

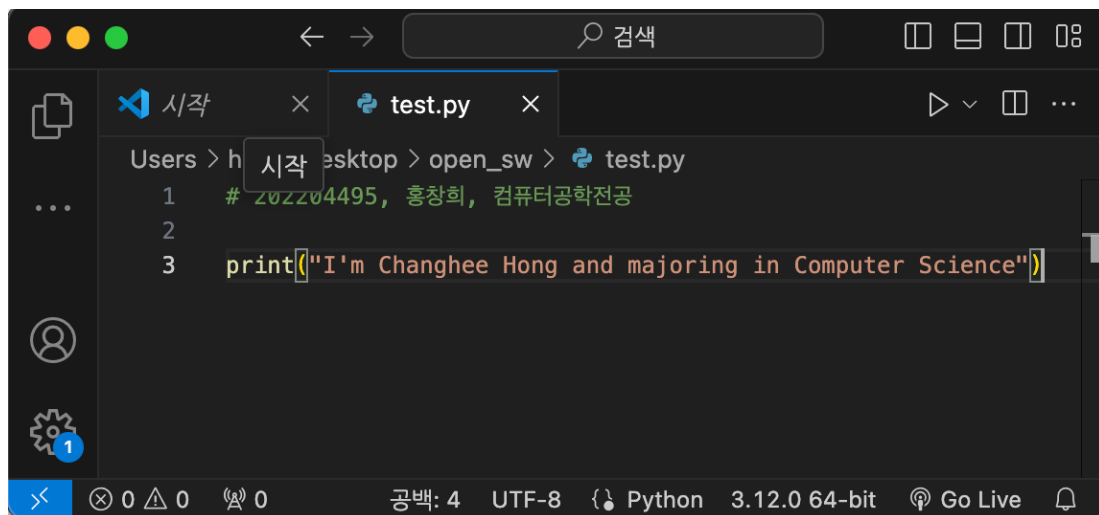
# 오픈소스SW – 5주차 실습

202204495 홍창희

## 실습 1

강의자료를 따라 터미널 또는 커맨드에서 커밋과 관련된 실습 절차를 그대로 수행하시오.

```
[hch@HCHui-MacBookPro ~ % cd desktop
[hch@HCHui-MacBookPro desktop % mkdir open_sw
[hch@HCHui-MacBookPro desktop % cd open_sw
[hch@HCHui-MacBookPro open_sw % git init
Initialized empty Git repository in /Users/hch/Desktop/open_sw/.git/
[hch@HCHui-MacBookPro open_sw % code test.py
[hch@HCHui-MacBookPro open_sw %
```



```
[hch@HCHui-MacBookPro open_sw % git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    test.py

nothing added to commit but untracked files present (use "git add" to track)
[hch@HCHui-MacBookPro open_sw %
```

→ 터미널에서 open\_sw 폴더를 만들고, 이동 후 저장소를 깃으로 초기화 시켰고, VS Code를 사용하여 test.py 파일을 작성했습니다. 파일 생성 후에 git status 명령어를 통해 unstaged 상태인 것을 잘 확인했습니다.

```
[hch@HCHui-MacBookPro open_sw % git add test.py
]
[hch@HCHui-MacBookPro open_sw % git status
]
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   test.py

hch@HCHui-MacBookPro open_sw % █
```

→ git add 명령어를 통해 stage에 등록했고 git status 명령어를 통해 staged 상태가 된 것을 잘 확인했습니다.

```
[hch@HCHui-MacBookPro open_sw % git rm --cached test.py
]
rm 'test.py'
[hch@HCHui-MacBookPro open_sw % git status
]
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    test.py

nothing added to commit but untracked files present (use "git add" to track)
hch@HCHui-MacBookPro open_sw % █
```

→ git rm --cached 명령어를 통해 stage에 등록된 파일만을 삭제했습니다. 그리고 git status 명령어를 통해 unstaged 상태가 된 것을 잘 확인했습니다.

```
[hch@HCHui-MacBookPro open_sw % git add test.py
]
hch@HCHui-MacBookPro open_sw % █
```

→ 다음 실습을 위해 다시 staged 상태로 변경해 놓았습니다.

```

[hch@HCHui-MacBookPro open_Sw % git add test.py
[hch@HCHui-MacBookPro open_Sw % git commit
[main (root-commit) 8141438] "This is Test Message"
 1 file changed, 3 insertions(+)
 create mode 100644 test.py
[hch@HCHui-MacBookPro open_Sw % git rm --cached test.py
rm 'test.py'
[hch@HCHui-MacBookPro open_Sw % git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        deleted:    test.py

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        test.py

hch@HCHui-MacBookPro open_Sw % █

```

→ rm 명령어와 reset 명령어의 차이점을 보이기 위해 test.py 파일을 한번 커밋한 후에 rm --cached 명령어를 통해 삭제해봤습니다. 그리고 status를 명령어를 사용하니 한 번도 커밋하지 않은 상태에서 rm --cached 명령어를 사용했을 때와 다른 상태인 것을 확인할 수 있었습니다.

이는 커밋을 한 번이라도 한 뒤에 rm 명령어를 사용해서 stage에서 없애버리면 "삭제"라는 것도 하나의 변동사항으로 인식한다는 걸 의미하는 것 같습니다.

그러므로, 파일을 등록한 후 커밋하지 않고 바로 삭제하려면 rm --cached 명령어를 사용하면 되고, 한 번이라도 커밋을 했다면 reset 명령어를 사용하여 커밋을 정리해 주어야 합니다.

```

[hch@HCHui-MacBookPro open_Sw % git reset HEAD test.py
[hch@HCHui-MacBookPro open_Sw % git status
On branch main
nothing to commit, working tree clean
hch@HCHui-MacBookPro open_Sw % █

```

→ 위에서 알게된 것을 확인하기 위해 git reset HEAD 명령어를 사용한 후 status 명령어를 통해 상태를 확인해보니 정상적으로 커밋이 정리된 것을 볼 수 있었습니다.

```

[hch@HCHui-MacBookPro open_Sw % git mv test.py name.py
[hch@HCHui-MacBookPro open_Sw % git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        renamed:    test.py -> name.py

hch@HCHui-MacBookPro open_Sw % █

```

→ 파일 이름을 변경하기 위해 git mv 명령어를 사용해서 파일 이름을 test.py에서 name.py로 바꿔주었습니다. 그 후 status 명령어를 통해 확인해보니 잘 변경된 것을 볼 수 있었습니다.

```

[hch@HCHui-MacBookPro open_Sw % git mv name.py test.py
[hch@HCHui-MacBookPro open_Sw % █

```

→ 다음 실습을 위해 다시 원래 이름으로 되돌려 놓았습니다.

※ 위에서 rm과 reset의 차이점을 보이기 위해서 커밋을 했던 과정을 세부적으로 기록하겠습니다.



```

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch main
#
# Initial commit
#
# Changes to be committed:
#   new file:   test.py
#
~
~
~
~
~
~
~
~
~
~
~/Desktop/open_sw/.git/COMMIT_EDITMSG" 11L, 227B

```

→ git commit 명령어를 입력하니 위와 같이 vi 에디터가 나타났습니다..



```
open_sw — vi — git commit — 80x18

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch main
#
# Initial commit
#
# Changes to be committed:
#   new file:   test.py
#

"This is Test Message"
~
~
~
-- INSERT --
```

→ 내용을 입력하기 위해 ESC를 누른 후 i를 눌러 INSERT 상태로 변경하고, 메시지를 남겨놓았습니다.

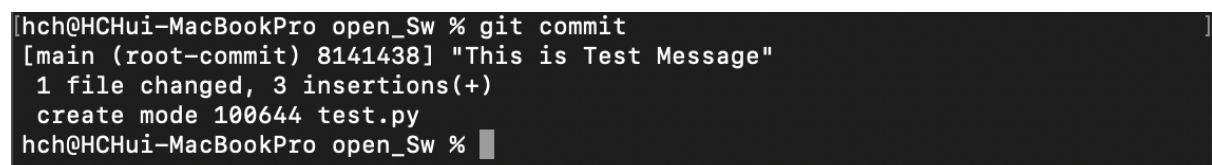


```
open_sw — vi — git commit — 80x18

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch main
#
# Initial commit
#
# Changes to be committed:
#   new file:   test.py
#

"This is Test Message"
~
~
~
:wq
```

→ 작성을 마친 후, 저장과 종료를 하기 위해 ESC를 누른 후 : 와 w, q 를 입력했습니다.



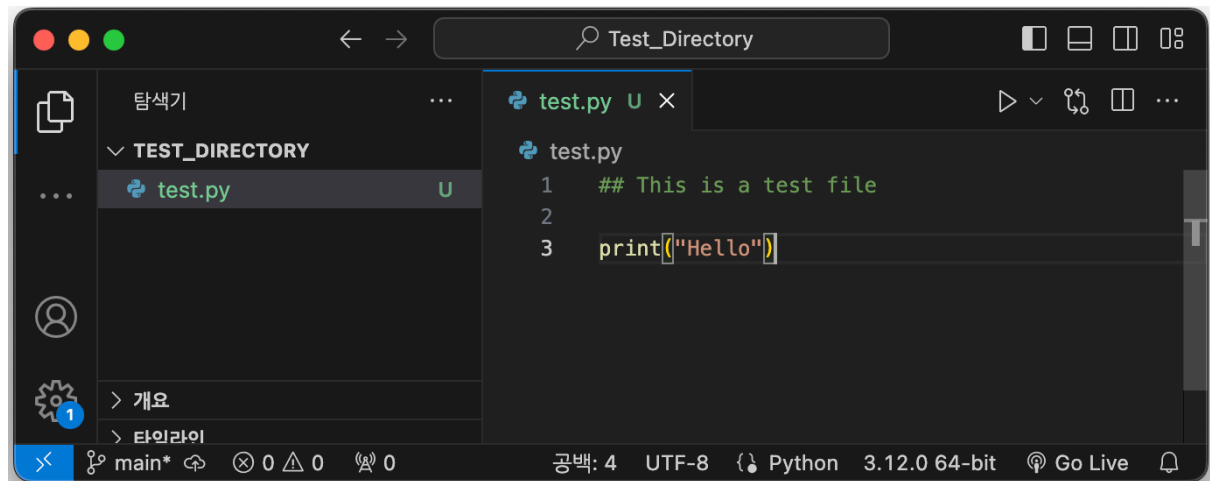
```
[hch@HCHui-MacBookPro open_Sw % git commit
[main (root-commit) 8141438] "This is Test Message"
 1 file changed, 3 insertions(+)
 create mode 100644 test.py
hch@HCHui-MacBookPro open_Sw %
```

→ vi 에디터가 종료된 후 터미널을 확인하니 성공적으로 커밋된 것을 볼 수 있었습니다.

## 실습 2

예를 들어 working directory에서 test.py라는 파일에 코드 작성을 완료하고 stage에 올려두었다. 그런데, 그 상태에서 내부에 수정해야할 것이 생각나서 다시 test.py 파일의 코드를 수정하였다. 이때 stage에 있는 test.py에 unstage를 시도하면 어떤 일이 발생하는가? 그리고 왜 이러한 일이 발생하는 것 같은지 작성하시오.

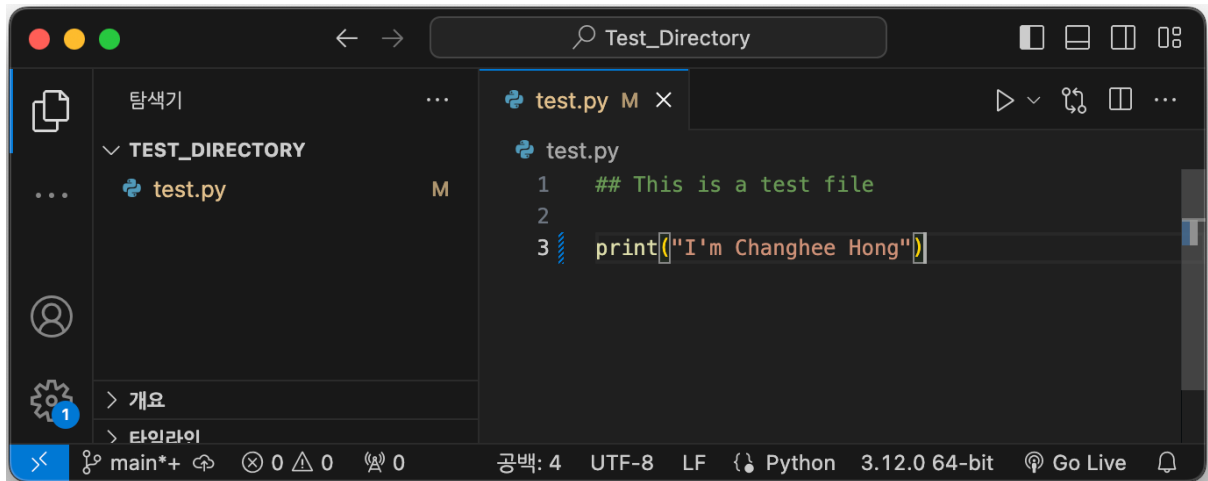
```
[hch@HCHui-MacBookPro ~ % cd desktop
hch@HCHui-MacBookPro desktop % mkdir Test_Directory
hch@HCHui-MacBookPro desktop % cd Test_Directory
hch@HCHui-MacBookPro Test_Directory % git init
Initialized empty Git repository in /Users/hch/Desktop/Test_Directory/.git/
hch@HCHui-MacBookPro Test_Directory % code test.py
hch@HCHui-MacBookPro Test_Directory %
```



→ Test\_Directory 폴더를 만들고, test.py 파일을 만든 후에 코드 작성을 마쳤습니다.

```
[hch@HCHui-MacBookPro Test_Directory % git add test.py
hch@HCHui-MacBookPro Test_Directory %
```

→ git add 명령어를 사용하여 test.py 파일을 stage에 올려두었습니다.



→ working directory에 있는 test.py 파일을 수정하였습니다.

```
[hch@HCHui-MacBookPro Test_Directory % git rm --cached test.py  
error: the following file has staged content different from both the  
file and the HEAD:  
    test.py  
(use -f to force removal)  
hch@HCHui-MacBookPro Test_Directory %
```

→ rm --cached 명령어를 사용하여 unstage를 시도하니 위와 같이 에러가 발생하며, HEAD 버전의 파일과 working directory에 있는 파일이 같지 않다는 경고를 보여줬습니다. 추가로, 강제로 unstage를 하려면 -f를 사용하라는 안내사항도 보여주는걸 확인했습니다.

→ 이러한 에러는 진행 중인 변경사항이 사라지는 것을 막기 위한 방어기제일 것이라고 생각합니다.

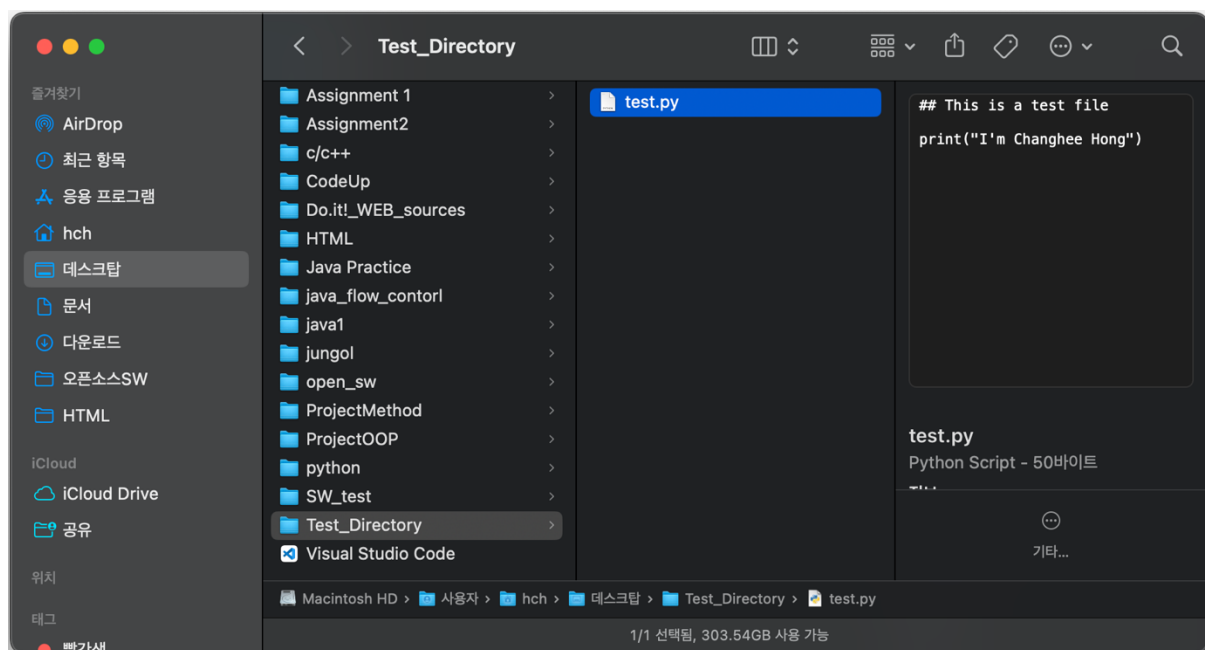


### 실습 3

이어서, 이를 강제로 `unstage`를 시도하는 경우 `stage`에 있는 `test.py`와 `working directory`에 있는 `test.py` 중 어떤 것이 우선시 되는가? 그것이 우선시 되는 이유는 무엇이라 생각하는지 쓰시오.

```
[hch@HCHui-MacBookPro Test_Directory % git rm -f --cached test.py  
rm 'test.py'  
hch@HCHui-MacBookPro Test_Directory % ]
```

→ `-f` 를 사용하여 강제로 `unstage`에 성공하였습니다.



→ 그 후 제 로컬에 있는 `working directory`에서 `test.py` 파일을 확인해봤더니, `stage`에 있던 HEAD 버전의 파일이 우선시 되어 파일이 변경되어 있는 것을 볼 수 있었습니다.

→ `stage`엔 보통 유의미한 변경사항이 생긴 파일들이 옮겨지기 때문에, `working directory`의 변경사항 보다 더 중요하다고 생각되어 우선시되는 것이라고 생각합니다.