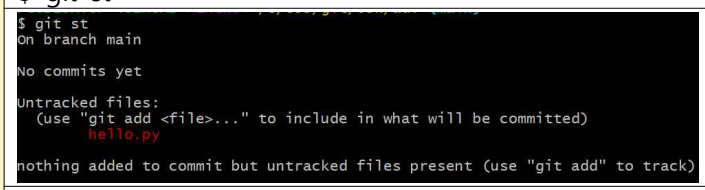
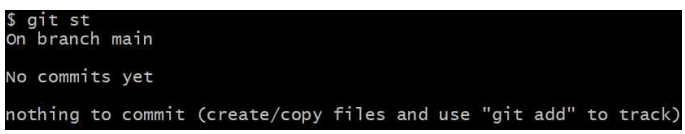
3주차 실습 시트

강 환수 교수

프로젝트	commit & log, clone & pull [교재 4장 5장]	
모듈1	□ 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	
모듈2	□ 다시 파일 생성해 추가한 후 stage에서 삭제, 복구 후 커밋	
모듈3	□ 파일 basic.py 총 3회 커밋, 이력 정보 확인	
모듈4	□ 다양한 커밋 정보 확인	
모듈5	□ 깃허브 원격저장소 복제와 원격저장소 수정내용 가져오기(pull)	
모듈6	□ 자신의 깃허브에 PAT(Personal Access Token) 생성	
준비	□ 다음 폴더를 생성 후 git/03w에서 git bash 실행 후 시작 □ vscode, github, source tree 준비	
모듈1 실습 (10min)	□ 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	
	$ git init adv $ cd adv	저장소 adv 생성

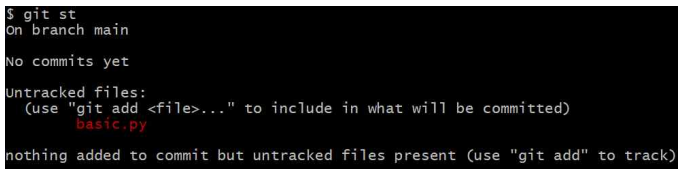
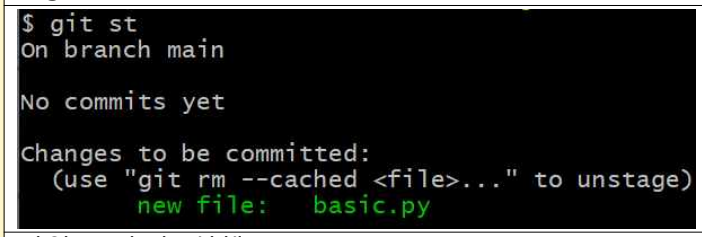
	$ git config user.name ai7dnn
	$ git config user.email ai7dnn@gmail.com
	
	$ git config core.autocrlf true
	$ git config core.safecrlf false
	$ git config core.editor "code --wait"
	$ git config --list
		사용자 환경 설정
		
		[교재 p41] 윈도와 맥의 캐리지 리턴과 라인 피드 변환 (윈도와 맥과 차이)
		편집기 환경 설정
		환경 설정 확인

	$ code hello.py
	print(list('python'))
	$ python hello.py
		vscode로 파이썬 파일 편집 후 저장 (ctrl + s) 후 실행

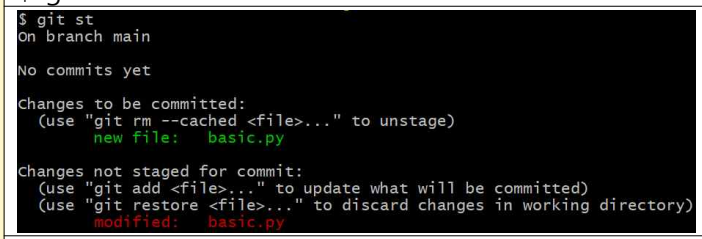
	$ git config --global alias.st 'status' $ git st	별칭을 생성해 실행
		새로 생성하면 untracked file
	$ rm hello.py $ git st	untracked file은 그대로 삭제 가능
		저장소에 커밋할 것이 없음

모듈2 실습 (15min)	□ 다시 파일 생성해 추가한 후 stage에서 삭제, 복구 후 커밋	

	$ code basic.py
	print(list(range(10)))
	$ python basic.py
	$ git st
		저장소 basic 생성해 저장(ctrl + s)
		
		깃 상태 보기

		untracked file: basic.py
	$ git add basic.py $ git st	
		처음으로 스테이지에 들어온 파일: new file: basic.py

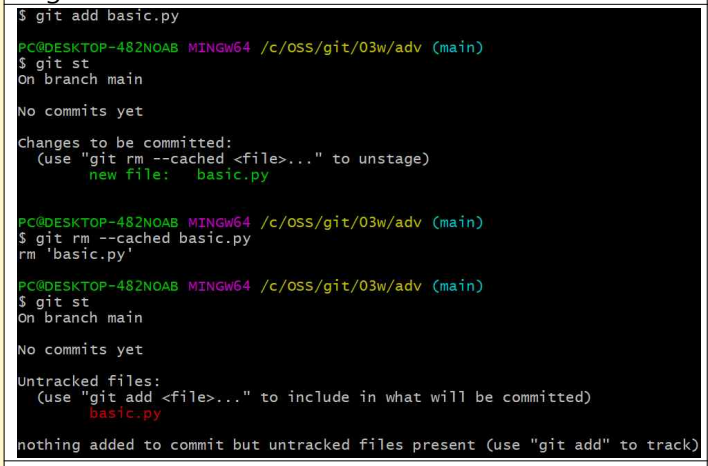
	파일 수정 후 실행
	$ code basic.py # 또는 vscode 직접 수정 후 저장
	$ python basic.py
	$ git st
		다음 코드 추가
		print(list(range(1, 20, 3)))

		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)))`

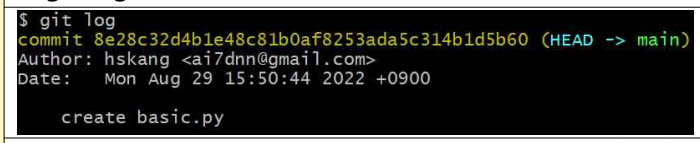
	PC@DESKTOP-482NOAB MINGW64 /c/OSS/git/03w/adv (main)
	$ git rm --cached basic.py
	error: the following file has staged content different from both the
	file and the HEAD:
	basic.py
	(use -f to force removal)
	
	PC@DESKTOP-482NOAB MINGW64 /c/OSS/git/03w/adv (main)
	$ git rm --cached -f basic.py
	rm 'basic.py'
	
	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)
	
	PC@DESKTOP-482NOAB MINGW64 /c/OSS/git/03w/adv (main)
	$ cat basic.py
	print(list(range(10)))
	print(list(range(1, 20, 3)))
		stage 내용 삭제
		
		현재는 head와 stage가 다르므로
		삭제가 불가하여, 무조건 삭제 옵션
		
		한 번도 커밋되지 않은 파일에서
		stage에서 제거하면
		다시 untracked file이 되며,
		내용은 위 modified file의 내용
		
		`print(list(range(10)))`
		`print(list(range(1, 20, 3)))`

	$ git add basic.py
	$ git st
	$ git config --global alias.sts 'status –s'
		[교재 102, 103]
		stage에서 취소
		stage -> untracked file

	$ git sts
	$ git rm --cached basic.py
	$ git st
	$ git sts	

		stage area와 WD의 파일이 같으므로 git rm —cached basic.py 이 순조로이 가능

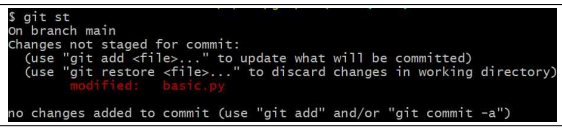
	$ git add basic.py
	$ git commit –m 'create basic.py'
	$ git log	

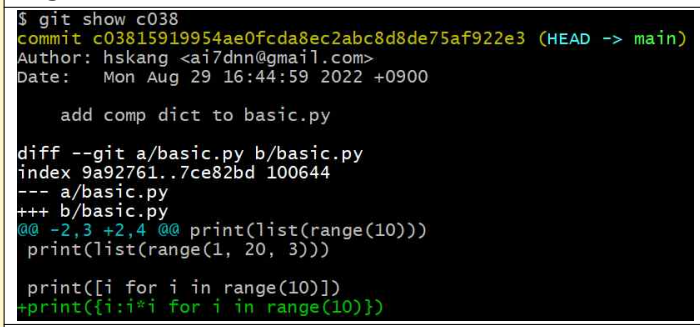
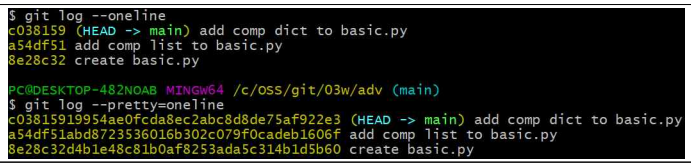
	PC@DESKTOP-482NOAB MINGW64 /c/OSS/git/03w/adv (main)
	$ git st
	On branch main
	nothing to commit, working tree clean
	
	PC@DESKTOP-482NOAB MINGW64 /c/OSS/git/03w/adv (main)
	$ git rm --cached basic.py
	rm 'basic.py'
	
	PC@DESKTOP-482NOAB MINGW64 /c/OSS/git/03w/adv (main)
	$ git st
	On branch main
	Changes 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 삭제
		
		SA에서 basic.py가 삭제되고
		
		원래대로 basic.py는 untracked가 됨

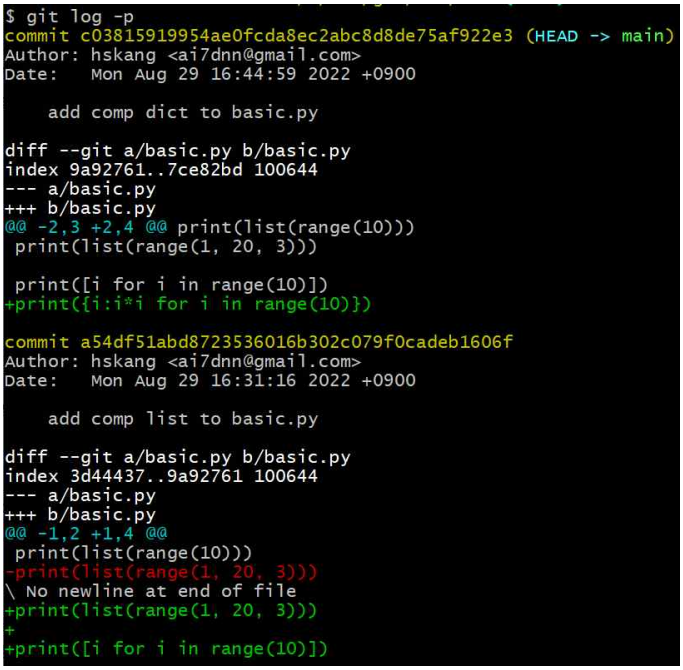
	PC@DESKTOP-482NOAB MINGW64 /c/OSS/git/03w/adv (main)
	$ git restore --staged basic.py
	
	PC@DESKTOP-482NOAB MINGW64 /c/OSS/git/03w/adv (main)
	$ git st
	On branch main
	nothing to commit, working tree clean
	
	PC@DESKTOP-482NOAB MINGW64 /c/OSS/git/03w/adv (main)
	$ cat basic.py
	print(list(range(10)))
	print(list(range(1, 20, 3)))
		다시 SA를 이전 상태로 바꾸는 명령, 결과적으로 이전 파일로 다시 SA가 됨

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 c038159191954ae0fcda8ec2abc8d8de75af922e3 (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 c038159191954ae0fcda8ec2abc8d8de75af922e3 (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 log --oneline $ git log --pretty=oneline	
		

	$ git log –p 또는 --patch	[교재 127] 모든 로그에서 수정한 줄을 비교
		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
모듈5 실습	□ 깃허브 원격저장소 복제와 원격저장소 수정내용 가져오기(clone, pull)	

	$ 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 02w에서 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.
		저장소와 동일 이름의 폴더 생성
		현재 폴더에 저장소 생성
		폴더 dname에 저장소 생성
		
		우리 수업 저장소 복제

	$ 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.
		서버 원격 저장소 복제 후 변화한 것이 없다면
		
		$ git pull
		Already up to date.

	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(-)	
모듈6 실습	□ 자신의 깃허브에 PAT 생성	
		암호를 파일에 저장
	□ 저장소 repo-test 생성 후, 지역저장소와 연동하기	

	$ git push -u https://ghp_GeU8yVKNmKdIpZ787LHDb6HARqF8h@github.com/lee7py/Python-Programming.git
	
	$ git push -u https://{token}@github.com/{username}/{repo_name}.git	