

CONCEPT & TUTORIAL

GIT & GITHUB

GIT & GITHUB

0		
	소개	
1	Git 설치 & 기본 설정	
2		로컬에 첫 Git 올리기
3		
	GitHub repository 생성	Github repository에 local git 연결
4		
5		Github에서 repository clone
6		
	Collaborator 초대하기	
7	초대된 collaborator 입장에서 push 해보기	
8		왜 push가 안될까요?
9		
	왜 pull이 안될까요?	Branch, merge
10	실수로 다 날려버렸어요	
11		

GIT & GITHUB

12

Git gui tool 소개

15

README.md

18

Github 말고 Gitlab도 있어요

13

Text-mode interface for git

16

거의 모든 Git 명령어

14

Open source에 기여하기

17

Git 때문에 생긴 파일

0. 소개 : GIT과 GITHUB의 관계

GIT

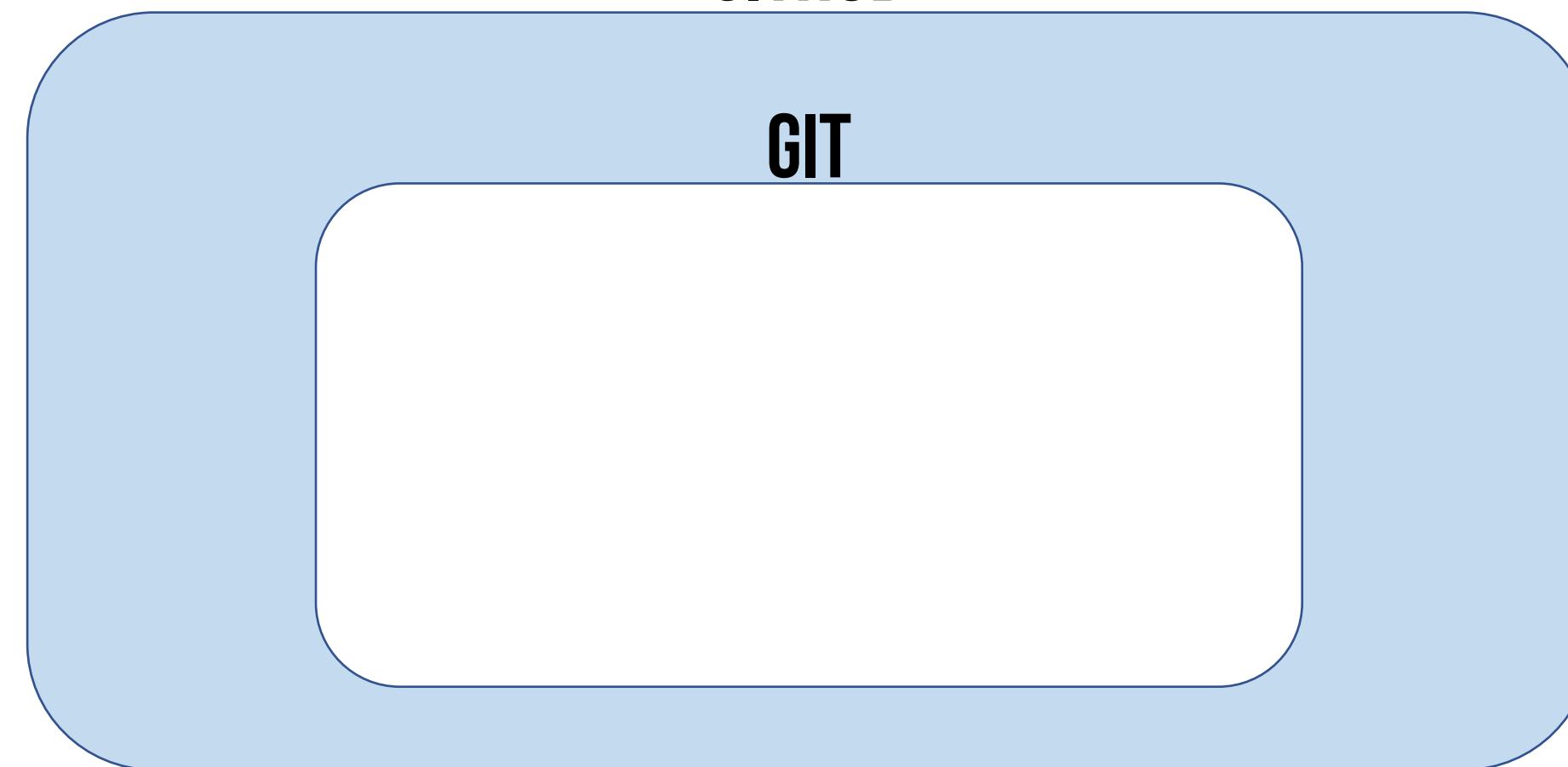
SVN, CVS 등과 유사한 분산 버전 관리 시스템의 한 종류이다. 소스 관리, 수정 내역이 commit 단위로 기록되며, git 소프트웨어에서 push/pull 명령어를 사용하여 원격 저장소에 접근하여 프로젝트 관리를 할 수 있는 기능을 제공한다.

GITHUB

git의 데이터 원격 저장소를 제공해주는 웹서비스, github를 사용할 시 git 데이터를 한 개의 local 컴퓨터에 종속되지 않게 해주어 소스 이동이 자유롭고, 배포도 용이하다.

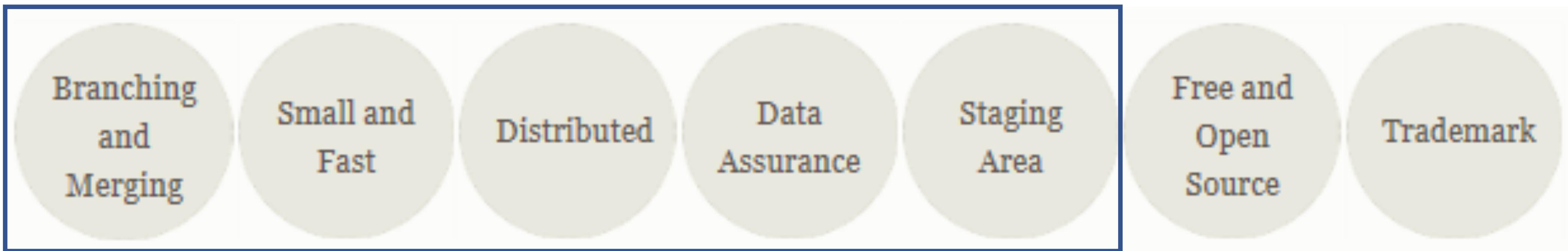
GITHUB

GIT



**GIT의 특징과 GITHUB의 특징을
구별해야한다.**

GIT : FEATURE



자세한 내용 : <https://git-scm.com>

GITHUB : FEATURE



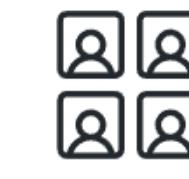
Code review



Project management



Integrations



Team management



Social coding



Documentation



Code hosting

자세한 내용 : <https://github.com>

1. GIT 설치 & 기본 설정

- Git 설치
\$ apt-get install git
- 이름과 이메일 설정(—은 - 2번)
\$ git config —global user.name “your name”
\$ git config —global user.email “your email”
- 설정 확인
\$ git config —list
- Git 도움말
\$ git help
- 명령어에 대한 상세 도움말
\$ git help <명령어>

2. 로컬에 첫 GIT 올리기: 기존 LOCAL 프로젝트를 GIT으로 관리하고 싶을 때

- .git 디렉토리 생성
\$ git init
- git으로 관리하고 싶은 파일/폴더 추가
\$ git add *.py
\$ git add /net
- 혹은 모든 폴더와 파일을 추가
\$ git add .
- 추가된 파일을 커밋 : 관리의 단위가 commit이기 때문에, commit을 해야지만 local에서 관리가 이루어짐
\$ git commit -m “First Commit”
- 추가 팁
모든 폴더와 파일을 추가하면서 커밋을 할 때 한 번에 처리하는 명령어(-a option 추가) :
\$ git commit -a -m “First Commit”

3. GITHUB REPOSITORY 생성

GitHub에서 새로운 repository 생성(Initialize this repository with a README 체크 해제)

Create a new repository

A repository contains all project files, including the revision history.

Owner

 heypaprika / _git&github_test 

Great repository names are descriptive and unique. Your new repository will be created as `_git-github_test supreme-computing-machine?`

Description (optional)

 Public

Anyone can see this repository. You choose who can commit.

 Private

You choose who can see and commit to this repository.

Initialize this repository with a README

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None ▾

Add a license: None ▾ 

Creating repository...

4. GITHUB REPOSITORY와 LOCAL GIT 연결: LOCAL 프로젝트를 GITHUB에서 관리하고 싶을 때

- 먼저 터미널을 이용하여 프로젝트 위치로 이동
 \$ cd project_dir
- 로컬에 .git 폴더가 있는 상황인지 \$ ls -al | grep .git로 확인
- commit을 한 내역이 있는지 \$ git log로 확인
- repository의 quick setup page의 https 주소 오른편 복사 버튼 클릭하여 복사.
- 이후 명령어 진행
 \$ git remote add origin <copied>
 \$ git push -u origin master

5. GITHUB에서 REPOSITORY CLONE

- Clone : Github에 있는 repository를 가져오는 명령어
- GitHub에서 clone or download 버튼을 클릭하면 project의 URL주소가 복사된다.
- 터미널에서 프로젝트를 둘 상위 디렉토리로 이동
`$ cd <upper_dir>`
- 클론
`$ git clone <URL>`
- 혹은 project의 dir 이름을 명시하여 클론
`$ git clone <URL> <dir>`

6. COLLABORATOR 초대하기

특정 repository에 대해서 collaborator가 아니면, 해당 repository에 push할 수 없다. 따라서 한 프로젝트를 같이 작업하기 위해서는 협업할 인원을 collaborator로 등록해주어야 한다.

The screenshot shows the GitHub repository settings page for 'heypaprika / github_page_sample'. The 'Settings' tab is selected. On the left, a sidebar menu includes 'Options', 'Collaborators' (which is currently selected), 'Branches', 'Webhooks', 'Notifications', 'Integrations & services', and 'Deploy keys'. The main content area is titled 'Collaborators' and contains the message: 'This repository doesn't have any collaborators yet. Use the form below to add a collaborator.' Below this is a search bar labeled 'Search by username, full name or email address' with the note: 'You'll only be able to find a GitHub user by their email address if they've chosen to list it publicly. Otherwise, use their username instead.' At the bottom right of the search bar is a 'Add collaborator' button.

7. 초대된 COLLABORATOR 입장에서 PUSH 해보기

- collaborator가 된 project를 clone하고, pycharm에서 해당 project를 열어서 파일 추가 혹은 수정해본다
- 이후의 명령어들로 push 해보기
 - \$ git commit -a -m “modified the file”
 - \$ git pull
 - \$ git push

8-1. 왜 PUSH가 안될까요?: COLLABORATOR가 아닌데, 다른 사람이 만든 REPOSITORY에 PUSH가 가능한가요?

- 불가능합니다.
- Collaborator 대신 contributor가 되는 방법은 12장에서 소개합니다.

8-2 왜 PUSH가 안될까요?: PULL과 PUSH의 밀접한 관계

- \$ git clone 과 \$ git push 사이 또는 이미 클론을 했으면 \$ git pull 과 \$ git push 사이에 다른 collaborator가 push를 먼저 했을 경우, \$ git push 명령어를 입력했을 때, pull을 먼저 하라는 메시지가 뜨게 된다.
- 이 메시지가 뜬다면 \$ git pull을 하여 다른 사람이 작업한 소스를 받은 후에 \$ git push로 자신의 코드를 올려야 한다.
- \$ git pull을 하고 \$ git push를 하였지만 push가 되지 않는다면, 충돌이 난 것이고 다음 장을 확인한다.

8-3. 왜 PUSH가 안될까요?: 동료와 동일한 부분을 수정했나봐요.

- \$ git push 가 될 때는, 동료의 소스와 나의 소스간에 충돌이 없는 경우라 이상이 없는 것으로 생각하여 넘어가면 되지만, \$ git push 가 되지 않을 때는 같은 소스 파일의 같은 부분을 수정하여 충돌이 생긴 것이다.
- 이렇게 충돌이 났을 때에는 먼저 push를 한 동료를 찾아가서 같이 소스를 보면서 작업하고 해당 부분을 어떻게 처리할 지 결정을 지어서 코드를 수정한다.
- 수정을 완료했으면, 이후 add 부터 다시 진행한다.

```
Auto-merging calculator/calc.py
CONFLICT (content): Merge conflict in calculator/calc.py
Automatic merge failed; fix conflicts and then commit the result.
```

pull 할 때 충돌된 경우 메시지

```
26     class trigonometric_function:
27         pass
28
29         <<<<< HEAD
30     class differential_integral:
31         def __init__(self):
32             self.differential = []
33             self.integral = []
34
35         ====
36     class arithmetic_operation:
37         pass
38     >>>>> 8d54a5ac5166711fd74fb8924154a988e55dad30
39
```

pull 할 때 충돌된 경우 결과

9. 왜 PULL이 안될까요?

- 해결 : stage
- Pull Error : Please, commit your changes or stash them before you can merge.
\$ git stash
\$ git pull
- Pull Error : Please move or remove them before you can merge.
\$ git add -A
\$ git stash
\$ git pull

10. 실수로 다 날려버렸어요

- \$ git add 취소하기 : git add 명령어로 추적을 원하지 않는 파일을 추가한 경우, 또한 아직 commit을 하지 않은 경우에 add한 파일이 있는 공간(staging area)에서 제외하는 것이 가능하다.
 - \$ git reset HEAD <filename>
- \$ git commit 취소하기 : 작업을 끝내지 않았는데 commit을 해버린 경우, 또한 add를 덜 한 경우에 commit을 취소하는 것이 가능하다. 옵션 3가지가 있다.
 - \$ git reset [–soft/–mixed/–hard] HEAD^
 - soft : add한 파일들을 staged 상태로 작업 폴더에 보존
 - mixed(default) : add한 파일들을 unstaged 상태로 작업 폴더에 보존
 - hard : add한 파일들을 unstaged 상태로 하고 작업 폴더에서 삭제
- \$ git push 취소하기 : 이 방법은 자신의 Local 내역을 원격 저장소에 강제로 덮어쓰기를 하는 방법이므로 주의해야 한다. 되돌아간 commit 이후의 모든 log 와 수정된 사항이 사라지게 된다.
우선 git commit 를 취소한다.
 - \$ git reflog 로 HEAD 번호 확인
 - \$ git reset HEAD@{<number>} 로 해당 commit으로 이동
 - \$ git commit -m “reset push”
 - \$ git push origin <branch_name> -f 또는 \$ git push origin +<branch_name>

10. 실수로 다 날려버렸어요 [잘 날리는 팁]

- \$ 원격 저장소에 잘못 올라간 파일 삭제하기 : push를 하였는데 잘못 포함한 파일이 있는 경우 그리고 전체를 되돌리기에는 무리인 경우, 해당 파일을 명령어로 제외할 수 있다.
 - 원격 저장소와 로컬 저장소 모두 해당 파일을 지운다.
`$ git rm <file_name>`
 - 원격 저장소에 있는 파일만 지운다.
`$ git rm --cached <file_name>`
 - 이후 commit과 push를 진행한다.
- + \$ untracked 파일 삭제하기 : 추적중이지 않은 파일을 자신의 작업 폴더에서 지울 수 있다. 기본적으로 gitignore에 명시된 파일은 삭제하지 않는다.
 - 디렉토리를 제외한 root에 있는 파일들만 삭제
`$ git clean -f`
 - 디렉토리까지 삭제
`$ git clean -f -d`
 - gitignore에 등록한 파일까지 삭제
`$ git clean -f -d -x`

11. BRANCH, MERGE

- 브랜치는 repository의 안정성을 보장할 수 있다. project를 완성하고 배포한 경우에 소스 수정 혹은 디버깅 해야하는 일이 생겼을 때, 완성된 소스를 직접 수정하기보다는 브랜치를 두어 프로젝트의 복사본을 만들어 작업을 하고, 작업이 완료된 브랜치를 메인에 merge 하는 방법이 서비스 측면에서 안정적인 방법이다.
- 브랜치 생성
\$ git branch <branch_name>
- 해당 브랜치로 전환
\$ git checkout <branch_name>
- 이후의 작업은 <branch_name>에서 이루어진다.
- add 와 commit 명령은 동일하다.
- push
\$ git push origin <branch_name>

11. BRANCH, MERGE

- master에 작업한 branch를 합치고(merge) 싶을 때는 마스터로 이동하고, merge 한다.
 \$ git checkout master
 \$ git merge <branch_name>
- 참고 : \$ git pull 명령어는 fetch + merge 이다.

12. GIT GUI TOOL 소개

- <https://git-scm.com/downloads/guis> 에 다수의 Gui tool이 소개되어 있다.
- GitHub desktop, source tree 등의 툴이 있다.
- git을 처음 사용하는 사람들에게 터미널보다는 더 직관적이다.
- 그러나 편하게 사용하기 위한 설정이 비교적 복잡한 편이다.

13. TEXT-MODE INTERFACE FOR GIT

- `tig` 설치하기

```
$ sudo apt install tig
```

git-github_test — tig — 80x24

2019-06-20 18:25 +0900 Unknown o Unstaged changes

2019-04-08 21:31 +0900 Chang Hyun Lee M [master] {origin/master} {origin/HEAD}

2019-04-08 21:19 +0900 CH Lee M Merge branch 'master' of https://git

2019-04-08 21:19 +0900 CH Lee o edited

2019-04-08 21:20 +0900 Chang Hyun Lee o commit

2019-04-08 21:14 +0900 Chang Hyun Lee M Merge branch 'master' of https://git

2019-04-08 21:13 +0900 CH Lee o modified calc

2019-04-08 15:38 +0900 CH Lee o test4

2019-04-08 15:36 +0900 CH Lee o test3

2019-04-08 15:29 +0900 CH Lee o test2

2019-04-08 15:29 +0900 CH Lee o test

2019-04-08 21:14 +0900 Chang Hyun Lee o modified calc

2019-04-08 15:20 +0900 yanghoJI o add README

2019-04-08 14:35 +0900 Chang Hyun Lee M merge col

2019-04-08 14:27 +0900 yanghoJI o add rectangle class

2019-04-08 14:28 +0900 Chang Hyun Lee o add trigonometric class

2019-04-08 14:22 +0900 Chang Hyun Lee o modified calc.py

2019-04-08 14:19 +0900 Chang Hyun Lee o modified calc.py

2019-04-08 10:41 +0900 Chang Hyun Lee o add torch_try37

2019-04-08 10:40 +0900 Chang Hyun Lee o add cal proj

2019-04-08 10:39 +0900 Chang Hyun Lee o deleted torch_ex3.py

2019-04-08 10:37 +0900 Chang Hyun Lee o rename

[main] Unstaged changes 95%

git-github_test — tig — 80x24

On branch master. Your branch is up-to-date with 'origin/master'.

Changes to be committed:

(no files)

Changes not staged for commit:

M pso.py

Untracked files:

(no files)

[status] Press u to stage 'pso.py' for commit 100%

git-github_test — tig — 80x24

commit ccbfee7e9a3cd540b125211f11860d4ec6a934a0

Refs: [master]

Author: Chang Hyun Lee <heypaprika@naver.com>

Date: Thu Jun 20 18:25:48 2019 +0900

modified pso.py

pso.py | 4 +-+
1 file changed, 2 insertions(+), 2 deletions(-)

commit 7e080441057ae4e751c98179c68fb90f8fe8a7c9

Refs: {origin/master}, {origin/HEAD}

Merge: b7f2ec1 8d54a5a

Author: Chang Hyun Lee <heypaprika@naver.com>

Date: Mon Apr 8 21:31:02 2019 +0900

solved conflict

calculator/calc.py | 4 +-+
1 file changed, 3 insertions(+), 1 deletion(-)

commit b7f2ec131b72ab91b024bf7dc1a5eb6fab31e557

[log] ccbfee7e9a3cd540b125211f11860d4ec6a934a0 - line 1 of 278 7%

Cannot move beyond the first line

git-github_test — tig — 80x24

Directory path /

drwxr-xr-x Chang Hyun Lee 2019-04-08 21:31 +0900 calculator

drwxr-xr-x Chang Hyun Lee 2019-04-08 10:41 +0900 torch_try37_NS_train_fin~

-rw-r--r-- Chang Hyun Lee 7 2019-04-08 14:22 +0900 .gitignore

-rw-r--r-- CH Lee 60 2019-04-08 21:13 +0900 README.md

-rw-r--r-- Chang Hyun Lee 3590 2019-04-06 14:22 +0900 ball_tracking.py

-rw-r--r-- Chang Hyun Lee 4460 2019-06-20 18:25 +0900 pso.py

-rw-r--r-- Chang Hyun Lee 356 2019-04-06 14:22 +0900 pyswarm.py

-rw-r--r-- Chang Hyun Lee 2857 2019-04-06 14:22 +0900 range-detector.py

-rw-r--r-- Chang Hyun Lee 572 2019-04-06 14:22 +0900 torch_ex.py

-rw-r--r-- Chang Hyun Lee 633 2019-04-06 14:22 +0900 torch_ex1.py

-rw-r--r-- Chang Hyun Lee 912 2019-04-06 14:22 +0900 torch_ex2.py

-rw-r--r-- Chang Hyun Lee 1050 2019-04-08 10:37 +0900 torch_ex33.py

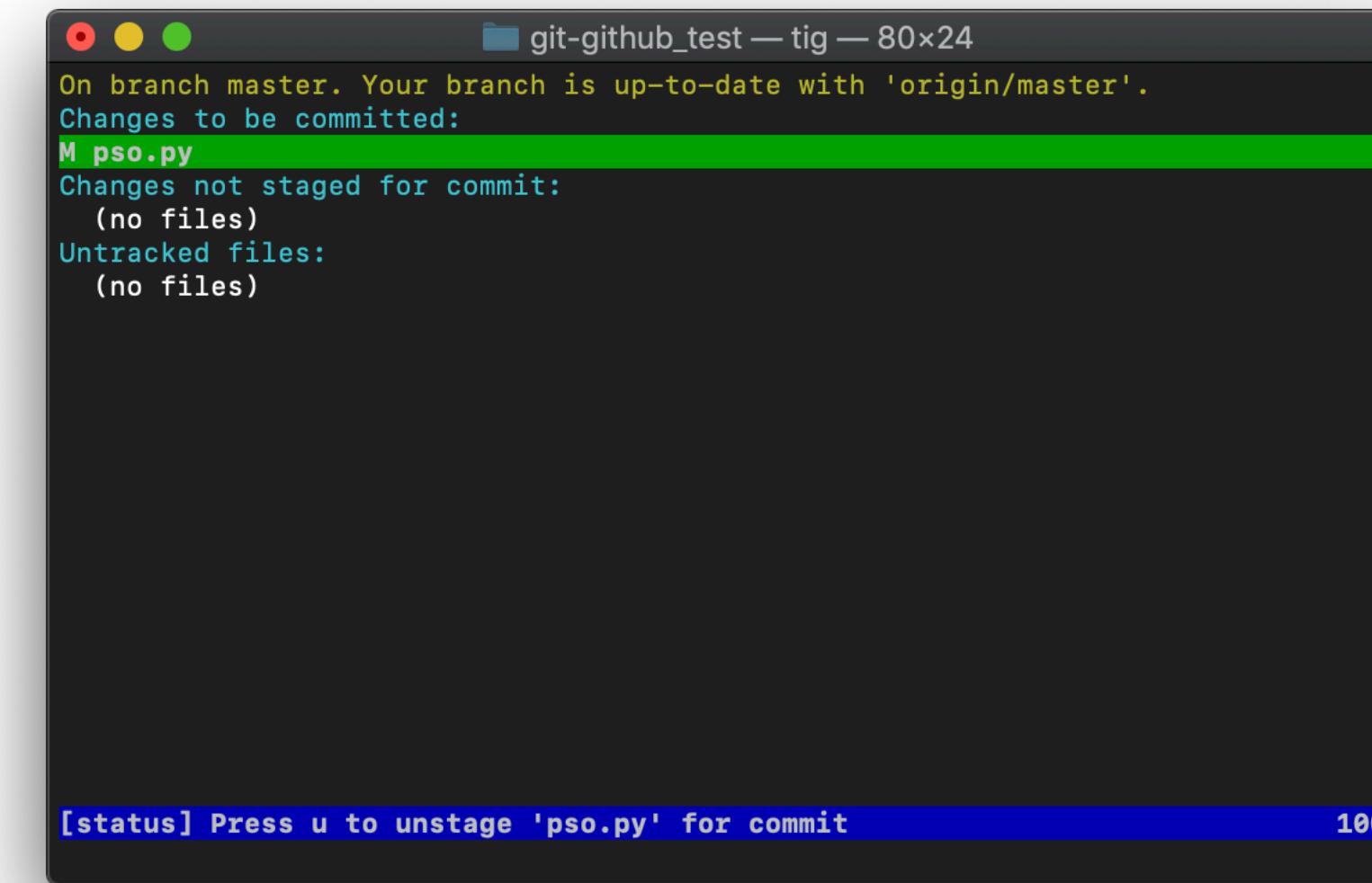
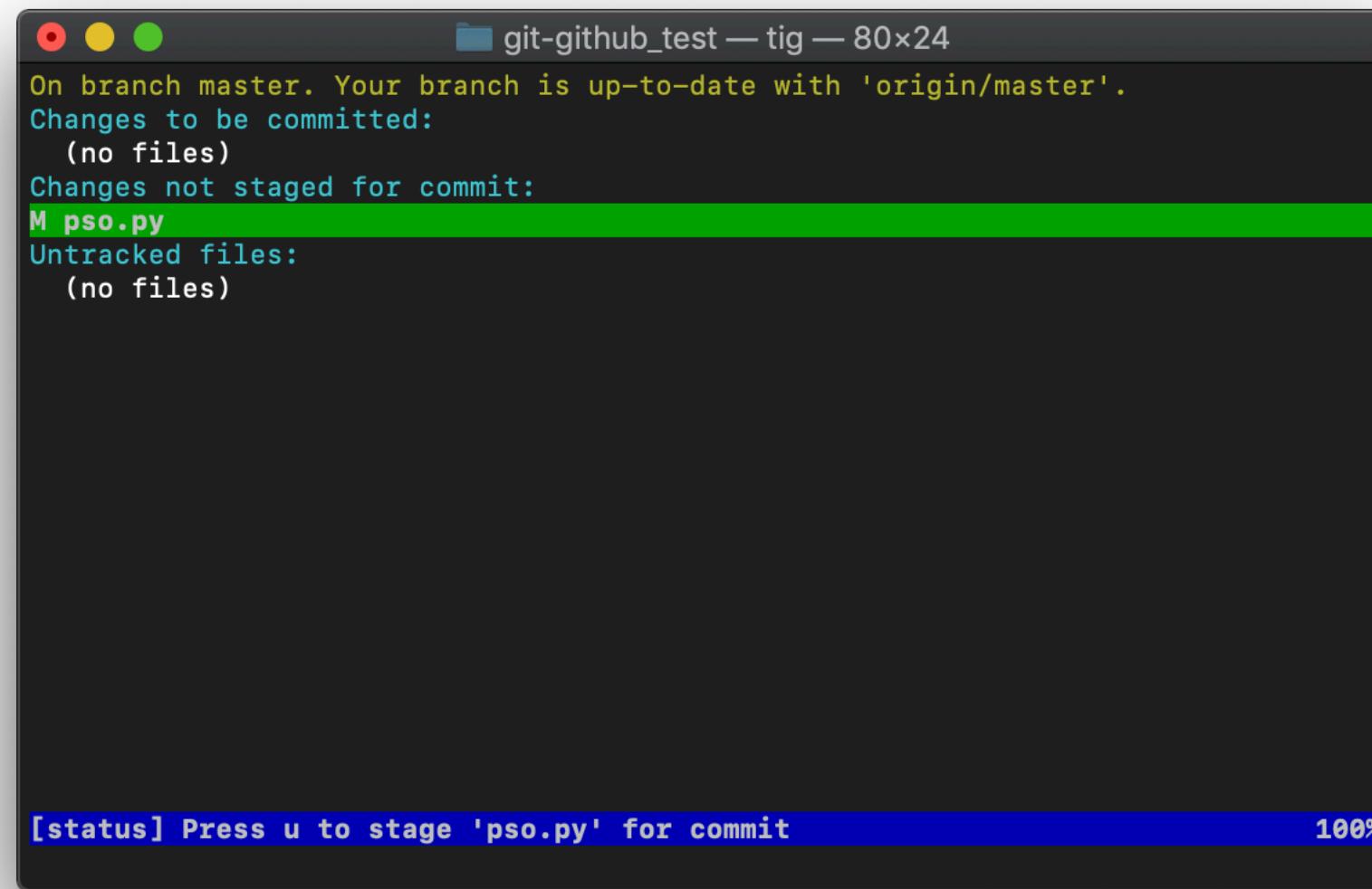
-rw-r--r-- Chang Hyun Lee 1398 2019-04-06 14:22 +0900 torch_ex4.py

[tree] b0108ca92ab9e8c32177e48de94db4a733266d12 - file 1 of 13 100%

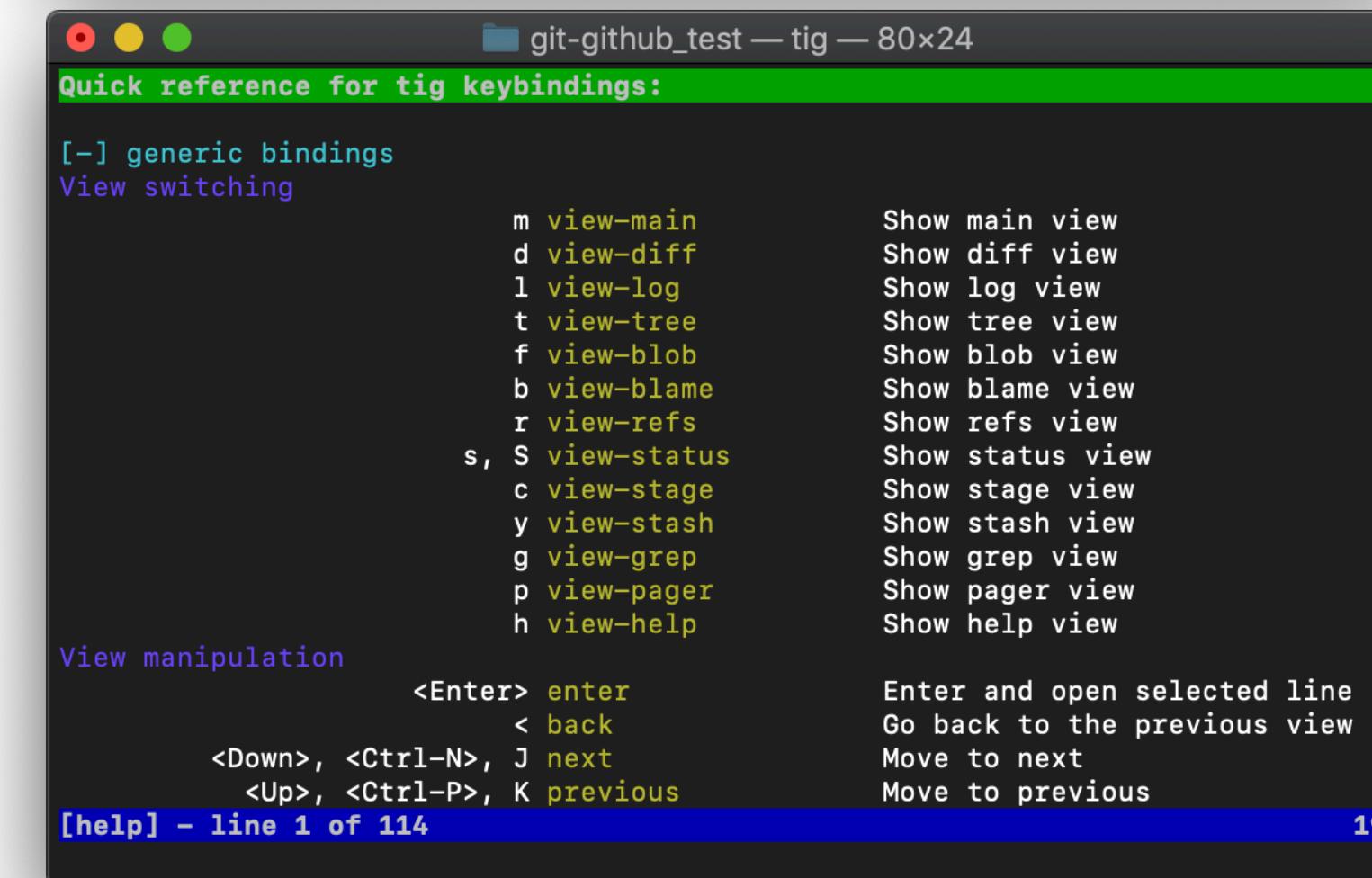
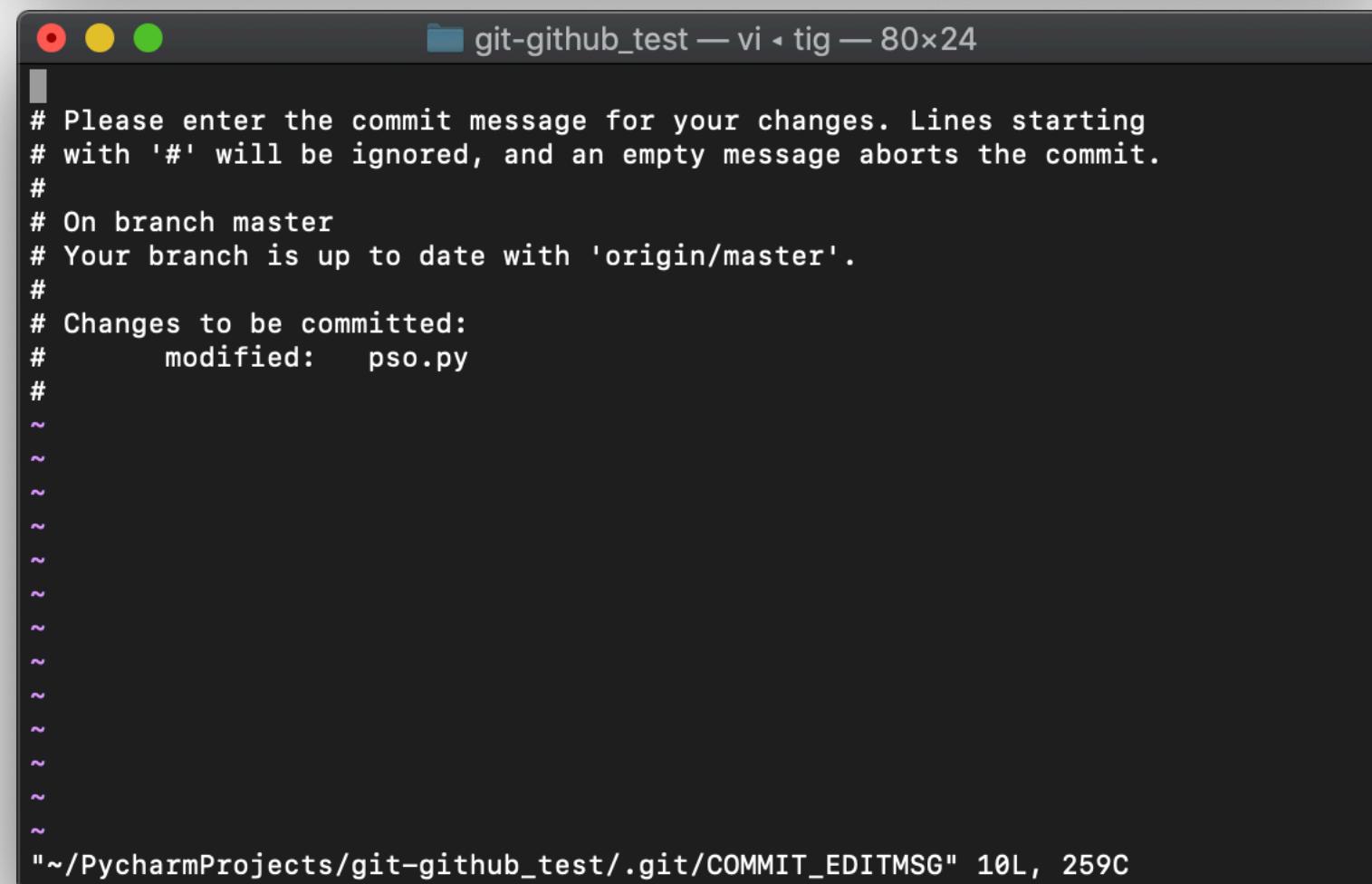
- Main 화면
- status 화면

- log 화면
- tree 화면

13. TEXT-MODE INTERFACE FOR GIT



- tig 로 add 하기
(status 화면에서 u)
 - tig 로 commit 하기
(status 화면 shift+c)
 - Help 화면



14. OPEN SOURCE에 기여하기

- Fork -> git clone -> 소스 수정 -> push

Mhtt2016 / Multi-Camera-Object-Tracking-via-Transferring-Representation-to-Top-View

Watch 3 Star 35 Fork 20

Code Issues 3 Pull requests 0 Projects 0 Wiki Security Insights

Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also compare across forks.

base repository: Mhtt2016/Multi-Camera-O... base: master head repository: Annoying-guys/Multi-Camer... compare: master

There isn't anything to compare.

Mhtt2016:master is up to date with all commits from Annoying-guys:master. Try switching the base for your comparison.

Showing 0 changed files with 0 additions and 0 deletions.

No commit comments for this range

Push 전 pull request 화면

Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also compare across forks.

base repository: Mhtt2016/Multi-Camera-O... base: master head repository: heypaprika/Multi-Camera-O... compare: master

Able to merge. These branches can be automatically merged.

Create pull request Discuss and review the changes in this comparison with others.

2 commits 1 file changed 0 commit comments 1 contributor

Commits on Jun 01, 2019

- heypaprika Update README.md
- heypaprika Update README.md

Showing 1 changed file with 2 additions and 1 deletion.

Unified Split

3 README.md

@@ -19,7 +19,8 @@ and [trained model](https://drive.google.com/file/d/1RgKUQt55CChsN0lX2JTxMdWBpz5

19 19

20 20 Download test videos "4 people indoor sequence"(4p-c0.avi, 4p-c1.avi, 4p-c2.avi, 4p-c3.avi) from ["EPFL" data set:
Multi-camera Pedestrian Videos

21 21](https://cvlab.epfl.ch/data/data-pom-index-php/) and detection files (4p-c0.pickle, 4p-c1.pickle, 4p-c2.pickle, 4p-c3.pickle) from [detection file for demo](https://drive.google.com/file/d/12MWB_CM0dDwfeG_ZxcwCYxI6sr_4vDKQ/view?)

Push 후 pull request 화면

14. OPEN SOURCE에 기여하기

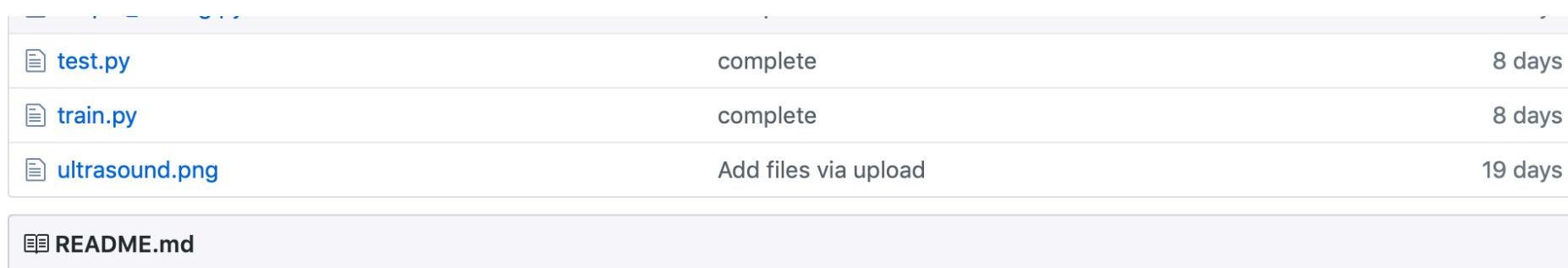
Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).

Pull request에 글이 게시된 후, 개설자와 이야기를 하면서 개설자는 pull 을 할 것인지 하지 않을 것인지 결정.

15. README.MD

- README.md 파일은 해당 project의 정보를 작성하여 이 repository를 사용 혹은 실행해보고자 하는 사용자들이나 수정 및 보완을 하려는 사용자들에게 repository 사용 방법 등을 알려줄 수 있는 시각적인 기능을 한다. example의 Raw file을 참고하여 작성한다.



Ultrasound with AI

Overview

Ultrasound with AI 은 광운대학교 전기공학과 ultrasound 수업에서 진행한 텁프로젝트입니다.

calculator 프로젝트를 보고 싶다면, [calculator repository](#)를 참고하세요.

이 프로젝트는 기존에 사용되던 의료 초음파 기술에 AI를 어디에 결합시킬 수 있을까 하는 궁금증으로 출발하였습니다. 기존의 의료 초음파를 사용하기 위한 장비에는 모니터, 파라미터를 조절하는 레버들, 프로브 등이 있어서 크기가 클 수 밖에 없었습니다. 크기가 커서 가정에서는 쉽게 사용할 수 없었는데, 요즘은 스마트폰에 probe를 연결하여 자가진단할 수 있도록 하는 application이 여럿 개발되었습니다.

이러한 여러 어플리케이션의 개발로 인하여 사용자는 많은 선택지를 가질 수 있게 되었지만 한 가지 문제점이 있는데, 바로 파라미터 값을 어떻게 설정해야 초음파 이미지의 화면이 뚜렷하게 나오는지를 잘 알지 못한다는 점입니다.

이러한 부분에 착안하여 저는 probe를 피부에 가져다 대었을 때, 어느 부위를 가져다 댈때에도 뚜렷한 화면을 보여주는 시스템을 만들고 싶었습니다. 그리고 이 repository가 그 시스템의 일부로서 동작할 것입니다.

Download and Install

- Download - git clone URL
- Install - (UBUNTU) : sh install.sh

What's included in the Project:

What	Description	link
1	file Description	link1
2	file Description	link2
3	file Description	link3
4	file Description	link4
5	file Description	link5

File example

하단의 snippet은 해당 모델을 training하는 소스입니다.

```
import torch
import torchvision.models as models
from linearRegression import linearRegression as Model
from dataset.ultrasound import Ultrasound_Dataset_for_dump as Dataset_dump
from dataset.ultrasound import Ultrasound_Dataset as Dataset

x_train = torch.randn(100,3,224,224)
```

- https://github.com/heypaprika/ultrasound_with_AI.

16. 거의 모든 GIT 명령어

- GitHub에서 Git cheat sheet pdf 파일을 만들어 제공하고 있다.
- <https://github.github.com/training-kit/downloads/github-git-cheat-sheet.pdf>
- 명령어와 간단한 설명으로 이루어져 있어서, 빠르게 명령어를 파악하는데 도움이 된다.

17. GIT 때문에 생긴 파일

- .git/ : \$ git init 명령어로 생기는 폴더이며, 이 폴더에서 해당 리포지토리의 소스 관리가 이루어진다.
- .gitignore : git이 관리하지 않을 파일이나 폴더를 지정해 둘 수 있다. GitHub에는 업로드 용량에 제한이 있어서, push할 용량이 크면 push 명령이 거절될 수 있다. 예로, 인공지능 특성상 데이터셋의 용량이 매우 커서, 데이터셋 폴더를 .gitignore에 등록하여 올리지 않도록 하는 편이다. <https://www.gitignore.io> 에서 자신의 os와 언어에 맞게 설정하여 .gitignore 파일을 초기화 할 수 있다.

18. GITHUB 말고 GITLAB도 있어요

- Gitlab은 free account도 privacy project에 collaborator를 무제한으로 추가할 수 있어서 GitHub의 3명으로 제한된 부분을 대체할 수 있다. Privacy project인데, 함께 하려는 사람이 3명 이상인 경우에 gitlab을 이용하면 된다.
- Gitlab은 GitHub와 크게 다르지 않아서 GitHub의 repository를 Gitlab으로 관리할 수 있도록 쉽게 이동할 수 있다.
- GitHub -> gitlab 이전 방법 :
- GitHub에서 project clone
\$ git clone <github_url>
- .git dir 제거
\$ rm -rf .git
- 이후의 명령어 진행
\$ git init
\$ git remote add origin <gitlab_url>
\$ git add .
\$ git commit -m "First Commit"
\$ git push -u origin master

GIT & GITHUB

THANK YOU!