

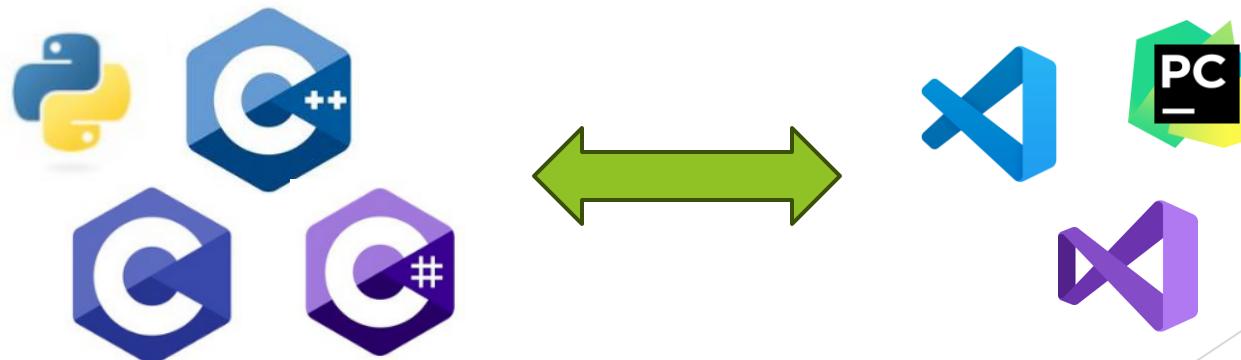
환경구성

환경(Environment)란?

- ▶ 환경이라 함은 우리는 인간 주변의 자연, 상태를 의미하는 단어입니다.
- ▶ 프로그래밍에서도 이 환경이라는 단어가 존재합니다.
- ▶ 간단하게 설명하면 '프로그래밍을 하기 위해 존재하는 통합 개발 환경, 프로그래밍 언어, 프레임워크/라이브러리를 아우르는 것'을 환경이라고 부릅니다.
- ▶ 그러나 우리는 이 '통합 개발 환경'과 '프로그래밍 언어', '프레임워크/라이브러리'에 대한 이해가 수반되어야만 이 환경을 이해할 수 있습니다.
- ▶ 때문에 환경 자체는 '우리가 프로그래밍 하기 위해 준비한 모든 것들, 컴퓨터 내부에 설치된 OS를 포함한 소프트웨어 전반'을 환경이라고 이해하시면 됩니다.
- ▶ 환경 자체는 이제 이해했으니 이를 더 간결하게 설명하기 위한, '통합 개발 환경'과 '프로그래밍 언어', '프레임워크/라이브러리'에 대해서 알아보겠습니다.

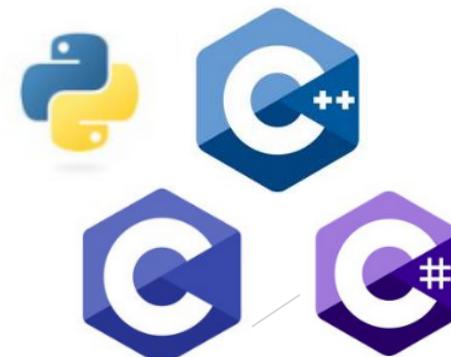
통합 개발 환경(Integrated Development Environment:IDE)

- ▶ 우선 'IDE'는 그 존재만으로 환경을 의미하는 것은 아닙니다.
- ▶ 그러나 이것들이 우리가 컴퓨터에서 환경이라고 부르는 것들을 아우를 수 있기 때문에 이를 부를 때 IDE라고 부릅니다.
- ▶ 아마 개발 공부 초기단계에는 흔히 하는 실수 중 하나가 바로 이 IDE와 프로그래밍 언어를 구분하는 것일 겁니다.
- ▶ 왜냐하면 프로그래밍 언어를 다루기 위해서 이 IDE를 설치하는 것을 항상 추천하기 때문에 IDE가 곧 프로그래밍 언어라고 착각하는 경우가 있습니다.
- ▶ 그러나 이 둘은 명확히 다릅니다.
- ▶ 그렇다면 무엇이 이 차이를 만들까요?



프로그래밍 언어(Programming Language)

- ▶ 우선 프로그래밍 언어에 대해서 이해할 필요가 있습니다.
- ▶ 프로그래밍 언어는 말그대로 '언어'입니다.
- ▶ 우리가 미국사람과 소통하기 위해서 써야하는 영어나 우리나라 사람들과 대화하기 위해 사용하는 한국어가 이에 해당합니다.
- ▶ 즉, 컴퓨터와 소통하기 위해 만들어진 언어입니다.
- ▶ 우리는 이 컴퓨터 언어로 컴퓨터와 상호작용하며 컴퓨터에 명령을 내리면 그 명령에 맞게 컴퓨터가 기능을 수행합니다.
- ▶ 그런데 왜 여러 종류의 언어가 존재할까요?
- ▶ 이는 사투리 혹은 집단별 은어와 같은 차이라고 생각하시면 됩니다.
- ▶ 같은 말 '그 애가 가씨였어?'라는 같은 말도 '가가 가가기'라는 경상도 사투리로 할 수 있듯이 같은 의미를 어떻게 명령내리고 사용할지의 차이로 생긴 것입니다.
- ▶ 물론 컴퓨터 쪽에서는 훨씬 복잡한 이유가 있습니다.

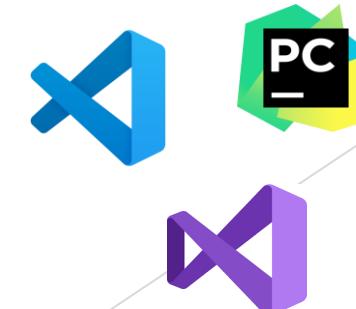


프로그래밍 언어(Programming Language)

- ▶ 이런 프로그래밍 언어들을 사용하기 위해서는 각 언어가 제공하는 방식으로 접근해야 합니다.
- ▶ 다만, 프로그래밍 언어들이 제공하는 기본 사용방식은 매우 '가볍다'라는 특징을 가지고 있습니다.
- ▶ 즉, 최소한의 기능을 가진 형태로 작동하기 때문에 실제 개발을 더 능률적으로 하기 위해 설정이 되어 있진 않습니다.
- ▶ 그렇기 때문에 실제 개발자들은 해당 언어를 더 능률적으로 작업하기 위해서 개발이라는 특수한 목적을 수행하기 위해 만들어진 프로그램을 사용합니다.
- ▶ 이 프로그램은 코드를 수정, 삭제하기 용이하며 작업한 파일들을 더 효과적으로 관리하고 한눈에 볼 수 있게 하며, 다른 서드파티 프로그램과 붙여 능률을 높일 때 더 쉽게 만들어줍니다.
- ▶ 그리고 이러한 개발을 위한 프로그램을 우리는 Integrated Development Environment, IDE, 통합 개발 환경이라고 부릅니다.

통합 개발 환경(Integrated Development Environment:IDE)

- ▶ 이런 개발이라는 특수한 목적에 고효율성을 위해서 통합적인 개발 도구와 기능을 미리 구성해 놓고 제공해주는 것이 IDE입니다.
- ▶ 이들은 더 쉽게 개발을 할 수 있게 하고, 문제를 더 쉽게 발견하고 수정하며, 개발 이외의 신경쓸 부분을 신경을 쓰지 않도록 만들어 줍니다.
- ▶ 즉, 환경 그 자체를 관리하고 문제가 없도록 만들어줍니다.
- ▶ 또한, 여러 다른 서드파티 프로그램들을 복잡하지 않게 연결을 도와줌으로써 개발 속도를 높여줍니다.
- ▶ 아마 Linux 개발을 쭉 해본 사람들은 알 것입니다.
- ▶ 개발의 모든 부분을 일일이 신경쓰고, 모든 명령어를 terminal에 일일이 적어가면서 개발을 하는 것이 얼마나 손이 많이 가는지.
- ▶ 그런 모든 신경쓸 부분을 모두 모아두고 한꺼번에 관리할 수 있게 하는 역활이 바로 IDE입니다.



IDE와 프로그래밍 언어의 비유

- ▶ 이를 현실에 비유하자면 한국어라는 프로그래밍 언어를 공부하기 위한 참고 서적이 IDE인 셈입니다.
- ▶ 이 참고 서적은 종류가 여러 종류가 있을 수 있습니다.
- ▶ 한국어는 한 종류이고 그 규칙은 이미 정해져 있기 때문에 하나의 규칙을 가지고 있지만 그 규칙을 한 책에 모두 담는 것은 실질적으로 어렵습니다.
- ▶ 그렇기 때문에 각 참고서적들은 자신이 원하는 스타일에 맞게 한국어를 공부할 수 있게 하고, 또한 공부하고자 하는 학생들의 수준에 맞게 정보를 차등 제공합니다.
- ▶ 그럼으로써 우리는 한국어를 더 빨리, 효율적으로 배울 수 있습니다.
- ▶ 또 경우에 따라서는 여기저기 퍼져있는 정보를 참고서적에 모아두어 한눈에 볼 수도 있으니 도움이 될 것입니다.
- ▶ 때문에 IDE와 프로그래밍 언어를 별개이면서 서로 연관이 깊은 존재임을 인지 해야 합니다.

프레임워크와 라이브러리

- ▶ 이제 IDE와 프로그래밍 언어에 대해 알아보았으니 환경을 설명하기 위한 마지막 프레임워크와 라이브러리에 대해서 알아보겠습니다.
- ▶ 이 둘은 크게 보면 서로 같아 보이지만 세부적으로는 서로 다른 의미를 가지고 있습니다.
- ▶ 우선 둘의 역할은 '이미 누군가가 만들어둔 기능'입니다.
- ▶ 즉 프로그래밍 언어를 다룰 때, 그냥 내가 처음부터 만드는 것이 아니라 누군가가 만든 기능을 가져와 사용함으로써 굳이 바닥부터 설계하고, 힘들게 개발하는 것이 아니라 가져와 사용하는 방식을 통해서 난이도를 낮추고 좀 더 복잡한 업무에 집중할 수 있도록 만듭니다.
- ▶ 마치 집을 짓기 위해서 망치가 필요하지만 전자동 망치를 통해서 더 쉽게 건물을 짓고, 망치를 치는 것에 집중할 시간을 줄임으로서 다른 곳에 더 집중하여 개발을 도와주는 역할을 합니다.
- ▶ 그렇다면 둘의 차이는 어떨까요?

프레임워크(Framework)

- ▶ 프레임워크는 기능의 덩어리이면서 동작의 제어를 프레임워크 자체가 통제합니다.
- ▶ 즉, 프레임워크라 함은 프레임워크라는 기능적 집합을 이용하여 기능을 가져와 사용하지만 동작의 전체적인 구조와 규칙은 프레임워크가 지정한 방식을 이용해야 하며, 모든 동작 또한 프레임워크가 제공하는 바를 통해서만 작동을 할 수 있다는 특징이 있습니다.
- ▶ 즉, 음식으로 비유하자면 '밀키트'와 같습니다.
- ▶ 이미 조리법이나 사용할 기본 재료 등을 제공하지만, 필요한 추가 소스나 양념, 추가 재료들을 이용하여 음식을 할 수 있습니다.
- ▶ 하지만 이러한 밀키트의 경우에는 이미 '어떤 요리가 될 것'이라는 것은 정해져 있습니다.
- ▶ 그렇기 때문에 쉽게 요리를 고급진 형태로 만들 수 있지만, 우리는 그 틀에서 벗어난 음식을 만들기는 어렵습니다.
- ▶ 프레임워크는 이처럼 만들 구조와 규격은 정해져있으며 고급 기능을 양질로 제공하지만 프레임워크의 규칙 내에서 제작해야한다는 특징을 가집니다.

라이브러리(Library)

- ▶ 라이브러리는 프레임워크와 다르게 정해진 구조와 규격을 제공하지 않습니다.
- ▶ 라이브러리는 그 자체가 어떤 기능을 가지고 있고 이를 가져와 사용할 수 있습니다.
- ▶ 때문에 사용자는 자신이 원하는 프로그램을 원하는 구조로 만든 곳에 사용하더라도 프레임워크와 다르게 크게 제약받지 않고 사용할 수 있습니다.
- ▶ 하지만 그만큼 고급 구조를 가지고 있지 않고 가장 기본적인 형태를 띠고 있기 때문에 어느정도 만들어진 구조의 프로그램을 만들려면 사용자가 이 전반적인 모든 지식을 가지고 있어야 합니다.
- ▶ 즉, 음식으로 비유하자면 라이브러리는 '이미 손질된 재료'에 해당합니다.
- ▶ 이미 손질된 재료를 바로 활용할 수 있지만, 결국 이를 요리하려면 우리는 그 요리의 레시피와 재료를 모두 알고 있어야 하고, 이를 위해서 필요한 다른 재료들 또한 모아야 합니다.
- ▶ 라이브러리는 이처럼 빠르게 사용자의 프로그램에 쉽게 기능을 추가할 수 있지만 거대한 구조를 기반하지 않기 때문에 모든 책임은 사용자가 진다는 특징이 있습니다.

프레임워크와 라이브러리가 의미하는 바

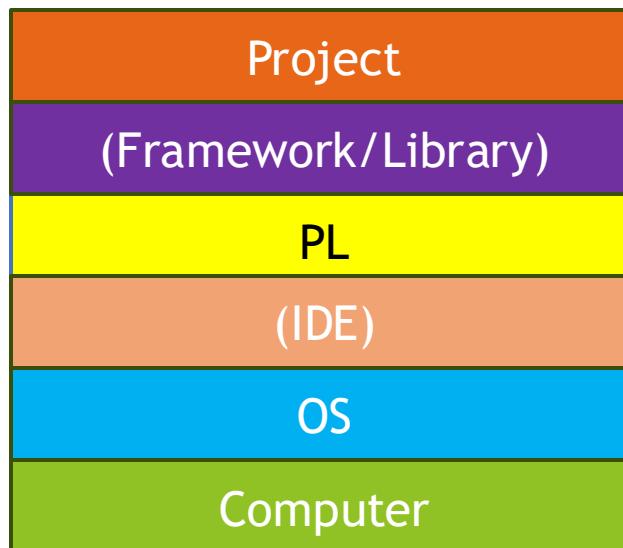
- ▶ 더불어 프레임워크와 라이브러리는 프로그래밍 언어별로 따로 존재합니다.
- ▶ 왜냐하면 이들은 결국 그 언어의 코딩 조각들이기 때문입니다.
- ▶ 물론 유명한 프레임워크나 라이브러리들은 서로 다른언어로 버전이 따로 존재하기 때문에 사용자가 용이하게 사용할 순 있습니다.
- ▶ 결국 IDE 또한 프로그래밍 언어를 이용하여 개발을 할 때, 효율과 능률을 높이기 위해서 사용됩니다.
- ▶ 이처럼 프레임워크와 라이브러리 또한 우리가 개발할 때, 바닥부터 개발하는 것이 아닌 이미 만들어진 형태의 코드와 기능을 가져와 활용함으로서 기본 틀과 완성된 파츠를 이용하여 더 쉽게 어려운 프로그래밍 구조를 개발하는 것에 그 의미를 두고 있습니다.
- ▶ 때문에 우리는 이러한 IDE, 프로그래밍 언어, 프레임워크/라이브러리를 준비함으로서 우리 원하는 개발 프로젝트를 수행할 환경을 조성할 수 있습니다.

환경 구성

- ▶ 이제 이러한 환경을 구성하여 본격적인 개발 프로젝트를 수행하게 될 것입니다.
- ▶ 그러나 이런 환경을 구성할 때 다음과 같은 문제를 겪게 됩니다.
- ▶ 독립적인 프로그램으로서 환경을 다루고 프로그래밍 언어를 다루는 **IDE**의 경우에는 어떤 프로그램을 이용하는지만 선택한다면 하나로 모든 프로젝트와 호환되어 사용할 수 있다는 장점이 있으나 다른 것들은 어떤 언어, 어떤 프레임워크, 라이브러리를 이용하는지 따라 큰 문제를 가지게 됩니다.
- ▶ 이를 쉽게 설명하기 위해 다음과 같이 만들어보겠습니다.

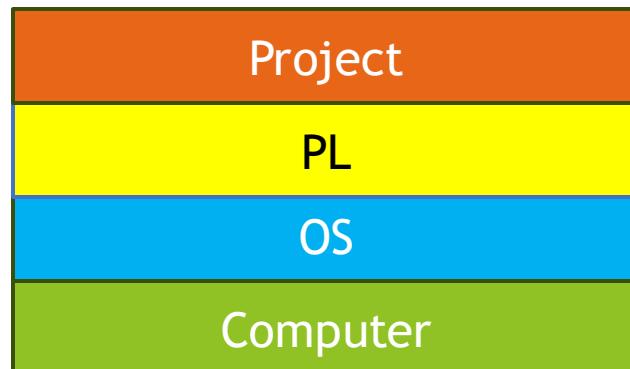
환경 구성

- ▶ 컴퓨터는 다음과 같은 구조적인 형태로 컴퓨터 상에서 작동합니다.
- ▶ 우선 컴퓨터와 직접 소통하고 작동하는 것은 프로그래밍 언어(**PL**)입니다.
- ▶ 이 때, **IDE**는 보조적인 역할을 하므로 실질적인 이 그림에서 없든 있든 크게 문제는 없습니다.
- ▶ 그러나 전체적인 환경을 다루는 학습 단계이기 때문에 차후엔 이들을 없이 표현하겠습니다.



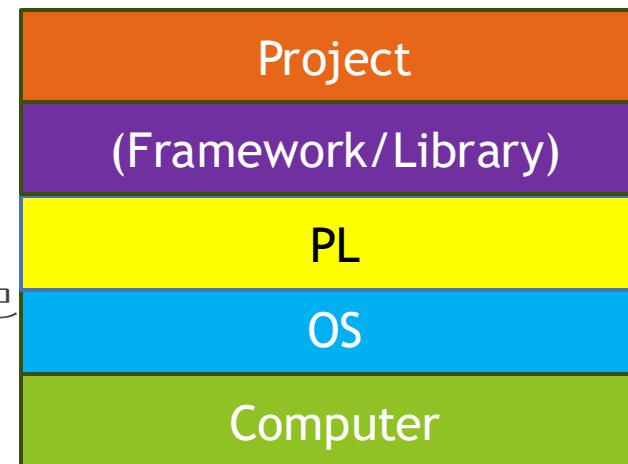
환경 구성

- ▶ 우리의 환경 구성의 목적은 어떤 프로젝트를 만들어내기 위하여 컴퓨터에 OS를 깔고 그 OS와 소통을 통하여 우리가 단순 명령어로 컴퓨터에 명령을 내릴 수 있는 PL을 설치하고 사용합니다.
- ▶ 그리고 PL을 통해서 작성된 프로젝트는 우리가 원하는 프로그램의 형태로 만들어지게 됩니다.
- ▶ 그런데 우리는 이런 환경에서 작업하기 위해서 여러가지 프레임워크와 라이브러리를 활용하려 합니다.
- ▶ 그럼 우리는 이 PL위에 여러 프레임워크와 라이브러리를 설치하게 될 것입니다.



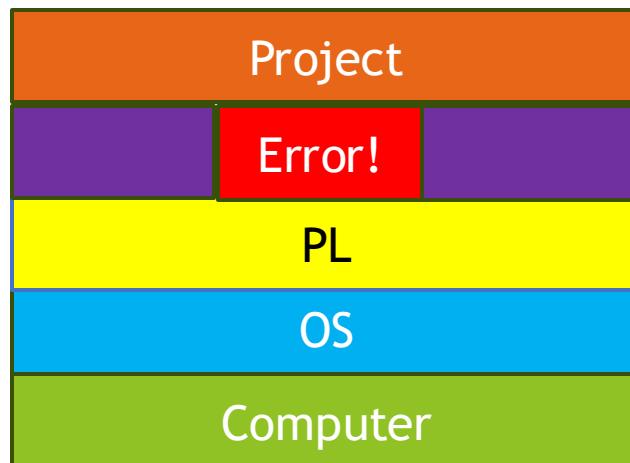
환경 구성

- ▶ 이렇게 하면 우리는 어떤 컴퓨터를 이용하여 프로그램을 만들 수 있습니다.
- ▶ 그러나 개발을 진행해보면 알겠지만 우리는 한 컴퓨터로 다수의 프로젝트를하게 됩니다.
- ▶ 굳이 매번 다른 프로젝트를 위해서 컴퓨터를 구매할 필요는 없습니다.
- ▶ 그러나 이미 이 컴퓨터는 문제가 있습니다.
- ▶ 우리가 이전 프로젝트를 작업하기 위하여 환경구성을 만들어놓았다는 것입니다.
- ▶ 사용하기 위해서는 설치를 해둬야 하고, 우리는 이미 설치가 된 환경에서 작업해야 합니다.
- ▶ 물론 우리가 똑같은 프로젝트를 한다면 좋겠지만, 대개 다른 프로젝트를 하게 될 거고, 프로젝트마다 필요로하는 프레임워크와 라이브러리가 다릅니다.
- ▶ 그러니 우리는 불필요하게 프레임워크나 라이브러리 사용할 수 밖에 없는 문제를 겪게 됩니다.



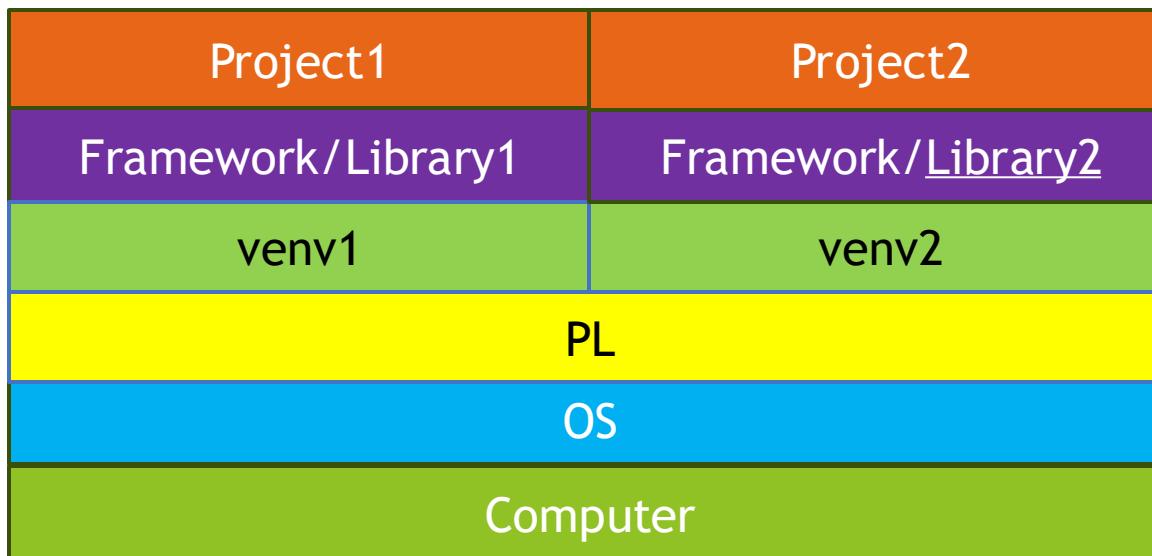
환경 구성

- ▶ 하지만 그런 방법은 썩 좋은 아이디어가 아닙니다.
- ▶ 특히, 불필요한 프로젝트가 있거나, 필요에 의해서 설치한 다른 프레임워크와 라이브러리가 그 존재만으로 서로 충돌한다면 우리는 다음 프로젝트를 시작하기도 전에 난관에 부딪치게 될 것입니다.
- ▶ 당연하게도 프레임워크와 라이브러리 또한 도움을 위해 존재하지만 이들끼리 복잡하게 얹힐 수록 예측하기 어려운 문제를 야기할 수 있다는 문제가 있습니다.
- ▶ 때문에 우리는 이를 '독립적'으로 관리할 필요가 있습니다.
- ▶ 이를 위해 우리는 '가상환경'이라는 개념을 이용할 수 있습니다.



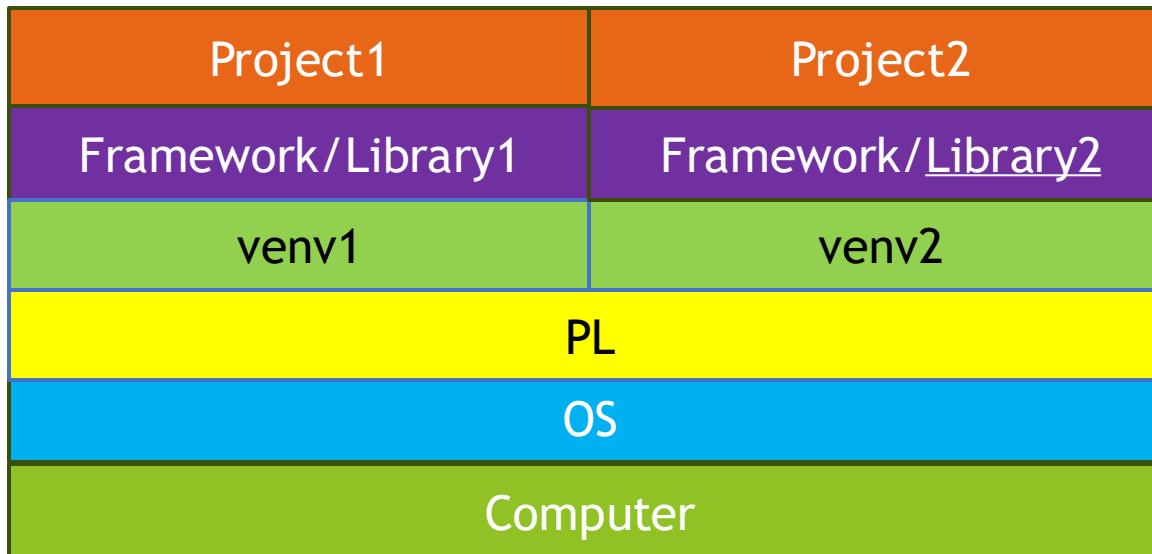
가상환경(Virtual Environment)

- ▶ 가상환경은 기존 프로그래밍 위에 또다른 환경을 만들 수 있도록 합니다.
- ▶ 하지만 이 환경은 실제 환경이 완전히 다시 조성되는 것은 아닙니다.
- ▶ 이 모든 시스템은 **OS**와 프로그래밍 언어에 묶여있어 기존 시스템의 흐름에 다른 흐름을 만들진 않습니다.
- ▶ 하지만 가상환경은 이 흐름을 '독립적'으로 작동하도록 묶어줍니다.
- ▶ 이렇게 함으로서 우리는 프로젝트마다 독립적인 환경을 제공해 줄 수 있습니다.



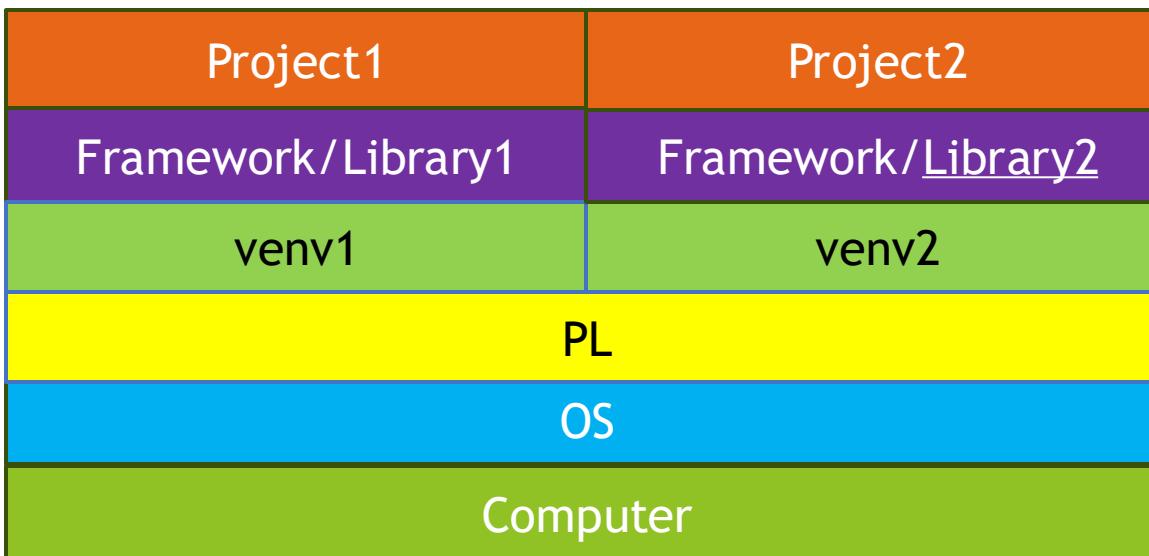
가상환경(Virtual Environment)

- ▶ 즉 가상환경은 일종의 '컨테이너'입니다.
- ▶ 우리가 원하는 프레임워크와 라이브러리와 같은 개발환경을 독립적으로 관리할 수 있게 하면서도, 전체적인 프로그래밍 언어는 그대로 사용합니다.
- ▶ 이는 용량 측면에서도 매우 효율적입니다.
- ▶ 독립적인 환경을 구성해주기 위해서 동일한 프로그래밍 언어를 독립적으로 설치하여 관리하는 것은 일관성에서도, 용량면에서도 비효율적인 일이니까 말입니다.



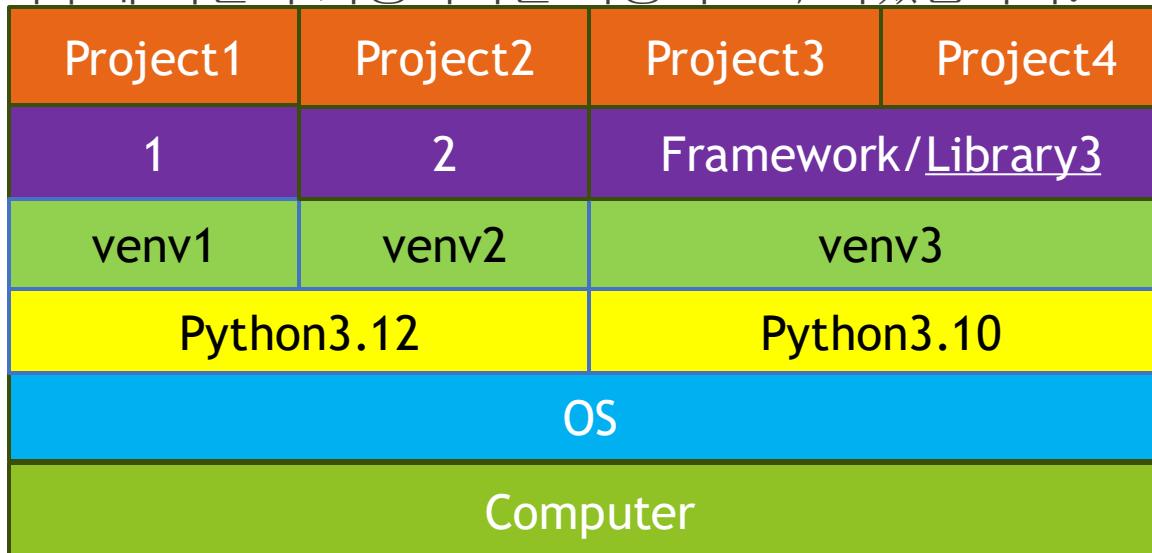
가상환경(Virtual Environment)

- ▶ 아래와 같이 프로그래밍 언어 위에 가상환경이 존재함으로서 모두가 같은 언어로 만들어진 프로젝트이지만 서로가 다른 프로젝트이며 이를 위해 구성된 환경 또한 독립적으로 관리되어 과하거나 부족한 환경을 겪을 필요가 없습니다.
- ▶ 다만 그 환경을 다루기 위한 기술이 필요하지만 프로그램이 커지면서 겪게될 문제보다는 오히려 쉬운 방법입니다.
- ▶ 이 방법은 차후에 여러 언어를 다룰 때도 유리합니다.



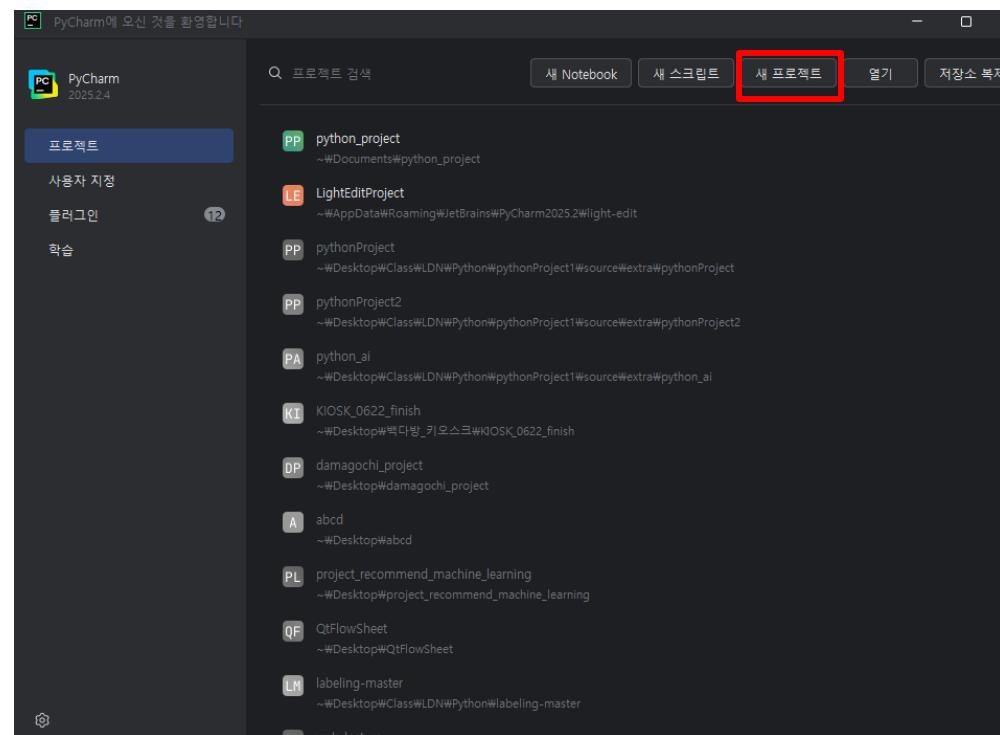
가상환경(Virtual Environment)

- ▶ 굳이 멀리 갈 필요 없이 Python 버전 3.12와 3.10을 동시에 다루게 된다면 어떨까요?
- ▶ 우리는 이런 경우에도 가상환경을 구분하여 사용하면 훨씬 독립적으로 관리할 수 있습니다.
- ▶ 이런 방식을 통해 차후엔 이 구성 자체를 하나로 '컨테이너'화 하여 관리하는 방법인 'Docker'를 이용할 수 있습니다.
- ▶ 물론 이는 차후에 시간이 가용하다면 사용하도록 하겠습니다.



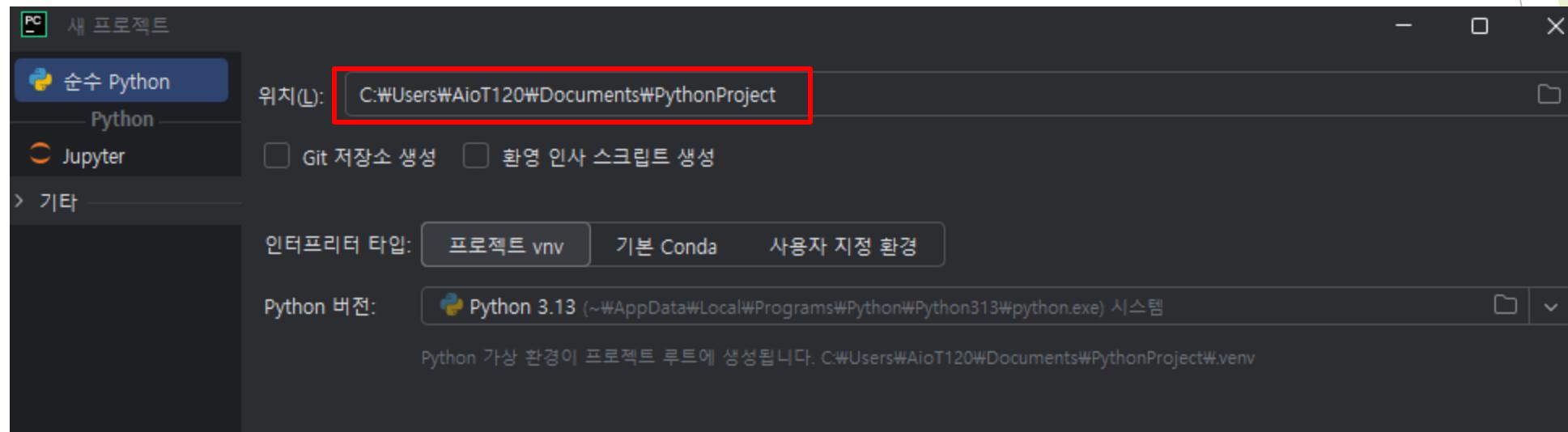
IDE에서 가상환경 만들기

- ▶ 그럼 우리가 Python 가상환경을 Python IDE인 Pycharm을 이용하여 만들어보겠습니다.
- ▶ IDE는 개발에 필요한 여러 어려운 작업을 쉽게 해주기 위해 존재하기 때문에 다음을 따라가면 손쉽게 가상환경을 만들 수 있습니다.
- ▶ 다음에서 '새프로젝트'를 선택하여 가상환경 생성을 시도합니다.



IDE에서 가상환경 만들기

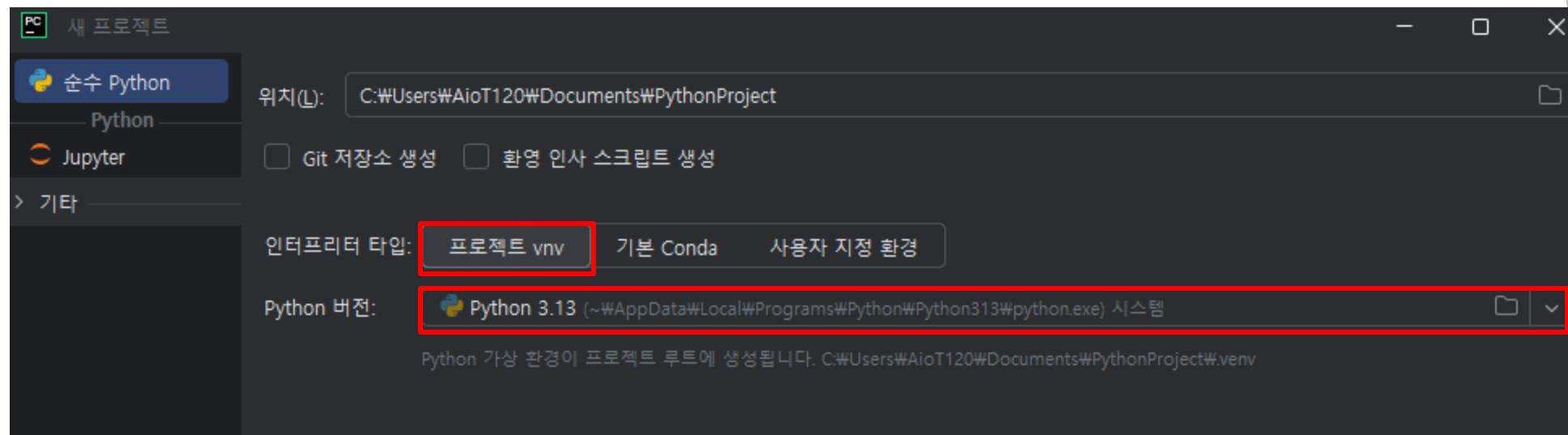
- ▶ 가상환경은 프로젝트 폴더 단위로 관리되므로 우선적으로 어디에 가상환경으로 만들어진 프로젝트를 생성할건지 묻습니다.
- ▶ 여러분이 관리하고 계속 접근할 위치를 선정하고 폴더를 생성합니다.



- ▶ 빨간 표시가 된 곳을 임의로 수정하거나 해당 위치를 기억해둡니다.

IDE에서 가상환경 만들기

- ▶ 기타 다른 설정들이 존재하지만 지금은 가상환경을 만드는 것 자체에 목적을 두고 있으므로 다음을 지정합니다.
- ▶ 가상환경 생성을 지정하고 이 가상환경이 가질 Python의 버전을 지정합니다.
- ▶ IDE의 기능으로 사전에 설치된 Python 버전들이 있다면 그 중 가장 최신인 버전이 선택될 것입니다.
- ▶ 이번에는 3.12버전을 선택하겠습니다.



IDE에서 가상환경 만들기

- ▶ 이후 기본 선택이 끝나면 생성을 누릅니다.
- ▶ 그렇게 되면 기본 환경을 만들기 위한 것들을 설치하고 해당 폴더에 가상환경이 구성되고 만들어지게 됩니다.
- ▶ 이렇게 가상환경을 만들어 프로젝트별로 관리하는 것은 개발자로서의 시작점이 라는 생각으로 항상 자신의 개발 환경에 신경써야합니다.

