**Github깃허브 사용법**

텍스트이(가) 표시된 사진

자동 생성된 설명

<https://tagilog.tistory.com/377>

1. 파일 넣기전 push일때 오류 생기면

토큰을 다시 받아서 최신화 시켜야 함.

<https://sosoeasy.tistory.com/536> 참조

**git 명령어 치면 사용법 나옴**

git -–version

🡪 버전 확인

/Users/kang0/Desktop/LAB/Lab-git 🡪폴더 위치.

git config –global user.name La-folia

gif config –global user.email kang0481@naver.com

먼저 git hub에서 경로를 복사해서, 저장해주고, 터미널에서 폴더로 이동하기 **CD 이용( ls는 list)**

이동 후

Git clone https://github.com/La-Folia/Numerical-Analysis.git

**파일 넣는법**

git add ~~~.txt

git commit -m”Add Text File [~~~.txt]”

🡪커밋을 진행하면서 실제로 파일이 업로드 한 시점을 스냅샷으로 찍어놓는 것. 나중에 프로젝트가 문제가 생겼을 때 원상태로 복구하는 목적으로 사용할 수 있음.

🡪아직 github에는 적용 X

그리고 git push를 이용해서 push로 원격 repository에 넣을 수 있음.

-Push한 이유에는 사이트를 새로고침하면, 특정한 파일이 올라간 것을 볼 수 있음. 방금전 업로드 한 것을 볼 수 있음. 이렇게 git hub의 프로젝트를 다 올려서 관리 가능.

local repository (내컴)

remote repository(github

**오픈 소스**

공개된 소스코드, 특정한 소프트웨어를 개발한 개발자의 권리를지키면서 누구나 소스코드 확인 가능

-대학교 시절 온갖 컴퓨터 과제의 도우미 같은 존재.

상당히 많은 라이브러리가 오픈 소스 🡪 부트스트랩,(예) 가장 유명한 웹 디자인 프레임 워크 중 하나.

Bootstrap

트위터에서 존재하는 부트 스트랩을 볼 수 있음. 단순하게 프로젝트에 포함시킬 뿐만아니라 부트스트랩 소스코드 자체도 공부하고 오픈소스에 기여할 수 있다는 것. 소스코드를 수정하거나 포함되어 있는 아이디어를 제공한다던지, 다양한 면에서 이 오픈소스에 기여를 할 수 있다는 것.

별이 많이 박힌 오픈소스일 수록 강력하고 좋은 소스가 굉장히 많기 때문에(1000개만 넘어도 정말 유명한 오픈 소스, 별때문에 자괴감이 있기도 함. 깃허브에는 다양한 프로젝트를 검색하고 확인하기만 해도 굉장히 많은 정보를 얻을 수 있다는 것.)

Clone해서 컴퓨터에 옮겨서 사용할 수도 있음. 유명한 open source의 경우 read.me라고 해서 처음 생김.

🡪read.me는 어떻게 설치, 어떻게 사용하는지 소스코드를 뜯어볼 수 있음

Contribution?

말그대로 기여한다는 의미, 기능을 추가하거나 보완 수정, 오타, 의견제시 등도 모두 contribution이라 할 수 있음. 누구나 할 수 있고 오픈 소스에 참여해서 실력을 늘리고 도움을 줄 수 있음

Why? Contribution?

많은 회사에서 뽑으때, git hub주소를 보내기도함, 등등

정말 많은 이유때문에 상당히 많은 사람들이 오픈소스를 하고 있음 여기서

Comitter?

관리하고 있는 오픈소스에 반영할지 안할지를 결정하는 사람

실제로는 정말 많은 contribution이 있지만, 정말 관리하는 사람은 따로 있음. 소스코드를 바꾸고 수정할 수 있으면, 위험하니, **몇십명 이내로 구성하는 것이 일산적.**

1. **사회공헌**
2. **소프트웨어의 품질 향상**
3. **어쩔 수 없이**

**MIT livense: 2차 저작물 공개 의무 없음, 무료, 배포**

**APahacee: 2차 저작물 공개 의무 없음, 무료, 배포**

**GPL은 다름, 무료, 배포 가능 하고 소스코드 수정이 가능하지만 2차 저작물 공개 의무 있음**

**Beerware:만나면 그냥 술이나 사주자.**

**Git 의 장점**

Git 하나의 프로젝트를 여러 사람들과 효율적 관리 가능. 여러면에서 프로젝트를 하면 충돌할 수 있지만, git은 병렬적으로 동시에 개발하게 해주면서, 협업할 수 있도록 도와줌. 본인이 활동한 것을 git에 나타나니까, 포트폴리오 관리에도 용이, 얼마나 주기적으로 commit을 했는지, 누구나 볼 수 있는 공간에 비치하면서 실제로 요즘에는 프로그래머들이 구인 구직 했을 때 git hub 계정명을 넣기도 함.

Git을 사용하지 않았을때의 문제점을 알려주면, 그것이 git의 장점이다. 깃을 사용하지 않으면 **소스코드를 USB에 담아서 다른 사람의 컴퓨터로 옮기는 방식 등으로 프로젝트를 공유해야함**. 동시에 프로젝트하기가 어렵다.

-**분산적인 개발**: 깃을 사용하는 전체 개발 내역을 각 개발자의 로컬 컴퓨터로 복사할 수 있음. 나중에 서로 수정된 내역을 합치기(**Merge**)할 수도 있으며 이 때 git의 고유한 프로토콜을 이용하게 된다.

-**효율적인 개발**: 깃은 일반적인 다른 버전 관리 시스템보다 성능이 뛰어남, 변경 이력 많아도 변경된 내용만 처리

-**비선형적인 개발**: **git**은 **branch**라는 개념이 사용되는데 프로젝트의 가지치기가 가능함. 트리구조, 다시 말해서 비선형적인 구조라고 볼 수 있음

-**변경 이력 보장**: 작업된 모든 내역들은 모두 별도의 영역에서 관리되어 안전하게 프로젝트를 운영할 수 있다.

**Git의 동작 원리**

파일 자체를 다 저장하기 보다 수정 내역 자체를 저장한다.

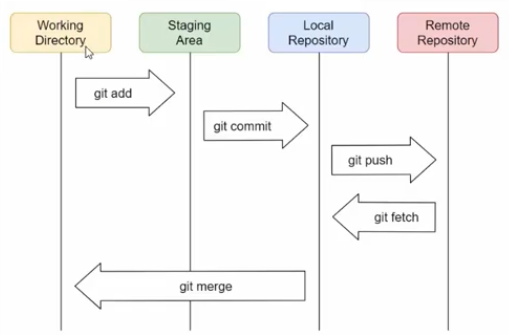
Git의 동작 원리를 바르게 이해하기 위해서는 세가지 구성 요소에 대하여 이해할 필요가 있음

**working** **Directory**:작업할 수 있는 디렉토리

**Staging** **Area**: **Commit**을 수행할 파일들이 올라가는 영역

실제로 commit을 수행할때, 작업한 내용을 반영하면 repository에 반영됨

**Git** **Directory**: git 프로젝트의 메타 데이터와 데이터 정보가 저장되는 디렉토리



만약 어떤 파일을 add해서 commit을 하면 git folder에 기록됨 컴퓨터 로컬에 기록, remote에는 반영이 안됨. 즉 commit은 로컬에만, push를 해야 remote repository에 반영이 된다.

**다른 사람이 만든 것을 다운 받고 적용하려면 어떻게 해야 하나?**

**🡪 git fetch**

그리고 특정한 파일에서 충돌이 발생하는 경우 merge를 하면서 컴퓨터와 repository가 동일화 됨.

다른 사람의 내용을 동기화 하려면 fetch와 merge를 사용해야함.

**그걸 각각 수행하면 귀찮기에 git pull을 일반적으로 많이 사용함.**

**Git Add, commit, push**

**Git pull**

**저장소 repository(repo)**

실제 소스코드가 담겨 있으면서 commit내역 등의 모든 작업 이력이 담겨 있는 공간. 환경설정과 관련한 폴더와 관련된 폴더들이 있음. 여실때는 에듀터 프로그램으로도 열 수 있음.

-각종 데이터와 해시 값등이 담겨져 있다.

**각각의 작업 내용을 구분하기 위한 목적, 내부적으로 특정한 작업값에 대해 hash값을 이용하는 것.**

**Git 명령어만 확실하게 사용할 수만 있으면 됨**

**Git과 관련된 reference를 공부하고 분석하는 것도 유익할듯.**

**소스코드를 수정해서 자신의 git 저장소에 저장하는 방법**

이번 시간에 보다 자세하게 소스코드를 수정해서 반영하는지를 알아 볼 것.

1. 해당 프로젝트에 소속된 사람이 아닌 경우

스스로 commit을 하여 저장소에 적용할 권한이 없다면, 소스코드를 수정하는 것에 제약이 있음. 이런 경우 PR(Pull Request)를 작성하여 오픈소스에 기여할 수 있다.

1. 해당 프로젝트에 소속된 사람인 경우

자신이 해당 프로젝트에 대한 권한을 가지고 있으므로 commit push해서 수정 내역이 반영이 됨.

**2번인 경우 어떻게 소스코드를 수정해서 반영하는지를 실습할 예정**

Git status로 git add로 올려져있는 것을 확인할 수 있음

텍스트이(가) 표시된 사진

자동 생성된 설명

Git reset 하면 add안되있게 만들 수 있음.

Git add . 하면 모든 프로젝트들이 다 올라서 반영됨

텍스트이(가) 표시된 사진

자동 생성된 설명

**텍스트이(가) 표시된 사진

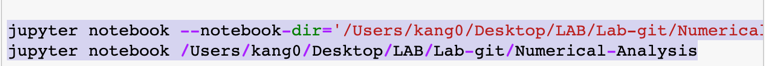
자동 생성된 설명**

git commit -m"Add my\_module [Add]"

하고 push 하면 됨.

**처음 주피터를 킬때부터 정리하자면,**

1. **터미널에서,**

**** jupyter notebook --notebook-dir='/Users/kang0/Desktop/Numerical-Analysis/Root\_Finding'

\*만약 여기서 폴더명을 바꿀려면 둘다 뒤에 새로운 디렉토리 폴더 명 기입하면 된다.

**입력. 🡪 주피터 실행**

1. **적절하게 코드 작성.**

jupyter nbconvert --to script example.ipynb

해주기.

1. 터미널에서 아래의 과정을 거치기.

git add ~~( . <- 전부 올리기)

git status로 확인해보기

git commit -m"Add my\_module [Add]"

git push

\*git checkout --이름

🡪수정된 것을 다시 원래 상태로 돌리기. 소스코드를 다시 돌리는 역할.

텍스트이(가) 표시된 사진

자동 생성된 설명

+나중에 commit내용을 amend로 수정할 수 있다.(이후에 나옴)

**commit을 되돌리는 법**



이걸로 commit을 볼 수 있음.

특정 프로젝트가 언제 수정됬는지 commit내용을 보면서 현재 포함되어 있는 개발자들이 파일을 수정했는지 확인할 수 있음

-즉, commit 내역들을 수정하면서 초기의 상태로도 돌릴 수 있다.

먼저 HTTPS에서 clone해서 해놓기.

텍스트, 테이블이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명

**수정해서 되돌리는 것 까지의 과정**

코드 수정하기.

git add .

git commit -m “add my\_module [mul]”

git status : commit은 저장되어 있고, push는 안한 것을 볼 수 있음.

git push 🡪 git hub 저장소에 수정된 내용 반영.

반영된 내역들은 git log를 보면 확인 할 수 있다.

텍스트이(가) 표시된 사진

자동 생성된 설명

앤터를 누르면 기록을 더 볼 수 있음

로그를 그만 보고 싶으면 q를 누르면 됨.

-각각의 commit내용은 각각의 해시값으로 매칭이 되고 있다.

-즉 해시값으로 그 commit을 지칭할 수 있다. 예전에 commit한 상황을 되돌리고 싶으면 해시값을 이용한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

**git reset – hard~~~**

hard와 해시를 이용해서, 그 전까지는 없던걸로 commit을 다 제거하는 능력.

다시한번 log를 찍으면

텍스트이(가) 표시된 사진

자동 생성된 설명

soft는 돌아가도 돌아가기 전까지 보관

mix도 존재 🡪 검색해서 확인할 것.

실제로 해당 사항으로 돌아간 것을 볼 수 있음

파이썬 코드를 추가했고 이런 것들은 다 사라지고 commit한 상황의 그 상태로 완전이 돌아가게 된다. 이 상태는 local 상태와 깃허브 상태가 전혀 다른 상태이기 때문에, 특정 지점에 돌아간 지점을 다시 git hub에 저장해야함.

🡪 이때 git push를 하면 오류 메세지가 뜸. 구성이 완전이 달라졌기 때문에, rejected가 뜬 것을 볼 수 있음

텍스트이(가) 표시된 사진

자동 생성된 설명

**git push -f**

🡪하지만 강제로 -f로 강제 푸쉬를 할 수 있음.

이러면 git hub 상태도 local상태처럼 이전 상태로 확 돌아갈 수 있음 commit내용이 싹 사라진 것을 볼 수 있음

document.txt(초기에 메모장을 넣었을때)처럼 초기 버전으로 돌아갈 수 있음.

**Commit 내용을 수정하는 방법**

코드를 수정하고나서 먼저 push를 해준다. (사이트에 먼저 적용, commit만 수정하면 내용은 수정되지 않는다.)

1. git commit –-amend

자동으로 특정한 editor이 설정됨. Unix 에디터가 됨. 윈도우는 기본적으로 수정모드와 관리자 모드로 나누어지는데, 수정모드는 E로 🡪I누르면 INSERT 모드가 됨.

텍스트이(가) 표시된 사진

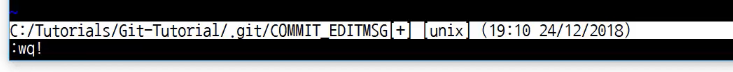
자동 생성된 설명

이때 수정하고

esc누르고



:wq!



Commit 메세지가 수정되고 원래 화면으로 넘어감.

git log로 add와 sub이 수정된 것을 볼 수 있음

텍스트이(가) 표시된 사진

자동 생성된 설명

그리고 다시 git push -f로 적용 🡪 git hub사이트도 변경된다.

git hub blog

글 올리는 법.

<https://theorydb.github.io/envops/2019/05/22/envops-blog-how-to-use-md/>

마크다운 사용법 참조.

먼저 주피터 파이썬 내용을 사이트에 올리는 방법을 정리해보겠다.

1. 먼저

jupyter nbconvert --to markdown notebook.ipynb

로 먼저 마크다운 만들어주면, 파일이 두개가 만들어지는데

* notebook.md : jupyter notebook 파일을 깃헙 블로그 양식에 맞춘 markdown 파일
* notebook\_files : jupyter notebook에서 사용되었던 모든 사진이 들어있는 파일 notebook.md 파일을 \_posts폴더로 옮긴다. 다음과 같이 title, data, categories등의 속성이 미포함된 상태이므로 상단에 양식에 맞춰 추가해준다.

1. 이때 md 파일은 \_posts로, notebook\_files(즉 안에 사진들어있는 파일)은 assets/img/blog에 넣어준다.
2. 그리고 notebook.md 파일에서 jpg나 png의 형식 앞에 이 주소 처럼 고쳐준다.

![png](/assets/img/blog/Matplotlib\_files/Matplotlib\_3\_0.png)

1. git add . / commit/ push 과정을 거친다.

git hub blog 사이트 수정 & 목차 추가

프로필 사진, 배경 사진 등 블로그 구조는

\_config.yml 로 수정하면 됨

만약 목차를 추가하고 싶다면, 저파일에서 목차를 추가시키고,

텍스트이(가) 표시된 사진

자동 생성된 설명

이렇게 사이드바 추가시킨다음,

올리고자 하는 파일의 md에서

텍스트이(가) 표시된 사진

자동 생성된 설명

permalink에다가 저 url을 적으면 끝.

Plotly같은 경우에는 그림 파일이 바로 본떠서 나오지 않기 때문에, 약간의 절차가 필요하다.

아래 사이트 참조

<https://davistownsend.github.io/blog/PlotlyBloggingTutorial/>