

[illegible]

한국전자통신연구원(ETRI) 인공지능연구소 지능정보연구본부  
 작성자: 박 준 용

# 수정 이력

버전	수정 날짜	수정자	수정 내용
1.0	2020.09.23	박준용	최초 작성
1.1	2020.09.24	신익희	conda install tensorflow==2.2.0에서 tensorflow-gpu==2.2.0으로 변경

# Step 1. Anaconda 소개

- Anaconda란?
  - <https://www.anaconda.com/>
  - 데이터 분석을 하기 위한 파이썬을 사용하는 개발 환경 통합 패키지
  - 독립적인 라이브러리 구성, 별도의 추가설치 필요 없음
  - 윈도우, 맥, 리눅스 전부 설치 가능
  - <https://www.anaconda.com/products/individual>

www.anaconda.com ▼

## Anaconda | The World's Most Popular Data Science Platform

**Anaconda** is the birthplace of Python data science. We are a movement of data scientists, data-driven enterprises, and open source communities.

You've visited this page 2 times. Last visit: 5/26/20

### Installation

Installing on Windows - Installing on macOS - Installing on Linux

### Individual Edition

Anaconda Individual Edition is the world's most popular Python ...

### Installing on Windows

Select an install for "Just Me" unless you're installing for all ...

[More results from anaconda.com »](#)

### Pricing

Products and pricing for every data scientist. Whether you are a ...

### Anaconda Individual Edition

Installation - User guide - Anaconda Enterprise 4 - ...

### Open Source

We are proud to distribute and contribute to a variety of open ...

## Anaconda Installers

Windows 

Python 3.8  
64-Bit Graphical Installer (466 MB)

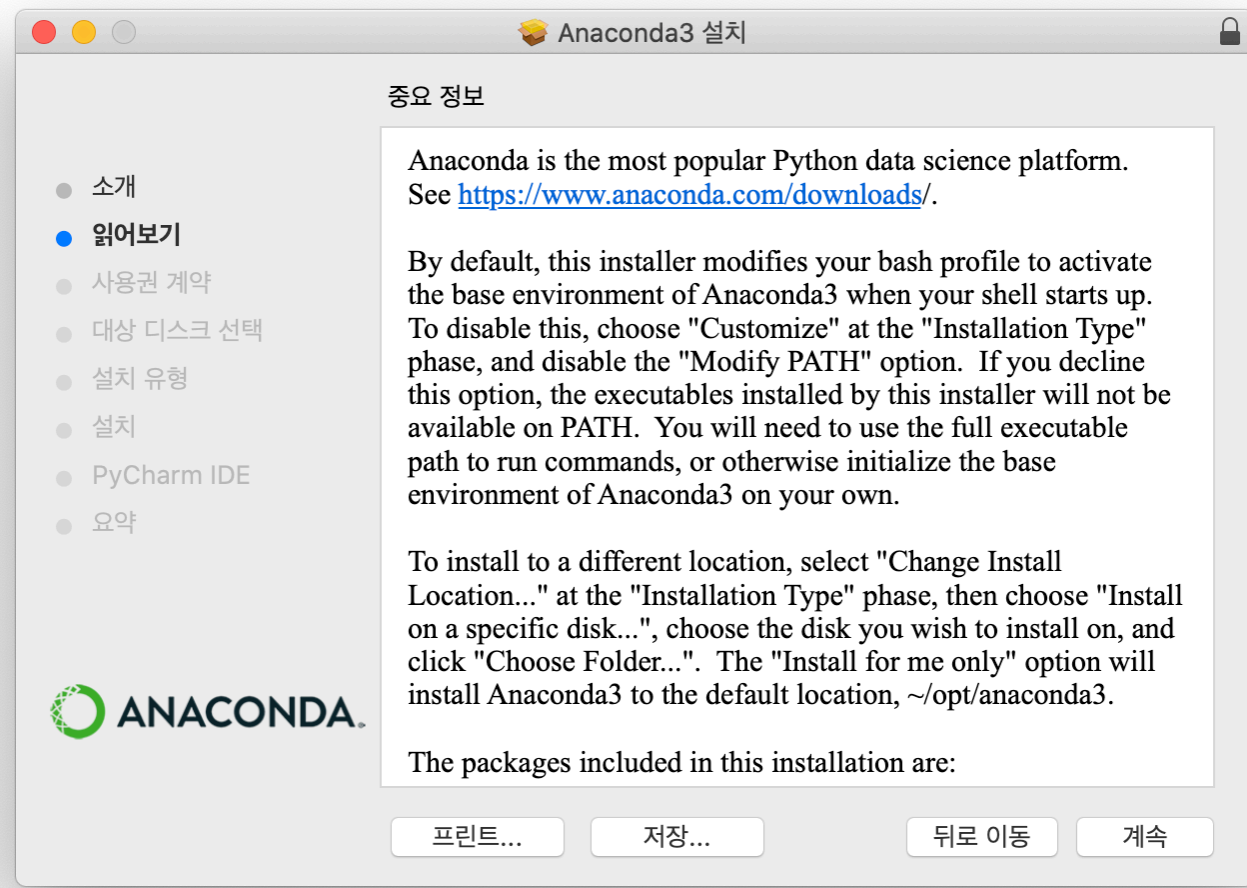
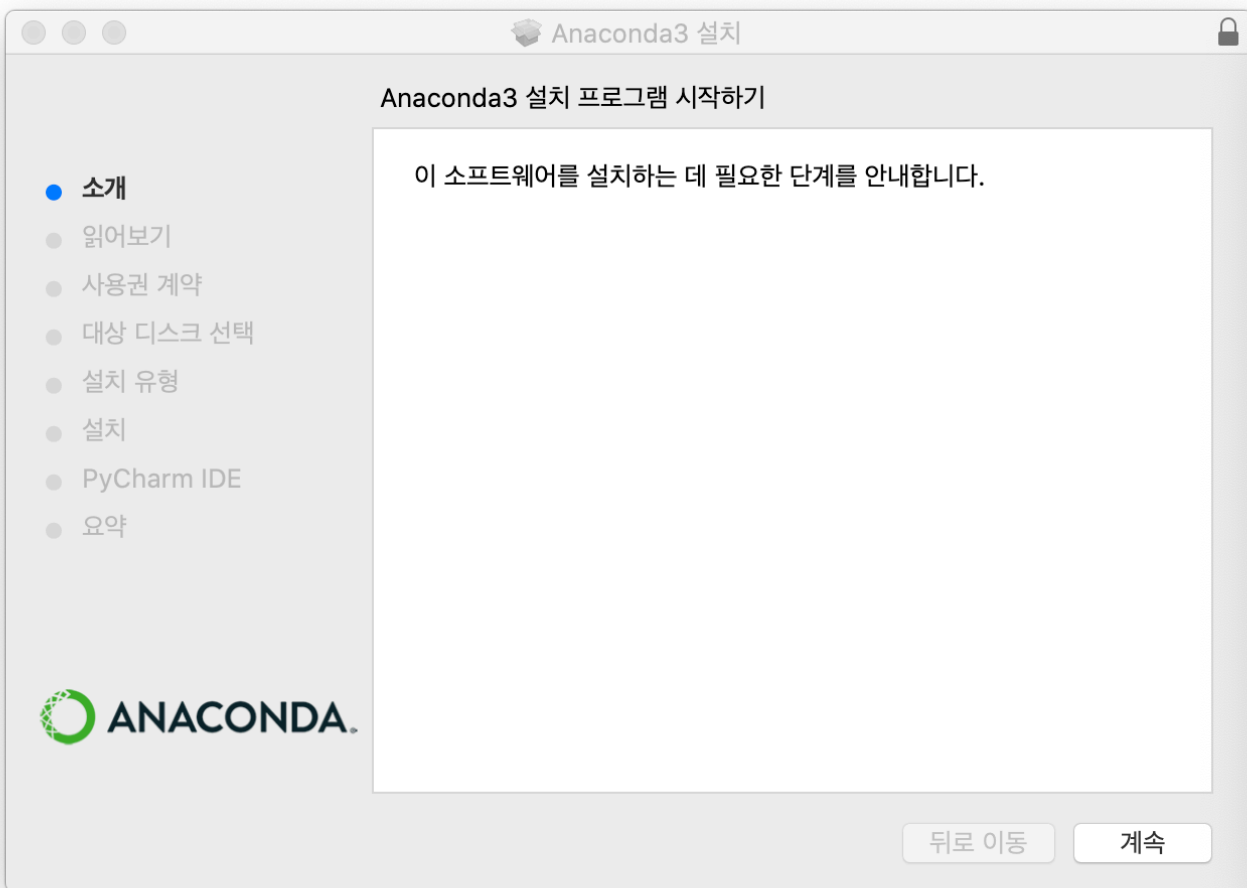
MacOS 

Python 3.8  
64-Bit Graphical Installer (462 MB)

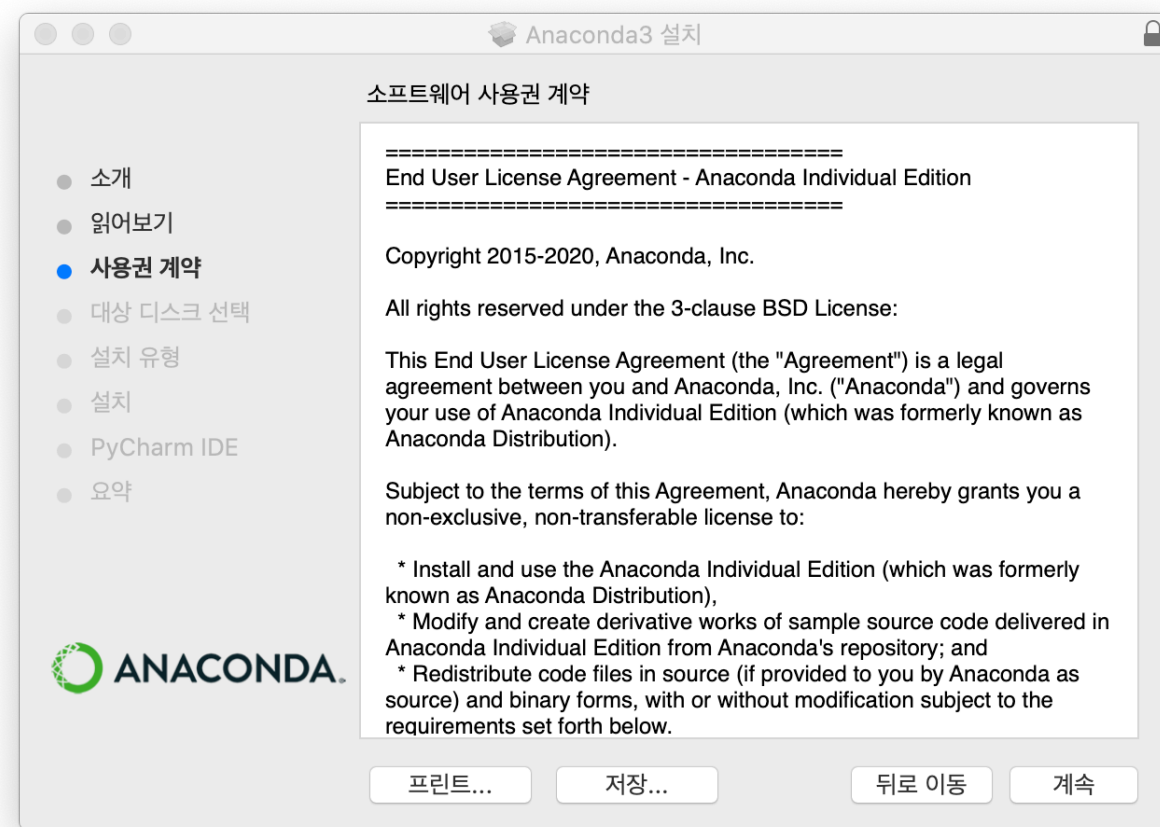
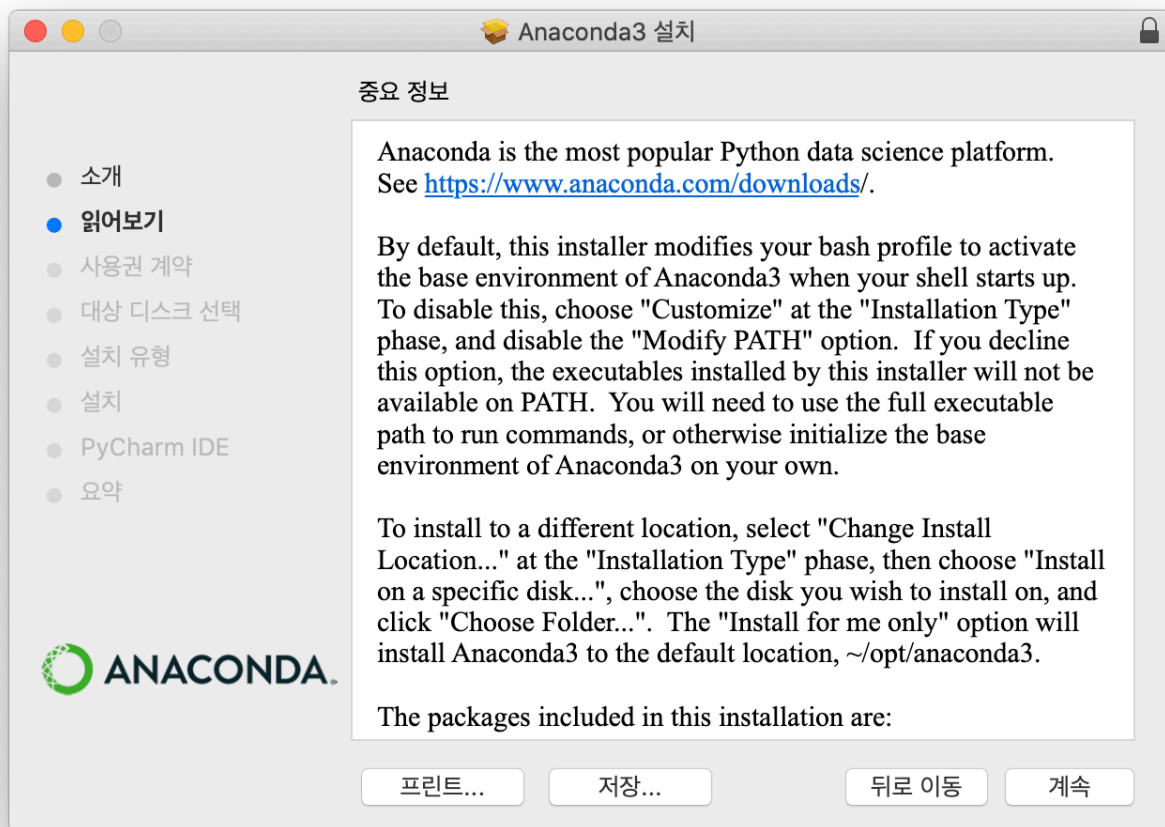
Linux 

Python 3.8  
64-Bit (x86) Installer (550 MB)

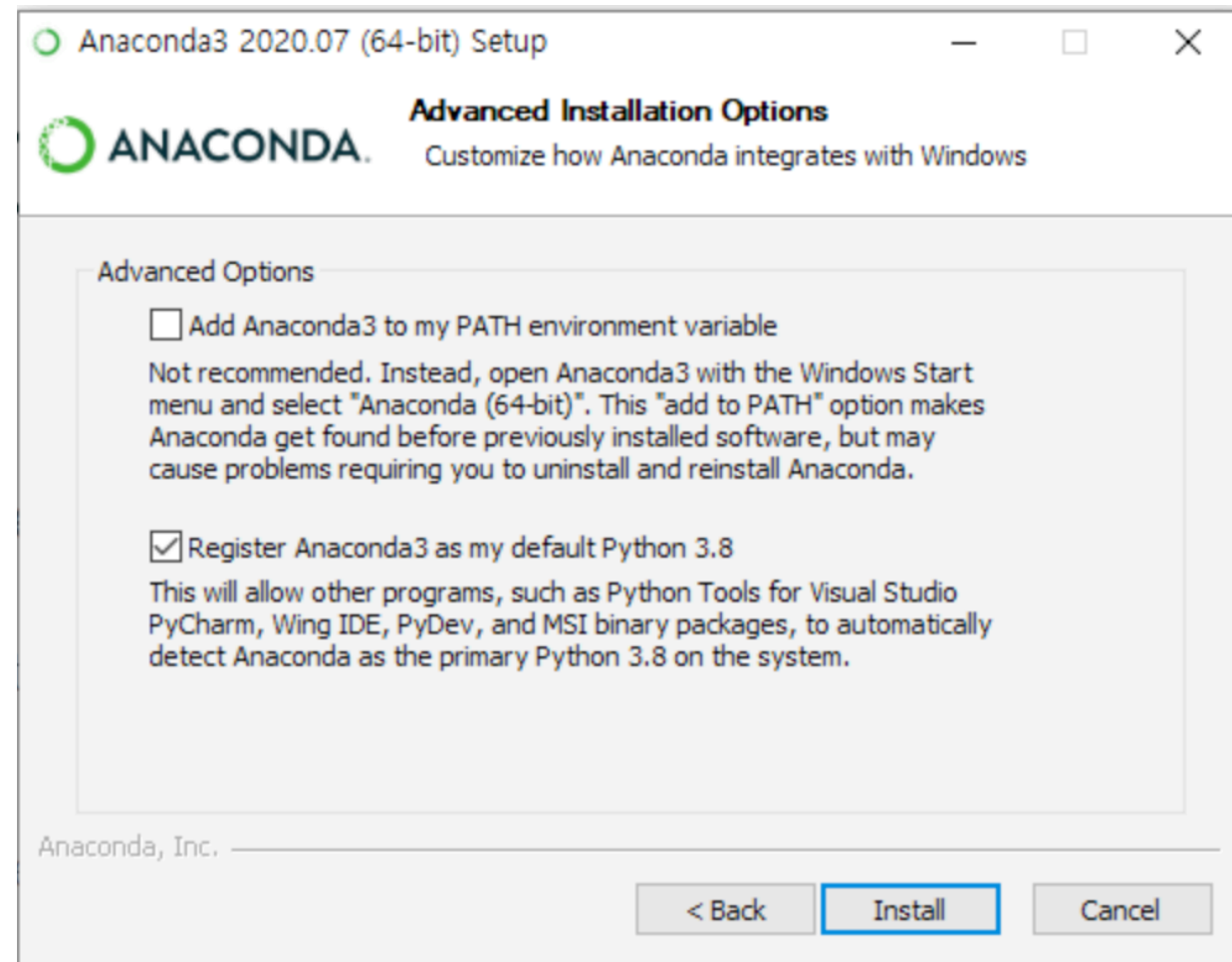
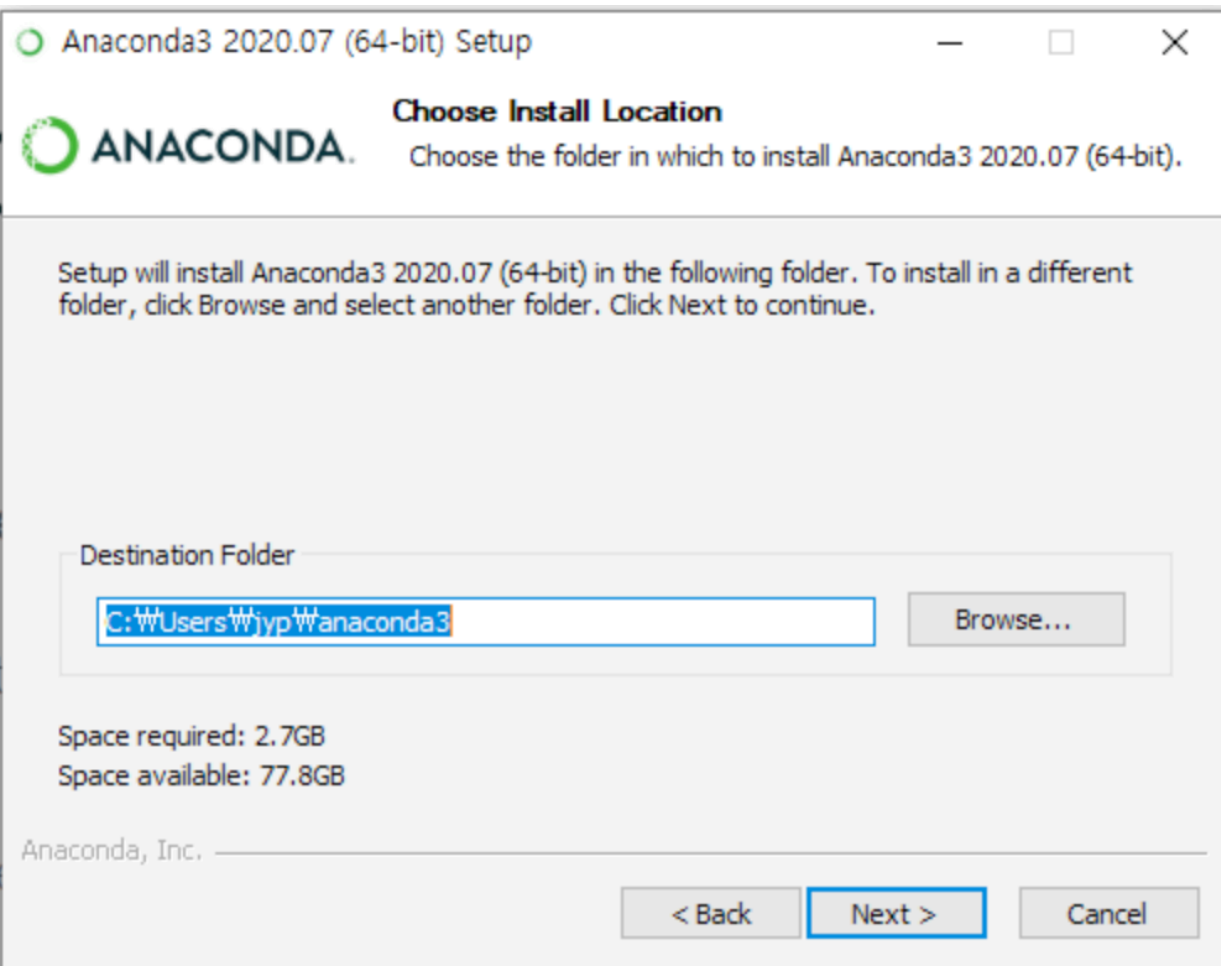
# Step 2. Anaconda 설치



# Step 2. Anaconda 설치

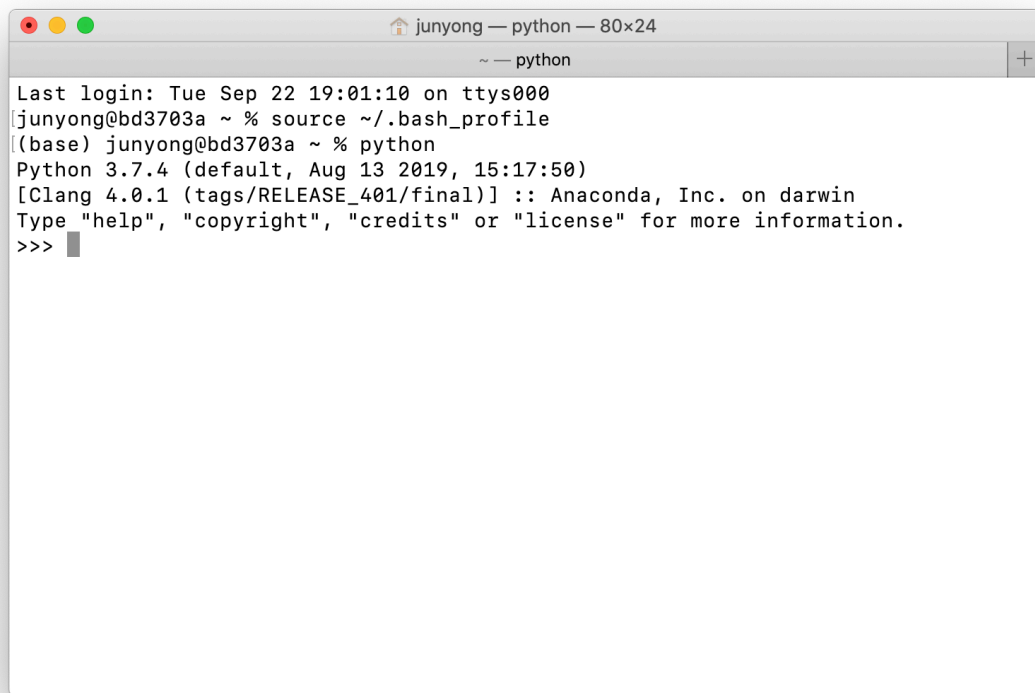


## Step 2. Anaconda 설치(Windows)



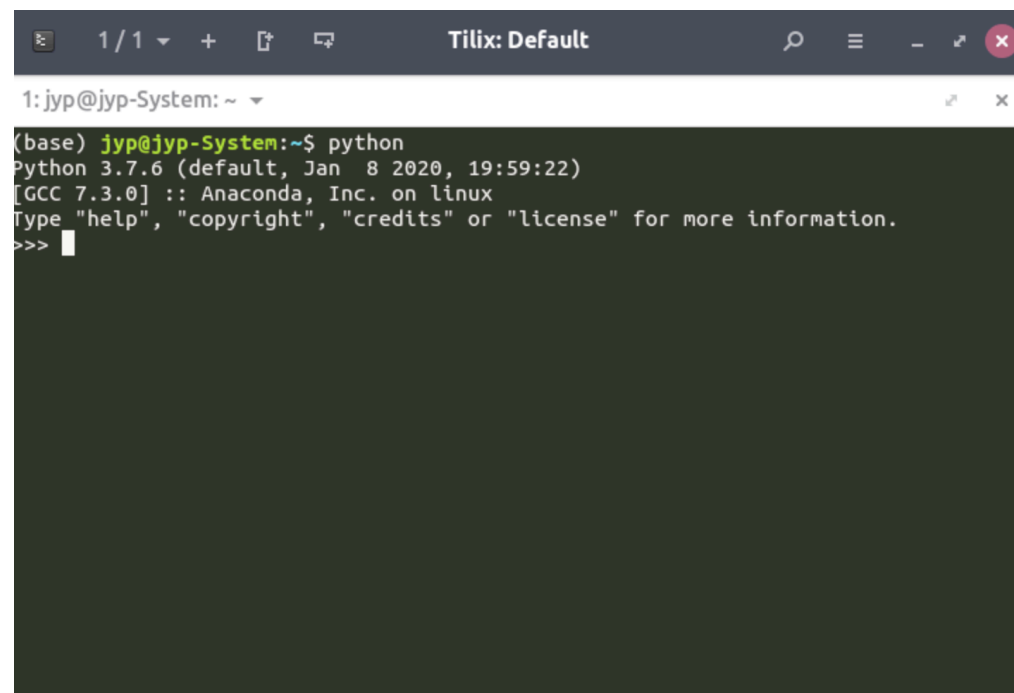
## Step 2. Anaconda 실행

- Windows: [시작 버튼] -> Anaconda Prompt 실행
- Mac OSX: Terminal 사용
- Linux: Terminal 사용



```
junyong — python — 80x24
~ — python

Last login: Tue Sep 22 19:01:10 on ttys000
[junyong@bd3703a ~ % source ~/.bash_profile
[(base) junyong@bd3703a ~ % python
Python 3.7.4 (default, Aug 13 2019, 15:17:50)
[Clang 4.0.1 (tags/RELEASE_401/final)] :: Anaconda, Inc. on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```



```
Tilix: Default
1: jyp@jyp-System: ~ ▾

(base) jyp@jyp-System:~$ python
Python 3.7.6 (default, Jan 8 2020, 19:59:22)
[GCC 7.3.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

## Step 2. Anaconda 실행

- Conda 확인: `conda info`
- Conda 업데이트: `conda update conda`
- 라이브러리 설치: `conda install [이름]`
- 라이브러리 업데이트: `conda update [이름]`

- 참고:

[https://docs.conda.io/projects/conda/en/4.6.0/\\_downloads/52a95608c49671267e40c689e0bc00ca/conda-cheatsheet.pdf](https://docs.conda.io/projects/conda/en/4.6.0/_downloads/52a95608c49671267e40c689e0bc00ca/conda-cheatsheet.pdf)



## Step 3. Virtual Environment 설치

- 가상환경(Virtual environment)이란 라이브러리와 버전을 따로 관리 할 수 있도록 가상의 고립된 개발환경 입
- 개발 환경 설정할 때와 패키지 관리하기에 만들어놓는 것이 중요
- 관리의 용이를 위해 여러 개 만들어놓을 수도 있음
- Conda 가상환경 새로 만들기(처음에만) : `conda create --name [이름] python=3.7`
- 가상환경에 접속하기 :
  - 윈도우: `activate [이름]`
  - Linux, Mac: `source activate [이름]`
- Conda 가상환경 목록: `conda env list`
- 참고: <https://docs.conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html>

## Step 4. 패키지 설치

- Conda 패키지 설치:
- `conda install tensorflow-gpu==2.2.0 pandas numpy numba matplotlib seaborn xgboost scikit-learn openpyxl xlrd tqdm pickle plotly plotly_express joblib`
- 기본적인 설치 명령어 이외에 추가적인 설치 명령어:
- `Conda install [이름]`
- 위와 같이 가상환경 및 패키지 설치가 끝나면 다음장에 실행 매뉴얼 진행

프  
- 토마토 수확

한국전자통신연구원(ETRI) 인공지능연구소 지능정보연구본부  
 작성자: 신 익 희

# 수정 이력

버전	수정 날짜	수정자	수정 내용
1.0	2020.09.24	신익희	최초 작성
1.1	2020.10.04	신익희	사용하지 않는 arguments 및 API들 제거

# 목차

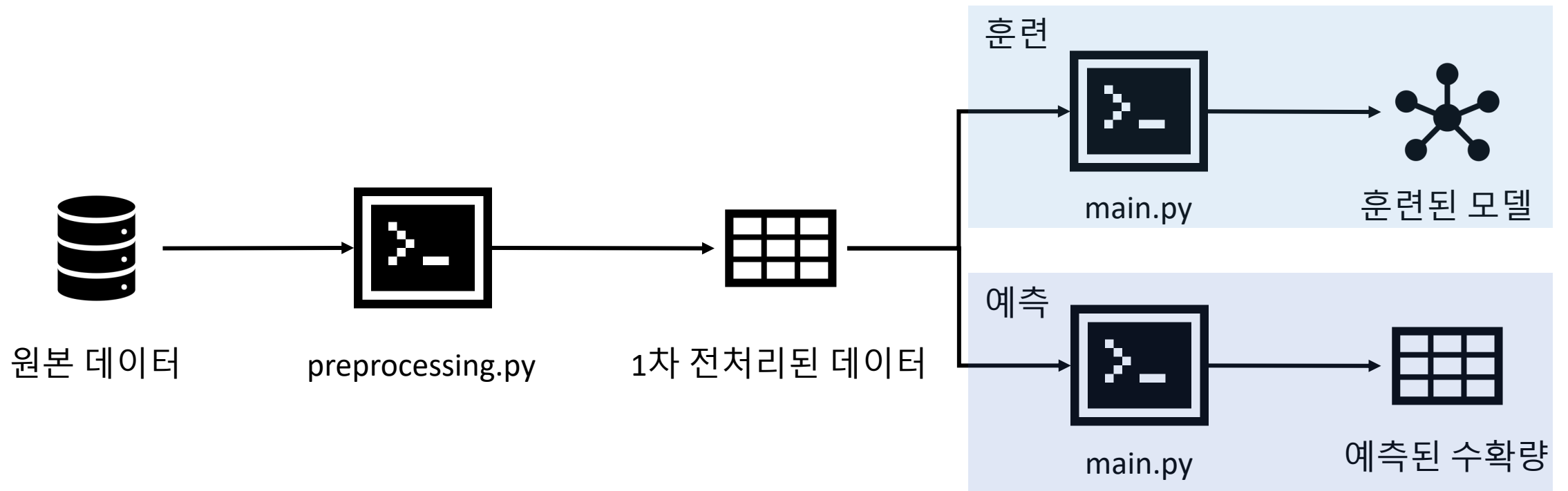
- 필요한 패키지 목록
- 프로그램 구성
- 데이터 전처리 프로그램
  - 프로그램 실행 방법
  - json 파일 규격 설명
- 모델 훈련/예측 프로그램
  - 생장 예측 모델
    - 프로그램 실행 방법
    - json 파일 규격 설명
  - 수확 예측 모델
    - 프로그램 실행 방법
    - json 파일 규격 설명
- API 설명

# 필요한 패키지 목록

- tensorflow-gpu 2.2.0
- pandas
- numpy
- numba
- matplotlib
- plotly
- plotly\_express
- xgboost
- scikit-learn
- openpyxl
- xlrd
- tqdm

# 프로그램 구성

- 스마트팜 토마토 수확량 예측 프로그램은 크게 데이터 전처리(1차), 모델 훈련, 예측으로 구성됨.
- 모델 훈련과 예측을 위해서는 데이터 전처리를 통해 데이터를 딥러닝 모델이 요구하는 형태로 맞추는 작업이 필요함.



# 프로그램 구성

- 프로그램 구조

- configs
- dataloader
- dataset
- experiments
- models
- optimizer
- utils
- main.py
- preprocessing.py
- runner.py



# 프로그램 구성

- 프로그램 구조

- configs

- └ evn\_config.json - 명성호 농장에 대해 환경 데이터만 사용해서 수확 예측
    - └ myeong.json - 명성호 농장 수확 예측
    - └ rf.json - 명성호 농장 생육 예측

- dataloader

- └ datautils.py - pandas DataFrame을 numpy array로 변환
    - └ loader.py - 모델에 맞는 dataloader 반환
    - └ lstm\_enc\_dec\_dataloader.py - 수확 예측을 위한 dataloader
    - └ rf\_dataloader.py - 생장 예측을 위한 dataloader

# 프로그램 구성

- 프로그램 구조

- dataset

- └ 2nd\_data

- └ beak - 백승호 농장 데이터

- └ beak\_env\_detail\_data.xlsx - 1차 전처리가 완료된 백승호 농장의 환경 데이터

- └ beak\_growth\_detail\_data.xlsx - 1차 전처리가 완료된 백승호 농장의 생육 데이터

- └ beak\_product\_1\_data.xlsx - 백승호 농장의 개화수 데이터

- └ beak\_product\_2\_data.xlsx - 백승호 농장의 착화수 데이터

- └ beak\_product\_3\_data.xlsx - 백승호 농장의 수확수 데이터

- └ beak\_product\_4\_data.xlsx - 백승호 농장의 수확 무게 데이터

- └ hwang - 황정현 농장 데이터

- └ kim - 김기성 농장 데이터

- └ myeong - 명성호 농장 데이터

- └ preprocessing - 데이터 전처리를 위한 샘플 데이터

# 프로그램 구성

- 프로그램 구조

- experiments

- └ lstm\_enc\_dec - 수확 예측 결과가 저장되는 디렉터리
    - └ rf - 생장 예측 결과가 저장되는 디렉터리

- models

- └ attention.py - attention 모델
    - └ lstm\_enc\_dec.py - 수확 예측 모델
    - └ rf.py - 생장 예측 모델
    - └ model.py - json 파일에 명시된 모델을 생성

# 프로그램 구성

- 프로그램 구조

- optimizer

- └ optimizer.py - 모델 학습에 사용할 최적화 알고리즘

- utils

- └ confloader.py - json 파일을 parsing해서 python object 로 반환

- └ csvutils.py - tensorflow의 tensor 데이터를 csv로 저장

- └ measures.py - 오차율 계산

- └ plotutils.py - 모델의 예측 결과를 해석하기 위한 그래프 생성

- └ rader.py - 생장 데이터에 대한 방사형 그래프 생성

# 프로그램 구성

- 프로그램 구조

- main.py - json 파일을 읽어 runner.py 실행
- preprocessing.py - 환경과 생육 데이터에 대한 1차 전처리 수행
- runner.py - 모델 생성, 훈련 및 예측을 수행

# 데이터 전처리 프로그램

- 프로그램 요약

- 환경과 생육 데이터에 대한 전처리를 실행
- 환경 데이터 - 평균, 표준편차, 최소, 최대, HD 관련 feature, 안좋은 환경 feature, 적정 환경 feature
- 생육 데이터 - 기본 생육 데이터, 생장 상태, 생장 상태 점수, 생장 구분

- 실행 방법

- `python preprocessing.py --config_path [your_config_json_file_path]`
- e.g., `python preprocessing.py --config_path ./configs/preprocessing.json`

# 데이터 전처리 프로그램

- json 파일 규격 설명

```
{
  "env":{
    "src_path": "./dataset/2nd_data/preprocessing",
    "dest_path": "./dataset/2nd_data/preprocessing",
    "names": ["HD_2018_01_Hwang.xlsx", "HD_2018_02_Hwang.xlsx", "HD_2018_03_Hwang.xlsx", "HD_2018_04_Hwang.xlsx", "HD_2018_05_Hwang.xlsx"],
    "save_name": "env_detail.xlsx"
  },
  "growth":{
    "src_path": "./dataset/2nd_data/preprocessing",
    "dest_path": "./dataset/2nd_data/preprocessing",
    "names": "growth_2nd.xlsx",
    "save_name": "growth_detail.xlsx",
    "farm_area": 7590,
    "planted_hills": 18600
  }
}
```

- 환경 데이터에 대한 설정과 생육 데이터에 대한 설정을 env, growth로 구분
- src\_path: 원본 데이터 경로
- dest\_path: 전처리한 데이터를 저장할 경로
- names: 원본 데이터 파일 이름
- save\_name: 전처리한 데이터를 저장할 파일 이름
- farm\_area: 엽면적지수 계산을 위한 농장 면적
- planted\_hills: 엽면적지수 계산을 위한 재식주수

# 모델 훈련/예측 프로그램

- **생장 예측 모델**

- 다음주 주간생육길이, 줄기굵기, 잎길이, 잎폭과 엽면적 지수를 예측
- 입력 데이터 - 사용자가 설정한 기간의 환경 데이터, 과거 주간생육길이, 줄기굵기, 잎길이, 잎폭과 엽면적 지수

- **수확 예측 모델**

- 화방당 평균 수확 무게를 예측
- 입력 데이터 - 사용자가 설정한 기간의 환경 데이터, 현재 생육 데이터, 현재 개화수 데이터, 현재 착화수 데이터

- **실행 방법**

- `python main.py --config_path [your_predict_config_json_file_path]`
- e.g.
  - `python main.py --config_path ./configs/myeong.json`
  - `python main.py --config_path ./configs/mlp.json`



# 생장 예측 모델

- json 파일 규격

```
"predict": "rf_train",  
"directory": "dataset/2nd_data/myeong/",  
"experiment": "experiments/rf",  
"model_file": "/models/growth_model.h5",
```

- predict: 모델 별 훈련/예측 구분
  - mlp\_train - 생장 예측 모델 훈련, mlp\_infer - 생장 예측
- directory: dataset 경로
- experiment: 모델의 훈련/예측 결과를 저장할 경로
- model\_file: 훈련이 완료된 모델이 저장된 경로

# 생장 예측 모델

- json 파일 규격

```
"train_data":{  
  "grw": "myeong_growth_detail_dataa.xlsx",  
  "env" : "myeong_env_detail_data.xlsx",  
  "product_1" : "myeong_product_1_data.xlsx",  
  "product_2" : "myeong_product_2_data.xlsx",  
  "product_3" : "myeong_product_3_data.xlsx",  
  "product_4" : "myeong_product_4_data.xlsx"  
},
```

- train\_data: 훈련용 데이터 파일명
  - grw: 생육 데이터 파일명
  - env: 환경 데이터 파일명
  - product\_1: 개화수 데이터 파일명
  - product\_2: 착과수 데이터 파일명
  - product\_3: 수확수 데이터 파일명
  - product\_4: 수확 무게 데이터 파일명

- json 파일 규격

```
"test_data":{  
  "directory":"dataset/2nd_data/myeong/",  
  "features" : "/test_dataset.csv",  
  "labels" : "/test_labels.csv"  
},
```

- test\_data: 생장 예측에 사용될 데이터
  - directory: 데이터의 위치
  - features: 데이터 파일명
  - labels: 정답 데이터 파일명

# 생장 예측 모델

- json 파일 규격

```
"output":{  
  "prediction":"/prediction.csv",  
  "score":"/score.csv",  
  "feature_importance":"/feature_importance.png"  
},
```

- output: 훈련/예측 결과물을 저장할 파일명
  - prediction: test data에 대한 모델의 예측 값을 저장할 파일명
  - score: test data에 대한 모델의 오차율을 저장할 파일명
  - feature\_importance: 모델의 예측 결과에 대한 feature importance 그래프 파일명

# 생장 예측 모델

- json 파일 규격

```
"model":{  
  "name":"rf"  
},
```

- model: 생장 예측 모델 생성에 필요한 입력 파라미터
  - name: 모델의 이름, 모델 생성시 필요

# 생장 예측 모델

- json 파일 규격

```
"util":{  
  "path": "./experiments/rf",  
  "model": "/rf_env2grw",  
  "train_stats":"/train_stats.csv"  
}
```

- util: 모델 훈련의 결과물을 사용하기 위한 입력 파라미터
  - path: 결과물이 저장될 경로
  - model: 모델 파일 이름
  - train\_stats: 훈련 데이터의 표준편차와 평균이 저장될 파일, test 데이터 정규화에 사용됨

# 수확 예측 모델

- json 파일 규격

```
"predict": "multi_encoder_train",
"data": {
  "train_data": {
    "path": "./dataset/2nd_data/myeong",
    "names": ["myeong_env_detail_data.xlsx", "myeong_growth_detail_data.xlsx", "myeong_product_1_data.xlsx", "myeong_product_2_data.xlsx"]
  },
  "label_data": {
    "path": "./dataset/2nd_data/myeong",
    "names": ["myeong_product_3_data.xlsx", "myeong_product_4_data.xlsx"]
  },
  "num_samples": 16,
  "seek_days": 43
},
```

- predict: 모델 별 훈련/예측 구분
  - multi\_encoder\_train - 수확 예측 모델 훈련, multi\_encoder\_infer - 수확 예측
- data: 모델의 입력으로 사용될 데이터
  - path: 훈련(train\_data)/정답(label\_data) 데이터 경로
  - names: 훈련(train\_data)/정답(label\_data) 데이터 파일 이름
  - num\_samples: 농장의 샘플 수
  - seek\_days: 기준일부터 과거 몇일까지 볼지 설정하는 값. 환경 데이터에만 쓰임

# 수확 예측 모델

- seek days 예시
  - seek\_days = 42

2020.1월

일	월	화	수	목	금	토
Memo			1 신정	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25 설날
26	27 대체 휴일	28	29	30	31	

2020.2월

일	월	화	수	목	금	토
Memo						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29

환경 데이터

생육 데이터의 날짜



# 수확 예측 모델

- json 파일 규격

```
"model":{
  "name":"lstm_enc_dec",
  "pretrained_path": "/home/ikhee/project/EdgeAnalytics/smartfarm-1/experiments/lstm_enc_dec/ckpt",
  "config": {
    "input_shapes": [],
    "output_shape": [],
    "explain": true,
    "env_only": false
  }
},
```

- model: 모델 생성에 필요한 입력 파라미터

- name: 모델의 이름, 모델 생성시 필요
- pretrained\_path: 이미 훈련이 끝난 모델이 저장된 경로 (예측에 사용됨)
- input/output\_shape: 비어 있는 리스트로 두면 됨. (프로그램 내부에서 자동으로 설정함)
- explain: true - 예측 결과에 대한 해석을 그래프로 제공, false - 예측 결과에 대한 해석을 제공하지 않음
- env\_only: true - 훈련/예측에 환경 데이터만 사용, false - 훈련/예측에 환경, 생육, 개화수, 착과수 데이터를 사용

# 수확 예측 모델

- json 파일 규격

```
"train":{  
  "epochs": 500,  
  "batch": 1,  
  "metric": "mse"  
},
```

- train: 모델 훈련 관련 설정
  - epochs: 훈련 반복 횟수
  - batch: 입력 한번에 사용될 데이터 수
  - metric: 정답 값과 예측 값의 오차를 계산할 알고리즘

# 수확 예측 모델

- json 파일 규격

```
"optimizer":{
  "algorithm": "sgd",
  "config": {
    "learning_rate": 0.0001,
    "momentum": 0.9,
    "nesterov": true
  }
},
```

- optimizer: 모델 훈련에 필요한 입력 파라미터

- algorithm: 최적화 알고리즘 선택
- config: 최적화 알고리즘의 입력 파라미터
  - learning\_rate: 최적화 알고리즘에 사용될 학습율
  - momentum: 최적화 알고리즘에 사용될 momentum 값 (일반적으로 0.9가 사용됨)
  - nesterov: true - nesterov accelerated gradient 사용, false - nesterov accelerated gradient 사용 안함

# 수확 예측 모델

- json 파일 규격

```
"util":{
  "save_path": "experiments/lstm_enc_dec",
  "env_heatmap": {
    "avail": true,
    "name": "env_heatmap.png",
    "x_labels": ["실내온도", "실내습도", "이슬점(Td)", "증산(HD)", "대기압", "PWS(?)", "수분량",
      "엔탈피", "비부피", "밀도", "PW(?)수증기", "절대습도", "절대AH(?)", "실내온도표준편차",
      "실내습도표준편차", "이슬점(Td)표준편차", "증산(HD)표준편차", "대기압표준편차", "PWS(?)표준편차",
      "수분량표준편차", "엔탈피표준편차", "비부피표준편차", "밀도표준편차", "PW(?)수증기표준편차", "절대습도표준편차",
      "절대AH(?)표준편차", "최소실내온도", "최소실내습도", "최소이슬점(Td)", "최소증산(HD)", "최소대기압",
      "최소PWS(?)", "최소수분량", "최소엔탈피", "최소비부피", "최소밀도", "최소PW(?)수증기", "최소절대습도",
      "최소절대AH(?)", "최대실내온도", "최대실내습도", "최대이슬점(Td)", "최대증산(HD)", "최대대기압",
      "최대PWS(?)", "최대수분량", "최대엔탈피", "최대비부피", "최대밀도", "최대PW(?)수증기", "최대절대습도",
      "최대절대AH(?)", "오전적정증산(HD)누적시간", "일출일몰적정증산(HD)누적시간", "하루적정증산(HD)누적시간",
      "오후심각증산(HD)누적시간", "오전심각증산(HD)누적시간", "12도이하온도누적시간", "30도이상온도누적시간",
      "적정온도누적시간", "주간평균온도", "오후부터일몰까지평균온도", "적정습도누적시간",
      "일출전후한시간평균온도", "일몰전후한시간평균온도", "적정온도변화폭(하위)", "적정온도변화폭(상위)", "야간평균온도",
      "일몰일출적합증산(HD)누적시간", "주야간온도차이", "주야간온도차8도이상인날수"]
  },
  "growth_heatmap": {
    "avail": true,
    "name": "growth_heatmap.png",
    "x_labels": ["샘플번호", "주간생육길이(cm)", "초장(cm)", "줄기굵기(mm)", "잎길이(cm)", "잎폭(cm)",
      "잎수(개)", "개화화방위치(cm)", "꽃과 줄기거리(cm)", "화방간거리(cm)", "화방경경(mm)", "엽면적지수",
      "주간생육길이_생육상태", "줄기굵기_생육상태", "잎길이_생육상태", "잎폭_생육상태", "잎수_생육상태",
      "엽면적지수_생육상태", "개화화방위치_생육상태", "꽃과줄기거리_생육상태", "생육상태점수", "생장구분"]
  }
},
```

# 수확 예측 모델

- json 파일 규격

- save\_path: 훈련이 완료된 모델, 예측 결과 csv, 그래프 등을 저장할 경로
- env\_heatmap: 환경 데이터에 대한 attention 알고리즘 기반 설명 그래프에 관한 설정
  - avail: true - heatmap 생성, false - heatmap 생성 안함
  - name: heatmap 저장 파일 명
  - x\_labels: heatmap의 x축에 넣을 feature 명
- growth\_heatmap: 생육 데이터에 대한 attention 알고리즘 기반 설명 그래프에 관한 설정
  - avail: true - heatmap 생성, false - heatmap 생성 안함
  - name: heatmap 저장 파일 명
  - x\_labels: heatmap의 x축에 넣을 feature 명

# 수확 예측 모델

- json 파일 규격

```
"device":{  
  "name": "/device:GPU:0"  
}
```

- device: 컴퓨팅 장치 설정

- name: 모델을 실행할 장치의 이름
- e.g.
  - “/device:GPU:0” - 0번 GPU에서 모델 실행
  - “/device:CPU:0” - 0번 CPU에서 모델 실행

- ❖ 아래의 코드로 tensorflow에서 실행 가능한 장치 목록을 얻을 수 있음

- from tensorflow.python.client import device\_lib
- print(device\_lib.list\_local\_devices())

# API 설명

- preprocessing.py

함수명	입력	출력
HD_cumulative_time	Pandas DataFrame	오전적정증산 누적 시간, 일출일몰적정증산 누적 시간, 하루 적정 증산 누적 시간, 오후 심각 증산 누적 시간, 오전 심각 증산 누적 시간
temperature_cumulative_time	Pandas DataFrame	12도 이하 온도 누적 시간, 30도 이상 온도 누적 시간, 적정 온도 누적 시간
temperature_average	Pandas DataFrame	주간 평균 온도, 오후부터 일몰까지 평균 온도
suitable_info	Pandas DataFrame	적정 습도 누적 시간, 일출 전후 한시간 평균 온도, 일몰 전후 한시간 평균 온도, 적정 온도 변화폭(하위), 적정 온도 변화폭(상위)
nighttime_info	Pandas DataFrame	야간 평균 온도, 야간 적합 HD 누적 시간
env_data_preprocessing	json arguments	원본 환경 데이터에 대한 평균, 표준 편차, 최소, 최대, 주야간 온도 차이, 주야간온도차8도이상인날수 추가 환경 전처리 데이터와 병합
growth_data_preprocessing	json arguments	생장 상태, 생장 상태 점수, 생장 구분 원본 생육 데이터와 병합

# API 설명

- runner.py

(클래스) 함수명	입력	기능
(Runner) train	-	생육/수확 예측 모델 훈련 함수 호출
(Runner) infer	-	생육/수확 예측 함수 호출
(Runner) run	-	train/infer 함수 호출
rf_train	json arguments	생육 예측 모델 훈련
rf_infer	json arguments	생육 예측
multi_encoder_train	json arguments	수확 예측 모델 훈련
multi_encoder_infer	json arguments	수확 예측