START CO-OPerative with Github:github로 시작하는 협업 첫 걸음

1. 파일 버전 관리
    (1) Repository
        a. repository == directory
            - github:repositories → new → name →commit
    (2) Remote & Local
        a. local & remote
            - local to remote:push
            - remote to local:pull, clone
    (3) Branch
        a. branch:분리된 작업 영역 개념
            - git의 장점 중 하나
            - conflict를 해소하기 위한 개념
    (4) Merge & fetch
        a. marge:서로 다른 코드를 융합
        b. cherry-picking
            - “최선 또는 가장 바람직한 선택”
            - 타 branck에 있는 일부 commit 가져오기
    (5) Commit
        a. commit ==메시지, 기록, footprint →수정 메시지 남기는 일
        b. stage 상태의 tracking file의 변경 이력을 ‘저장’
        c. commit message
            - 변경 내용 요약 //제목
            - 변경 사유 //본문(선택)
            - 참고사항 //꼬리말(선택)
        d. 추후 유지 보수 용이
        e. 모두가 이해 가능한 용어
    (6) 실습
        a. ‘git init’ 현 디렉토리 초기화
        b. ‘git remote add <name:user> <url:repo>’ remote branch 연결
        c. ‘git add (-p)’
        a. untracking file to tracking file
        b. unstage file to statge
        d. ‘git commit (-m “message”)’ staging 된 file commit
        e. ‘git push <remote> <branch>’로 최종 push

2. 협업의 버전 관리
    (1) Fork:제 3자(혹은 조직) 의 repo →나의 repo로 branch
    → 필수 동반(issue, pull request or approve, merge)
    (2) Issue & TODO
        a. Issue ==이슈, 논점
        b. ★절대 투두리스트에 이슈 넣지 말것,,,
    (3) Project
    (4) Pull Request:main branch에 코드 삽입 가능 여부 묻기
    (5) CO-OP Branch & MY Branch →검사받고 올릴 것
        a. CO-OP시 Main Branch == Forked 한 Main Branch
        b. CO-OP시 Local Branch == 내 Main Branch
3. 상호검증

(1) Fort & make a pull request
    a. base repo선택: base:main ← head repo선택 compare:main
    b. reviewers선택:짝꿍(이창현 선배error 403)
    c. labels:documentation
    d. 상호검증 알림 메시지
    e. comment 수정사항 등 안내
    f. 검증 완료시 approve submit
(2) conflict
    a. marge시 충돌:수정사항이 동시다발적일 경우 →수정사항 찾아 반영