프로젝트 개요

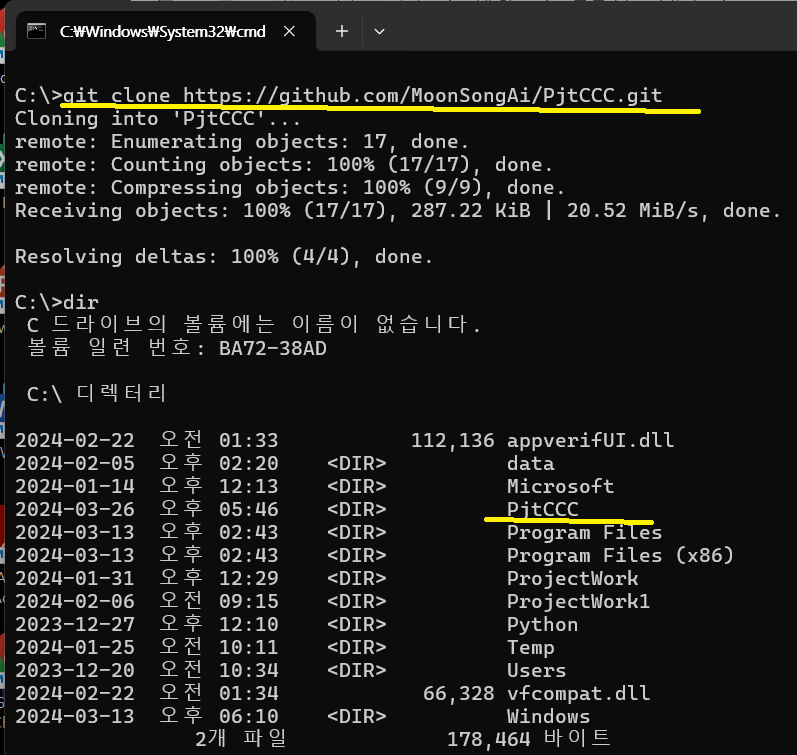
표시 디자인 이미지분석……………..

프로젝트 이름은 ClearCut Check (triple C)라한다.

약어로는 “CCC” 라 하고 가상 환경명 등으로 사용한다

# 프로젝트 환경 구축 Step by step

## MoonSongAi/PjtCCC.git에서 폴더구조 복사 하기



C:/ root에 clone으로 최초 폴더구조 등을 복사한다

Git clone https://guthub.com/moonSongAi/PjtCCC.git

Main 과 master 두개의 branch가 복사되었을 것이다.

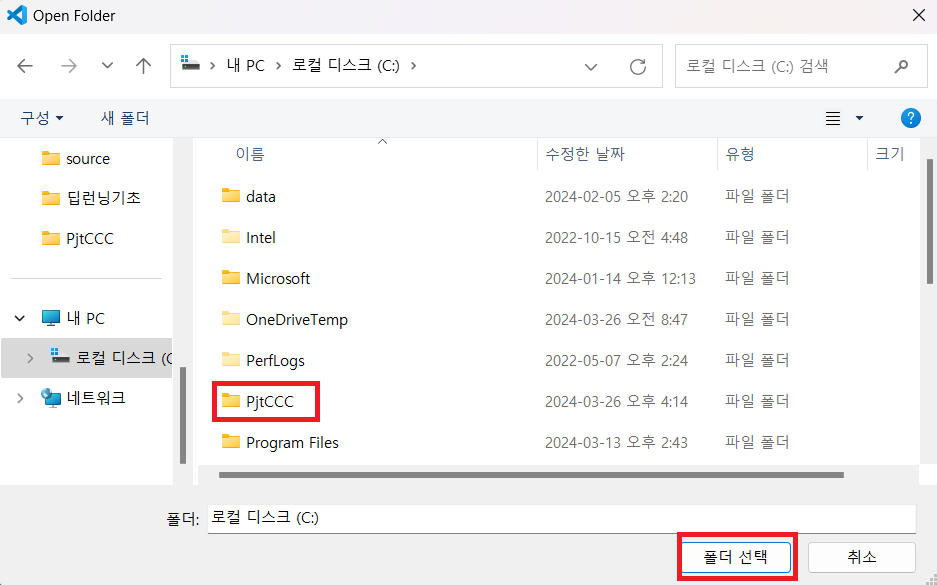
Git checkout main , or master 로 브랜치를 선택 하면 내용 확인 가능

## 가상환경 만들기( CCC)

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명VS Code miniConda 환경에서

C:\PjtCCC 폴더를 선택 한다

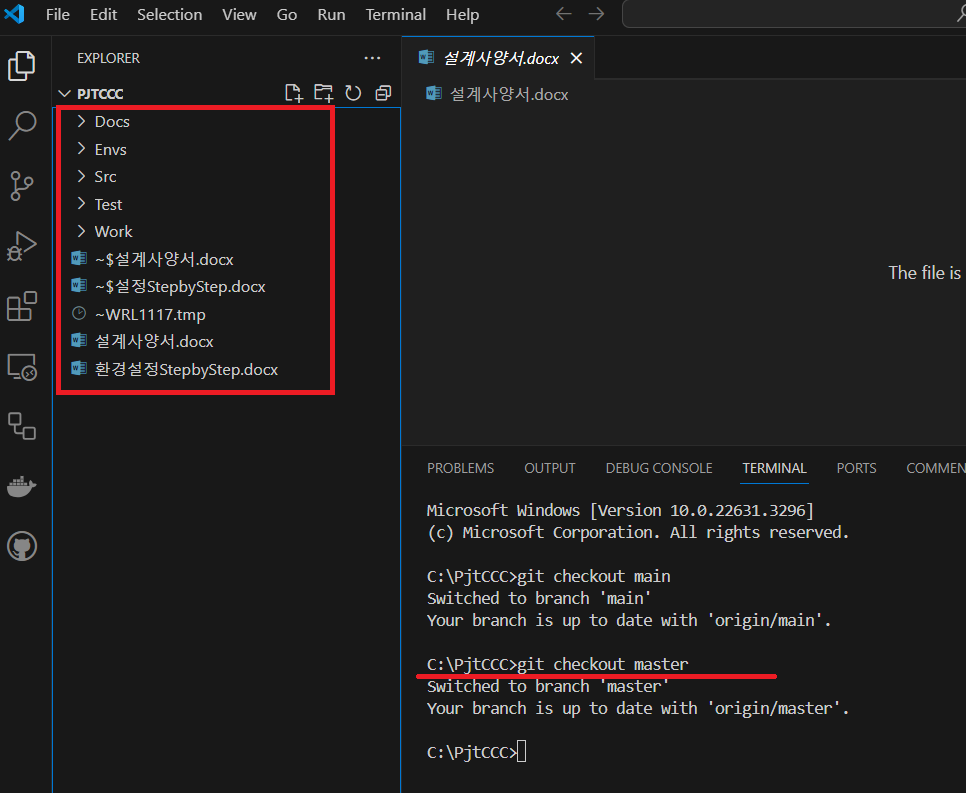


텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

Terminal -> New Terminal 로 CMD창을 아래 하단에 연다



Git checkout master 해야 왼쪽의 폴더 구조가 보인다(브랜치를 main -> master로 변경).

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

miniConda 가상환경에서 동작 할 수 있도록 conda 명령어로 가상환경을 “CCC”로 만든다.

**Python 버전은 3.11**로 통일한다

Conda env list 로 “CCC”라는 가상환경이 miniconda3 가상환경 하위로 생성된 것을 확인 할수 있다.

텍스트, 스크린샷, 디자인이(가) 표시된 사진

자동 생성된 설명

## 개인 환경 추가하기

폴더 또는 파일 추가는 Vs Code 왼쪽 Explorer 화면의 아이콘을 통해 생성한다

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

Work 폴더 밑에 본인 작업용 폴더 생성

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

파일 추가는 폴더를 선택 후 파일추가 아이콘 클릭 후 파일명을 입력 한다

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

## 원격저장소와 내 PC의 Vs Code환경을 연결하기

본 프로젝트의 원격저장소 주소는 [**https://github.com/MoonSongAi/PjtCCC.git**](https://github.com/MoonSongAi/PjtCCC.git) 입니다.

* VS Code를 열고, 소스 코드가 있는 폴더를 엽니다.
* 왼쪽 사이드바에서 소스 제어 아이콘(분기 모양)을 클릭합니다.
* '소스 제어' 패널에서 '저장소 초기화'를 선택하고, 로컬 폴더를 Git 저장소로 초기화합니다.
* 터미널을 열고(`Ctrl + ``), 다음 명령어로 저장소를 GitHub 저장소와 연결합니다:

**git remote add origin** [**https://github.com/MoonSongAi/PjtCCC.git**](https://github.com/MoonSongAi/PjtCCC.git)

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

Remote 연결이 되어있지 않은 경우에 위 명령어를 사용해 연결설정을 해준다

**Git remote -v** 명령어로 remote 설정 값을 확인할 수 있다

원격저장소에 Push 하기위해서는 자격증명이 필요하다

* Windows 검색 바에서 '**자격 증명 관리자**'를 검색하고 열어줍니다.
* 'Windows 자격 증명'을 선택하고 '일반 자격 증명 추가'를 선택합니다.
* 인터넷 또는 네트워크 주소에 **git: https://github.com**을 입력합니다.
* 사용자 이름에 GitHub의 사용자 이름을 입력하고(MoonSongAi), 비밀번호에는 Personal Access Token을 입력한 다음 저장합니다. ( 9MR5)

ghp\_4v1suaZWcMPKB1J1Qi9g6J1LIk4lXE3Q\*\*\*\*

* 이제 VS Code에서 GitHub를 사용할 때, 이 자격 증명을 사용하여 인증을 시도할 것입니다.

## 매일 해야 하는 버전관리 루틴들…

출근 해서 작업 시작 전… 마음을 다 잡고 .. 원격저장소를 내 PC에 동기화 시키다.

Source control 메뉴에서 동기화 방법

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

* **터미널 사용:**
  + VS Code의 하단에 있는 터미널을 엽니다(**Ctrl** + `` 키로 열 수 있음).
  + 현재 Git 브랜치가 올바른지 확인하고, 필요한 경우 **git checkout [master]** 명령어로 브랜치를 변경합니다.
  + **git pull** 명령어를 입력하여 원격 저장소의 변경 사항을 로컬 저장소로 가져옵니다.
* **소스 제어 옵션 사용:**
  + 소스 제어 패널에서, '...' 버튼(더 보기 액션)을 클릭합니다.
  + 나타나는 메뉴에서 'Pull '을 선택합니다.
  + 원격 저장소와 브랜치를 선택하면 VS Code가 해당 브랜치의 변경 사항을 로컬 저장소로 가져옵니다.

Vs Code 왼쪽 하단에 동기화된 브랜치 와 상태 정보가 보인다

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

###### 작업을 신나게 하고 하루 일과를 마칠 때 또는 그냥 서버에 백업 하고 싶을 때….

작업한 로칼 저장소를 원격저장소에 백업 한다… 그 절차는 다음과 같다

**1. 소스 제어 패널 열기:**

* VS Code의 사이드바에서 소스 제어 아이콘(보통 분기 모양 아이콘)을 클릭하여 소스 제어 패널을 엽니다.

**2. 변경 사항 스테이징하기:**

* 변경된 파일들이 소스 제어 패널에 나열됩니다. 변경 사항을 스테이징하기 위해 각 파일 옆의 '+' 버튼을 클릭하거나, 모든 변경 사항을 한 번에 스테이징하기 위해 패널 상단의 '+' 버튼을 클릭합니다.

**4. 변경 사항 커밋하기:**

* 스테이징된 변경 사항에 대한 커밋 메시지를 입력합니다. 이는 변경 사항에 대한 간단한 설명이어야 합니다.
* 메시지를 입력한 후, 소스 제어 패널 상단의 체크마크(√) 아이콘을 클릭하여 커밋합니다.

**5. 변경 사항 푸시하기:**

* 소스 제어 패널의 '...' 버튼(더 보기 옵션)을 클릭하고, 'Push'를 선택합니다.
* 만약 여러 원격 저장소가 있다면, 'Push To...'를 선택하고 푸시할 원격 저장소를 선택할 수 있습니다.
* VS Code가 설정된 인증 방식에 따라 로그인을 요구할 수 있습니다. GitHub를 사용한다면, GitHub 계정 정보 또는 Personal Access Token이 필요할 수 있습니다.

**6. 푸시 진행 상태 확인:**

* VS Code 하단의 상태 바에서 Git 푸시의 진행 상태를 확인할 수 있습니다. 완료되면, 로컬 변경 사항이 원격 저장소에 성공적으로 푸시된 것입니다.

**주의사항:**

* 원격 저장소에 푸시하기 전에 항상 로컬 저장소를 최신 상태로 유지하기 위해 **Pull** 또는 **Fetch**를 먼저 수행하는 것이 좋습니다. 이는 충돌을 방지하고, 푸시 과정에서 발생할 수 있는 문제들을 최소화하는 데 도움이 됩니다.

이 단계들을 따라하면, VS Code에서 Windows 환경 기준으로 로컬 저장소의 변경 사항을 원격 저장소에 push할 수 있습니다.

## 프로젝트의 일관성과 재현성 유지를 위해

여러 사용자가 같은 운영 환경(예: 파이썬 버전, 설치된 패키지 등)을 유지하는 것은 프로젝트의 일관성과 재현성을 위해 매우 중요합니다

본 프로젝트는 이를 위해 가상환경을 적용 하겠습니다

**단계 1: 필요한 패키지 설치**

1. **의존성 파일 확인:** 프로젝트와 함께 **requirements.txt** 파일이 제공되어야 합니다. 이 파일에는 프로젝트 실행에 필요한 모든 Python 패키지와 버전이 명시되어 있습니다.
2. **패키지 설치:** **pip install -r requirements.txt** 명령어를 사용하여 **requirements.txt**에 명시된 모든 패키지를 설치합니다.

**pip install -r requirements.txt**

**단계 2: 가상 환경에서 작업**

* 가상 환경이 활성화되면, 해당 환경 내에서 Python 코드를 실행하거나 추가 패키지를 설치할 수 있습니다. 이때 설치된 패키지는 가상 환경에만 영향을 미치며 시스템의 다른 부분에는 영향을 주지 않습니다.

**단계 3: 의존성 관리**

* 프로젝트에서 새로운 패키지를 사용하는 경우, 가상 환경이 활성화된 상태에서 해당 패키지를 설치하고, **pip freeze > requirements.txt** 명령어로 새로운 requirements.txt 파일을 생성하여, 모든 사용자가 같은 패키지를 사용할 수 있도록 합니다.

모든 개발자 개인은 본인의 개인 작업 폴더에 **requirements.txt 를 관리 하고 일정 시점에 통합한다**