목차

[**[R 설치 ]** 2](#_Toc32761988)

[**R 설치** 2](#_Toc32761989)

[**RStudio 설치** 2](#_Toc32761990)

[**Java 설치 및 환경설정** 2](#_Toc32761991)

[**RStudio활용** 2](#_Toc32761992)

[**ANACONDA 설치** 3](#_Toc32761993)

[**CoLab** 4](#_Toc32761994)

# **[R 설치 ]**

## **R 설치**

1. 윈도우 설치

. 위치 : <https://www.r-project.org/> -> Download의 CRAN클릭 -> Korea 위치에 여러 사이트 중 하나를 선택(나는 <https://ftp.harukasan.org/CRAN/>) -> Download and Install R 에서 [Download R for Windows](https://ftp.harukasan.org/CRAN/bin/windows/) 를 클릭 -> base 클릭 -> Download R 3.6.2 for Windows 클릭 -> 다른 이름으로 저장

## **RStudio 설치**

1. R-Studio 설치 (R 설치후에 설치해야 함)

. 위치 : <https://rstudio.com/> -> 맨위에 Download 클릭 -> -> OS가 Windows 10/8/7 인 라인의 설치파일 다운 (유분투 용등 다양하네)

## **Java 설치 및 환경설정**

1. Java 설치

. 위치 : <https://java.com/ko/download/> -> “무료 Java 다운로드” 클릭 -> “동의 및 무료 다운로드 시작” 클릭

1. Java 환경변수 설정

. 위치 : 제어판\시스템 및 보안\시스템 -> 고급시스템설정 클릭 -> 팝업에서 -> 환경변수 클릭 -> 사용자 변수 박스의 path를 클릭 후 새로만들기 버튼 클릭 -> 변수이름에 JAVA\_HOME, 변수값에 자바 설치 위치 C:\Program Files (x86)\Java\jre1.8.0\_241 를 입력(이 부분은 확인이 필요함)

## **RStudio활용**

1. Script 창 : Ctrl + Shift + N 키 눌러 뛰운다

<https://cran.r-project.org/manuals.html>

1. Ctrl + L console 창 지우기,
2. Cntl + Entet 라인에 두고 하면 라인별로 수행 됨

## **ANACONDA 설치**

* <https://www.anaconda.com/> -> 우측상단 다운로드 버튼 -> Windows Installer Python 3.7 version 다운로드 버튼

명령어

$conda install numpy #a Package install

$conda install numpy scipy pandas # many Package install

$conda install numpy = 1.10 # specific ver. install

$conda remove package\_name

$conda update package\_name

$conda update --all

$conda list # installed all package list

$conda seach search\_term # search package for installing

=================================

conda create --name tf2.0-gpu python=3.7.4 (tf2.0-gpu :가상환경이름)

conda activate tf2.0-gpu

conda info –envs

conda install ipykernel jupyter

python -m ipykernel install --user --name tf2.0-gpu --display-name "tf-gpu"

jupyter notebook

pip install tensorflow-gpu

pip install tensorflow

conda install pandas matplotlib numpy

import tensorflow as tf

print(tf.\_\_version\_\_)

tf.test.is\_gpu\_available()

conda deactivate

conda update conda

conda update anaconda

conda update python

1. Numpy

<https://docs.scipy.org/doc/numpy-1.13.0/reference/>

## **CoLab**

* <https://colab.research.google.com/>

1. linux 명령어 실행 (!~)

!cat /etc/issue.net -- colab OS

!head /proc/cpuinfo

!head –n 3 /proc/meminfo

!df –k

!pwd

!ls –ltr

!python --version

pip install tensorflow-hub

GPU : 런타임 -> 런타임유형변경 -> 하드웨어 가속기에 GPU 선택

1. 파일 관리(로컬)

ㅇ 파일 -> 노트 업로드

from google.colab import files

uploaded = files.upload()

colab에 google drive연결

from google.colab import drive

drive.mount('/content/working\_drive')

<https://www.kaggle.com/c/bike-sharing-demand/team>

박성엽phjpsy

%tensorflow\_version 2.x

 %tensorflow\_version 1.x

## **GIT**

**커맨트 라인(Command Line):** 깃 명령어를 입력할 때 사용하는 컴퓨터 프로그램. 맥에선 터미널이라고 한다. PC에선 기본적인 프로그램이 아니어서 처음엔 깃을 다운로드해야 한다(다음 섹션에서 다룰 것이다). 두 경우 모두 마우스를 사용하는 것이 아닌 프롬프트로 알려진 텍스트 기반 명령어를 입력한다.

**저장소(Repository):** 프로젝트가 거주(live)할 수 있는 디렉토리나 저장 공간. 깃허브 사용자는 종종 “repo”로 줄여서 사용한다. 당신의 컴퓨터 안의 로컬 폴더가 될 수도 있고, 깃허브나 다른 온라인 호스트의 저장 공간이 될 수도 있다. 저장소 안에 코드 화일, 텍스트 화일, 이미지 화일을 저장하고, 이름붙일 수 있다.

**버전관리(Version Control):** 기본적으로, 깃이 서비스되도록 고안된 목적. MS 워드 작업할 때, 저장하면 이전 화일 위에 겹쳐쓰거나 여러 버전으로 나누어 저장한다. 깃을 사용하면 그럴 필요가 없다. 프로젝트 히스토리의 모든 시점의 “스냅샷”을 유지하므로, 결코 잃어버리거나 겹쳐쓰지 않을 수 있다.

**커밋(Commit):** 깃에게 파워를 주는 명령이다. 커밋하면, 그 시점의 당신의 저장소의 “스냅샷”을 찍어, 프로젝트를 이전의 어떠한 상태로든 재평가하거나 복원할 수 있는 체크포인트를 가질 수 있다.

**브랜치(Branch):** 여러 명이 하나의 프로젝트에서 깃 없이 작업하는 것이 얼마나 혼란스러울 것인가? 일반적으로, 작업자들은 메인 프로젝트의 브랜치를 따와서(branch off), 자신이 변경하고 싶은 자신만의 버전을 만든다. 작업을 끝낸 후, 프로젝트의 메인 디렉토리인 “master”에 브랜치를 다시 “Merge”한다.

**주요 명령어**

깃은 리눅스와 같은 큰 프로젝트를 염두에 두고 디자인되었기 때문에, 깃 명령어는 아주 많다. 그러나, 깃의 기본을 사용할 때에는 몇 개의 명령어만 알면된다. 모두 “git”이란 단어로 시작된다.

**git init:** 깃 저장소를 초기화한다. 저장소나 디렉토리 안에서 이 명령을 실행하기 전까지는 그냥 일반 폴더이다. 이것을 입력한 후에야 추가적인 깃 명령어들을 줄 수 있다.

**git config:** “configure”의 준말, 처음에 깃을 설정할 때 가장 유용하다.

**git help:** 명령어를 잊어버렸다? 커맨드 라인에 이걸 타이핑하면 21개의 가장 많이 사용하는 깃 명령어들이 나타난다. 좀 더 자세하게 “git help init”이나 다른 용어를 타이핑하여 특정 깃 명령어를 사용하고 설정하는 법을 이해할 수도 있다.

**git status:** 저장소 상태를 체크. 어떤 화일이 저장소 안에 있는지, 커밋이 필요한 변경사항이 있는지, 현재 저장소의 어떤 브랜치에서 작업하고 있는지 등을 볼 수 있다.

**git add:** 이 명령이 저장소에 새 화일들을 추가하진 않는다. 대신, 깃이 새 화일들을 지켜보게 한다. 화일을 추가하면, 깃의 저장소 “스냅샷”에 포함된다.

**git commit:** 깃의 가장 중요한 명령어. 어떤 변경사항이라도 만든 후, 저장소의 “스냅샷”을 찍기 위해 이것을 입력한다. 보통 “git commit -m “Message hear.” 형식으로 사용한다. -m은 명령어의 그 다음 부분을 메시지로 읽어야 한다는 것을 말한다.

**git branch:** 여러 협업자와 작업하고 자신만의 변경을 원한다? 이 명령어는 새로운 브랜치를 만들고, 자신만의 변경사항과 화일 추가 등의 커밋 타임라인을 만든다. 당신의 제목이 명령어 다음에 온다. 새 브랜치를 “cats”로 부르고 싶으면, git branch cats를 타이핑한다.

**git checkout:** 글자 그대로, 현재 위치하고 있지 않은 저장소를 “체크아웃”할 수 있다. 이것은 체크하길 원하는 저장소로 옮겨가게 해주는 탐색 명령이다. master 브랜치를 들여다 보고 싶으면, git checkout master를 사용할 수 있고, git checkout cats로 또 다른 브랜치를 들여다 볼 수 있다.

**git merge:** 브랜치에서 작업을 끝내고, 모든 협업자가 볼 수 있는 master 브랜치로 병합할 수 있다. git merge cats는 “cats” 브랜치에서 만든 모든 변경사항을 master로 추가한다.

**git push:** 로컬 컴퓨터에서 작업하고 당신의 커밋을 깃허브에서 온라인으로도 볼 수 있기를 원한다면, 이 명령어로 깃허브에 변경사항을 “push”한다.

**git pull:** 로컬 컴퓨터에서 작업할 때, 작업하고 있는 저장소의 최신 버전을 원하면, 이 명령어로 깃허브로부터 변경사항을 다운로드한다(“pull”).

init은 “initialize(초기화)”를 뜻한다. 이 코드를 입력하면 이 디렉토리를 로컬 깃 저장소라고 컴퓨터에게 말해주는 것이다.

git init

touch ppp.txt

git add ppp.txt

git commit -m “Add ppp.txt”

git remote add origin https://github.com/phjpsy/Colabb.git

git remote –v

git push

git config --global user.name "이름"

git config --global user.email "깃허브 메일주소" // 매번 물어보는 귀찮음을 피하기 위해 설정.

mkdir ~/MyProject // 로컬 디렉토리 만들고

cd ~/myproject // 디렉토리로 들어가서

git init // 깃 명령어를 사용할 수 있는 디렉토리로 만든다.

git status // 현재 상태를 훑어보고

git config --global phjpsy.email "phjpsy@naver.com"

git add 화일명.확장자 // 깃 주목 리스트에 화일을 추가하고 or

git add . // 이 명령은 현재 디렉토리의 모든 화일을 추가할 수 있다.

git commit -m “현재형으로 설명” // 커밋해서 스냅샷을 찍는다.

git remote add origin https://github.com/username/myproject.git // 로컬과 원격 저장소를 연결한다.

git remote -v // 연결상태를 확인한다.

git push origin master // 깃허브로 푸시한다.

$ touch initial

$ git add initial

$ git commit -m "inital commit"

$ git push -u origin master git remote add origin-push $(git config remote.origin.url)

$ git fetch origin-push

$ git push --force-with-lease origin-push

$ git fetch

git remote add origin-push $(git config remote.origin.url)

git fetch origin-push

git fetch # update 'master' from remote

git tag base master # mark our base point

git rebase -i master # rewrite some commits

git push --force-with-lease=master:base master:master

**아래로 하면 되네,, 위는 도치 모르겠음당**

$ git add .

$ git commit -m "inital commit"

$ git push --force-with-lease origin-push

1. GitBash 설치

<https://gitforwindows.org>

1. New Repository 생성

<https://github.com>

git init

git add README.md

git commit -m "first commit"

git remote add origin https://github.com/phjpsy/VUE.git

git push -u origin master

1. Root Folder 연결

D:\Study위치의 \VUE 폴드에 마우스 놓고 오른쪽 클릭하여 “git bash here” 클릭하여

git bash창 뛰움

$ git init

$ git remote add origin https://github.com/phjpsy/VUE.git

$ git status

$ git add .

$ git commit -m "first make 20200226"

$ git push origin master

다른 분 것도 가져와 보자

$ git init

$ git remote add origin <https://github.com/joshua1988/tacademy-vue>

$ git pull origin master

$ git status

$ git add .

$ git commit -m "first make 20200226"

$ git push origin master

1. ㅇ7
2. ㅇ

## **VUE**

Visual Studio에서 1~7 수행…

1. <https://code.visualstudio.com/> visual studio 설치후 D:\study\tacademy-vue-master 폴더를 그대로 댕겨오면 visual studio 올라 감
2. in Visual Studio , 좌측에 확장을 눌러서 검색한 후 눌러 설치한다

* liver server
* vetur -- html편하게 작성 도구
* material icon theme
* night owl
* eclipse keymap

1. cli : cntl + shift + p

>color theme 치면 아래 리스트 나오고 그중 선택하면 화면 색등이 바뀜

cli : cntl + shift + p

>file icon 리스트에서 선택

1. <https://nodejs.org/ko/> 설치 ,,, 아래는 단순히 node기능을 확인 한 것

node.js 실행 :

$ node -- node mode in

> 1 + 2 -- 노드 REPL(read-eval-print loop), 자바스크립트 코드를 입력, 즉시 결과 확인할 수 있는 프로그램

> .exit -- node mode out

C:\Users\phjps>node –version

C:\Users\phjps>npm

Usage: npm <command> …..

C:\Users\phjps>npm –version

C:\Users\phjps>npm init -- 프로젝트 생성

실행한 폴더에 pacakge.json 파일이 생성

1. Node.js 설치후 path 에 C:\Program Files\nodejs 를 추가 해주어야

Visual Studio에서 node -v, npm -v 가 실행됨 (확인이 되어야 6번 설치 가능)

1. <https://angular.io/guide/setup-local> 에 접속하면 npm install -g @angular/cli

이 나오는데, 이걸 복사해서, 메뉴에서 Terminal -> new terminal을 열어 복사하여 실행 한다

1. ng new my-angular-app -- C:\study\tacademy-vue-master\frameworks> 로 이동하여 프로젝트 생성 (윈도우 탐색에서 powershell 쳐서 .. 나온 것 중 권리자 권한으로 실행을 하여 아래 처름 권한 변경 후 해야 설치됨)

PS C:\Windows\system32> Set-ExecutionPolicy RemoteSigned – 권한 변경

(base) PS C:\Windows\system32> Get-ExecutionPolicy – 권한변경 확인

1. cd my-angular-app

ng serve --open -- 서버 오픈하면 화면이 뜨고 F12를 누르면, 소스가 보임

C:\study\tacademy-vue-master\frameworks\my-angular-app\src\app에서 app.component.html을 모두 지우고 간단이 <div> Hello </div>만 해서 다시 저장하면 웹화면이 바뀐 걸 볼 수 있음

app.component.html 을 <div> {{title}} </div> 로 수정도 해 보자

(app.component.ts에export class AppComponent 클래스에 있는 title 이 나옴)

[Angular 시작하기]

1. npm install -g @angular/cli
2. cd C:\study\tacademy-vue-master\frameworks
3. ng new my-angular-app -- project 생성
4. cd my-angular-app
5. ng serve --open -- 서버 오픈/요걸해야 바뀌네
6. app/app.component.ts, app/app.component.html 파일 수정하여 Hello World 출력

<div> Hello </div>만 or app.component.html 을 <div> {{title}} </div> 로 수정도

[React 시작하기]

1. frameworks 폴드에서 우클릭, new Folder 선택, ‘my-react-app’를 생성
2. my-react-app 폴드에서 우클릭, new File 선택, ‘index.html’을 생성
3. index.html에 아래 복사하여 붙여 넣음

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" />

<title>Hello React!</title>

<script src="https://unpkg.com/react@16/umd/react.development.js"></script>

<script src="https://unpkg.com/react-dom@16/umd/react-dom.development.js"></script>

<script src="https://unpkg.com/babel-standalone@6.26.0/babel.js"></script>

</head>

<body>

<div id="root"></div>

<script type="text/babel">

class Root extends React.Component {

constructor() {

super();

this.state = {

message: 'hi'

}

}

render() {

return <h1>{this.state.message}</h1>

}

}

ReactDOM.render(<Root />, document.getElementById('root'));

</script>

</body>

</html>

1. 파일에서 우클릭하여 ‘open with live server’ 선택, 웹을 오픈

[Vue.js 시작하기]

1. <https://kr.vuejs.org/v2/guide/index.html> -> 죄측 Getting Started 클릭, 개발용 소스 위치 <https://cdn.jsdelivr.net/npm/vue/dist/vue.js> 를 url에 붙이면 소스가 나옴
2. frameworks 폴드에서 우클릭, new Folder 선택, ‘my-vue-app’를 생성
3. my-vue-app 폴드에서 우클릭, new File 선택, ‘index.html’을 생성
4. index.html에 아래 복사하여 붙여 넣음

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>Hello Vue</title>

</head>

<body>

<div id="app">

{{ str }}

</div>

<script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>

<script>

new Vue({

el: '#app',

data: {

str: 'Hello Vue!'

}

})

</script>

</body>

</html>

1. 파일에서 우클릭하여 ‘open with live server’ 선택, 웹을 오

크롬에서 F12눌러 소스등 볼 때, VUE 기능도 같이 하려면

<https://chrome.google.com/webstore/detail/vuejs-devtools/nhdogjmejiglipccpnnnanhbledajbpd>에 접속하여 “Chrome에 추가” 버튼을 클릭 한다 (나중에 debugger등에 유용)

// ctrl+shift+c

사용법 등 - Google에 Object.defineProperty mdn 로 검색(항시 mdn을 붙인다)

* Vue.JS 코어
* Vue Router
* Vuex
* Vue CLI
* Vue Devtools

[Vue본격학습]

1. vue폴드 생성, old.html 생성
2. ! key in후 Tab 누르면 기본 뼈대 생성 됨, body에 div+Tab,

**커맨트 라인(Command Line):** 깃 명령어를 입력할 때 사용하는 컴퓨터 프로그램. 맥에선 터미널이라고 한다. PC에선 기본적인 프로그램이 아니어서 처음엔 깃을 다운로드해야 한다(다음 섹션에서 다룰 것

1. ㅇㅇ