3주차 실습 시트

강 환수 교수

<table>
<tr><th>프로젝트</th><th colspan="2">commit & log, clone & pull [교재 4장 5장]</th></tr>
<tr><td>모듈1</td><td colspan="2">☐ 깃에서 전체 필요한 설정과 확인
❍ line feed / carriage return 자동 변환
❍ 이름과 메일 주소 설정, 편집기, 별칭 명령어 설정
☐ 파일 hello.py 생성 및 실행, 삭제
❍ 추가되지 않은 파일은 탐색기에서 그대로 삭제 가능</td></tr>
<tr><td>모듈2</td><td colspan="2">☐ 다시 파일 생성해 추가한 후 stage에서 삭제, 복구 후 커밋</td></tr>
<tr><td>모듈3</td><td colspan="2">☐ 파일 basic.py 총 3회 커밋, 이력 정보 확인</td></tr>
<tr><td>모듈4</td><td colspan="2">☐ 다양한 커밋 정보 확인, --patch --stat</td></tr>
<tr><td>모듈5</td><td colspan="2">☐ 깃허브 원격저장소 복제와 원격저장소 수정내용 가져오기(pull)</td></tr>
<tr><td>모듈6</td><td colspan="2">☐ 자신의 깃허브에 PAT(Personal Access Token) 생성, push와 pull</td></tr>
<tr><td>준비</td><td colspan="2">☐ 다음 폴더를 생성 후 git/03w에서 git bash 실행 후 시작
☐ vscode, github, source tree 준비</td></tr>
<tr><td rowspan="6">모듈1
실습
(15min)</td><td colspan="2">☐ 깃에서 전체 필요한 설정과 확인
❍ line feed / carriage return 자동 변환
❍ 이름과 메일 주소 설정, 편집기 설정
☐ 파일 hello.py 생성 및 실행, 삭제
❍ 추가되지 않은 파일은 탐색기에서 그대로 삭제</td></tr>
<tr><td>$ git config --global user.name ai7dnn
$ git config --global user.email ai7dnn@gmail.com

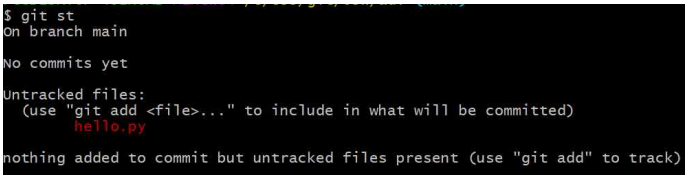
$ git config --global core.autocrlf true
$ git config --global core.safecrlf false

$ git config --global core.editor “code --wait”
$ git config --global init.defaultbranch main

$ git config --global alias.st status
$ git config --global alias.sts ‘status –s’

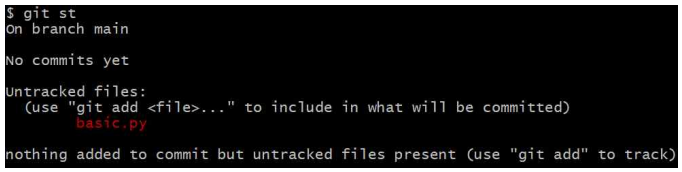
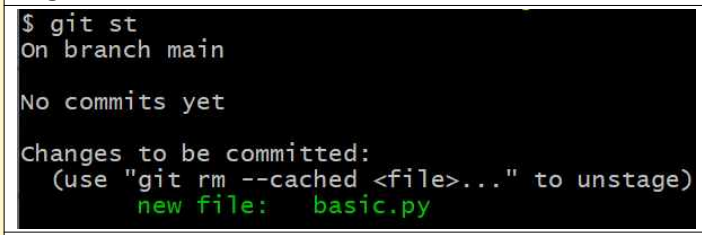
$ git config --global --list</td><td>사용자 환경 설정

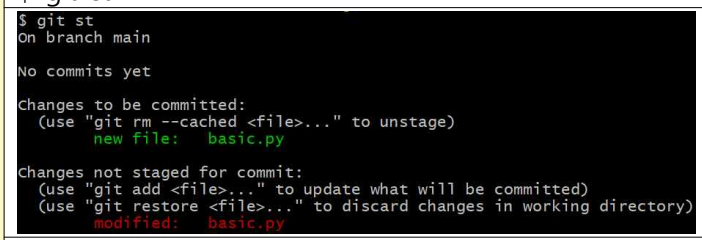
[교재 p41] 윈도와 맥의 캐리지 리턴과 라인 피드 변환 (윈도와 맥과 차이)
편집기 환경 설정
환경 설정 확인</td></tr>
<tr><td>$ git init adv
$ cd adv</td><td>저장소 adv 생성</td></tr>
<tr><td>$ code hello.py

$ python hello.py</td><td>vscode로 파이썬 파일 편집 후 저장
print(list(‘python’))
(ctrl + s) 후 실행</td></tr>
<tr><td>$ git config --global alias.st ‘status’
$ git st</td><td>별칭을 생성해 실행</td></tr>
<tr><td> 
</td><td>새로 생성하면 untracked file</td></tr>
</table>

<table>
<tr><td></td><td>$ rm hello.py
$ ls
$ git st</td><td>untracked file은 그대로 삭제 가능</td></tr>
<tr><td></td><td> 
</td><td>저장소에 커밋할 것이 없음</td></tr>
<tr><td rowspan="8">모듈2
실습
(20min)</td><td colspan="2">□ 다시 파일 생성해 추가한 후 stage에서 삭제, 복구 후 커밋</td></tr>
<tr><td>$ code basic.py

$ python basic.py
$ git st</td><td>저장소 basic 생성해 저장(ctrl + s)
print(list(range(10)))

깃 상태 보기</td></tr>
<tr><td> 
</td><td>untracked file: basic.py</td></tr>
<tr><td>$ git add basic.py
$ git st</td><td></td></tr>
<tr><td> 
</td><td>처음으로 스테이지에 들어온 파일:
new file: basic.py</td></tr>
<tr><td>파일 수정 후 실행
$ code basic.py # 또는 vscode 직접 수정 후 저장

$ python basic.py
$ git st</td><td>다음 코드 추가
print(list(range(1, 20, 3)))</td></tr>
<tr><td> 
</td><td>하나의 파일이 2개로 나뉨
new file: basic.py(stage 위치파일)
print(list(range(10)))

modified: basic.py(WD 위치파일)
print(list(range(10)))
print(list(range(1, 20, 3)))</td></tr>
<tr><td>PC@DESKTOP-482NOAB MINGW64 /c/OSS/git/03w/adv (main)
$ git rm --cached basic.py
error: the following file has staged content different from both the
file and the HEAD:
basic.py
(use -f to force removal)

PC@DESKTOP-482NOAB MINGW64 /c/OSS/git/03w/adv (main)
$ git rm --cached -f basic.py
rm 'basic.py'

PC@DESKTOP-482NOAB MINGW64 /c/OSS/git/03w/adv (main)
$ git st
On branch main

No commits yet

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

nothing added to commit but untracked files present (use "git add" to track)</td><td>$ git rm --cached fname
작업공간에는 남겨두고 스테이징 영역에서만 파일 삭제
파일 fname이 untracked 됨

현재는 basic.py 가 WD와 stage가 다르므로 삭제가 불가하여, 무조건 삭제 옵션 –f 사용 삭제

stage에서 제거하면
다시 untracked file이 되며,
내용은 위 modified file의 내용

print(list(range(10)))
print(list(range(1, 20, 3)))</td></tr>
</table>

	PC@DESKTOP-482NOAB MINGW64 /c/OSS/git/03w/adv (main) $ cat basic.py print(list(range(10))) print(list(range(1, 20, 3)))	

	$ git add basic.py
	$ git st
	$ git rm --cached basic.py
	$ git st
	$ git sts
		[교재 102, 103]
		stage에서 취소
		stage -> untracked file

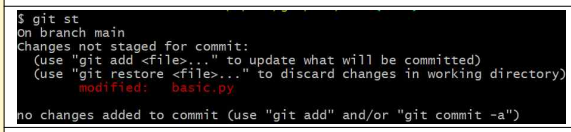
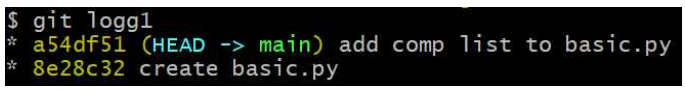
	$ git add basic.py
	
	PC@DESKTOP-482NOAB MINGW64 /c/OSS/git/03w/adv (main)
	$ git st
	On branch main
	
	No commits yet
	
	Changes to be committed:
	(use "git rm --cached <file>..." to unstage)
	new file: basic.py
	
	PC@DESKTOP-482NOAB MINGW64 /c/OSS/git/03w/adv (main)
	$ git rm --cached basic.py
	rm 'basic.py'
	
	PC@DESKTOP-482NOAB MINGW64 /c/OSS/git/03w/adv (main)
	$ git st
	On branch main
	
	No commits yet
	
	Untracked files:
	(use "git add <file>..." to include in what will be committed)
	basic.py
	
	nothing added to commit but untracked files present (use "git add" to track)
		stage area와 WD의 파일이 같으므로 git rm —cached basic.py 이 순조로이 가능

	$ git add basic.py
	$ git commit –m 'create basic.py'
	$ git log	

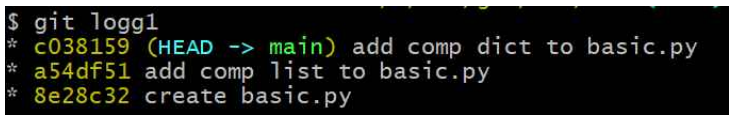
	$ git log commit 8e28c32d4b1e48c81b0af8253ada5c314b1d5b60 (HEAD -> main) Author: hskang <ai7dnn@gmail.com> Date: Mon Aug 29 15:50:44 2022 +0900 create basic.py	

	PC@DESKTOP-482NOAB MINGW64 /c/OSS/git/03w/adv (main)
	$ git st
	On branch main
	nothing to commit, working tree clean
	
	PC@DESKTOP-482NOAB MINGW64 /c/OSS/git/03w/adv (main)
	$ git rm --cached basic.py
	rm 'basic.py'
	
	PC@DESKTOP-482NOAB MINGW64 /c/OSS/git/03w/adv (main)
	$ git st
	On branch main
	Changes to be committed:
	(use "git restore --staged <file>..." to unstage)
	deleted: basic.py
	
	Untracked files:
	(use "git add <file>..." to include in what will be committed)
	basic.py
	
	PC@DESKTOP-482NOAB MINGW64 /c/OSS/git/03w/adv (main)
	$ git sts
	D basic.py
	?? basic.py
		커밋 이후 stage에서 삭제하면 다음 상태가 됨
		D basic.py # SA엔는 삭제된 상태(D)
		?? basic.py # WD에는 남아 untracked로 있음
		
		파일이 삭제되었다는 내역이 Staging Area에 추가됨과 동시에 원본 파일이 디스크에만 남게되고 추적에서 제외되었기 때문에 untracked 됨
		
		https://dololak.tistory.com/310

	PC@DESKTOP-482NOAB MINGW64 /c/OSS/git/03w/adv (main)
	$ git restore --staged basic.py
	
	PC@DESKTOP-482NOAB MINGW64 /c/OSS/git/03w/adv (main)
	$ git st
	On branch main
	nothing to commit, working tree clean
	
	PC@DESKTOP-482NOAB MINGW64 /c/OSS/git/03w/adv (main)
	$ cat basic.py
		다시 SA를 이전 상태로 바꾸는 명령, 결과적으로 이전 파일로 다시 SA가 됨

	print(list(range(10))) print(list(range(1, 20, 3)))	
모듈3 실습 (15min)	□ 파일 basic.py 총 3회 커밋, 이력 정보 확인	
	$ # 편집 $ git st	파일 basic.py , 다음 추가 편집 저장 print([i for i in range(10)])
		
	$ git commit –am 'add comp list to basic.py' $ git st	스테이지(stage)에 저장(tracked file)과 커밋을 한 번에
		깃 저장소(git repository) 저장, commit: '~에 적어두다'를 의미
	$ git config --global alias.logg1 'log --oneline --graph' $ git logg1	로그 확인
		깃 커밋 이력 정보 commit ID(7개의 16진수), (HEAD -> 브랜치명) 커밋메시지
	$ # 편집 후 저장 $ python basic.py	파일 basic.py 편집 저장 print({i:i*i for i in range(10)})

	$ git commit –am 'add comp dict to basic.py'
	$ git log
	$ git log basic.py
	$ git logg1
		커밋 전체 로그
		파일에 대한 커밋 전체 로그
		커밋 전체 로그 한 줄씩

		3개의 커밋이 보임
모듈4 실습	□ 다양한 커밋 정보 확인	
	$ git log commit c03815919954ae0fcda8ec2abc8d8de75af922e3 (HEAD -> main) Author: hskang <ai7dnn@gmail.com> Date: Mon Aug 29 16:44:59 2022 +0900 add comp dict to basic.py	커밋ID 저자 날짜 커밋메시지 $ git log head^ 이전 커밋부터 모두 표시, 그러므로 2개만 표시
	$ git log --pretty=short commit c03815919954ae0fcda8ec2abc8d8de75af922e3 (HEAD -> main) Author: hskang <ai7dnn@gmail.com> add comp dict to basic.py	[교재 126] 날짜만 없음
	$ git show head $ git show c038	[교재 127] head: 가장 최근 커밋 ID 4개 자리 이상 기술
		a/basic.py: 이전 파일 b/basic.py: 현재(커밋) 파일 이전 파일 줄 2에서, 3개 줄 이후(현재) 파일 줄 2에서, 4개 줄 @@ -2,3 +2,4 @@ 추가된 줄 +print({i:i*i for i in range(10)})

```
$ git show c038
commit c03815919954ae0fcda8ec2abc8d8de75af922e3 (HEAD -> main)
Author: hskang <ai7dnn@gmail.com>
Date:   Mon Aug 29 16:44:59 2022 +0900

    add comp dict to basic.py

diff --git a/basic.py b/basic.py
index 9a92761..7ce82bd 100644
--- a/basic.py
+++ b/basic.py
@@ -2,3 +2,4 @@ print(list(range(10)))
 print(list(range(1, 20, 3)))

 print([i for i in range(10)])
+print({i:i*i for i in range(10)})
```

$ git log --oneline
$ git log --pretty=oneline

```
$ git log --oneline
c038159 (HEAD -> main) add comp dict to basic.py
a54df51 add comp list to basic.py
8e28c32 create basic.py

PC@DESKTOP-482NOAB MINGW64 /c/OSS/git/03w/adv (main)
$ git log --pretty=oneline
c03815919954ae0fcda8ec2abc8d8de75af922e3 (HEAD -> main) add comp dict to basic.py
a54df51abd8723536016b302c079f0cadeb1606f add comp list to basic.py
8e28c32d4b1e48c81b0af8253ada5c314b1d5b60 create basic.py
```

$ git log –p 또는 --patch

[교재 127]
모든 로그에서 수정한 줄을 비교

```
$ git log -p
commit c03815919954ae0fcda8ec2abc8d8de75af922e3 (HEAD -> main)
Author: hskang <ai7dnn@gmail.com>
Date:   Mon Aug 29 16:44:59 2022 +0900

    add comp dict to basic.py

diff --git a/basic.py b/basic.py
index 9a92761..7ce82bd 100644
--- a/basic.py
+++ b/basic.py
@@ -2,3 +2,4 @@ print(list(range(10)))
 print(list(range(1, 20, 3)))

 print([i for i in range(10)])
+print({i:i*i for i in range(10)})

commit a54df51abd8723536016b302c079f0cadeb1606f
Author: hskang <ai7dnn@gmail.com>
Date:   Mon Aug 29 16:31:16 2022 +0900

    add comp list to basic.py

diff --git a/basic.py b/basic.py
index 3d44437..9a92761 100644
--- a/basic.py
+++ b/basic.py
@@ -1,2 +1,4 @@
 print(list(range(10)))
-print(list(range(1, 20, 3)))
\ No newline at end of file
+print(list(range(1, 20, 3)))
+
+print([i for i in range(10)])
```

a/basic.py(이전'앞' 파일): 이전 커밋
b/basic.py(이후'뒤' 파일): 현재 커밋
@@ −2,3 +2,4 @@
-이전파일 2: 2줄(줄 번호)에서 다음 3개의 줄을 참조
+이후파일 2: 2줄(줄 번호)에서 다음 4개의 줄을 참조

```
PC@DESKTOP-482NOAB MINGW64 /c/OSS/git/03w/adv (main)
$ git log --stat
commit 5c4d581dd76a1f91b022164d1b7d0a2818a018b7 (HEAD -> main)
Author: ai7dnn <ai7dnn@gmail.com>
Date:   Sat Sep 17 11:41:54 2022 +0900

add comp dict to basic.py

basic.py | 3 ++-
 1 file changed, 2 insertions(+), 1 deletion(-)

commit d354b85c936d8364ced53a1c67d6b209ce33e321
Author: ai7dnn <ai7dnn@gmail.com>
Date:   Sat Sep 17 11:39:57 2022 +0900

add comp list to basic.py

basic.py | 4 +++-
 1 file changed, 3 insertions(+), 1 deletion(-)

commit 2546149b2676e7fe6caa62807a600a7be92a0a7c
Author: ai7dnn <ai7dnn@gmail.com>
Date:   Sat Sep 17 11:07:20 2022 +0900
```

간단한 통계 정보 표시
$ git log --stat

<table>
<tr><td></td><td>create basic.py

basic.py | 2 ++
1 file changed, 2 insertions(+)</td><td></td></tr>
<tr><td></td><td>$ git log --stat -2</td><td>최근 n개 커밋 이력 보기
$ git log -3</td></tr>
<tr><td rowspan="3">모듈5
실습</td><td colspan="2">□ 깃허브 원격저장소 복제와 원격저장소 수정내용 가져오기(clone, pull)</td></tr>
<tr><td>$ git clone 주소_URL # 동일 폴더 생성해 복제
$ git clone 주소_URL . # 현재 폴더에 복제
$ git clone 주소_URL dname # dname 폴더 생성해 복제

# 폴더 03w에서 작업
$ cd ..
$ git clone https://github.com/ai7dnn/OSS-lect.git

$ git clone https://github.com/ai7dnn/OSS-lect.git
Cloning into 'OSS-lect'...
remote: Enumerating objects: 84, done.
remote: Counting objects: 100% (84/84), done.
remote: Compressing objects: 100% (82/82), done.
remote: Total 84 (delta 50), reused 5 (delta 2), pack-reused 0
Receiving objects: 100% (84/84), 7.57 MiB | 3.79 MiB/s, done.
Resolving deltas: 100% (50/50), done.</td><td>저장소와 동일 이름의 폴더 생성
현재 폴더에 저장소 생성
폴더 dname에 저장소 생성

우리 수업 저장소 복제</td></tr>
<tr><td>$ cd oss-lect # 폴더 이동

$ git remote # 원격 저장소 별칭 이름 조회
$ git remote –v # 원격 저장소 별칭 이름 세부 조회

$ git pull # 수정된 내용 다시 가져오기

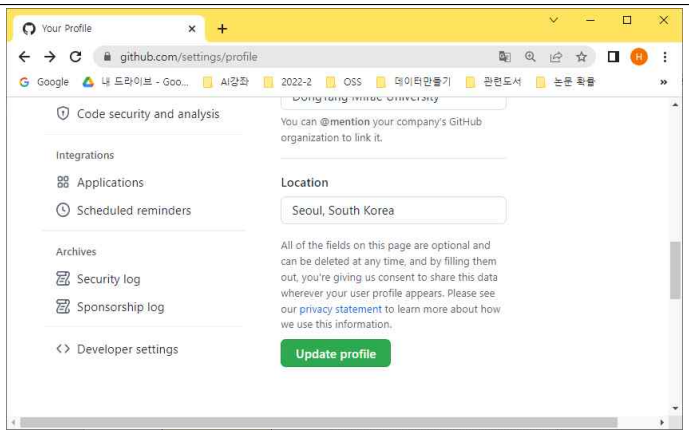
$ git remote
origin

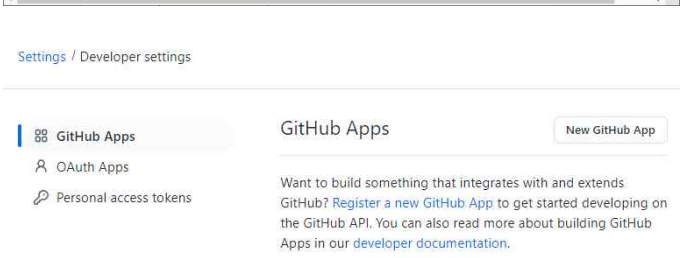
$ git remote -v
origin https://github.com/ai7dnn/OSS-lect (fetch)
origin https://github.com/ai7dnn/OSS-lect (push)

$ git pull
Already up to date.

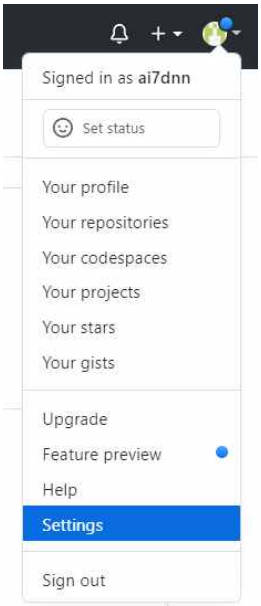
$ git pull
remote: Enumerating objects: 10, done.
remote: Counting objects: 100% (10/10), done.
remote: Compressing objects: 100% (8/8), done.
remote: Total 9 (delta 1), reused 9 (delta 1), pack-reused 0
Unpacking objects: 100% (9/9), 738.45 KiB | 1.99 MiB/s, done.
From https://github.com/ai7dnn/OSS-lect
f3d36c7..ab5064c main -> origin/main
Updating f3d36c7..ab5064c
Fast-forward
.../02\354\243\274 add commit log.pdf" | Bin 0
-> 407519 bytes
...4\230\201 \352\263\204\355\232\215\354\204\234.pdf" | Bin 0
-> 72208 bytes
...4\227\205 \354\204\244\352\263\204\354\204\234.pdf" | Bin 0
-> 165100 bytes
...4\235\230 \352\263\204\355\232\215\354\204\234.pdf" | Bin 0
-> 93314 bytes
...4\235\230 \354\204\244\352\263\204\354\204\234.pdf" | Bin 0
-> 112599 bytes
5 files changed, 0 insertions(+), 0 deletions(-)</td><td>서버 원격 저장소 복제 후 변화한 것이 없다면

$ git pull
Already up to date.</td></tr>
<tr><td>모듈6
실습</td><td colspan="2">□ 자신의 깃허브에 PAT 생성</td></tr>
</table>





사용자 하부에서 settings



암호를 파일에 저장

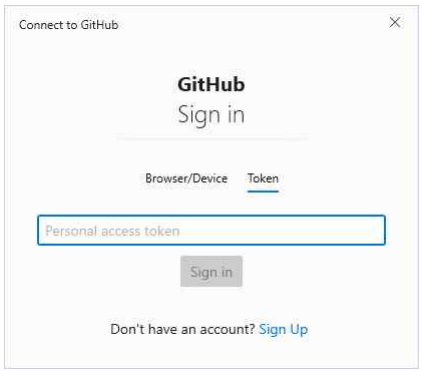
□ 저장소 repo-test 생성 후, 지역저장소와 연동하기

$ git clone 주소

파일 하나 생성

$ git push

다음 화면 표시



만일 현재 로그인되어 있다면 바로 Browser/Device 클릭

Token 누르고 PAT 정확히 복사 후 sign in 클릭

$ git push −u https://ghp_GeU8yVKNmKdIpZ787LHDb6HARqF8h@github.com/lee7py/Python−Programming.git

명령어에서 암호를 직접 지정하는 방법
$ git push -u https://{token}@github.com/{username}/{repo_name}.git

□ 깃과 깃허브 연동 실습

1. 자신의 깃허브에 저장소 myrepo 생성
2. 지역 저장소로 clone
3. 깃허브에서 파일 README.md 파일 수정
4. 지역 저장소로 pull
5. 지역 저장소에서 파일 hello.py 생성
6. 지역 저장소의 변화 내용(파일 hello.py 생성)을 깃허브로 push

$ git clone URL

$ git pull

$ git push