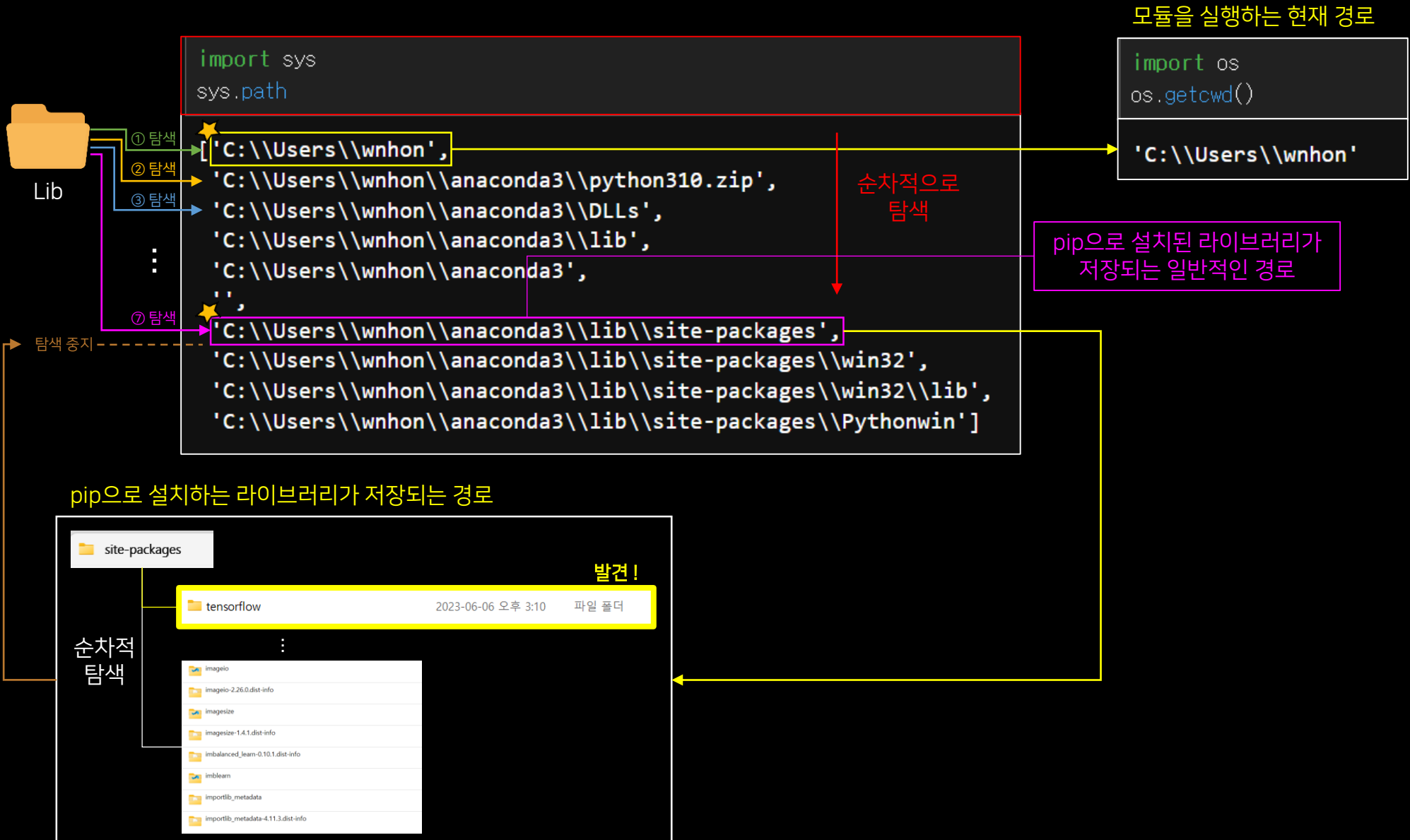


라이브러리 import 작동 원리



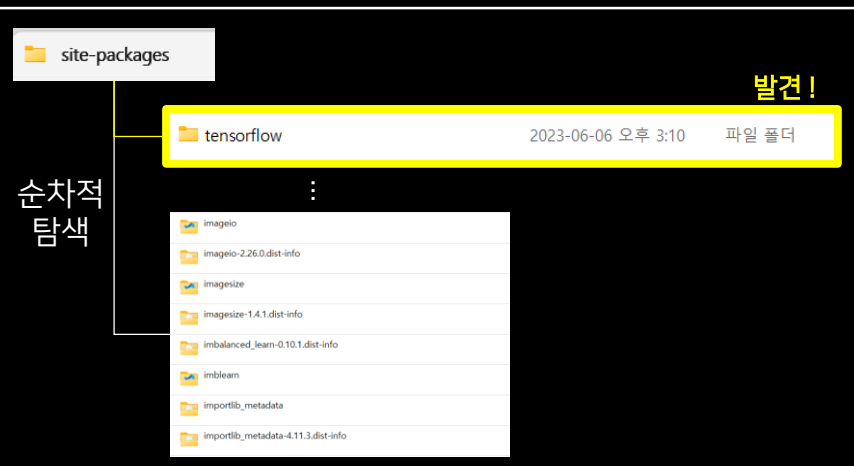
모듈을 실행하는 현재 경로

```
import os
os.getcwd()
```

'C:\\Users\\wnhon'

pip으로 설치된 라이브러리가
저장되는 일반적인 경로

pip으로 설치하는 라이브러리가 저장되는 경로



라이브러리 import 경로를 추가하는 법

방법 ① : 애플리케이션 내부에서 추가해주는 법

```
import sys
sys.path
```

```
['C:\\Users\\wnhon',
 'C:\\Users\\wnhon\\anaconda3\\python310.zip',
 'C:\\Users\\wnhon\\anaconda3\\DLLs',
 'C:\\Users\\wnhon\\anaconda3\\lib',
 'C:\\Users\\wnhon\\anaconda3',
 '',
 'C:\\Users\\wnhon\\anaconda3\\lib\\site-packages',
 'C:\\Users\\wnhon\\anaconda3\\lib\\site-packages\\win32',
 'C:\\Users\\wnhon\\anaconda3\\lib\\site-packages\\win32\\lib',
 'C:\\Users\\wnhon\\anaconda3\\lib\\site-packages\\Pythonwin']
```

```
type(sys.path)
```

```
list
```


리스트는 삽입, 삭제, 탐색이 가능한 자료구조

리스트의 첫 번째 인덱스에 lib이 포함되어 있는 경로를 insert 해준다

```
sys.path.insert(0, "<경로>")
```

다른 라이브러리를 찾으면 탐색을 멈추므로, 반드시 첫 인덱스에 삽입

pip(python package manager)로 설치되는 폴더

 **tensorflow/tensorflow** 프로젝트

An Open Source Machine Learning Framework for Everyone

[python](#) [machine-learning](#) [deep-neural-networks](#) [deep-learning](#) [neural-network](#)

● C++ · ☆ 175k · Updated 12 minutes ago

pip으로 설치된 라이브러리는
github에 프로젝트 전체가
아닌, 특정 폴더 임을 이해해야 한다

tensorflow-gardener [XLA:CollectiveMatmul] Convert emit_windowed_d... 5594a84 12 minutes ago 148,874 commits		
📁 .github	[Linaro:ARM_CI] Allow all jobs in matrix to complete	2 weeks ago
📁 tensorflow	[XLA:CollectiveMatmul] Convert emit_windowed_dot_general into a real ...	12 minutes ago
📁 third_party	Integrate LLVM at llvm/llvm-project@7d9677a9bd4f	23 minutes ago
📁 tools	Merge pull request #25673 from Ryan-Qiyu-Jiang:env_capture_script_m...	4 years ago
📄 .bazelrc	Updates Bazel version from 5.3.0 to 6.1.0	2 weeks ago
📄 .bazelversion	Updates Bazel version from 5.3.0 to 6.1.0	2 weeks ago
📄 .clang-format	[clang-format] Init @ root	2 years ago
📄 .gitignore	Ignore CoreML BUILD files which are generated by the configure script	3 years ago
📄 .pylintrc	Add soft-link to pylintrc to project root	4 years ago
📄 .zenodo.json	Add .zenodo.json for clean automated DOI numbers.	2 years ago
📄 AUTHORS	Add Arm Ltd to AUTHORS	last year
📄 BUILD	[NFC, internal change] Polish copybara workflow file.	3 years ago
📄 CITATION.cff	Add CITATION.cff	2 years ago
📄 CODEOWNERS	Remove myself from CODEOWNERS as I will no longer be in TF.	last year
📄 CODE_OF_CONDUCT.md	Update CODE_OF_CONDUCT and fix broken sync	3 years ago
📄 CONTRIBUTING.md	Fix formatting of the CONTRIBUTING.md section on PRs.	2 months ago
📄 ISSUES.md	Internal change	2 years ago
📄 ISSUE_TEMPLATE.md	Merge pull request #40033 from AbdulBaseerMohammedKhan:master	3 years ago
📄 LICENSE	Remove duplicate LICENSE information for third_party dependencies an...	2 years ago
📄 README.md	Fix badge style on README	last month
📄 RELEASE.md	Add 'auto' steps_per_execution tuner parameter for Keras compile.	11 hours ago
📄 SECURITY.md	Announce sunset of TFSAs, moving to rely solely on CVEs.	2 months ago
📄 WORKSPACE	Prevent buildifier warning in WORKSPACE file.	3 years ago
📄 arm_compiler.BUILD	Update the compiler_pieces of the RPi ARM compiler.	2 years ago

라이브러리 import 경로를 추가하는 법

방법 ② : OS의 환경변수에 추가해주는 법

Python도 OS 위에서 구동되는 프로그램

사용자 홈 디렉터리의 환경변수 파일에 변수를 추가하여
import 경로를 추가도 가능하다

홈디렉터리 환경변수 파일 [OS : Amazon Linux release 2 (Karoo)]

- .bashrc
- .bash_profile

```
export PYTHONPATH="<추가할 경로>:${PYTHONPATH}"
```

방법 ③ : python 실행시 PYTHONPATH 옵션 넣어주기

설명

```
python -h
```

```
PYTHONPATH : ';'-separated list of directories prefixed to the  
            default module search path. The result is sys.path.
```

실행

```
PYTHONPATH=/path/to/module python my_script.py
```

잠깐, 지금 방법으로 하시면 안 됩니다.

`sys.path.insert()` 로는 배포하는 코드는 실제 배포/이관하는 코드에 넣으시면 안됩니다.

왜 일까요?

당신이 추가한 경로는 당신의 컴퓨터 환경에만 맞춰져 있기 때문입니다.

그 코드가 다른 사람의 컴퓨터에 들어가서도 돌아갈 것 같나요?

당신이 이관해주는 그 코드가 다른 사람의 수정을 통해야만 돌아가야 하나요?

프로그래밍에 하드 코딩을 이용하는 것은 좋지 않은 방법입니다. 명시적인 부분을 써야한다면, 환경변수를 설정하거나 config를 설정할 수 있는 파일(json, ini, cfg, .env 등)을 따로 분리하여 관리하는 게 정석입니다. 심지어는 SQL도 .sql 이나 .hql의 별도 스크립트 파일로 분리하여 사용하는 경우가 많습니다. 오픈소스 플랫폼의 내부 구조와 기술 블로그, 컨퍼런스, 다른 개발자들의 github을 많이 보시면서 어떻게 구성하고 코드를 짜는 지 많이 보면 좋습니다.

Airflow, Spark, Hadoop(HDFS, MapReduce, YARN), Hive 어떤 플랫폼 다 마찬가지입니다.

개발자 이외에 플랫폼 사용자가 직접적인 코드 수정을 제한하게 하고, 추가 해야한다면 상속이나 플러그인 코드를 구현하게 하여 유저 커스터마이징을 구현하게 하지 직접적인 코드 수정을 하지는 않습니다.

세계 최고의 개발자들은 어떻게 import 할까?

세계 최고의 파이썬 개발자들은 import 를 어떻게 쓰고 있을까요?

pandas / pandas / core / series.py

```
import numpy as np

from pandas._config import (
    get_option,
    using_copy_on_write,
)

from pandas._libs import (
    lib,
    properties,
    reshape,
)
```



spark / python / pyspark / sql / dataframe.py

```
from pyspark import copy_func, _NoValue
from pyspark._globals import _NoValueType
from pyspark.context import SparkContext
from pyspark.errors import PySparkTypeError, PySparkValueError
from pyspark.rdd import (
    RDD,
    _load_from_socket,
    _local_iterator_from_socket,
)
```



airflow / airflow / models / baseoperator.py

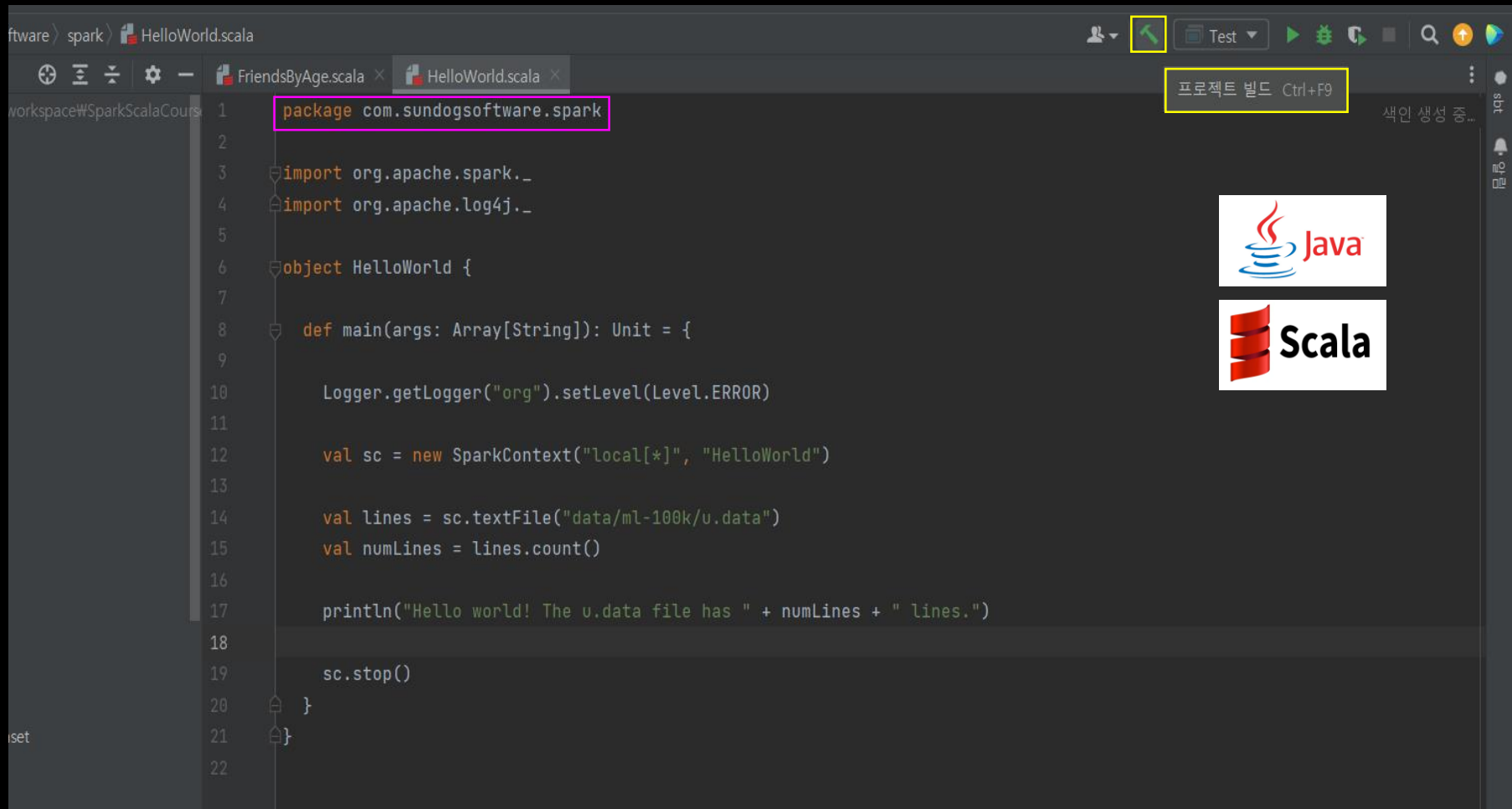
```
from airflow.configuration import conf
from airflow.exceptions import AirflowException, DagInvalidTriggerRule, RemovedInAirflow3Warning, TaskDeferred
from airflow.lineage import apply_lineage, prepare_lineage
```



패키지가 시작되는 특정 폴더(pandas, pyspark, airflow)에서 절대 경로로 접근 합니다.
그런데 여러 분들의 코드에서 저렇게 써도 동작이 안 됩니다.
여러분의 코드는 빌드라는 과정을 거치지 않았으니까요.

다른 언어의 빌드에 대해서

다른 언어의 IDE를 써보시면 **Build** 라는 과정을 거칩니다. 또, **패키지의 최상위 경로를 지정**도 해줍니다. 물론 python에도 pycharm이라는 IDE가 자동으로 빌드 해주는 기능이 있는 것 같습니다. 하지만, 보통 jupyter를 이용해서 분석 작업을 하기 때문에 스스로 build를 해봅시다.



The screenshot shows an IDE window with a Scala file named `HelloWorld.scala`. The code defines a package `com.sundogsoftware.spark` and an object `HelloWorld` with a `main` method. The IDE interface includes a toolbar with a 'Test' button and a '프로젝트 빌드 Ctrl+F9' button. On the right side, there are logos for Java and Scala.

```
1 package com.sundogsoftware.spark
2
3 import org.apache.spark._
4 import org.apache.log4j._
5
6 object HelloWorld {
7
8   def main(args: Array[String]): Unit = {
9
10     Logger.getLogger("org").setLevel(Level.ERROR)
11
12     val sc = new SparkContext("local[*]", "HelloWorld")
13
14     val lines = sc.textFile("data/ml-100k/u.data")
15     val numLines = lines.count()
16
17     println("Hello world! The u.data file has " + numLines + " lines.")
18
19     sc.stop()
20   }
21 }
```

Java

Scala

파이썬에서 빌드하기

파이썬에서 빌드를 하려면 setup.py 파일이 필요합니다.

setup.py 파일은 패키지 내부(source code)부분이 아니라 외부인 프로젝트 폴더 안에 있어요.

[pandas](#) / [pandas](#) / [core](#) / [series.py](#) <https://github.com/pandas-dev/pandas>

■ pandas	REF: Separate groupby, rolling, and window agg/apply list/dict-like (#...	20 hours ago
■ scripts	CoW: Add warning for chained assignment with fillna (#53779)	4 days ago
■ typings	STYLE start enabling TCH (#51687)	5 months ago
■ web	DOC: Updated pandas extension list (#53960)	3 days ago
□ .devcontainer.json	Streamline docker usage (#49981)	8 months ago
□ .gitattributes	Ci: use versioneer, for PEP440 version strings #9518	8 years ago
□ .gitignore	Improved documentation for extension development (#53380)	2 months ago
□ .gitpod.yml	Install pre-commit automatically in gitpod (#52856)	3 months ago
□ .libcst.codemod.yaml	TYP: Autotyping (#48191)	10 months ago
□ .pre-commit-config.yaml	CLN: Consolidate Dependencies (#53863)	3 days ago
□ AUTHORS.md	CI add end-of-file-fixer (#36826)	3 years ago
□ CITATION.cff	GH: Add CITATION.cff (#47710)	last year
□ Dockerfile	Ci: Use regular solver for docker build (#50341)	7 months ago
□ LICENSE	Update copyright year (#51065)	6 months ago
□ MANIFEST.in	Ci: Build wheel from sdist (#53087)	last month
□ README.md	DOC: Added Nav Links to Discussion and Development section on READ...	last month
□ codecov.yml	CI/DOC: replace master -> main branch (#45336)	last year
□ environment.yml	CLN: Consolidate Dependencies (#53863)	3 days ago
□ generate_pxi.py	BLD: Setup meson builds (#49115)	2 months ago
□ generate_version.py	BLD: Setup meson builds (#49115)	2 months ago
□ meson.build	BLD: Setup meson builds (#49115)	2 months ago
□ pyproject.toml	CLN: Consolidate Dependencies (#53863)	3 days ago
□ pyright_reportGeneralTypeIssues.json	TYP: type all arguments with str default values (#48508)	10 months ago
□ requirements-dev.txt	CLN: Consolidate Dependencies (#53863)	3 days ago
□ setup.py	Refactor Extension Modules (#53346)	2 months ago

패키지

```
from setuptools import (
    Command,
    Extension,
    setup,
)

if __name__ == "__main__":
    # Freeze to support parallel compilation when using
    # spawn instead of fork
    multiprocessing.freeze_support()
    setup(
        version=versioneer.get_version(),
        ext_modules=maybe_cythonize(extensions,
        compiler_directives=directives),
        cmdclass=cmdclass,
    )
```

setup.py 내부 구현은 프로젝트마다 상이합니다.
다만, 프로그램 setup 메서드를 호출하는 부분의 파라미터에
name, version, install_requires, python_requires 등을 작성합니다.

setup.py

다른 프로젝트 구조 염탐하기

그럼 이 앞 부분은 무엇 일까? 이 앞에 어떤 공통적인 파일들이 있는 지 직접 비교해서 보세요.
제가 알려주는 건 의미 없습니다. 직접 파악하시고 비교하시고 공통점을 찾아보세요 :)

`pandas` / `pandas / core / series.py` <https://github.com/pandas-dev/pandas>

`spark / python` / `pyspark / sql / dataframe.py` <https://github.com/apache/spark/tree/master/python>

`airflow` / `airflow / models / baseoperator.py` <https://github.com/apache/airflow>

그래야 앞으로 프로젝트 구조를 보는 눈이 생기고 어떻게 구성해야 하는 지 포맷이 그려집니다.
당신의 코드를 공유해야 하는 상황이라면, 기본적인 설명과 포맷도 없는 결과물의 코드를 과연
어떤 개발자가 이해할 수 있을까요?

이건 암묵적인 약속입니다. 이걸 개발자 사이의 의사소통 할 수 있는 도구입니다.
아무것도 모르겠다면 일단 README.md 부터 써보기 시작하면서 점차 범위를 늘려보세요.

스마트시티 프로젝트에서 구현한 setup.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
```

셔뱅(!)
인코딩 설정

```
import sys
from setuptools import setup, find_packages

try:
    exec(open("sooss/_version.py").read())
except IOError:
    print(
        "You must be in SOSS's python dir.",
        file=sys.stderr,
    )
    sys.exit(-1)
```

VERSION = __version__ 프로그램 버전 (1.0.0.dev0)

with open("README.rst") as f:
 long_description = f.read()
README 파일을 읽어서
long_description 설정

```
if __name__ == "__main__":
```

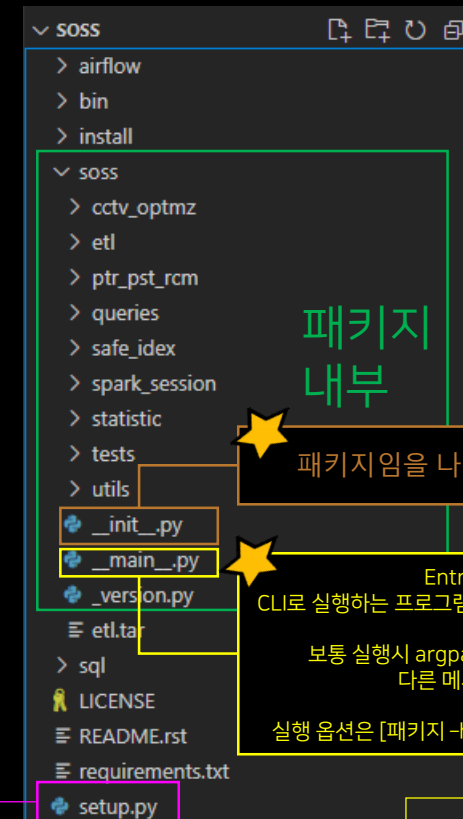
```
    setup(
        name="sooss",
        version=VERSION,
        description='Sooss Analysis Python Program',
        long_description=long_description,
        author="B2EN",
        author_email="b2en@b2en.com",
        license="Your License (BSD, MIT, Apache ..)",
        packages=find_packages(),
        install_requires=[
            "numpy",
            "pandas==1.4.1",
            "geopandas",
            "tdqm",
            "joblib",
            "scikit-learn",
            "xgboost",
            "lightgbm",
            "haversine",
            "pyspark==3.3.2",
            "apache-airflow[postgres]==2.6.2"
        ],
        python_requires=">=3.6"
```

setup.py 로 설치할 패키지

```
python setup.py install
python setup.py develop
```

을 사용하면 설치 가능
(setup 버전에 따라 설치 커맨드가
달라질 수 있습니다)

필요 파이썬 버전



패키지
내부

패키지임을 나타내는 __init__.py

Entry Point 입니다.
CLI로 실행하는 프로그램이 아니라면 없는 경우도 있습니다.

보통 실행시 argparse에 따라 실행 옵션을 받아
다른 메서드를 호출합니다.

실행 옵션은 [패키지 -help]로 보통 찾아볼 수 있습니다.

```
[ec2-user@safe-service etl]$ airflow --help
usage: airflow [-h] GROUP_OR_COMMAND ...

positional arguments:
  GROUP_OR_COMMAND

Groups:
  celery      Celery components
  config      View configuration
  connections Manage connections
  dags        Manage DAGs
  db          Database operations
  jobs        Manage Jobs
  kubernetes  Tools to help run the KubernetesExecutor
  pools       Manage pools
  providers   Display providers
  roles       Manage roles
  tasks       Manage tasks
  users       Manage users
  variables   Manage variables
```

빌드가 됐다면? (1)

`pip list | grep <패키지명>` 으로 내 패키지가 설치됐음을 볼 수 있습니다.

```
[icitydatahub@icsossai ~]$ pip list | grep soss  
soss 1.0.0.dev0 /home/icitydatahub/soss
```

`pip show <패키지명>` 을 치면 내 패키지에 대한 설명도 나옵니다.

```
[icitydatahub@icsossai ~]$ pip show soss  
Name: soss  
Version: 1.0.0.dev0  
Summary: Soss Analysis Python Program  
Home-page:  
Author: B2EN  
Author-email: b2en@b2en.com  
License:  
Location: /home/icitydatahub/soss  
Editable project location: /home/icitydatahub/soss  
Requires: apache-airflow, geopandas, haversine, joblib, lightgbm, numpy, pandas, pyspark, scikit-learn, tqdm, xgboost  
Required-by:
```

빌드가 됐다면? (2)

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

"""
안 전 지 수 운 영 모 들
"""

from __future__ import annotations

from datetime import datetime, timedelta
import logging
import os
import time
from typing import (
    Tuple,
    List,
    Union,
)

import joblib
import numpy as np
import pandas as pd
from pyspark.sql.utils import ParseException
from py4j.protocol import Py4JJavaError
from sqlalchemy import create_engine
from sqlalchemy.orm import Session

from soss.spark_session.db_connector import SparkClass
from soss.queries.safe_idx_query import SafeIdxQuery
from soss.utils.config_parser import *
```

그리고 무엇보다 내 패키지의 루트부터
절대 경로로 접근 가능합니다.
(/home/user/soss/soss 로 접근 안해도 됩니다.)

마무리

ppt 안에 담기에는 내용이 너무 많습니다. 설명도 많습니다.

그래서 다소 많은 설명이 누락되었습니다만, 생각을 넓히는 자료가 되었으면 합니다.

가장 추천하는 건, 직접 도큐먼트를 읽어가면서 만들어보는 것입니다.

항상 그렇지만, 아는 것과 해보는 것은 차이가 있습니다.

기본 개념을 많이 알더라도 실제 해보면서 trouble-shooting을 하면서 배우는 것은 다른 일입니다.

많이들 도전하시고, 실패도 겪어보고, 또 해내셨으면 좋겠습니다.

다음에 또 문서를 작성할 일이 있다면 test code 작성(pytest)에 대해 써볼 까 합니다.

그럼 다음에 뵙겠습니다.