

# 강의 1-1

# 온보딩, 의사 코드 기반 코딩

Python 입문 과정 3주 프로그램

한날, 2024-03-04, copyright RealWorldPudding all rights reserved.



PuddingCamp

<https://pudding.camp>

# 캡프 소개



# 푸딩캠프

The proof of the **pudding** is in the eating.



- 백문이 불여일견
- 학습자 자신의 학습법을 구축하도록 코칭하여, 배우고자 하는 주제를 효과적으로 학습하도록 돋는 캠프
- 2024년 4월 초 정식 출시 예정
- 3월 체험 프로그램은 이례적인 이벤트



# 캠프 프로그램



PuddingCamp

<https://pudding.camp>

# 전문가가 되는 4가지 방법 : 인식, 인지

1만 시간만으로는 전문가가 되지 못한다?

- 피드백이 있는 반복
- 타당한 환경
- 적절한 시점에 이뤄지는 피드백
- 의도적인 반복

<https://youtu.be/yiG0n0K7woU>



# 프로그램에서 기대할 수 있는 것

## Python 중급이나 라이브러리 활용할 단계에 입문하기

- 기대
  - 학습 노트 관리기 동작에 필요한 요소를 Python으로 실습하고 구현
  - Python 코드 이해
  - Python으로 코딩할 기반 다지기
- 욕심 : 학습 노트 관리기 구현
  - 구현체를 엮어 실제 동작하는 학습 노트 관리기 구현
  - 머신러닝, A.I.를 적용한 키워드 추출, 문서 요약 기능



# 프로그램에서 기대할 수 없는 것

- Python 전문가 되기
- 프로그래머 전직 성공
- 포트폴리오 마련
- 참가비에 대한 이자



# 프로그램 커리큘럼

진행 상황에 따라 추가로 다루는 컨텐츠는 따로 명시 안 함

- 1-1 : 의사 코드 기반 코딩, 테스트 자동화
- 1-2 : 함수-1, 객체와 객체 지향-1
- 2-1 : 기본 문법, 내장 자료형, 흐름 제어)
- 2-2 : 이름 공간, 모듈과 패키지
- 3-1 : 주요 내장 모듈과 패키지, 예외 처리
- 3-2 : 함수-2, 객체와 객체 지향-2



# 캠프 프로그램 주간 일정

월	화	수	목	금	토	일
대뜸 퀴즈	복습 퀴즈 배포	과제 제출 마감 	대뜸 퀴즈	복습 퀴즈 배포	과제 제출 마감 	팀 회고 제출 마감 
복습 퀴즈 풀이			복습 퀴즈 풀이			학습 노트 제출 마감 
라이브 강의			라이브 강의			학습자 공유회
과제 출제			과제 출제			코칭 데이 1 (피드백)
						코칭 데이 2 (이 주의 컨텐츠)



# 디스코드 채널 이용 안내

- **#python-beginner-pre** : 본 프로그램의 참가자 전원의 공간. 공지 등 알림 사항은 이곳에 올라갑니다.
- **#community-rules** : 푸딩캠프의 디스코드 서버를 이용하기 위한 규칙/행동강령을 안내합니다.
- **#introduce** : 자기소개 채널입니다. 함께 학습 여정을 떠날 동료들에게 여러분을 소개해보세요.
- **#Study Room** : 각 팀 별 보이스 채널입니다. 입장하면 바로 음성/영상 대화가 시작되며, 전화기 수화기 아이콘을 누르면 종료합니다. 팀 별로 자유롭게 이용하세요.
- **#Classroom** : 강의가 이뤄지는 채널입니다.
- **#homework-help** : 과제에 대한 문의나 도움을 요청하는 곳. 학습 주제에 대한 Q&A 아니예요.
- **#모래상자** : 디스코드가 생소하시다면 이 채널에서 부담없이 테스트하세요.



# 학습 노트 작성 방법

- 개인 별 학습노트 폴더에 저장합니다.
- “학습 노트 템플릿”을 복사한 뒤 다음 형식으로 이름을 지으세요.
  - 주차-회차 : 1-1, 1-2, 2-1, 2-2, 3-1, 3-2
- 학습 노트 템플릿은 권장하지만, 짹꿍(팀 동료)들의 피드백을 담는 내용이 있으면 어떤 양식이든 무방합니다.
- 각 학습자는 팀의 동료들 모두의 학습 노트에 피드백을 남기고, 받습니다.
  - 팀의 마감시점을 정하면 좀 더 책임감이 들어 좋습니다.
  - 질문, 의견같은 피드백을 남기되, 정 남길 피드백이 없으면 칭찬이나 응원이라도 남겨주세요.

## 학습을 위해 제안하는 노트 작성 방식

- 강의 중에는 키워드나 문장만 간략히 적습니다.
  - 강의가 끝난 후 이를 기반으로 학습자 개인의 이해와 관점으로 설명을 채웁니다.
  - 틀리거나 제대로 이해하지 못해도 무방합니다. 이해하고 알고 있는 선에서 다른 이에게 설명하듯 내용을 채웁니다.
  - 코치는 학습 노트를 보고 학습자의 학습 상태를 파악하고, 이에 맞춰 멘토링과 피드백을 합니다.
- 녹취하듯 강의를 노트로 작성하는 건 추천하지 않습니다.
  - 강의를 듣지 못합니다.
  - 이해한 느낌에 속아서 학습 기회를 놓칩니다.
  - 강의 후 약식 버전으로 강의 슬라이드를 제공합니다.



# 실습 코드 관리 방법

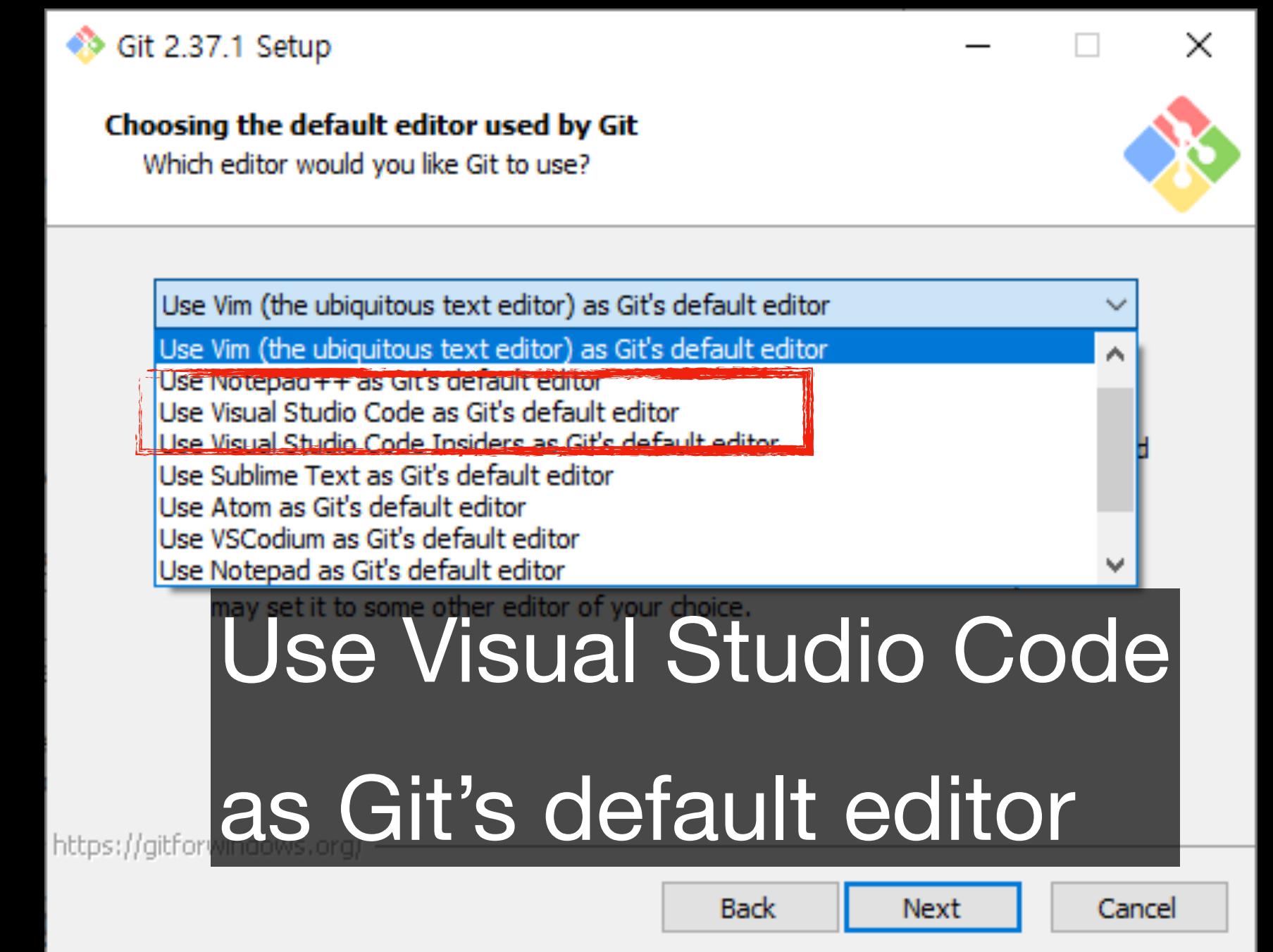
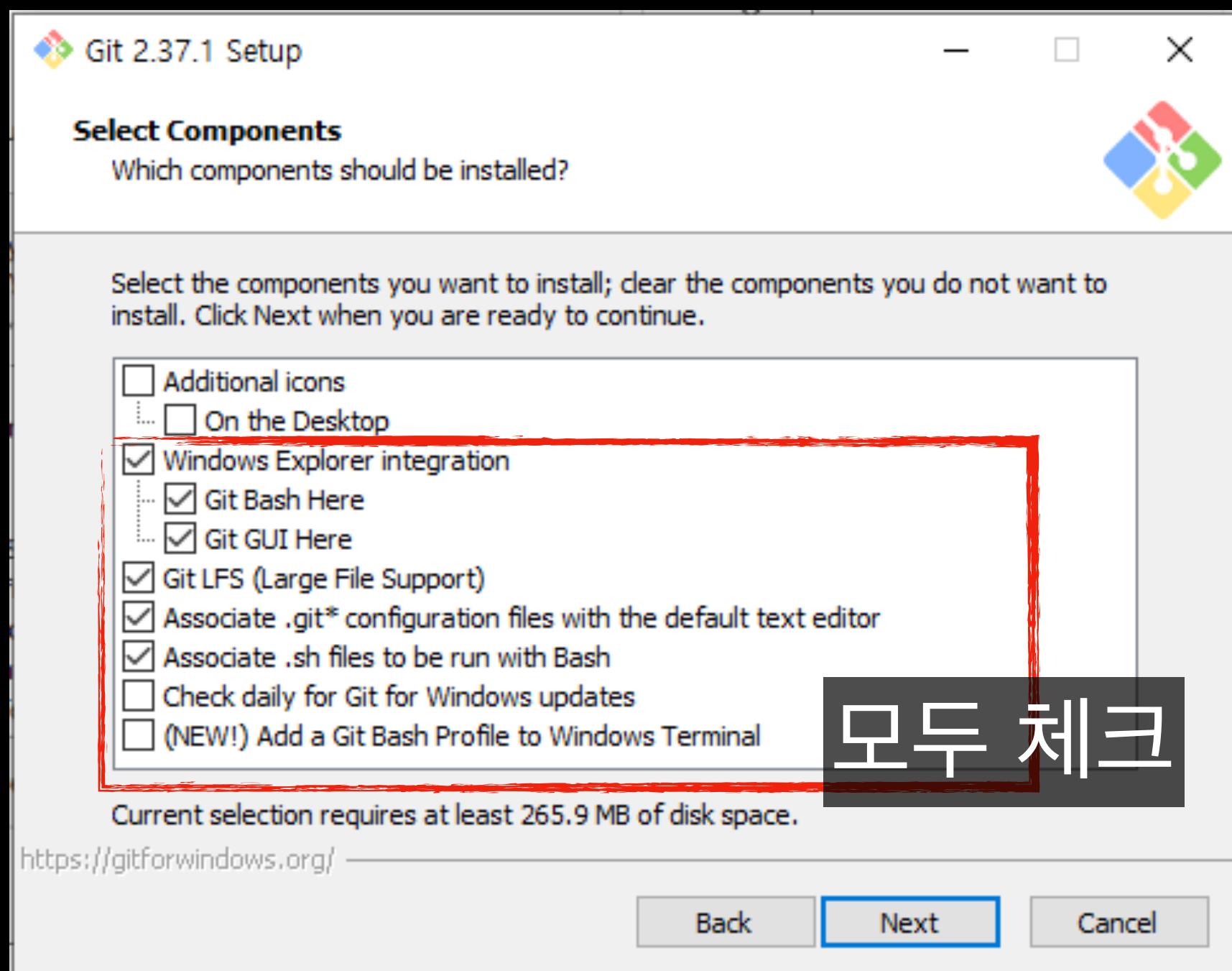
Git을 들어봤고 조금이라도 경험해보고 싶은 분 대상

- Git을 활용하여 코드 변경 이력을 남기기(commit)
- 각 커밋에 제목과 설명을 기록.
  - 복습할 미래의 자신에게 제공한다는 의도로 작성.
- 맥OS : Git이 내장되어 있어 설치 불필요. Git GUI 프로그램 설치로 바로 진행.
- 윈도우 : Git 설치 필요 후 Git GUI 프로그램 설치.



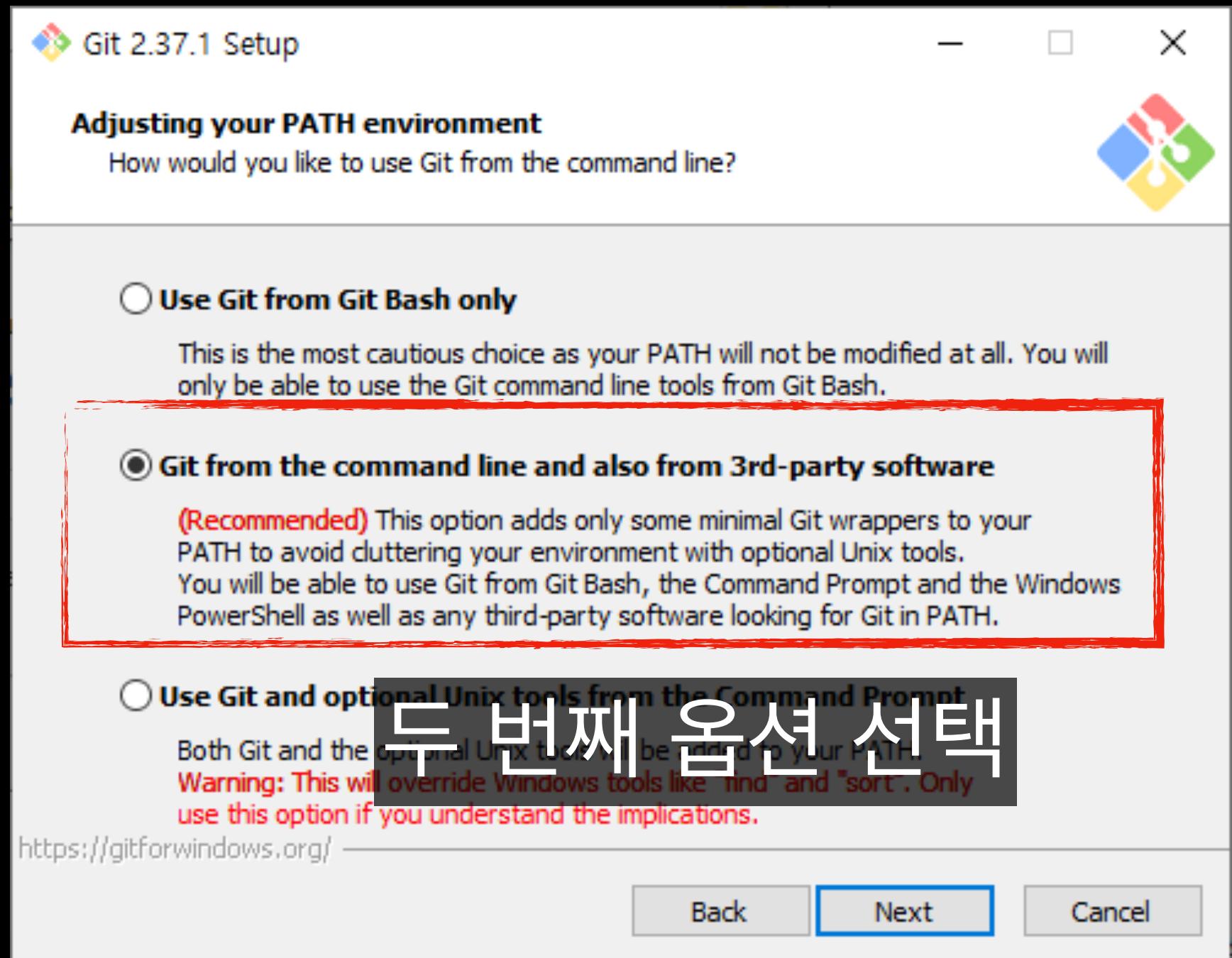
# 윈도우에 Git 설치

- 맥OS : 내장되어 있어서 설치 불필요.
- 윈도우 : <https://git-scm.com/download/win>에서 운영체제에 맞는 파일 다운로드한 후 실행.

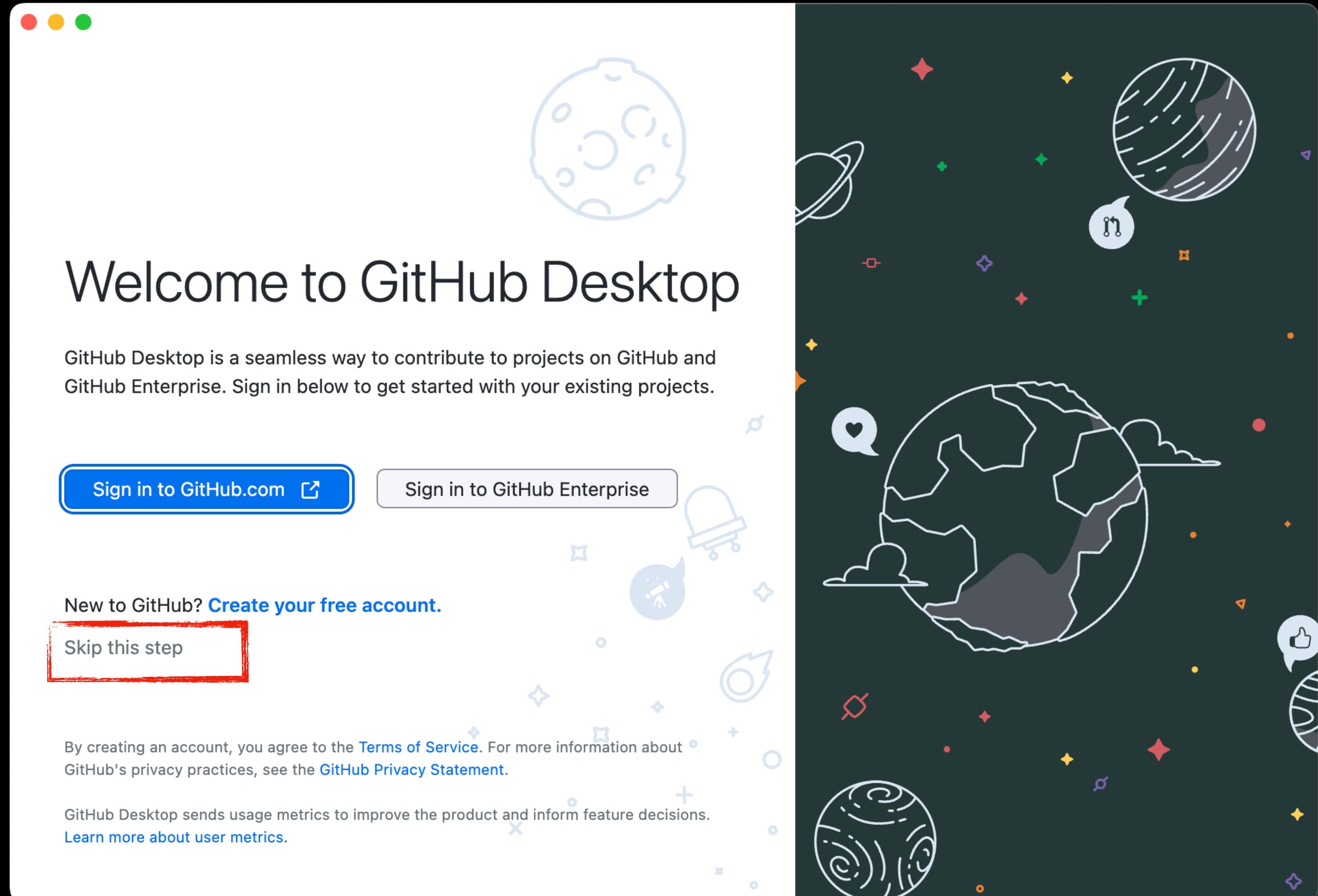


# 윈도우에 Git 설치

- 이외의 옵션은 기본 옵션을 사용.



# GitHub Desktop 사용

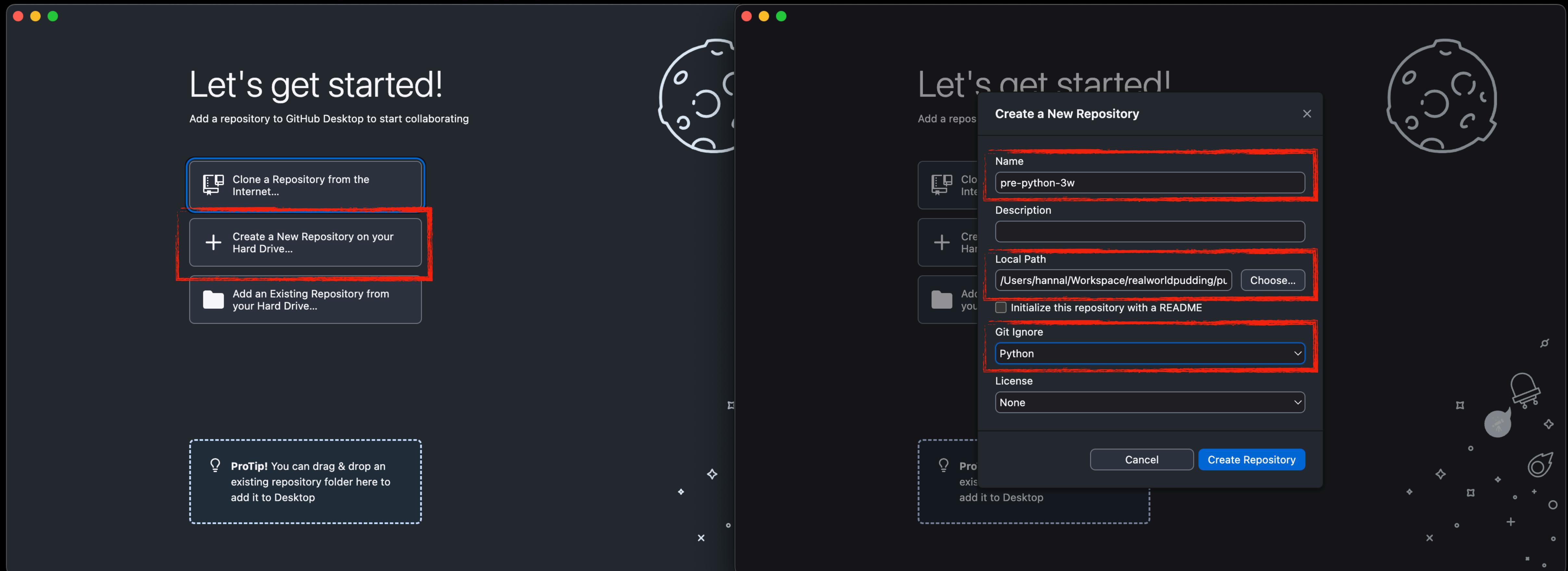


- Git GUI 프로그램 :  
[https://  
desktop.github.com/](https://desktop.github.com/)
- GitHub 계정 없으면  
Skip this step 클릭.



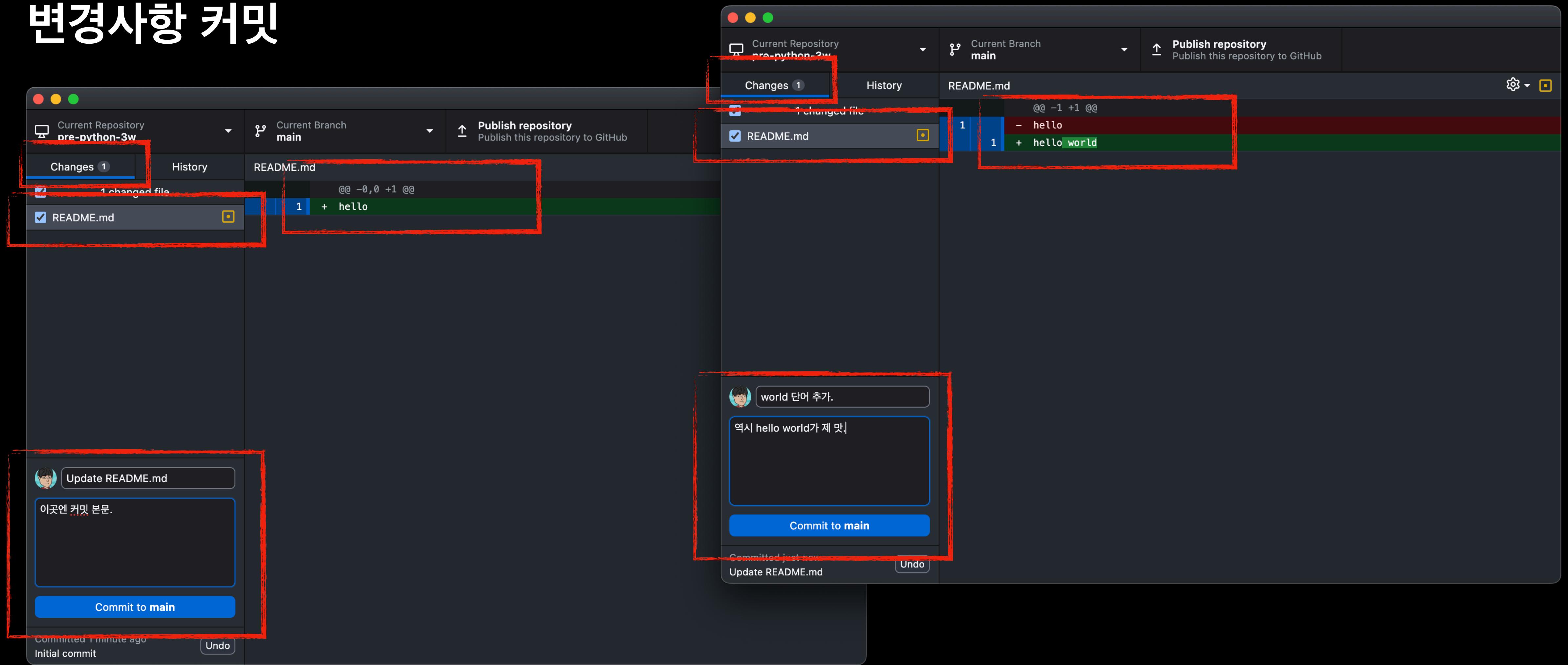
# GitHub Desktop 사용

## 기존 경로를 Git 저장소로 설정



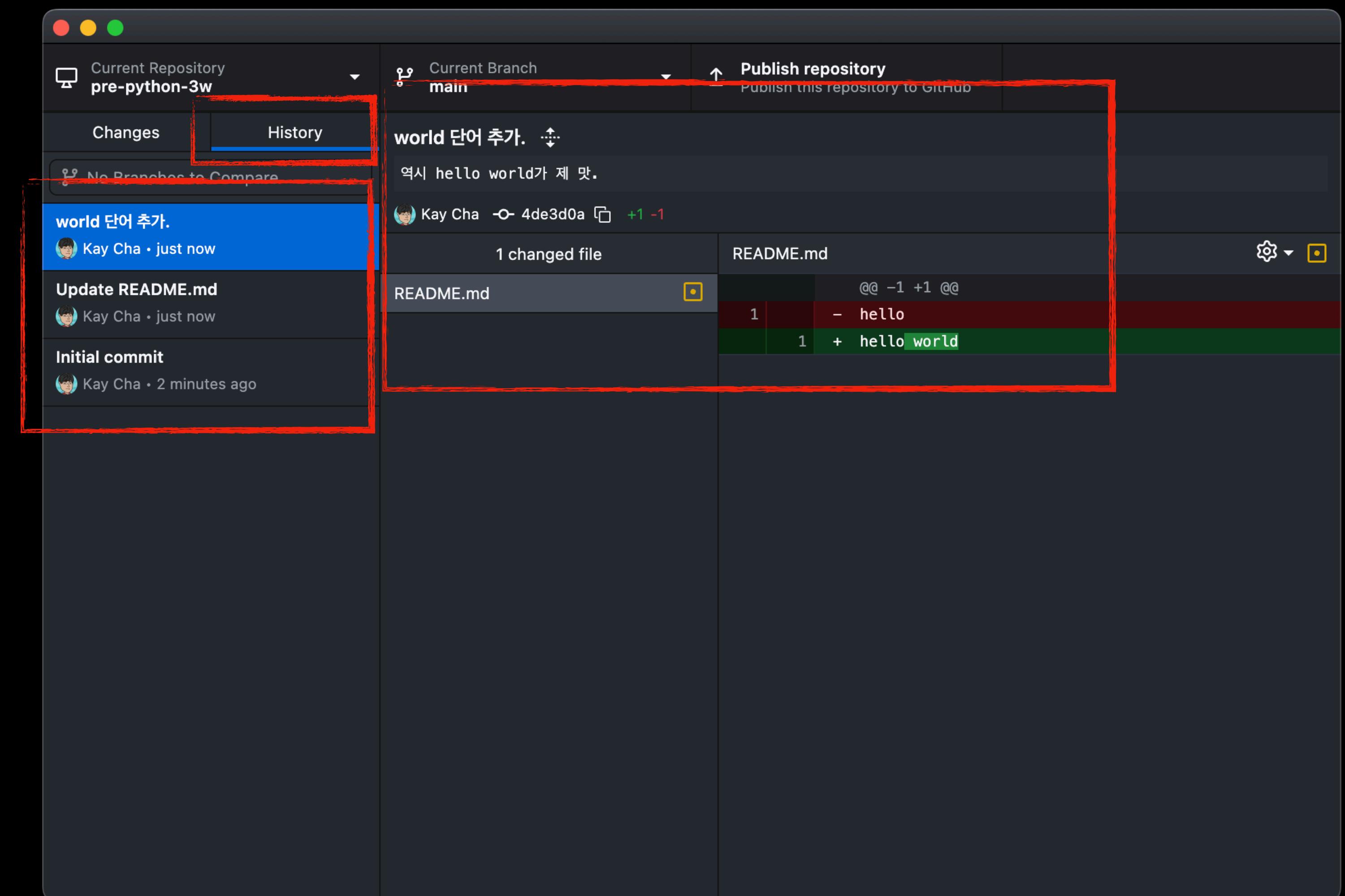
# GitHub Desktop 사용

## 변경사항 커밋



# GitHub Desktop 사용

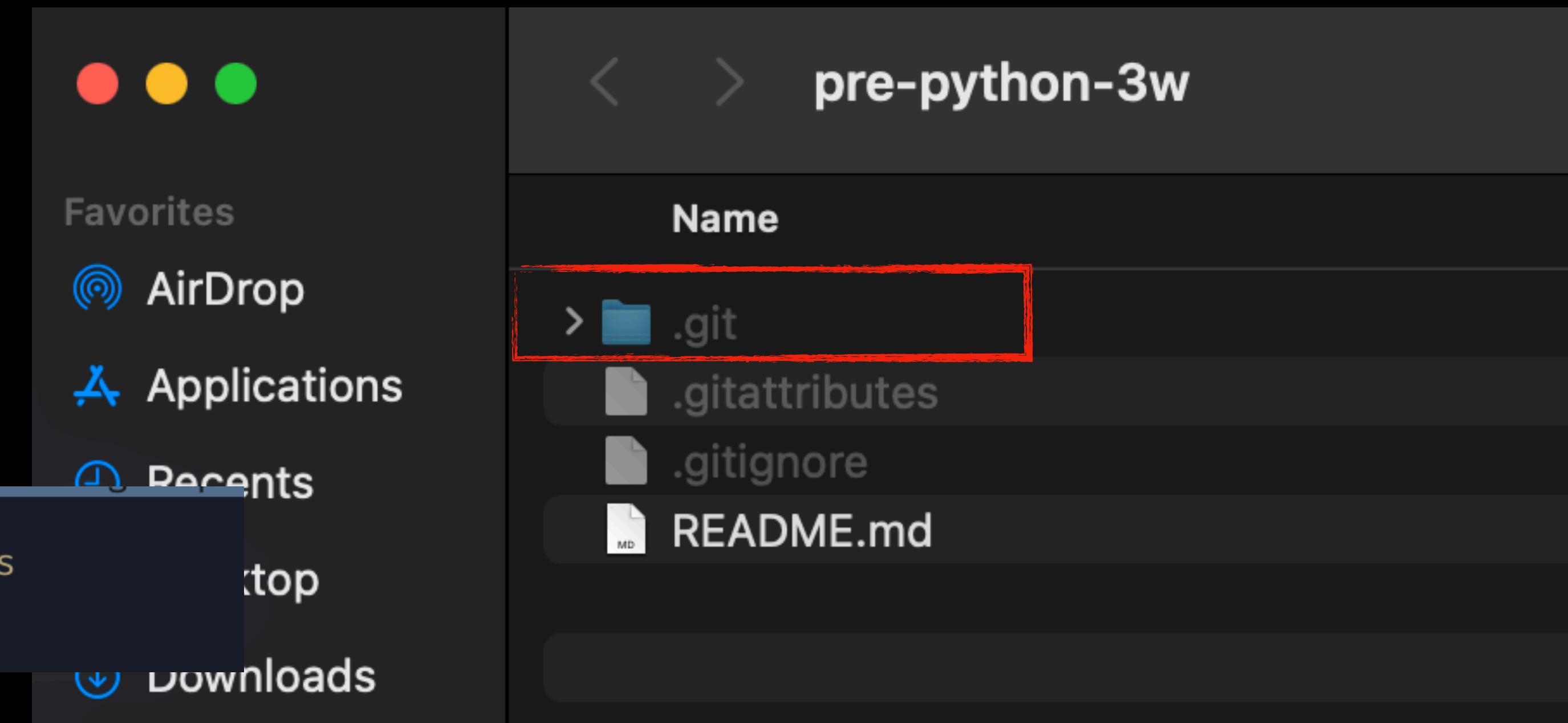
## 커밋 확인



# Git 저장소를 과제로 제출 가능

- .git 디렉터리를 통채로 압축해서 제출
- 코치가 변경 이력을 똑같이 볼 수 있으므로 학습자의 코드 작성 과정을 볼 수 있어서 상황과 맥락 이해에 도움이 됨.
- 학습자 여러분이 그렇게 하시면 좋다는 것이지 꼭 이렇게 하실 필요 없어요.

```
drwxr-xr-x@  - hannal 2 Mar 20:03 .git  
.rw-r--r--@ 66 hannal 2 Mar 20:01 .gitattributes  
.rw-r--r--@ 2.8k hannal 2 Mar 20:01 .gitignore  
.rw-r--r--@ 12 hannal 2 Mar 20:02 README.md
```



# 과제 제출 방법

- 팀 별 과제 폴더(구글 드라이브)에 제출
- “<이름>-과제코드”로 폴더 생성
  - 한날-20240307
  - 해당 폴더 안에 과제 제출.
  - 파일명 지켜주세요.



주상에서 구체로



# 추상화에서 구체화로

알아서 하는 건 인공지능이 또는 인간지능이 합니다.

- 문제 -> 문제 정의 -> 문제 이해 -> 구체화
- 아는 만큼 구체화가 가능한 역설
- 아는 것 vs 모르는 것 = 구체화 가능한 것 vs 구체화하지 못하는 것
- 인간 언어 - 고급(high level) 프로그래밍 언어 - 저급(low level) 프로그래밍 언어
- 개념과 언어를 추상 영역에서 구체 영역으로 동시에 전환하는 건 어렵고 힘든 게 당연.
  - 이런 작업을 이쪽 세계 용어로 뭐라고 부를까요?

[https://www.youtube.com/watch?v=cDA3\\_5982h8](https://www.youtube.com/watch?v=cDA3_5982h8) (1:00)



# 의사 코드 주도 코딩

개념과 언어 중 언어만이라도 추상 단계에서 접근하자

- 의지력을 동시에 두 가지에 쓰지 말고 하나라도 익숙한 걸로 접근해서 하나씩 부러뜨리기
  - 개념은 구체
  - 언어는 추상(자연어)



# 라면 끊이기

실습-20240304-1-1-1

- 개인 별 학습 노트에 접속하세요(구글 닉스).
- 한 번에 하나의 의미만 담는 짧은 문장 : 현재형, 조건문은 사용 불가.
- 상황 가정 : 글 이해만 가능한 어린 아이가 나열된 문장을 그대로 따라하여 라면 끊이면 성공.
- 무작위로 2인 선정하여 발표.
- 제한 시간 : 5분



# 유치원생, 푸딩캠프의 디스코드 서버에 초대하기

실습-20240304-1-1-2

- 짹수 팀 : PC Desktop 사용 기준.
- 홀수 팀 : 스마트폰 사용 기준.
- 키보드와 마우스, 스마트폰은 다룰 줄 알지만, 디스코드는 모르는 유치원생 3년 차.
- 현재형 문장, 조건문은 사용 불가.
- 2개팀 발표.
  - 자발적 발표 또는 무작위 선정.
- 팀 보이스채널에서 팀 별로 진행.
- 7분 : 각자 실습
- 23분 : 팀 내 발표와 피드백



# 언어 구체화하기

😊 실습 아니예요.



문서화



# 코드와 문서를 가까이

## 주석

- 한 번에 구현(구체)으로 넘어가지 말고, 손이 코드를 작성하는 과정을 거치며 단계를 밟아 가세요.
- 주석으로 의사 코드를 작성하세요.
- 주석을 다음 단계 구체 코드로 전환하세요.
- 코드를 보고 바로 작성 의도가 떠오르지 않으면 주석으로 작성 의도를 남기세요.
- 실무 프로젝트가 아니니 코드에 주석을 적극 남기세요.
- 코딩 테스트에서도 이 방식은 매우 유용해요.



# 코딩 테스트에 활용 예시

긴장하면 아는 것도 까먹고, 까먹으면 당황해서 다 까먹는다.

```
def quick_sort(items: list[int]) -> list[int]:
    # 배열이 비어있거나 1개의 요소만 가지고 있다면, 정렬할 필요가 없다.

    # 배열을 반으로 나누고, 피벗을 선택한다.

    # 배열을 순환하면서 피벗보다 작은 요소들을 작은 배열에,
    # 큰 요소들을 큰 배열에, 같은 요소들을 같은 배열에 넣는다.

    # 작은 배열과 큰 배열을 재귀적으로 정렬하고, 합쳐서 반환한다.
```

```
def quick_sort(items: list[int]) -> list[int]:
    # 배열이 비어있거나 1개의 요소만 가지고 있다면, 정렬할 필요가 없다.
    if len(items) <= 1:
        return items

    # 배열을 반으로 나누고, 피벗을 선택한다.
    pivot = items[len(items) // 2]

    # 배열을 순환하면서 피벗보다 작은 요소들을 작은 배열에,
    # 큰 요소들을 큰 배열에, 같은 요소들을 같은 배열에 넣는다.
    for num in items:
        if num < pivot:
            less.append(num)
        elif num > pivot:
            greater.append(num)
        else:
            equal.append(num)

    # 작은 배열과 큰 배열을 재귀적으로 정렬하고, 합쳐서 반환한다.
    return quick_sort(less) + equal + quick_sort(greater)
```

```
def quick_sort(items: list[int]) -> list[int]:
    # 배열이 비어있거나 1개의 요소만 가지고 있다면, 정렬할 필요가 없다.
    if len(items) <= 1:
        return items

    # 배열을 반으로 나누고, 피벗을 선택한다.
    pivot = items[len(items) // 2]

    # 배열을 순환하면서 피벗보다 작은 요소들을 작은 배열에,
    # 큰 요소들을 큰 배열에, 같은 요소들을 같은 배열에 넣는다.
    less = []
    greater = []
    equal = []
    for num in items:
        if num < pivot:
            less.append(num)
        elif num > pivot:
            greater.append(num)
        else:
            equal.append(num)

    # 작은 배열과 큰 배열을 재귀적으로 정렬하고, 합쳐서 반환한다.
    return quick_sort(less) + equal + quick_sort(greater)
```



# 컴퓨터가 문서를 활용할 수 있게

## docstring

```
def join_puddingcamp(name: str, value: int) -> bool:  
    """푸딩캠프에 참가하는 함수입니다.  
  
    Args:  
    -----  
        name (str): 참가자 이름.  
        value (int): 참가자 번호.  
  
    Returns:  
    -----  
        bool: 참가 처리 여부.  
  
    >>> join_puddingcamp("홍길동", 1)  
True  
"""  
return True
```

```
join_puddingcamp  
value (int): 참가자 번호.  
(name: str, value: int) -> bool  
name (str): 참가자 이름.  
푸딩캠프에 참가하는 함수입니다.  
  
Args:  
name (str): 참가자 이름.  
value (int): 참가자 번호.  
  
Returns:  
join_puddingcamp("홍길동", 1)  
True  
You, 1 minute ago
```



# 컴퓨터에게 피드백 받기



# 에러나 경고는 우리의 벗이자 피드백 제공자

- “왜 안 돼” <<<< \* 100 “왜 되지?”
- 에러나 경고가 싫은 이유
  - 불친절, 영어, 기껏 했는데 에러라니!, 싫음 아무튼 싫음
- Psychological victory를 성취하여 싫은 요인을 제거할 수 있다면?
  - 에러를 예상 범위 내로 품기, 정확한 것이 친절이라 여기기, 뻔한 영어 문장 패턴
  - 행동으로 극복할 수 있다면? 잣은 테스팅
    - 명확한 걸 반복하기 => 컴퓨터가 짱짱 잘하는 일

- 피드백이 있는 반복
- 타당한 환경
- 적절한 시점에 이뤄지는 피드백
- 의도적인 반복



# 원인과 피드백의 관계를 생각해보세요.

- 음식을 흘림.
- B : “너 반찬 흘렸어”
  - A : 어? 나한테 뭐라고 하네? 기분 나쁘네? 버럭!
  - A : 그래? 어디? 이거 젓가락으로 잡기 어렵다.
    - B : 이건 숟가락으로 먹는 게 편하더라고.
    - A : 그러네. 옴뇸뇸.

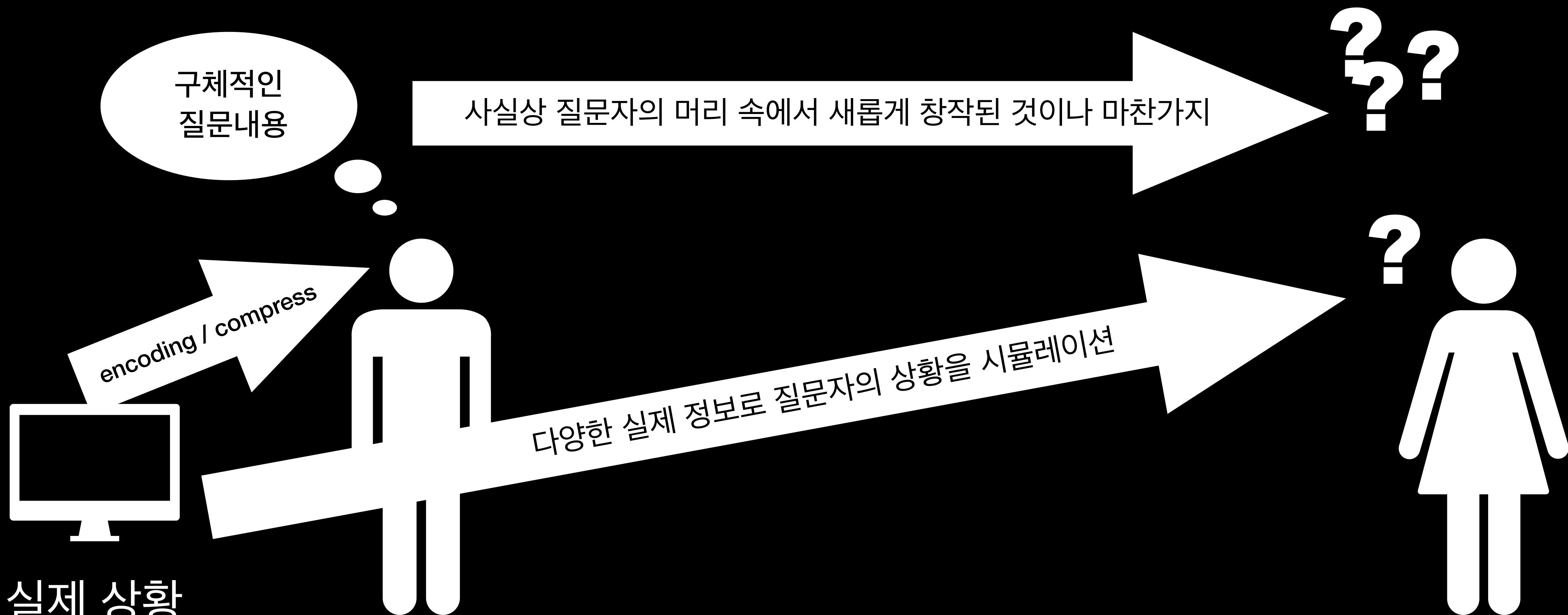


# 질문하며 학습하기



# 여러분은 질문 컨텐츠 크리에이터!

집중! 패널티 팀 과제로 “질문하기”가 있습니다.



# 질문 컨텐츠 작성 양식

## 구체적인 문제 구체화 장치

- 증상, 의문사항 (\*)
- 동작 또는 재현 상황, 환경, 조건 (\*)
- 기대하거나 예상한 동작 또는 결과 (\*)
- 실제 동작 또는 결과 (\*)
- 발생한 오류 또는 출력 메시지 (\*)
- 시도하거나 겪어본 것
- 추측하는 원인이나 문제 (\*)
- 참고한 자료
- 추가 정보 (코드 등)

[https://pudding.camp/~pybgnpreqna](https://pudding.camp/~/pybgnpreqna)



# 질문이 컨텐츠가 되는 예



# 질문이 컨텐츠/정보가 되는 예

2024년 2월 20일

sooyoon 작성자 오늘 오후 2:44  
[상황]

AS-IS (without trailing slash)

```
path('<int:id>/something', ProfilePasswordView.as_view(),  
name='password')
```

TO-BE (append trailing slash)

```
path('<int:id>/something/', ProfilePasswordView.as_view(),  
name='password')
```

제가 예전에 어떤 API를 만들 때, 끝에 / 슬래시를 붙이지 않고 만들었는데, N개월 뒤에 서버 팀 내에서 끝에 슬래시를 추가해달라는 요청이 있어서 바꾸게 되었습니다. 그랬더니, 정상 동작하던 기능이 안되더라고요. 원인은 POST API가 301 Moved Permanently가 되면서 GET 요청으로 바뀌면서 안되는 것 같더라고요. 우선은 핫픽스로 다시 룰백을 해서 / 를 뺏더니, 이번에는 아래와 같은 에러가 납니다.

You called this URL via POST, but the URL doesn't end in a slash and you have APPEND\_SLASH set. Django can't redirect to the slash URL while maintaining POST data. Change your form to point to `api-some-url.com/111111/something/` (note the trailing slash), or set APPEND\_SLASH=False in your Django settings.

[여기서 제가 궁금한 점]  
기존에 잘 동작하던 것이 왜 갑자기 안될까요?  
룰백을 했는데 안돌아가면, 기존에도 안됐어야했던 것이 아닐까요?  
기존에는 잘 동작하다가 룰백을 하니까 안된다고 하는 게 이해가 안가서요.  
혹시 비슷한 이슈를 겪어본 적이 있으신가요?

[의심되는 지점]

- N개월 사이에 파이썬 버전을 3.8 -> 3.11로 올렸습니다.



# 질문 컨텐츠 예) 변경 전과 기대 결과

2024년 7월 20일

sooyoon 작성자 오늘 오후 2:44  
[상황]

AS-IS (without trailing slash)

```
path('<int:id>/something', ProfilePasswordView.as_view(),  
name='password')
```

TO-BE (append trailing slash)

```
path('<int:id>/something/', ProfilePasswordView.as_view(),  
name='password')
```

제가 예전에 어떤 API를 만들 때, 끝에 / 슬래시를 붙이지 않고 만들었는데, N개월 뒤에 서버 팀 내에서 끝에 슬래시를 추가해달라는 요청이 있어서 바꾸게 되었습니다. 그랬더니, 정상 동작하던 기능이 안되더라고요. 원인은 POST API가 301 Moved Permanently가 되면서 GET 요청으로 바뀌면서 안되는 것 같더라고요. 우선은 핫픽스로 다시 룰백을 해서 / 를 뺏더니, 이번에는 아래와 같은 에러가 납니다.

You called this URL via POST, but the URL doesn't end in a slash and you have APPEND\_SLASH set. Django can't redirect to the slash URL while maintaining POST data. Change your form to point to api-some-url.com/111111/something/ (note the trailing slash), or set APPEND\_SLASH=False in your Django settings.

[여기서 제가 궁금한 점]

기존에 잘 동작하던 것이 왜 갑자기 안될까요?

룰백을 했는데 안돌아가면, 기존에도 안됐어야했던 것이 아닐까요?

기존에는 잘 동작하다가 룰백을 하니까 안된다고 하는 게 이해가 안가서요.

혹시 비슷한 이슈를 겪어본 적이 있으신가요?

[의심되는 지점]

- N개월 사이에 파이썬 버전을 3.8 -> 3.11로 올렸습니다.



# 질문 컨텐츠 예) 시도한 것, 시도한 것에 대한 피드백

2024년 7월 20일

sooyoon 작성자 오늘 오후 2:44  
[상황]

AS-IS (without trailing slash)

```
path('<int:id>/something', ProfilePasswordView.as_view(),  
name='password')
```

TO-BE (append trailing slash)

```
path('<int:id>/something/', ProfilePasswordView.as_view(),  
name='password')
```

제가 예전에 어떤 API를 만들 때, 끝에 / 슬래시를 붙이지 않고 만들었는데, N개월 뒤에 서버 팀 내에서 끝에 슬래시를 추가해달라는 요청이 있어서 바꾸게 되었습니다. 그랬더니, 정상 동작하던 기능이 안되더라고요. 원인은 POST API가 301 Moved Permanently가 되면서 GET 요청으로 바뀌면서 안되는 것 같더라고요. 우선은 핫픽스로 다시 룰백을 해서 / 를 뺏더니, 이번에는 아래와 같은 에러가 납니다.

You called this URL via POST, but the URL doesn't end in a slash and you have APPEND\_SLASH set. Django can't redirect to the slash URL while maintaining POST data. Change your form to point to api-some-url.com/111111/something/ (note the trailing slash), or set APPEND\_SLASH=False in your Django settings.

[여기서 제가 궁금한 점]

기존에 잘 동작하던 것이 왜 갑자기 안될까요?

룰백을 했는데 안돌아가면, 기존에도 안됐어야했던 것이 아닐까요?

기존에는 잘 동작하다가 룰백을 하니까 안된다고 하는 게 이해가 안가서요.

혹시 비슷한 이슈를 겪어본 적이 있으신가요?

[의심되는 지점]

- N개월 사이에 파이썬 버전을 3.8 -> 3.11로 올렸습니다.



# 질문 컨텐츠 예)

## 문제 정의 : 정확하진 않음

2024년 2월 20일

sooyoon 작성자 오늘 오후 2:44  
[상황]

AS-IS (without trailing slash)

```
path('<int:id>/something', ProfilePasswordView.as_view(),  
name='password')
```

TO-BE (append trailing slash)

```
path('<int:id>/something/', ProfilePasswordView.as_view(),  
name='password')
```

제가 예전에 어떤 API를 만들 때, 끝에 / 슬래시를 붙이지 않고 만들었는데, N개월 뒤에 서버 팀 내에서 끝에 슬래시를 추가해달라는 요청이 있어서 바꾸게 되었습니다. 그랬더니, 정상 동작하던 기능이 안되더라고요. 원인은 POST API가 301 Moved Permanently가 되면서 GET 요청으로 바뀌면서 안되는 것 같더라고요. 우선은 핫픽스로 다시 룰백을 해서 / 를 뺏더니, 이번에는 아래와 같은 에러가 납니다.

You called this URL via POST, but the URL doesn't end in a slash and you have APPEND\_SLASH set. Django can't redirect to the slash URL while maintaining POST data. Change your form to point to api-some-url.com/111111/something/ (note the trailing slash), or set APPEND\_SLASH=False in your Django settings.

[여기서 제가 궁금한 점]

기존에 잘 동작하던 것이 왜 갑자기 안될까요?

룰백을 했는데 안돌아가면, 기존에도 안됐어야했던 것이 아닐까요?

기존에는 잘 동작하다가 룰백을 하니까 안된다고 하는 게 이해가 안가서요.

혹시 비슷한 이슈를 겪어본 적이 있으신가요?

[의심되는 지점]

- N개월 사이에 파이썬 버전을 3.8 -> 3.11로 올렸습니다.



# 질문 컨텐츠 예) 예상/추정하는 것, 원인 제공 요인



hannal 오늘 오후 3:31

view단 url을 수정 안 했고,  
django settings도 안 고쳤는데,  
python version만 올린 거라면 의아한 상황이긴 한데, `from django.conf import settings` 했  
을 때 3.8에서 APPEND\_SLASH와 3.11에서 APPEND\_SLASH가 어떻게 나오는지 봄보세요.  
동일하다면 python 버전 문제는 아닐 거예요.



sooyoon 작성자 오늘 오후 2:44  
[상황]

AS-IS (without trailing slash)

```
path('<int:id>/something', ProfilePasswordView.as_view(),  
name='password')
```

TO-BE (append trailing slash)

```
path('<int:id>/something/', ProfilePasswordView.as_view(),  
name='password')
```

제가 예전에 어떤 API를 만들 때, 끝에 / 슬래시를 붙이지 않고 만들었는데, N개월 뒤에 서버 팀 내에서 끝에 슬래시를 추가해달라는 요청이 있어서 바꾸게 되었습니다. 그랬더니, 정상 동작하던 기능이 안되더라고요. 원인은 POST API가 301 Moved Permanently가 되면서 GET 요청으로 바뀌면서 안되는 것 같더라고요. 우선은 핫픽스로 다시 룰백을 해서 / 를 뺏더니, 이번에는 아래와 같은 에러가 납니다.

You called this URL via POST, but the URL doesn't end in a slash and you have APPEND\_SLASH set. Django can't redirect to the slash URL while maintaining POST data. Change your form to point to `api-some-url.com/111111/something/` (note the trailing slash), or set APPEND\_SLASH=False in your Django settings.

[여기서 제가 궁금한 점]

기존에 잘 동작하던 것이 왜 갑자기 안될까요?

룰백을 했는데 안돌아가면, 기존에도 안됐어야했던 것이 아닐까요?

기존에는 잘 동작하다가 룰백을 하니까 안된다고 하는 게 이해가 안가서요.

혹시 비슷한 이슈를 겪어본 적이 있으신가요?

[의심되는 지점]

- N개월 사이에 파이썬 버전을 3.8 -> 3.11로 올렸습니다.



# 질문 컨텐츠 예)

문제 정의를 받쳐주는 정보가 해결에 도움



hannah 오늘 오후 3:31

sooyoon 작성자 오늘 오후 4:30

아까전에 revert로 급하게 롤백을 했었는데, 지금 해당 PR을 다시 보니,  
목록에 롤백한 commit은 있는데 file changed에는 코드가 없더라고요 00  
뭔가... 깃헙의 버그인걸까요?

그동안은 revert로 해왔던 것 같은데, 제가 git을 잘 몰라서 그런건지 😅  
'기존에 잘 동작하던 것이 왜 갑자기 안될까요?'에 대한 것은 Python이나 Django의 문제가 아  
니라 롤백이슈가 맞는 것 같습니다! (수정됨)

2024년 7월 20일

sooyoon 작성자 오늘 오후 2:44  
[상황]

AS-IS (without trailing slash)

```
path('<int:id>/something', ProfilePasswordView.as_view(),  
name='password')
```

TO-BE (append trailing slash)

```
path('<int:id>/something/', ProfilePasswordView.as_view(),  
name='password')
```

제가 예전에 어떤 API를 만들 때, 끝에 / 슬래시를 붙이지 않고 만들었는데, N개월 뒤에 서버  
팀 내에서 끝에 슬래시를 추가해달라는 요청이 있어서 바꾸게 되었습니다. 그랬더니, 정  
상 동작하던 기능이 안되더라고요. 원인은 POST API가 301 Moved Permanently가 되면서  
GET 요청으로 바뀌면서 안되는 것 같더라고요. 우선은 핫픽스로 다시 롤백을 해서 / 를 뺏더  
니, 이번에는 아래와 같은 에러가 납니다.

You called this URL via POST, but the URL doesn't end in a  
slash and you have APPEND\_SLASH set. Django can't redirect  
to the slash URL while maintaining POST data. Change your  
form to point to api-some-url.com/111111/something/ (note  
the trailing slash), or set APPEND\_SLASH=False in your  
Django settings.

[여기서 제가 궁금한 점]

기존에 잘 동작하던 것이 왜 갑자기 안될까요?

롤백을 했는데 안돌아가면, 기존에도 안됐어야했던 것이 아닐까요?

기존에는 잘 동작하다가 롤백을 하니까 안된다고 하는 게 이해가 안가서요.

혹시 비슷한 이슈를 겪어본 적이 있으신가요?

[의심되는 지점]

- N개월 사이에 파이썬 버전을 3.8 -> 3.11로 올렸습니다.

출처 : 한날의 디스코드 서버

<https://pudding.camp>



PuddingCamp

# 문제 범위 좁하기



# 문제 범위 좁하기

## 가설 확인

- Q : Python 환경설정할 때 Visual Studio Code의 Terminal에서 python -m venv .venv를 입력하면 오류가 나옵니다.

The screenshot shows a dark-themed terminal window with a red rectangular highlight around the error output. The terminal tabs at the top are labeled '문제', '출력', '디버그 콘솔', '터미널', and '포트'. The '터미널' tab is selected. The terminal output is as follows:

```
>>> quit()
-MacBookPro Python % python -m venv .venv
zsh: command not found: python
-MacBookPro Python % python3
Python 3.12.2 (main, Feb 6 2024, 20:19:44) [Clang 15.0.0 (c)
Type "help", "copyright", "credits" or "license" for more information
>>> python -m venv
File "<stdin>", line 1
  python -m venv
          ^
SyntaxError: invalid syntax
```

- Python이 없다고 나오네?
- 설치가 안 됐나?
- python3는 실행 되고, 최신 버전인 걸 보니 이번에 갓 설치한 것 같군.
- python에 대한 link가 안 걸렸나?
- Homebrew로 설치했나 확인해보자.
- 내가 Clean 설치해보지 않은 맥OS에서 Python 설치법이 바뀌었나?
  - 맥OS 13버전 이후는 업그레이드로 설치만 함.
  - 맥OS 13버전은 설치하자마자 Python 사용 환경을 구성해서 맥OS 정책/설정 변화를 알지 못함.



# 문제 범위 좁하기

## 문제 환경을 재현해보기

- Q : Homebrew로 설치했어요. 안내대로 unlink, link 했는데 차이가 없어요.

```
-MacBookPro ~ % which python  
not found  
-MacBookPro ~ % which python3  
homebrew/bin/python3  
-MacBookPro ~ % brew unlink python@3.12 && brew link python@3.12  
  /opt/homebrew/Cellar/python@3.12/3.12.2_1... 25 symlinks removed.  
  /opt/homebrew/Cellar/python@3.12/3.12.2_1... 25 symlinks created.  
-MacBookPro ~ % which python  
not found  
-MacBookPro ~ % which python3  
homebrew/bin/python3  
-MacBookPro ~ %
```

- Homebrew로 설치는 정상인 것 같네.
- Homebrew의 link 명령어도 정상 동작했고.
- Homebrew script에서도 python으로는 link(alias)를 걸지 않았군.  
<https://bit.ly/3uTerhQ>
- Homebrew 문제는 아니고, 맥OS에 python에 대해 link가 안 걸려있나?



# 문제 범위 좁하기

문제 원인 파악 -> 해결

```
The default interactive shell is now zsh.  
To update your account to use zsh, please run `chsh -s /bin/zsh`.  
For more details, please visit https://support.apple.com/KMHT20050.  
:~ root# which python  
:~ root# which python3  
/usr/bin/python3  
:~ root#
```

- pyenv가 설치되어 있지 않은 root 계정에도 python에 대한 link가 걸려있지 않군.
- 최근 맥OS에서 Python 설정을 뭔가 바꿨나보다.
- python2를 사용할 일은 없을테니 python 이름에 대해 symbolic link 하는 방법을 알려드리자.



# 문제 정의를 잘하면 좋은 점



# 그래서 뭐가 좋은데요?

- A.I : 프롬프트 엔지니어링
- 구글링 : 검색 결과 품질 상승
- 질문자 : 질문을 정의하고 구체화하는 과정에서 **aha! moment**를 만나 문제 실마리를 찾기도 해요.
- 답변자 : 문제 범위가 좁혀져 있어서 깊고 구체적이며 실용적인 해결책이나 코칭 가능
  - “제가 짜는 코드엔 왜 이리 버그가 많을까요?” –> 자기계발서 엔딩
  - “제가 짜는 코드엔 비즈니스 로직과 관련된 버그가 많습니다. 요구사항대로 구현했고 단위 테스트도 작성해서 구현 단계에서는 문제가 없는 걸 확인했는데, 왜 검수할 땐 버그가 많이 발견되는 걸까요?” –> 실무적인 코칭, 멘토링



# 과제



과제-20240304-1



# 의사 코드로 코딩하기

## 문서 관리기 기본 기능

- 구현할 기능
  - 지정한 디렉터리에 있는 문서 파일을 나열해 보여주고, 목록에서 하나를 선택하면(클릭, 엔터) 문서 파일의 내용을 보여준다.
  - 기능은 향후 추가되는 걸 염두.
- 최대한 자세하게.
  - 프로그래밍 구현을 설명하기 어렵다면 최대한 아는 지식을 동원하여 “논리적”으로 말이 되게 서술하면 돼요.
  - 깊은 구현단까지 들어가는 게 아니라 과정대로 개발하면 동작하는 것에 초점.
- Python을 접해본 분은 Python 의사 코드로 전환까지 해보세요. (선택)
- 보는 것과 달리 생각보다 어려워요. 집중력 좋은 시간을 할애하시길 권장합니다.
  - 장담하건데, 이 과정에 능숙해지면 코딩하실 수 있어요.



과제-20240304-2



# Python 학습 환경 구성

## Python VirtualEnv, Visual Studio Code에 Python 설정

- 강의 1-2부터 Python 실습을 합니다.
- 학습에 사용할 PC에 Python 실습 환경을 준비해오세요.
  - Python 3.10 이상
  - 3주 동안 실습할 코드를 저장할 디렉터리 생성 (이하 작업 디렉터리)
  - 작업 디렉터리 안에 Python VirtualEnv 가상 환경 만들기
    - Python에 내장된 `venv` 사용 권장
    - VirtualEnv 디렉터리 이름은 `.venv` (쩜 브이이엔브이)
  - 작업 디렉터리에 `src` 디렉터리 만들기 (이곳에 실습 코드를 작성)
  - 가상 환경에 진입하기, 빠져나오기
  - Visual Studio Code에서 사용할 Python 인터프리터를 Python 가상환경에 있는 Python으로 지정하기
  - 다른 코드 에디터를 사용하고 다룰 줄 알면 그 도구를 사용하셔도 됩니다.



한국  
한국

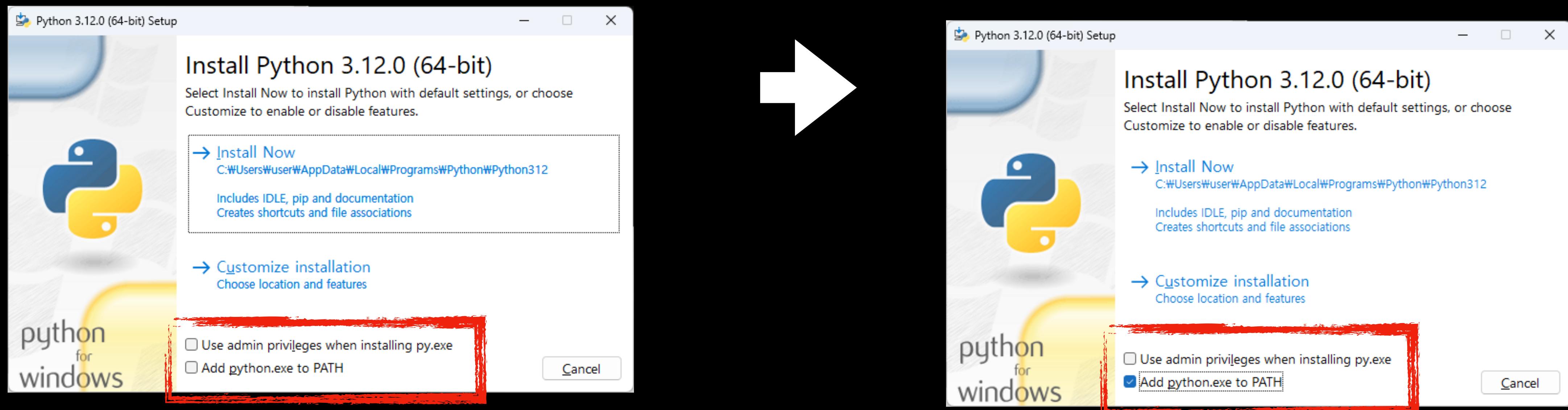


# Python 설치와 설정



# 윈도우에 Python 설치

- 공식 홈페이지( <https://python.org> )에서 최신 버전 내려받은 후 설치
- 명령 프롬프트를 연 뒤에 python 명령어를 실행했을 때 not found 류 메시지 나오면 리부팅 필요.

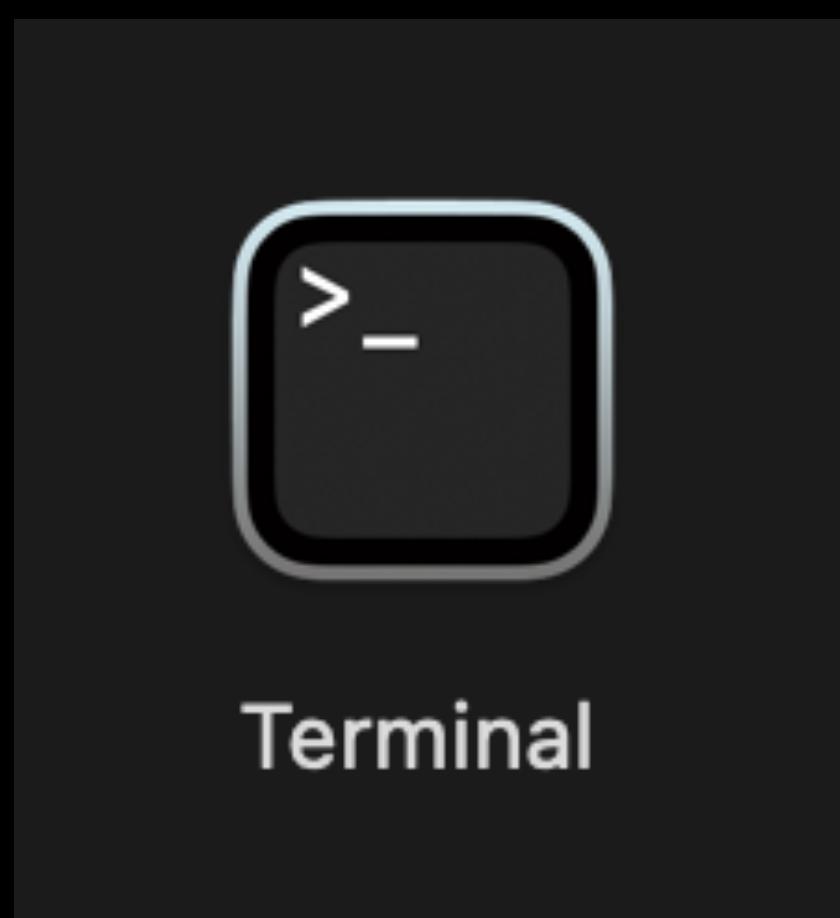


# 맥OS에 설치

Homebrew 설치를 안 했다면 Homebrew부터 설치



- <https://brew.sh> 접속
- 빨간 표시한 아이콘을 클릭하여 설치 스크립트를 클립보드에 복사



- 맥OS에 내장된 터미널 앱을 실행
- 응용 프로그램 > 유ти리티 > 터미널

- 클립보드에 복사한 스크립트를 터미널 앱에 붙여넣은 후 실행

A screenshot of a Mac OS X terminal window. The title bar says "hannal -- zsh -- 80x24". The terminal shows the command "curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh | sh" being typed into the text area. The command has just been completed, as indicated by the cursor at the end of the line.

# 맥OS에 설치

## Homebrew로 Python 설치

- 터미널 앱에서 다음 명령어 실행
  - `brew install python@3.12`
  - `brew not found` 라고 나오면 터미널 앱을 재실행 필요 (맥OS 리부팅 불필요)
  - `python3` 로는 Python이 실행이 되지만, `python` 으로는 실행이 되지 않을 수 있음.



# Python VirtualEnv 생성하기 1/2

맥OS, 윈도우 공통

1. 비주얼 스튜디오 코드(이하 vscode) 실행
2. 작업 기준 디렉터리 생성 (<https://pudding.camp/@/b40959ce> 참고)
3. vscode의 터미널 패널 열기 (<https://pudding.camp/@/03374bba> 참고)
  - 커맨드 팔레트를 열고 terminal 타이핑하면 나옵니다.  
(<https://pudding.camp/@/942a72a0> 참고)
  - 원도우 : Ctrl+Shift+P
  - 맥OS : Command+Shift+P



# Python VirtualEnv 생성하기 2/2

## 맥OS, 윈도우 공통

### 1. 터미널 패널에서 다음 명령어 실행

- `python3 -m venv .venv`
- `-m venv` : venv 모듈을 사용한다는 의미
- `.venv` : 생성할 VirtualEnv 디렉터리 이름

The screenshot shows a terminal window with the following text:  
hannal@hannalmba ~ / Workspace / realworldpudding / codingdude-contents / guide-to-python > enhance-puddingnote |  
enhance-puddingnote | 1001 20:06:06

Below the terminal is a code editor window titled "app.py — guide-to-python". The code is a Python application for a note-taking application. It includes imports for Pathlib, Typing, Textual, and PuddingNote. The main class, PuddingNote, contains methods for handling content screens and managing metadata. The code editor has a sidebar showing project files like app.py, credits.py, checker04.py, and main.py.



# vscode에서 Python VirtualEnv 지정하기

## 1. 확장기능(Extensions)에서 Python 확장 기능 설치

- <https://pudding.camp/@/7820f953>에서 동영상 참고

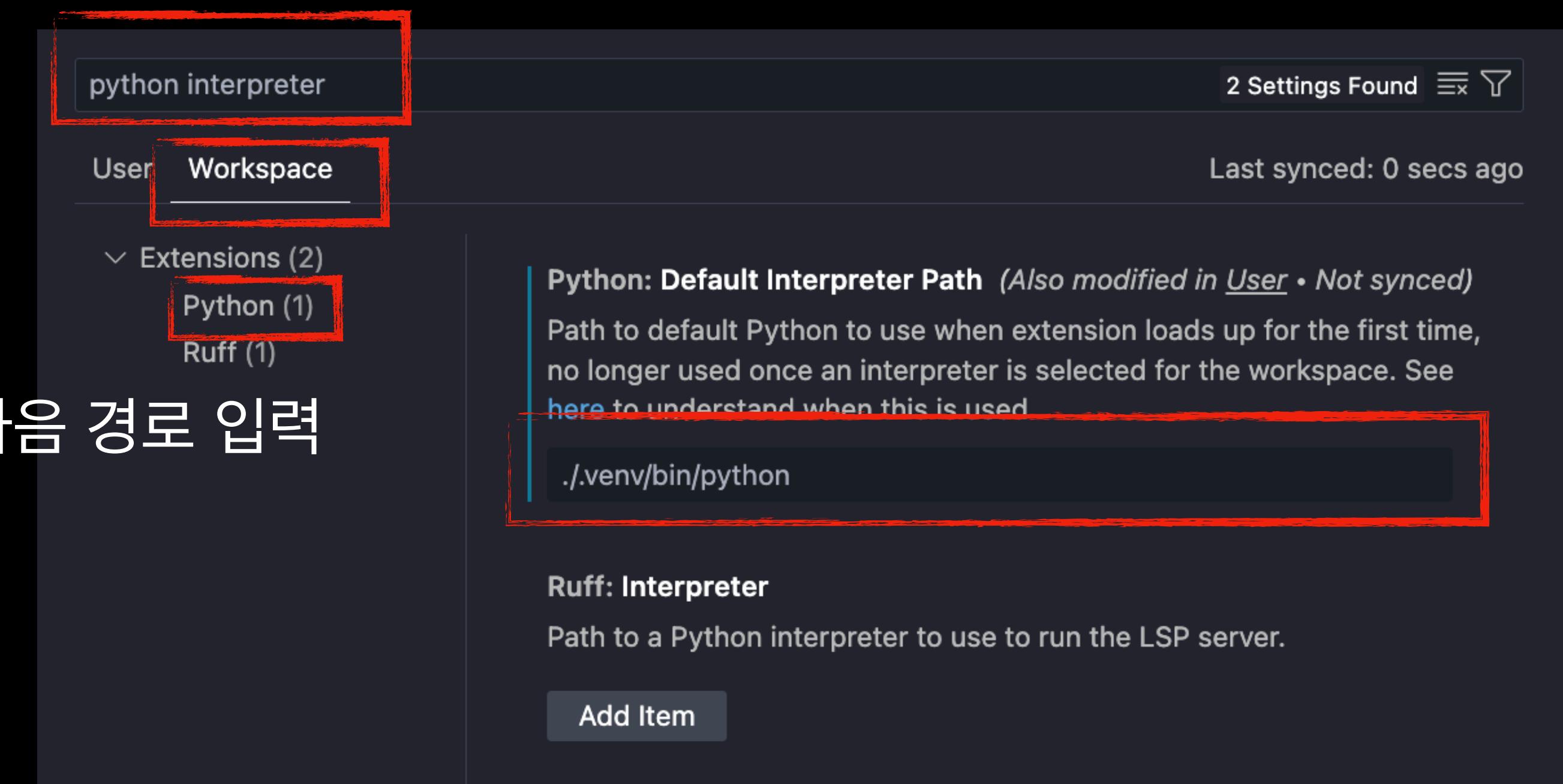
## 2. vscode에서 설정 화면

## 3. 설정 검색란에 python interpreter 입력

## 4. Workspace 탭 선택

## 5. Python: Default Interpreter Path 설정란에 다음 경로 입력

- 맥OS : `./.venv/bin/python`
- 윈도우 : `.\.venv\bin\python`
- \ 문자 : 역슬래시 (엔터 바로 위에 있는 키)



# 생성한 가상 환경에 진입하기

vscode 터미널 패널에선 해당 사항 없음

1. .venv 디렉터리가 있는 경로에 진입
2. 맥OS : `source ./venv/bin/activate`
3. 윈도우
  - PowerShell : `.\venv\Scripts\Activate.ps1`
  - 명령 프롬프트 : `.\venv\Scripts\activate`  
또는 `.\venv\Scripts\activate.bat`
4. 정상 진입하면 Shell의 프롬프트 부분이 변합니다.
5. `deactivate` 을 실행하면 다시 빠져나옵니다.

