깃과 깃허브 학습 내용 요약 - 브랜치 병합 및 충돌 해결

1. 다양한 브랜치 병합

- 브랜치 병합(merge):
  - 두 브랜치를 하나로 합치는 과정.
  - Fast-forward 병합: 일렬 상태에서 병합, 새로운 커밋 생성 없음.
  - 3-way 병합: 조상이 다른 두 브랜치 병합, 새로운 병합 커밋 생성.
  - Squash 병합: 여러 커밋을 하나로 합쳐 병합.
- 명령어:
  - $ git merge [브랜치명]: 일반 병합.
  - $ git merge --no-ff [브랜치명]: 강제로 3-way 병합 수행.

2. 병합 충돌과 해결

- 병합 충돌 발생:
  - 두 브랜치가 동일한 파일의 다른 부분을 수정한 경우.
  - 충돌 발생 시 파일 내부에 <<<<<<<, =======, >>>>>>> 표기로 구분.
- 충돌 해결 과정:
  1. 충돌한 파일 수정 후 저장.
  2. $ git add [파일명]으로 수정사항 반영.
  3. $ git commit 으로 병합 완료.
- 명령어:
  - $ git merge --abort: 병합 취소.
  - $ git revert [커밋 ID]: 특정 커밋 되돌리기.

3. 브랜치 리베이스(rebase)

- Rebase 란?

  - 브랜치의 Base 를 변경해 이력을 선형으로 정리.
  - 복잡한 병합 이력을 간소화하고, Fast-forward 병합에 유리.
- 충돌 해결:
  - 충돌 발생 시 충돌 파일 수정 후 $ git add 및 $ git rebase --continue.
- 주의:
  - 원본 이력이 변경되므로 협업 중에는 주의 필요.

4. 커밋 이력 수정

- 명령어:
  - $ git commit --amend: 최신 커밋 메시지 수정.
  - $ git rebase -i HEAD~[숫자]: 이전 여러 커밋 수정 및 병합(squash).
- 실습 예제:
  - 최신 커밋에 파일 추가 후 $ git commit --amend 로 반영.
  - Squash 를 활용해 여러 커밋을 하나로 병합.

5. 비주얼 스튜디오 코드에서 깃 활용

- 주요 기능:
  - Changes(작업 디렉토리), Staged Changes(스테이징 영역) 시각적 관리.
  - 확장 기능(Git Graph) 설치로 이력 확인 및 비교.
- 명령어 통합:
  - $ git restore, $ git add, $ git commit 과 같은 작업을 UI 로 실행.

6. 버전 되돌리기 (Reset)

- Reset 의 3 가지 옵션:
  - --hard: 작업 디렉토리, 스테이징 영역, 저장소 모두 되돌림.
  - --mixed: 스테이징 영역과 저장소만 되돌림.

- --soft: 저장소만 되돌림.

- 명령어:

- $ git reset [옵션] [커밋 ID].

- $ git reset --hard ORIG_HEAD: 초기 상태로 복원.

7. 커밋 취소(Revert)

- Revert 란?

- 지정된 커밋의 변경사항을 되돌리고, 새로운 커밋 생성.

- 기존 이력을 유지하므로 협업에 적합.

- 명령어:

- $ git revert [커밋 ID]: 일반 Revert.

- $ git revert --no-edit: 메시지 자동 추가.

8. 임시 저장(Stash)

- Stash 활용:

- 작업 내용을 임시 저장 후 복원 가능.

- $ git stash 로 저장, $ git stash apply 로 복원.

- 다양한 명령어:

- $ git stash list: 저장된 목록 확인.

- $ git stash drop: 특정 임시 저장 삭제.

++기말 시험

19 – 30 강 주로 보기 하지만 중간거가 안 나오는건 아님

o/x -> 2 개

괄호 채우기 -> 3 개

객관식 -> 5 개

실습코드문제 -> 5~7 개