 3

- 파이썬 인터프리터로 간단한 연산을 할 수 있다

```
(.venv) C:\... python  
Python 3.10.11 ...
```

```
>>> 3 + 3  
6  
>>> 3 - 3  
0  
>>> 3 * 3  
9  
>>> 30 - 2*6  
18  
>>> (30 - 2*6) / 3  
6.0  
>>> 17 / 3  
5.666666666666667  
>>> 17 // 3  
5  
>>> 17 % 3  
2  
>>> 2 ** 5  
32
```

# Hello, Python! 3

- 코드로 구현하고 싶다면 print() 함수를 사용할 수 있다

```
1 print(3 + 3)
2 print(3 - 3)
3 print(3 * 3)
4 print(30 - 2 * 6)
5 print((30 - 2 * 6) / 3)
6 print(17 / 3)
7 print(17 // 3)
8 print(17 % 3)
9 print(2 ** 5)
```

# Hello, Python! 4

- 코드에 주석을 남길 수 있다
- # (한줄 주석)
- """ ... """ (여러줄 주석)

```
1 print("Hello world 1!")
2 # print("Hello world 2!")
3
4 """
5 이것도
6 마찬가지로
7 """
```

```
Hello world 1!
```