

AI 인공지능 코딩 구현 홀로서기 5기

[4주차, 주말 수업] Debug, Git

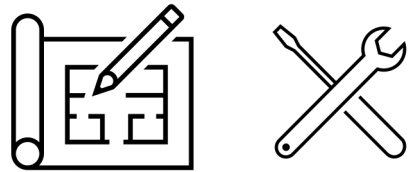
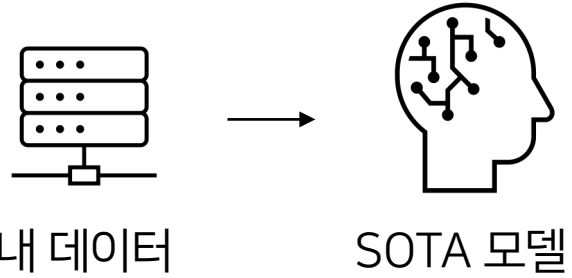
정 정 민

강의 목차

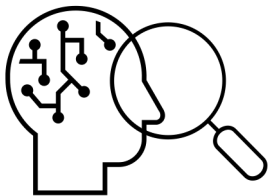
1. Debug Mode
2. Github과 repo
3. VScode와 Github

Debug Mode

크게 다를 바 없어 보이는 일..



스스로 직접 모델을 구현



다른 사람이 짠 모델 확인



큰 집도 작은 망치질로 부터



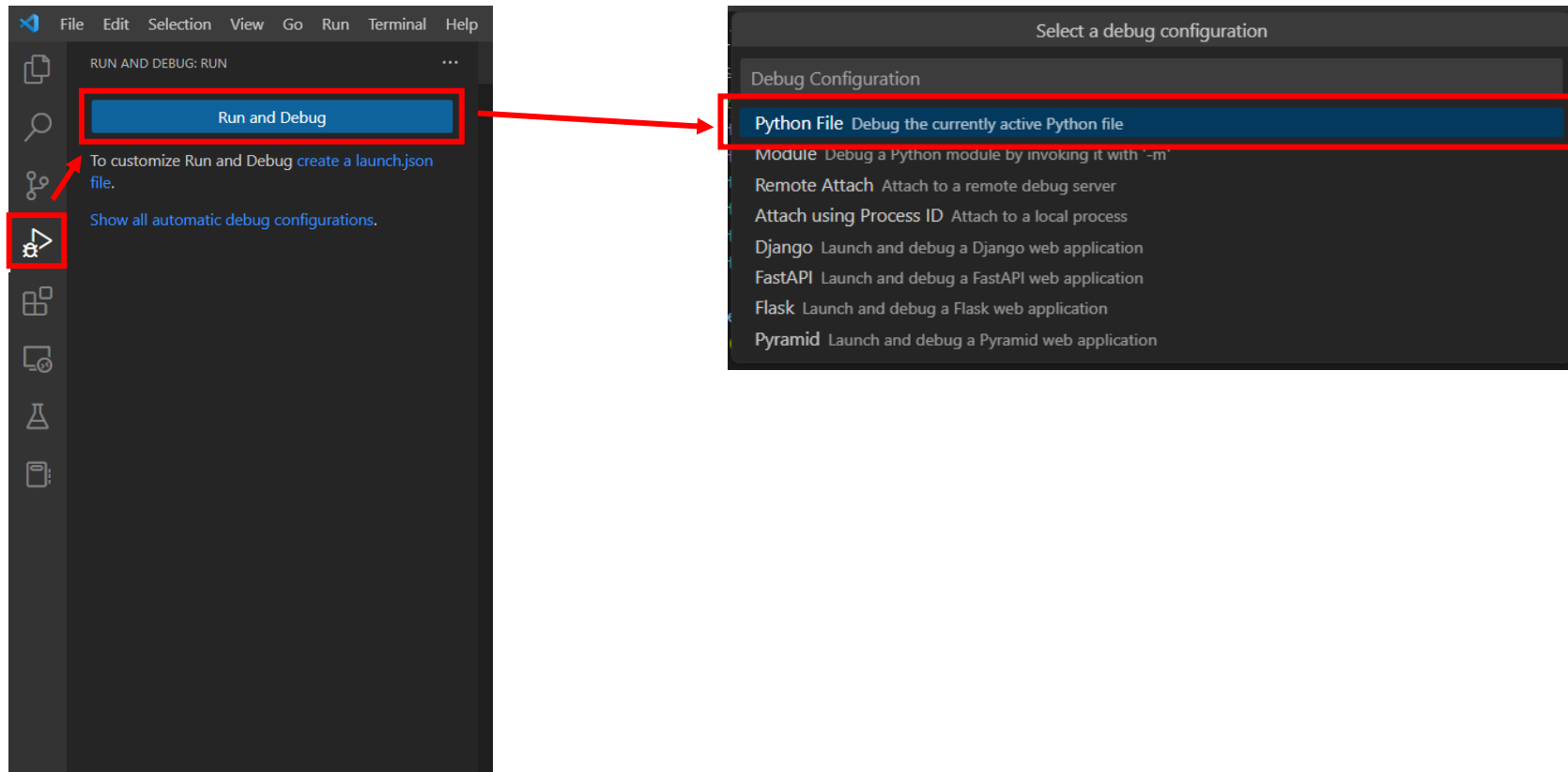
X 100...0

Why Debug Mode?

- AI 구현에서 홀로 서기 위한 최고의 도구
- 단순히 버그와 오류를 찾고 이것을 고치는 것을 넘어 실제 코드의 동작 과정을 이해할 수 있음
- 문제를 더 작고 명확하게 만들어 줌
- 컴퓨터가 프로그램 코드를 실행시킬 때, 강제로 중간 과정에서 작동을 멈추게 한 후 아래 작업을 할 수 있음
 - 변수의 값과 attribute 확인, 앞으로 실행될 코드의 결과를 미리 확인
 - Tensor의 값, shape, gradient 연결, 타입 등
 - 특정 변수와 연산이 되는지, 함수나 메소드를 적용 가능 여부, 시각화 등
- 단점?
 - 세팅에 시간과 노력이 좀 들 수 있음

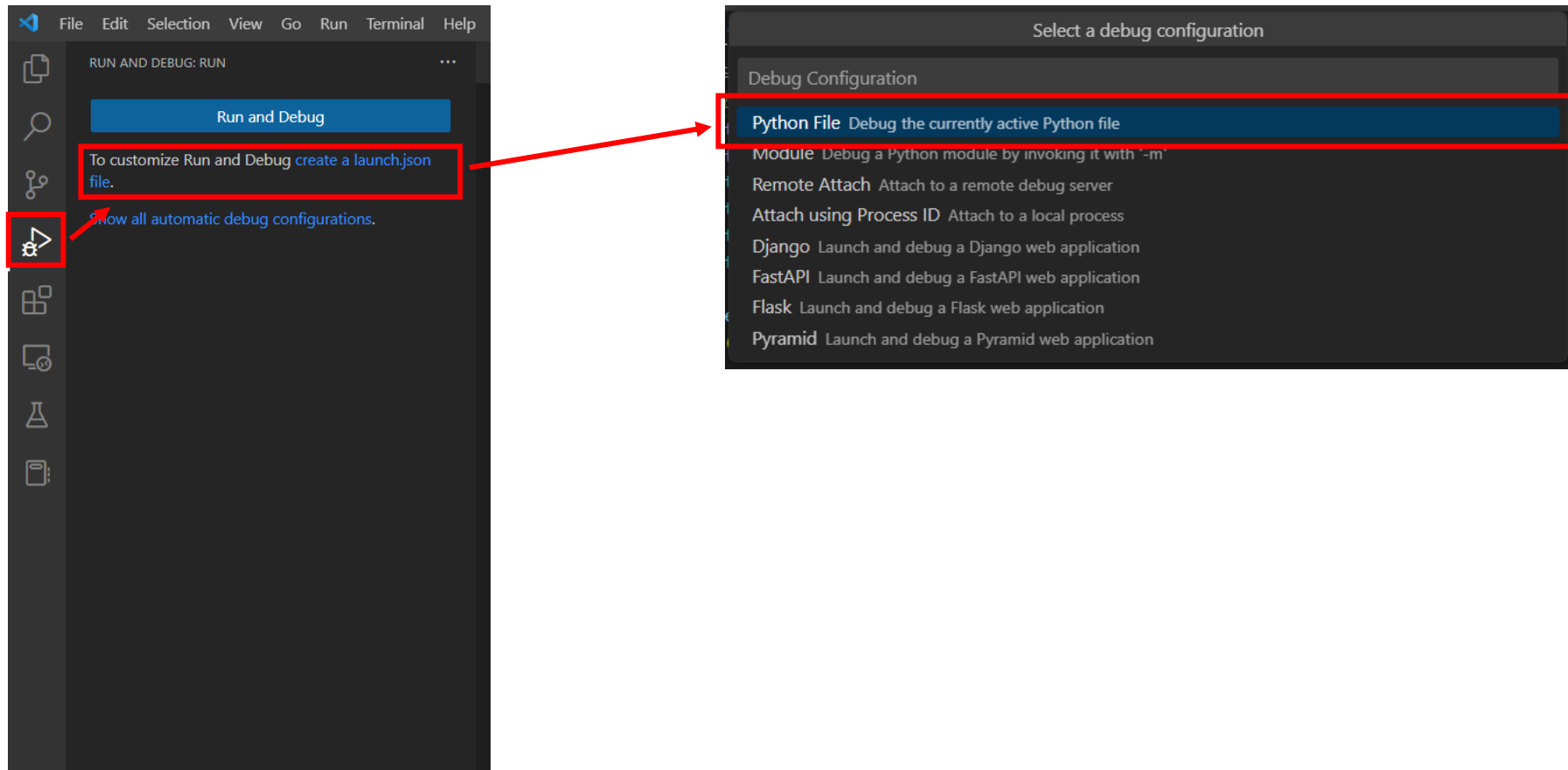
VScode Debug Mode. 1) 지금 열린 파일 디버깅

- Python extension이 깔려 있어야 함
- 좌측 패널의 디버그 컨트롤 버튼 클릭 > Run and Debug > Python File
- 많이 안 씀



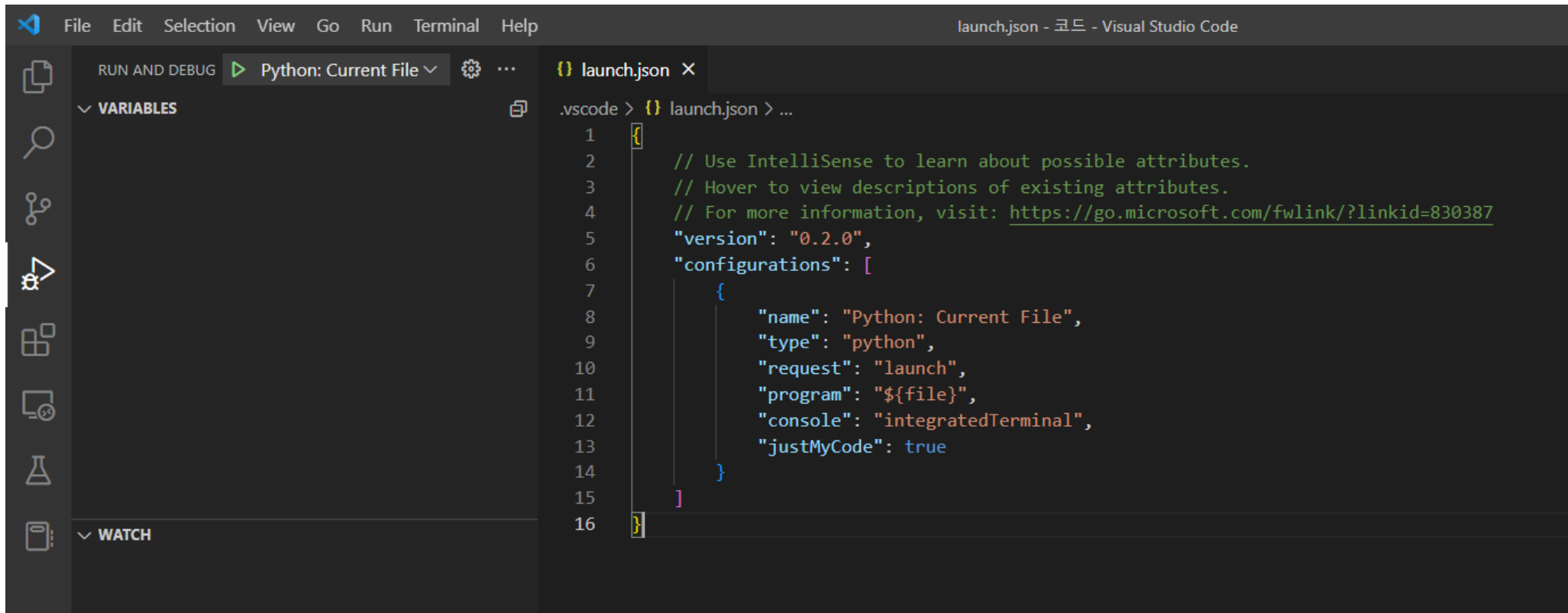
VScode Debug Mode. 2) 디버깅 컨트롤 파일 생성

- 좌측 패널의 디버그 컨트롤 버튼 클릭 > 'create a launch.json file' > Python File
- 많이 사용. 수업에도 이것을 활용 예정



VScode Debug Mode. 2) 디버깅 컨트롤 파일 생성

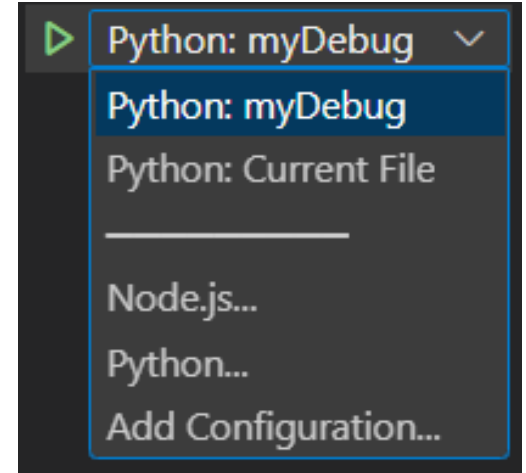
- 이 파일의 위치는 .vscode/lauch.json
- 여러 디버깅 환경을 저장해 두고 필요에 따라 원하는 디버깅 조건을 실행
- 아래의 디버깅 세팅은..



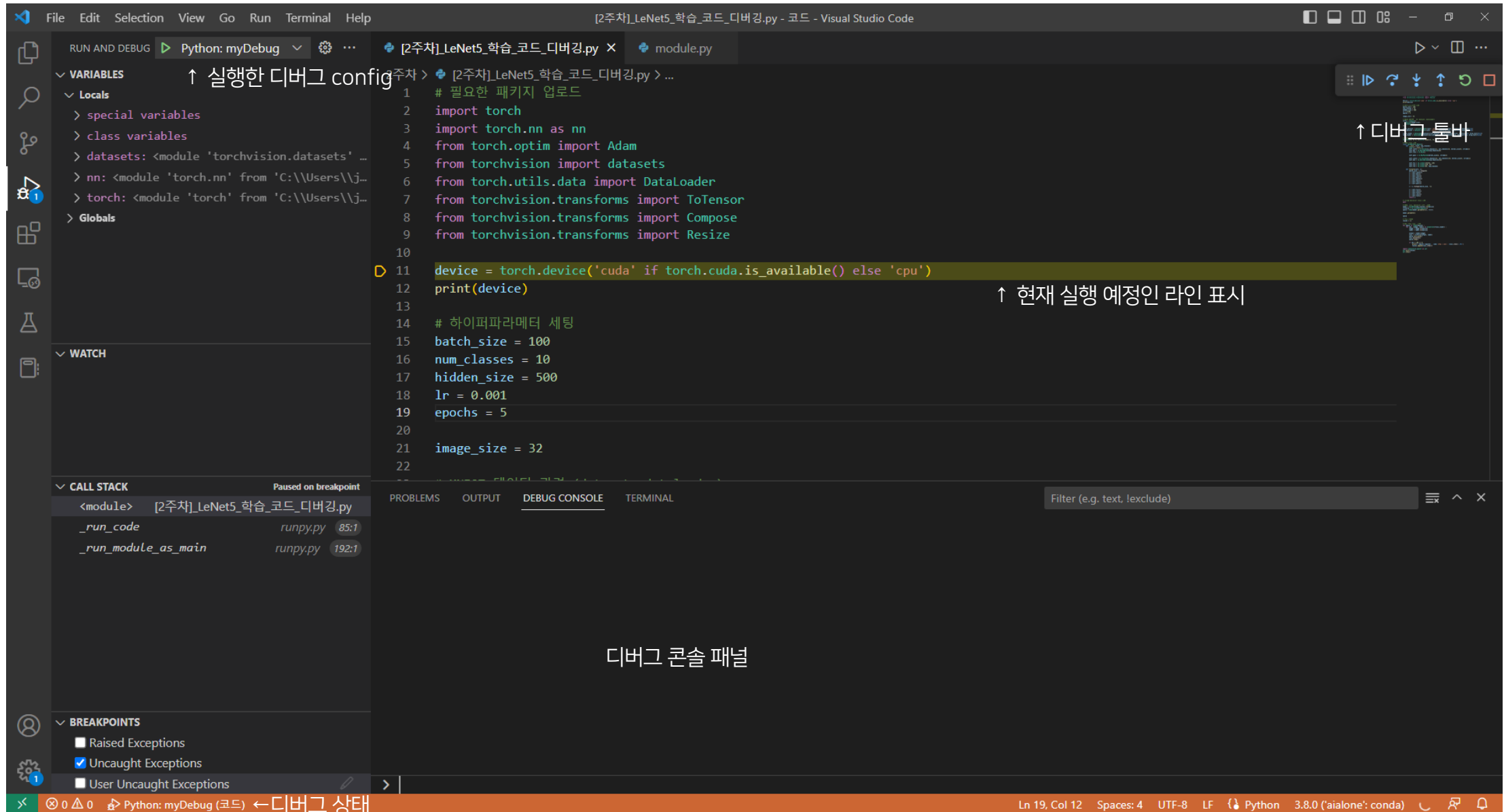
```
1  {
2      // Use IntelliSense to learn about possible attributes.
3      // Hover to view descriptions of existing attributes.
4      // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387
5      "version": "0.2.0",
6      "configurations": [
7          {
8              "name": "Python: Current File",
9              "type": "python",
10             "request": "launch",
11             "program": "${file}",
12             "console": "integratedTerminal",
13             "justMyCode": true
14         }
15     ]
16 }
```

Basic Configuration

```
"configurations": [  
  {  
    "name": "Python: myDebug",  
    "type": "python",  
    "request": "launch",  
    "program": "${workspaceFolder}${pathSeparator}[파일까지 경로] ",  
    "console": "integratedTerminal",  
    "justMyCode": false,  
    "cwd": "${workspaceFolder}",  
    "env": {"CUDA_VISIBLE_DEVICES": "0"}, // GPU 개수 안으로  
    "args": [  
      "--arg1", "arg1_value",  
      "--arg2", "arg2_value",  
    ],  
  },  
]
```

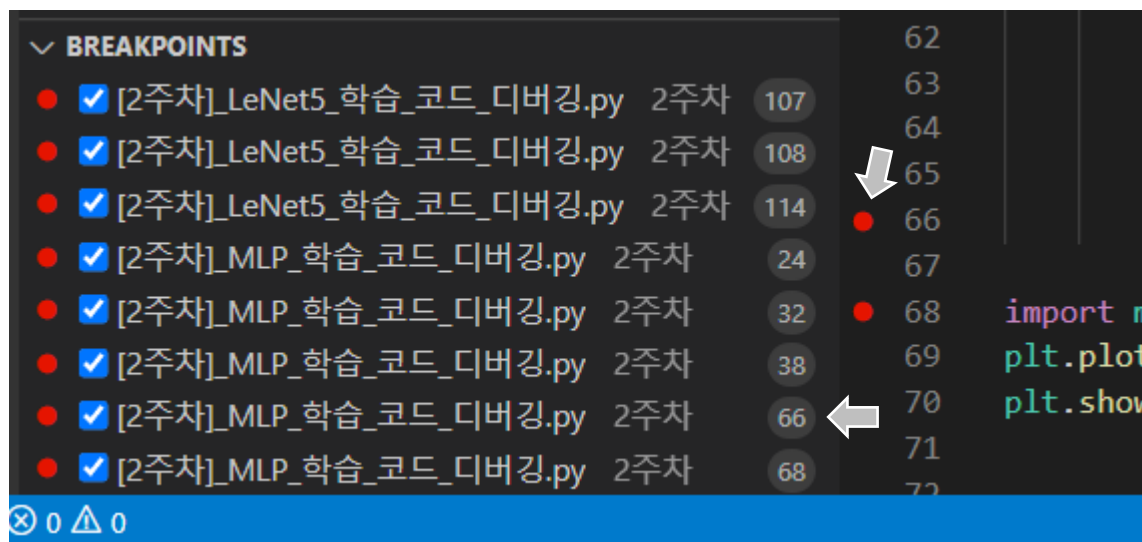


Debug Mode 화면 모습



Debug Mode Break Points

- 코드 실행을 일시 중단하는 지점
 - 지정한 break point 까지 실행된 프로그램의 상태를 검사하고 디버깅 할 수 있음
 - 변수 값, 함수 호출 스택, 조건 등
 - 복수개의 break point도 사용 가능
-
- 코드 편집기의 일시 중단할 코드 줄 왼쪽 여백을 클릭
 - 단축키 : Ctrl + F9, Command + F9



Debug Mode Debug Toolbar

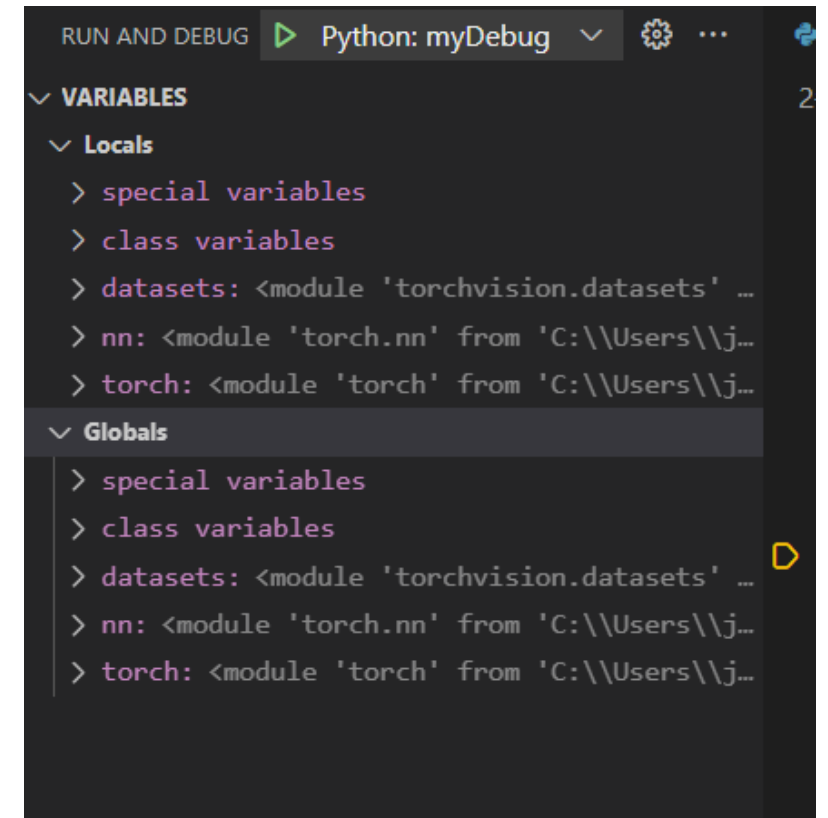


- 디버깅 중인 프로그램의 상태 제어하는 시각적 툴바

단축키	액션	설명
F5	Continue / Pause	Continue : 중지된 프로그램을 시작 (until 끝까지 or 다음 break point 까지) Pause : 실행 중인 지점에서 프로그램을 일시 중지
F10	Step Over	지금 실행 중인 라인을 실행시키고 다음 줄로 넘어감
F11	Step Into	해당 코드를 실행시키는 하위 실행 환경으로 접근
Shift + F11	Step Out	어떤 함수나 하위 실행 환경에 있을 때, return 까지의 코드를 실행시키고 상위 과정으로 바로 빠져나옴
Ctrl + Shift + F5	Restart	디버그 모드 다시 시작, 변경된 코드가 적용된 상태로 다시 시작
Shift + F5	Stop	디버그 모드 강제 종료

Debug Mode Variables

- 디버그 상태에서 접근 가능한 모든 변수의 값을 확인할 수 있음
 - Locally & Globally
- 심지어 import 한 패키지도 접근이 가능
- 어떠한 변수가 있는지를 볼 수 있는 용도 말고는 거의 사용하지 않음
- 추가로) 편집기 위에 마우스를 올려도 확인 가능



Debug Mode Debug console

- 디버그 상태에서 코드 실행 상태를 확인하는 도구
- 변수 값 확인, 함수 적용, 프로세스 실행의 결과를 보여줌
- 커맨드 창과 비슷한 역할을 수행
- 새로운 변수를 만들고 저장할 수 있음

```
78 # 모델, Loss, Optimizer 객체 만들기
79 model = myLeNet5(num_classes).to(device)
80 criteria = nn.CrossEntropyLoss()
81 optim = Adam(model.parameters(), lr=lr)
82
83 model.parameters
84
```

PROBLEMS DEBUG CONSOLE TERMINAL

```
→ image_size
32
→ train_dataset[0][0].shape
> torch.Size([1, 32, 32])
→ model
myLeNet5(
  (conv1): Conv2d(1, 6, kernel_size=(5, 5), stride=(1, 1))
  (bn1): BatchNorm2d(6, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (relu): ReLU()
  (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
> (conv2): Conv2d(6, 16, kernel_size=(5, 5), stride=(1, 1))
  (bn2): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (fc1): Linear(in_features=400, out_features=120, bias=True)
  (fc2): Linear(in_features=120, out_features=84, bias=True)
  (fc3): Linear(in_features=84, out_features=10, bias=True)
)
→ new_var = torch.randn((1,2,3,4,5))
→ new_var.shape
> torch.Size([1, 2, 3, 4, 5])
```

Debug Mode watch

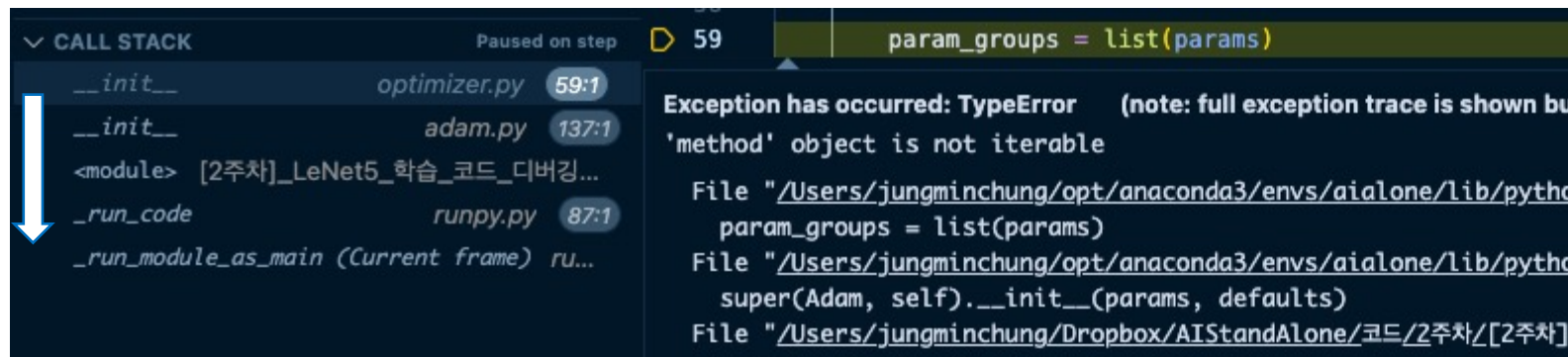
- Debug console 과 비슷한 부분이 있음
- 원하는 변수, 함수 실행 결과 등의 값을 볼 수 있음
 - 보기 용이므로 import 는 할 수 없음
- 하지만 말 대로 지금 멈춰 있는 디버그 상태에서 내가 설정한 값을 보려고 함
 - 찾지 못하는 변수면 값을 볼 수 없음
 - 멈추는 시점에 따라 그 값이 달라짐
- 매 시점 해당 코드를 실행시키고 보여주는 형태
 - Loop 문이라면 정지 시점 마다 값이 변화
 - 그림을 저장하는 문구라면 매번 다시 저장
- 따라서, 실행에 오래 걸리는 함수나 프로세스는 따로 적지 않는 말 것

```
✓ WATCH
import os: SyntaxError: invalid syntax (<s...
> model: myLeNet5(
criteria: NameError: name 'criteria' is no...
77
78 # 모델, Loss, Optimizer 객체 만들기
79 model = myLeNet5(num_classes).to(device)
80 criteria = nn.CrossEntropyLoss()
81 optim = Adam(model.parameters(), lr=lr)
82
```

```
✓ WATCH
import os: SyntaxError: invalid syntax (<s...
> model: myLeNet5(
> criteria: CrossEntropyLoss()
77
78 # 모델, Loss, Optimizer 객체 만들기
79 model = myLeNet5(num_classes).to(device)
80 criteria = nn.CrossEntropyLoss()
81 optim = Adam(model.parameters(), lr=lr)
82
```


Debug Mode Call Stack

- 현재 멈춰 있는 디버그 상태까지 어떤 경로로 시스템이 타고 들어왔는지 경로를 볼 수 있음
- Error가 났다면 문제가 있는 지점 상위의 함수나 프로세스에 어떠한 값이 들어가고 무엇때문에 문제를 일으켰는지를 확인할 때 용이



```
CALL STACK
Paused on step 59
__init__ optimizer.py 59:1
__init__ adam.py 137:1
<module> [2주차]_LeNet5_학습_코드_디버깅... 87:1
_run_code runpy.py 87:1
_run_module_as_main (Current frame) ru... 87:1
```

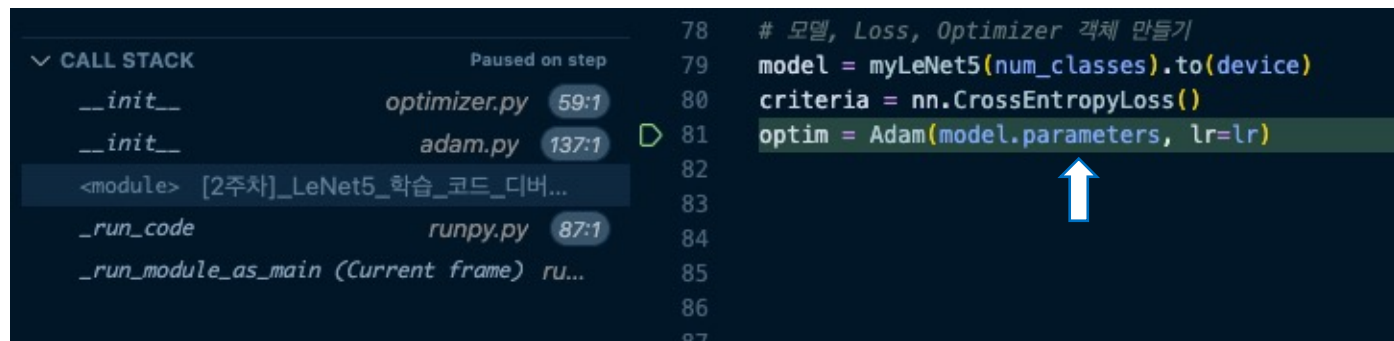
Exception has occurred: TypeError (note: full exception trace is shown below)

'method' object is not iterable

File "/Users/jungminchung/opt/anaconda3/envs/aialone/lib/python3.6/site-packages/torch/optim/adam.py", line 137, in __init__
param_groups = list(params)

File "/Users/jungminchung/opt/anaconda3/envs/aialone/lib/python3.6/site-packages/torch/optim/adam.py", line 137, in __init__
super(Adam, self).__init__(params, defaults)

File "/Users/jungminchung/Dropbox/AIStandAlone/코드/2주차/[2주차]_LeNet5_학습_코드_디버깅.py", line 87, in _run_code



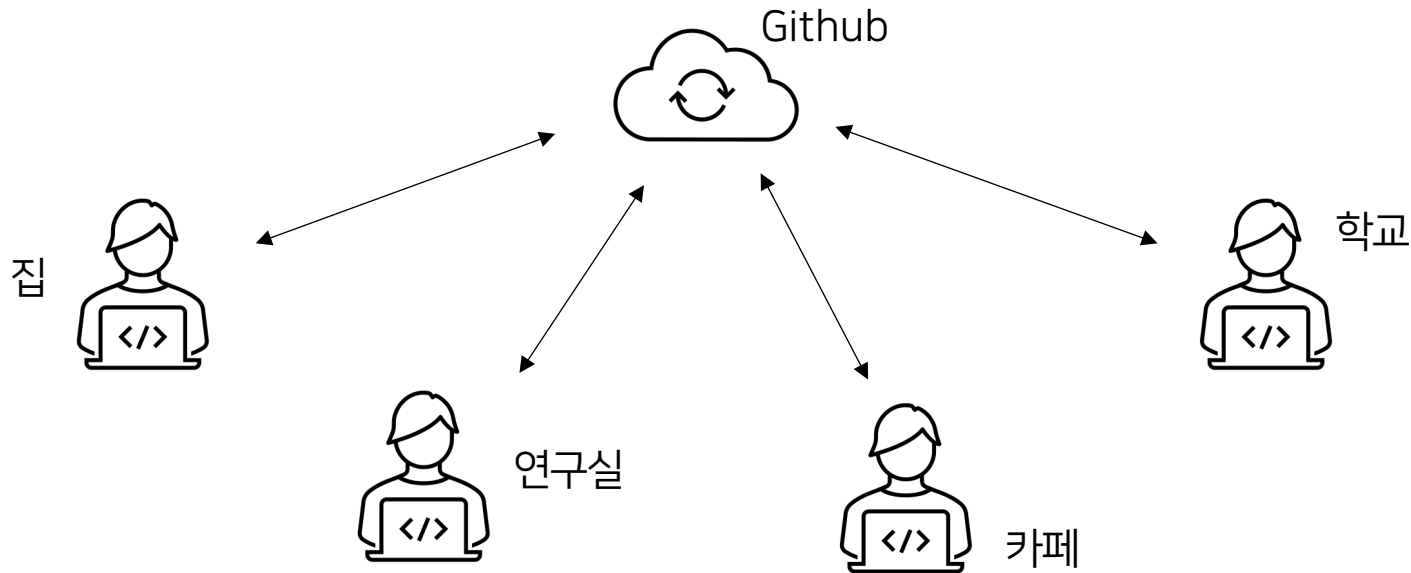
```
CALL STACK
Paused on step 59
__init__ optimizer.py 59:1
__init__ adam.py 137:1
<module> [2주차]_LeNet5_학습_코드_디버깅... 87:1
_run_code runpy.py 87:1
_run_module_as_main (Current frame) ru... 87:1
```

```
78 # 모델, Loss, Optimizer 객체 만들기
79 model = myLeNet5(num_classes).to(device)
80 criteria = nn.CrossEntropyLoss()
81 optim = Adam(model.parameters, lr=lr)
82
83
84
85
86
87
```

Github과 repo

코드 저장소 : Github

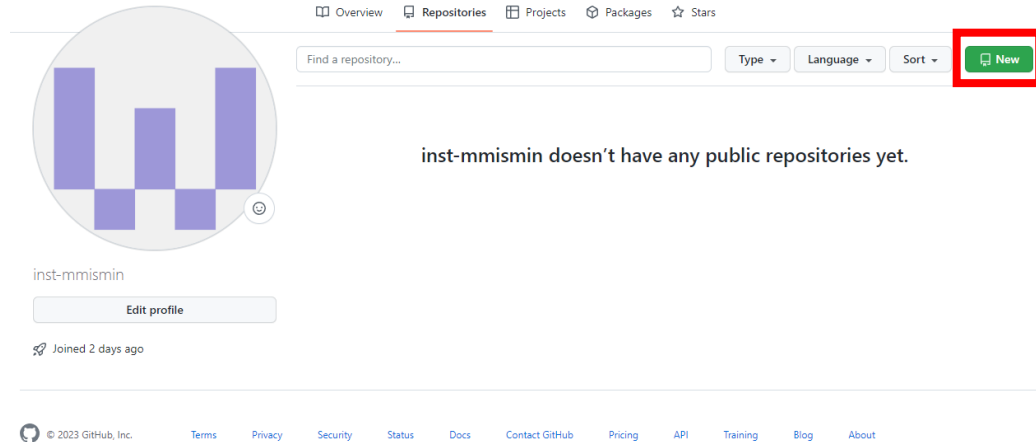
- 프로젝트 안에 코드가 비대해지고, 여러 프로젝트를 다뤄야 한다면
- 그리고 여러 작업 환경 (재택, 연구실, 회사, 카페 등)에서 작업을 한다면
- 코드 저장에 대한 필요성이 상승 → Git → 특히 클라우드 저장의 욕구 → Github



- 클라우드 기반이라 협업이 가능하나, **홀로서기 정신에 맞춰 개인 코드 정비 및 관리용**으로 사용

Github repository

- Github은 repo 단위로 코드 관리
- Repo (repository) : 내 코드를 담고 있는 저장소
 - 로컬(local) repo : 내 컴퓨터에 저장된 프로젝트 & 코드
 - 원격(remote) repo : Github 서버에 저장된 프로젝트 & 코드
- 작업은 로컬에서 → 원격에 업로드
- 필요시 원격에서 로컬로 다운로드



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner * inst-mmismmin / Repository name * AIStandAlone ✓

Great repository names are short and memorable. Need inspiration? How about [animated-octo-engine](#)?

Description (optional)

- ☒ **Public**
Anyone on the internet can see this repository. You choose who can commit.
- ☐ **Private**
You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

- ☐ **Add a README file**
This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore

Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: **None**

Choose a license

A license tells others what they can and can't do with your code. [Learn more.](#)

License: **None**

ⓘ You are creating a public repository in your personal account.

Create repository

Git 설치 확인

- 정상 설치 시 git 입력에 따라 아래와 같은 설명이 나와야 함



```
C:\Users\Wjungminchung>git
usage: git [-v | --version] [-h | --help] [-C <path>] [-c <name>=<value>]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p | --paginate] [-P | --no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           [--super-prefix=<path>] [--config-env=<name>=<envvar>]
           <command> [<args>]

These are common Git commands used in various situations:


start a working area (see also: git help tutorial)
  clone      Clone a repository into a new directory
  init       Create an empty Git repository or reinitialize an existing one


work on the current change (see also: git help everyday)
  add        Add file contents to the index
  mv         Move or rename a file, a directory, or a symlink
  restore    Restore working tree files
  rm         Remove files from the working tree and from the index

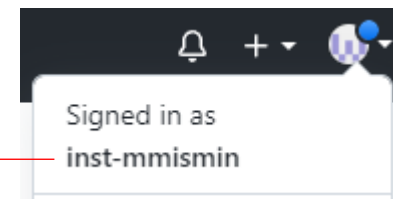
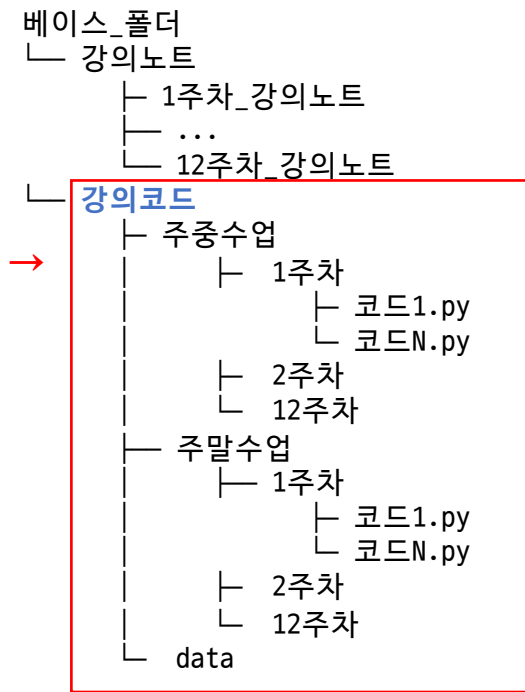

examine the history and state (see also: git help revisions)
  bisect     Use binary search to find the commit that introduced a bug
  diff       Show changes between commits, commit and working tree, etc
  grep       Print lines matching a pattern
  log        Show commit logs
  show       Show various types of objects
  status     Show the working tree status
```

내 작업 공간에 config 설정하기

- 로컬 repo까지 이동 (cd 명령어 활용)
- **Git을 쓰겠다는 선언**을 해야 함 : `git init`
- 나의 **신원을 설정하기 위해 config**를 넣어야 함
 - 이름(user.name)과 이메일(user.email) 설정
 - User.name : 원하는 대로.. (Github의 프로필 이름과는 달라도 되지만 이를 추천)
 - User.email : **Github 로그인 아이디!!** (신원 확인에 사용됨)
 - 영향력 범위를 주의
 - Global config : 컴퓨터 모든 repo에 영향 (우선순위 ↓)
 - 잘 생각해서 써야 함. Github 계정이 2개 이상이거나, 서버에서는 사용 금지
 - Local config : 지금 있는 위치의 repo에만 영향 (우선순위 ↑)
 - 이걸로 설정 시 `git init`은 우선되어야 함

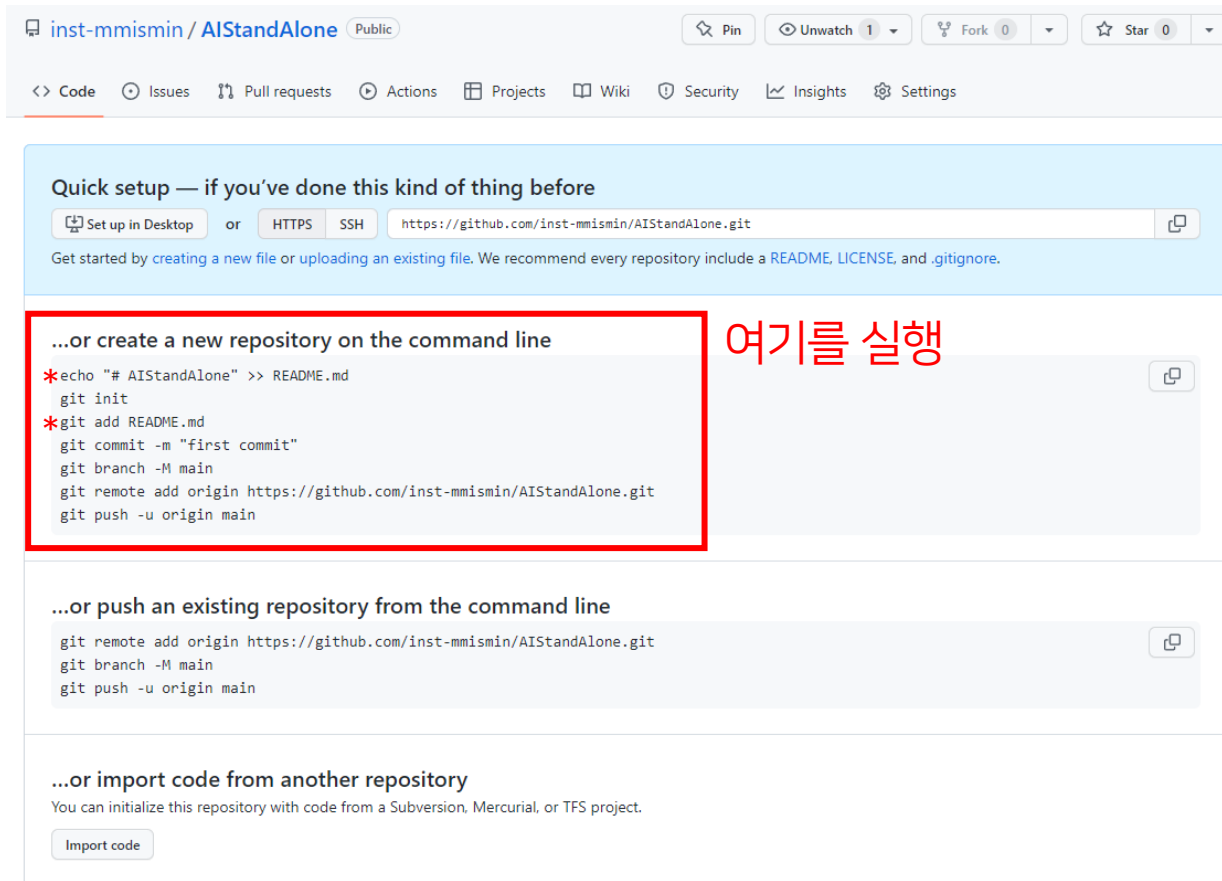
```
git config (--global) user.name "inst-mmismin"
git config (--global) user.email "inst.mmismin@gmail.com"
```

로컬 repo 범위 →
데이터 제거!!
.vscode도 없음 주의



Github에 내 작업 공간 올리기

- 맨 처음에는 원격 repo를 만들면 나오는 명령어를 거의 그대로 치면 됨
- 아래 * 옆의 명령어는 필수 아님



inst-mmismmin / AIStandAlone Public

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Quick setup — if you've done this kind of thing before

Set up in Desktop or HTTPS SSH <https://github.com/inst-mmismmin/AIStandAlone.git>

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

```
*echo "# AIStandAlone" >> README.md
git init
*git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/inst-mmismmin/AIStandAlone.git
git push -u origin main
```

여기를 실행

...or push an existing repository from the command line

```
git remote add origin https://github.com/inst-mmismmin/AIStandAlone.git
git branch -M main
git push -u origin main
```

...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

Import code

```
echo "# AI StandAlone" >> README.md
```

```
git init
```

```
git add README.md
```

```
git commit -m "first commit"
```

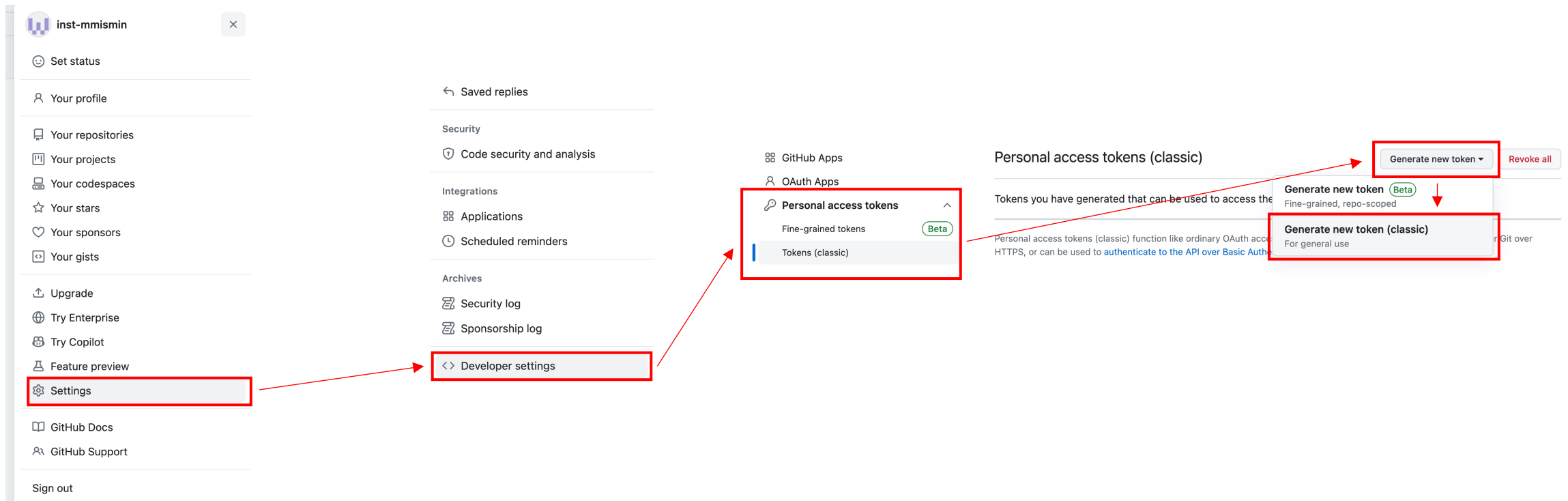
```
git branch -M main
```

```
git remote add origin [본인_Git_repo_주소].git
```

```
git push -u origin main
```

Push 전용 PW token 설정 (1/2)

- 최근 보안 강화를 목적으로 Log-In 과정에서 사용하는 PW만으로는 Push 과정에서 **Permission Error**가 남
- 원격 Repo와 로컬 Repo의 "관리자인 내가 Push를 하고 있음"을 알려주는 PW token 를 써야 함
- Token 설정 후 잘 가지고 있다가 필요시 복.불으로 사용
 - 처음 설정한 token는 재확인 불가
 - Token를 잊어버리면 다시 생성 필요



Push 전용 PW token 설정 (2/2)

- 최근 보안 강화를 목적으로 Log-In 과정에서 사용하는 PW만으로는 Push 과정에서 **Permission Error**가 남
- 원격 Repo와 로컬 Repo의 "관리자인 내가 Push를 하고 있음"을 알려주는 PW token 를 써야 함
- Token 설정 후 잘 가지고 있다가 필요시 복.붙으로 사용
 - 처음 설정한 token는 재확인 불가
 - Token를 잊어버리면 다시 생성 필요

The image shows the GitHub 'Personal access tokens (classic)' interface and a mobile notification. The interface is divided into two main sections: 'New personal access token (classic)' and 'Personal access tokens (classic)'. The 'New' section includes a 'Note' field, an 'Expiration' dropdown set to 'No expiration', a warning box about expiration, a 'Select scopes' section with 'repo' selected, and a 'Generate token' button. The 'Tokens' section shows a list of generated tokens, with one token highlighted and its value partially visible as 'ghp_...Ws9'. A red arrow points from the 'Generate token' button to the mobile notification. The notification is a yellow box with the text '출로서기 4,5기 강의 token' and the token value 'ghp_...Ws9'.

New personal access token (classic)

Personal access tokens (classic)

Generate new token ▼ Revoke all

Tokens you have generated that can be used to access the [GitHub API](#).

Make sure to copy your personal access token now. You won't be able to see it again!

Copied!

ghp_...Ws9 ✓ Delete

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

출로서기 4,5기 강의 token

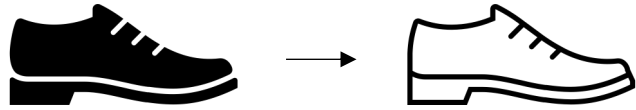
ghp_...Ws9

오전 7:18

전송

Git add, commit, push

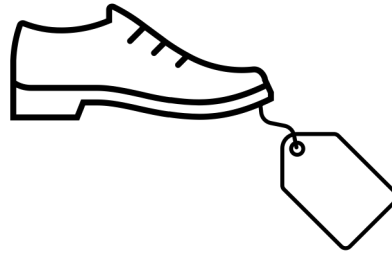
Add



본사에서 새로운 상품 중 출시할
상품을 선택 & 출시 준비

내 Local에서 변경된 부분 중
Remote에 보낼 코드 선택 & 준비

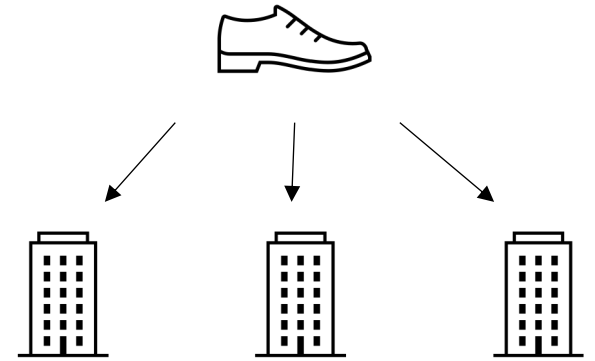
Commit



출시할 상품의 설명 달기
가격, 소재 등등

올릴 코드 설명

Push



본사의 출시 상품과 동일한 제품을
매장에 공급 & 최신 상품 동기화

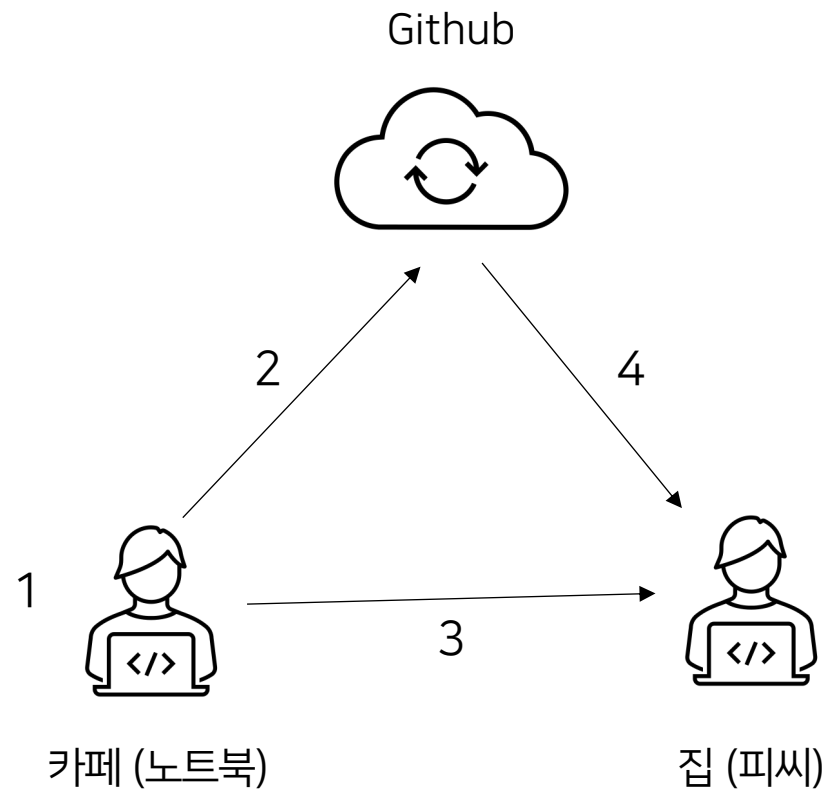
코드 업데이트
Local과 Remote 코드 동기화

Commit message

- 본인이 어떤 것을 변화 시켰는지를 파악하기 위해 알아보기 쉽게 적을 것
 - 영어로 적을 필요 X
- 특정 조직마다 commit message 규칙이 존재
 - 지금은 알 수 없으니, 일단 본인의 코드 변화를 잘 적는 연습을 하는 것이 필요
- 커밋의 제목과 본문 중 일단 제목만 잘 적기
- 간단한 추천으로는
 - "Add : ~ 기능을 하는 모듈 / 함수 / 파일 등을 추가함"
 - "Modify : ~ 기능 추가 / 업데이트 등을 위해 코드 변경"
 - "Fix : ~ 에러를 수정. 단순 typo / 논리 재구조화 / 입, 출력 변경"

Git Pull

- 원격 repo의 내용을 로컬 repo로 가지고 와서 덮어 씌우는 과정
- 항상 새로운 환경(다른 장소 & 다른 시간)에서 작업 전, Pull 받는 습관이 필요
- Pull 받기 전에 코드 변경 시 Pull이 안될 수 있음



Github에 올릴 때 주의사항 : 용량 제한

- 50MB 이상을 Add 하고 Push 하면 경고
- 100MB 이상이면 에러
- 용량이 큰 파일(학습된 모델, 데이터, 중간 결과물 (json, csv 등)은 다른 경로(구글 드라이브 등)로 제공

Github 로컬 repo 정리 맛보기

- 정답은 없음. 아래는 예시이지만 일반적으로 많이 사용하는 틀
- 하지만 실제로 우리가 사용할 구조와는 좀 다름
 - 우리는 주차별로 서로 다른 코드가 존재하므로

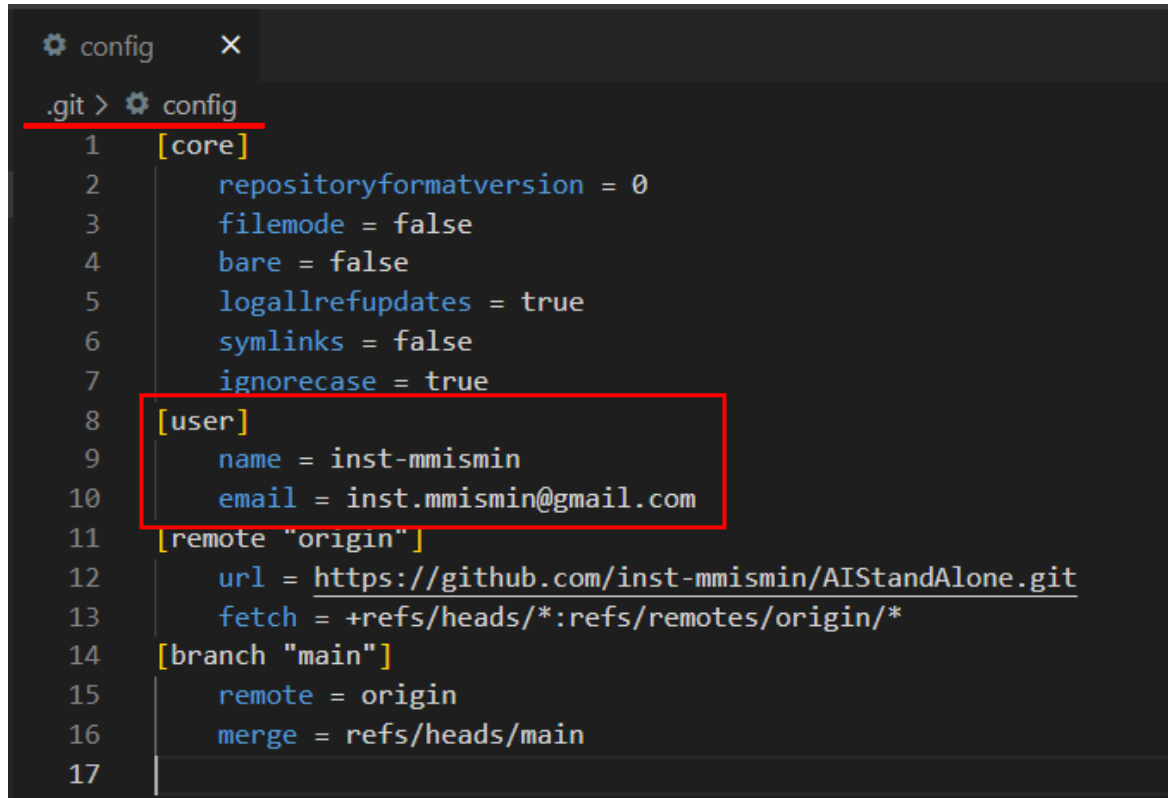
Root

```
|— utils/ # [폴더] 자주 공통적으로 사용되는 코드나 클래스 (전처리 등)
|— mics/ # [폴더] 특정 목적으로만 사용하는 코드나 클래스 (시각화, 데이터 분석 등)
|— modules/ # [폴더] Loss, Optimizer, dataset, collator 등
|— networks/ # [폴더] AI model 클래스
|— train.py # 학습 코드
|— eval.py # 평가 코드
|— inference.py # 추론 코드
|— requirements.txt # Dependency 명세
|— .gitignore # github 추적 금지 파일
└─ README.md # 리드미 파일
```

VScode와 Github

VScode에서 Github 연결

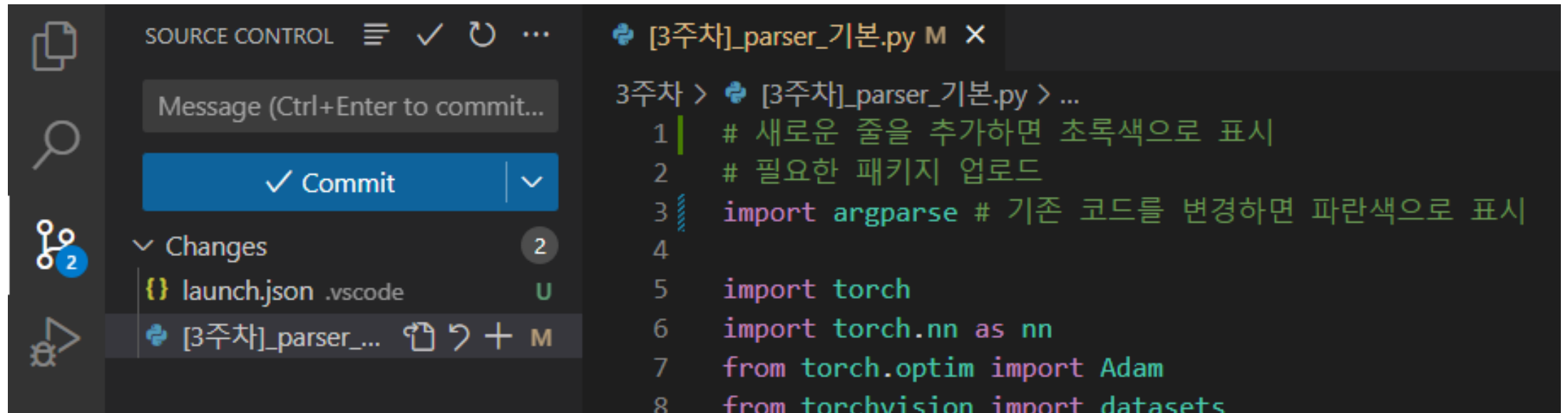
- `git init` 한 폴더에서 vscode를 열면 자동으로 연결
 - `.git` 폴더 속 config의 `user.email` 과 `user.name`을 참조



```
config x
.git > config
1 [core]
2     repositoryformatversion = 0
3     filemode = false
4     bare = false
5     logallrefupdates = true
6     symlinks = false
7     ignorecase = true
8 [user]
9     name = inst-mmismin
10    email = inst.mmismin@gmail.com
11 [remote "origin"]
12     url = https://github.com/inst-mmismin/AIStandAlone.git
13     fetch = +refs/heads/*:refs/remotes/origin/*
14 [branch "main"]
15     remote = origin
16     merge = refs/heads/main
17
```

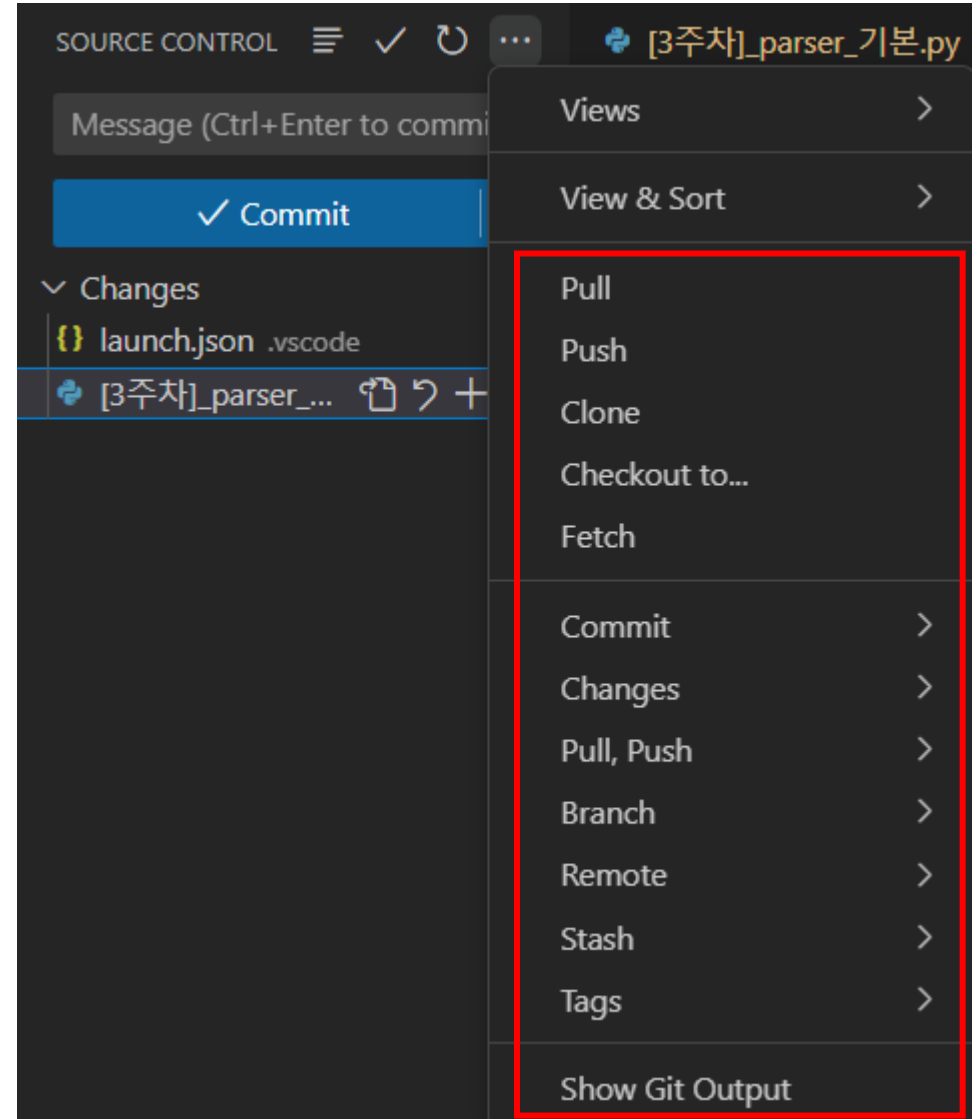

VScode에서 Git이 연결되면..

- Add, Commit, Push를 vscode에서 진행할 수 있음
- 코드를 변경하면 에디터 안에서 시각적으로 보여줌
- Git 페이지에서 변경 내용 표시됨



VScode에 add, commit, push

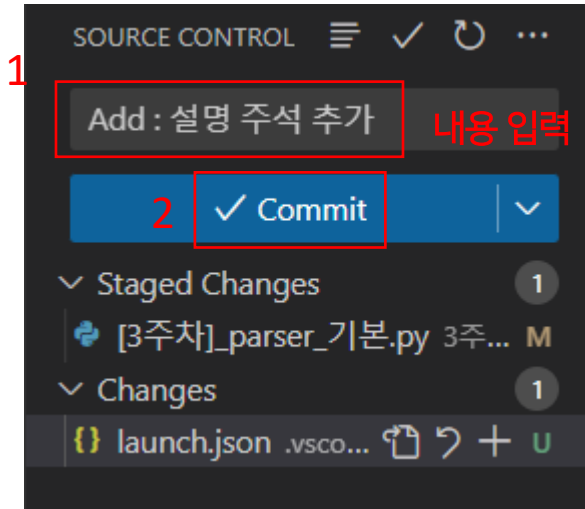
- 기존 command line 기반의 git 명령어에서
- 클릭 기반으로 git 명령어를 수행할 수 있음
- 외에도 pull, add 취소, commit 취소 등을 할 수 있음



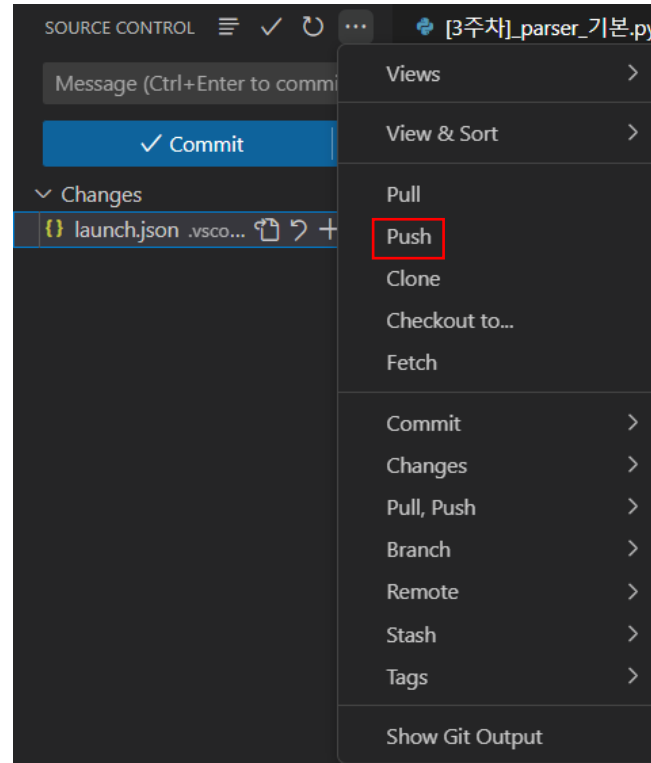
VScode에 add, commit, push



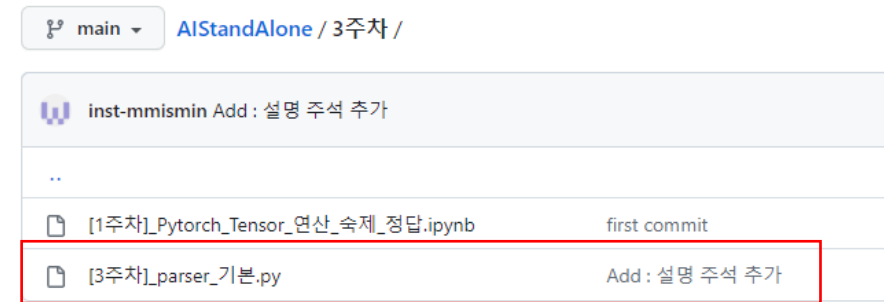
Add



Commit



Push



원격 저장소
변경 결과

과제

- Debug
 - 각자의 개발 환경에 Debug 세팅을 진행하고
 - 11페이지 (Debug Mode 화면 모습)과 같은 실행 중간 모습을 캡처해서 보내기
- Github
 - 본인 repo에 코드를 업로드 한 뒤, 저에게 Manage Access를 주세요. (ID : inst-mmismin)
 - 주중, 주말 코드 모두 업데이트
 - Data는 업로드 하지 않도록 주의!!