

Hongik Univ.

Tesla Project “첫걸음”

@SEUNGWOO LEE

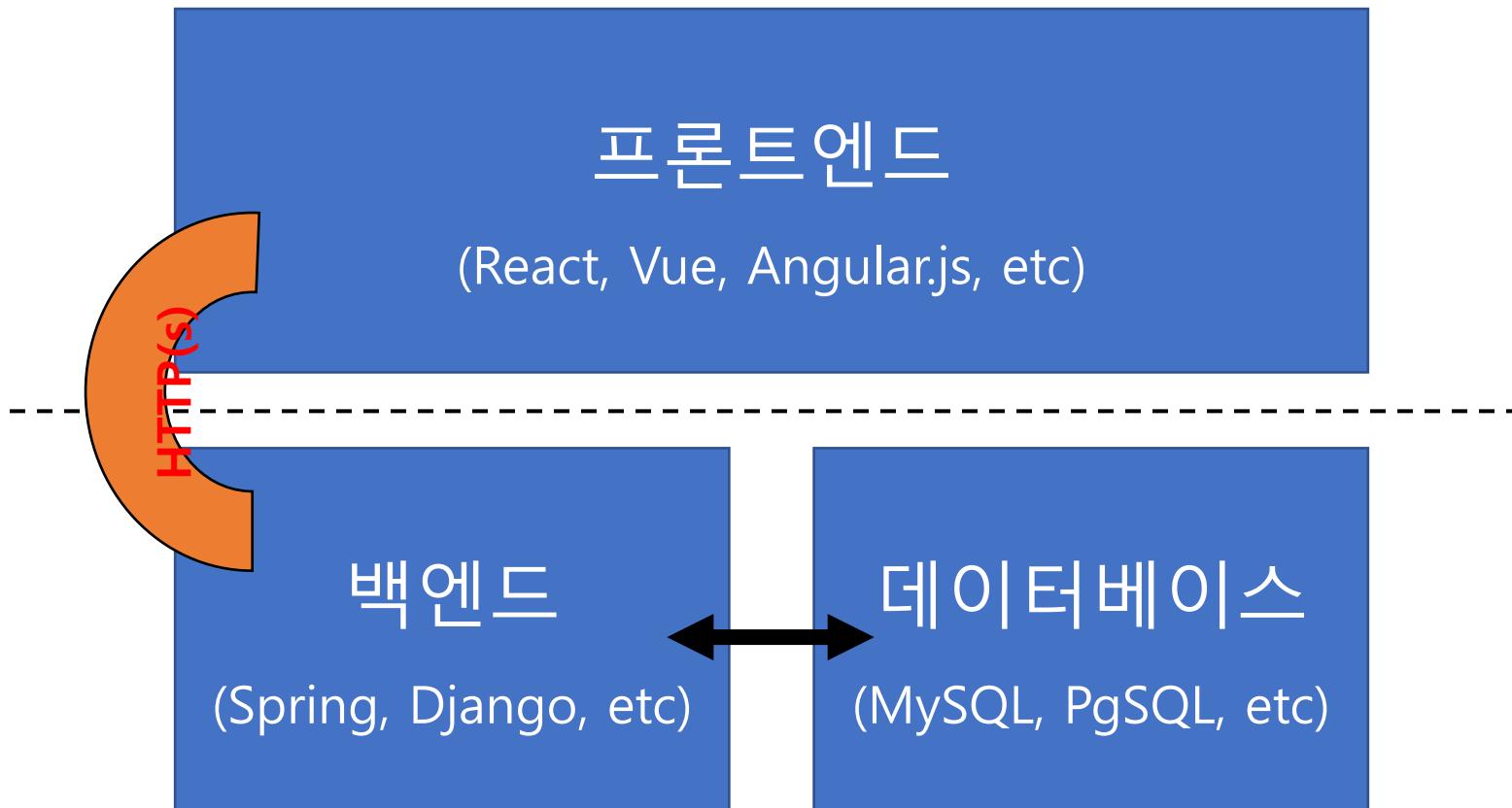
어떻게 작업할지 토의

=Brainstorming

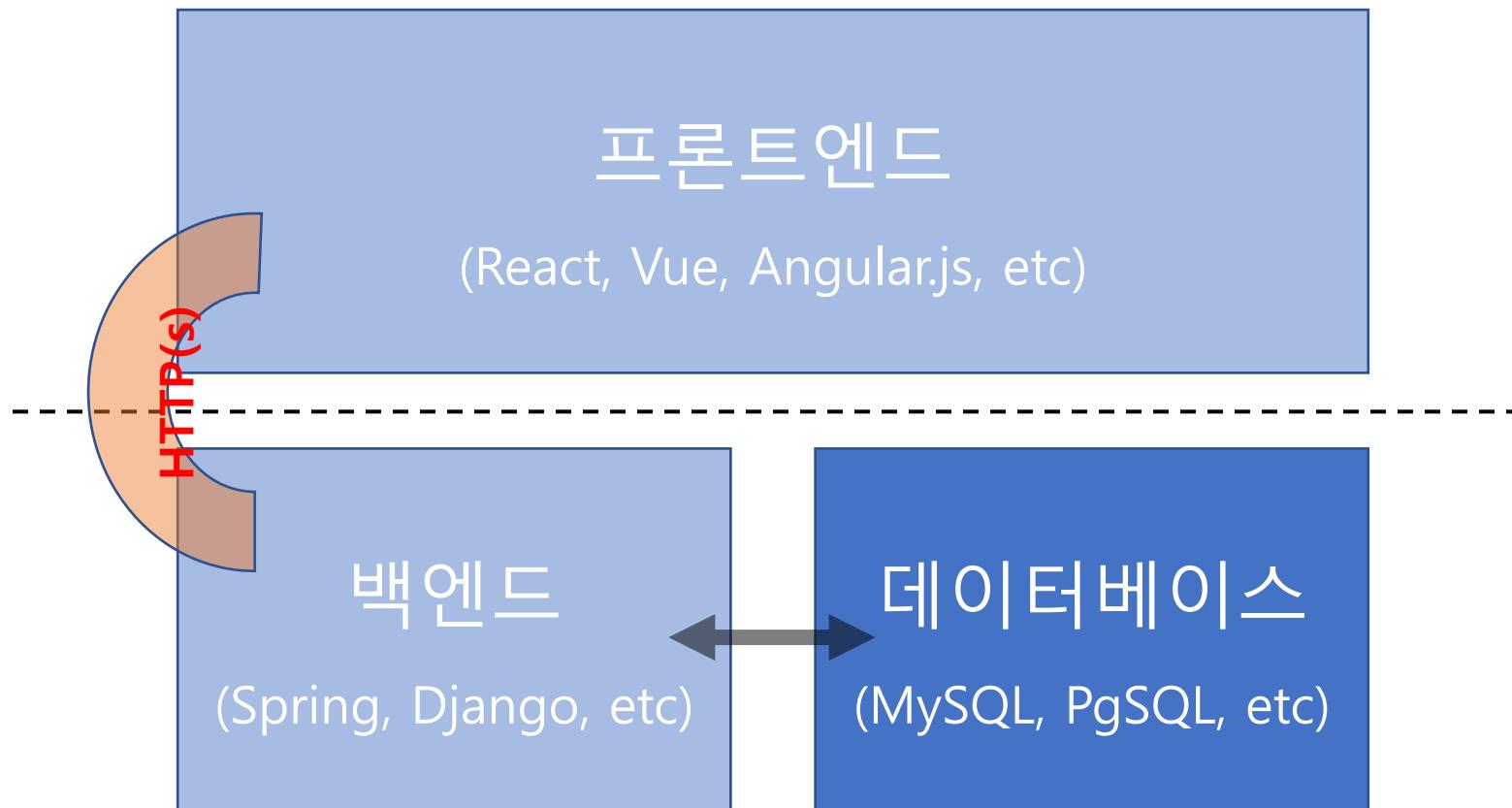
“목적”

스크랩핑 + a

Typical Web(or App) Services



Typical Web(or App) Services



데이터베이스

1. 뭘 담을 것인가

3. 어떻게
자동화할 것인가?

2. 어떻게 담을 것인가

데이터베이스
(MySQL, PgSQL, etc)

데이터베이스

1. 뭘 담을 것인가

3. 어떻게
자동화할 것인가?

2. 어떻게 담을 것인가

데이터베이스
(MySQL, PgSQL, etc)

데이터베이스

1. 뭘 담을 것인가



- 테슬라 자동차 관련된 Metrics
 - 1. 뭘 어떻게 주행하였는가?
 - 2. 사용자의 운전습관
 - 3. 운전 당시의 날씨
 - 4. etc

데이터베이스

(MySQL, PgSQL, etc)

사례: awesome-tesla

The screenshot shows a GitHub repository page for `rjohnson3/awesome-tesla`. The repository has 15 commits, 1 branch, 0 packages, 0 releases, and 1 contributor. The README.md file contains the following content:

```
Awesome Tesla

A curated list of awesome resources for Tesla vehicles.

Want to contribute? Fork this repository, add your resources and send us a PR.

Table of Contents

• Mobile Apps
  • Official
  • Controls
  • Statistics
• In-Car Browser Apps
• Web Apps
```

<https://github.com/rjohnson3/awesome-tesla>

사례: awesome-tesla



Web Statistics

- [TeslaFi](#) - Incredible wealth of stats for your Tesla.
- [Teslastics](#) - Teslastics is a platform to monitorize your Tesla.

Awesome Tesla

A curated list of awesome resources for Tesla vehicles.

Want to contribute? Fork this repository, add your resources and send us a PR.

Table of Contents

- [Mobile Apps](#)
 - [Official](#)
 - [Controls](#)
 - [Statistics](#)
- [In-Car Browser Apps](#)
- [Web Apps](#)

<https://github.com/rjohnson3/awesome-tesla>

데이터베이스

1. 뭘 담을 것인가

3. 어떻게
자동화할 것인가?

2. 어떻게 담을 것인가

데이터베이스
(MySQL, PgSQL, etc)

Tesla API (Un-Official)

The screenshot shows a web browser displaying the [Tesla API \(Un-Official\)](https://www.teslaapi.io) website. The page has a header with navigation icons and a search bar. The main content area features a sidebar with a tree-view navigation menu and several sections of text. A footer at the bottom right contains a link to the site's URL.

Header: https://www.teslaapi.io

Page Title: Tesla API

Header Links: Model S API TMC Thread, Tesla Referral Code

Search Bar: Search...

Sidebar (Left):

- Tesla API** (highlighted)
- AUTHENTICATION
 - OAuth
 - User
- PRODUCTS
 - List
- VEHICLES
 - List
 - State And Settings
 - Commands
- POWERWALLS
 - State And Settings
 - Commands
- ENERGY SITES
 - State And Settings
 - Commands
- PREFERENCES

Powered by GitBook

Main Content Area:

Tesla API

How is this site organized?

This site is broken into sections for different API information:

- Authentication - API commands to generate an `{access_token}` to communicate with your vehicle.
- Vehicles - API commands to communicate between your vehicle and your client.
- Codes - Localized Tesla vehicle option codes for the [Model S](#) and [Model X](#)

What Tesla products are supported?

- Tesla Model S
- Tesla Model X
- Tesla Model 3

What is teslaapi.io?

This is a community of developers who are reverse engineering Tesla's API.

What are the client_id and client_secret values?

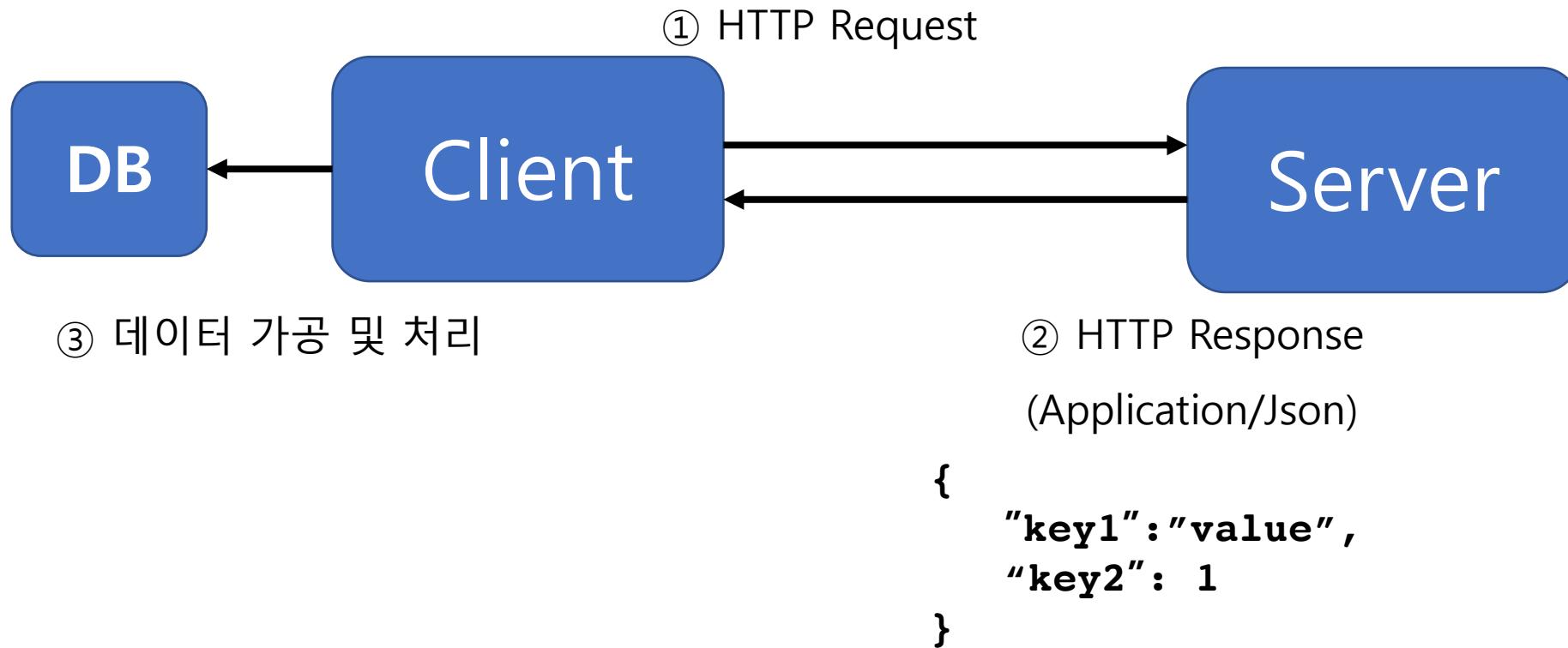
Authentication to the Tesla API is done through OAuth. These values were retrieved from somebody's

Right Sidebar:

- CONTENTS
 - How is this site organized?
 - What Tesla products are supported?
 - What is teslaapi.io?
 - What are the client_id and client_secret values?
 - Is this API official?

<https://www.teslaapi.io>

Typical API Requests



데이터 요청시 사용해보면 좋은 툴

HTTP Clients

Libraries for working with HTTP.

- [grequests](#) - requests + gevent for asynchronous HTTP requests.
- [httplib2](#) - Comprehensive HTTP client library.
- [httpx](#) - A next generation HTTP client for Python.
- [requests](#) - HTTP Requests for Humans.
- [treq](#) - Python requests like API built on top of Twisted's HTTP client.
- [urllib3](#) - A HTTP library with thread-safe connection pooling, file post support, sanity friendly.

<https://github.com/vinta/awesome-python#http-clients>

데이터 가공시 사용해보면 좋을 툴

ORM

Libraries that implement Object-Relational Mapping or data mapping techniques.

- Relational Databases
 - [Django Models](#) - The Django ORM.
 - [SQLAlchemy](#) - The Python SQL Toolkit and Object Relational Mapper.
 - [awesome-sqlalchemy](#)
 - [dataset](#) - Store Python dicts in a database - works with SQLite, MySQL, and PostgreSQL.
 - [orator](#) - The Orator ORM provides a simple yet beautiful ActiveRecord implementation.
 - [orm](#) - An async ORM.
 - [peewee](#) - A small, expressive ORM.
 - [pony](#) - ORM that provides a generator-oriented interface to SQL.
 - [pydal](#) - A pure Python Database Abstraction Layer.
- NoSQL Databases
 - [hot-redis](#) - Rich Python data types for Redis.
 - [mongoengine](#) - A Python Object-Document-Mapper for working with MongoDB.
 - [PynamoDB](#) - A Pythonic interface for [Amazon DynamoDB](#).
 - [redisco](#) - A Python Library for Simple Models and Containers Persisted in Redis.

<https://github.com/vinta/awesome-python#orm>

데이터베이스

1. 뭘 담을 것인가

3. 어떻게
자동화할 것인가?

2. 어떻게 담을 것인가

데이터베이스
(MySQL, PgSQL, etc)

데이터베이스

귀찮은 일을
어떻게 하면
손쉽게 처리할 수 있을까?

3. 어떻게
자동화할 것인가?

“Write Once, Use Many”

Celery



Celery

DOCS AND SUPPORT

INSTALL

TUTORIALS

COMMUNITY

SOURCECODE

Celery: Distributed Task Queue



Celery is an asynchronous task queue/job queue based on distributed message passing. It is focused on real-time operation, but supports scheduling as well.

The execution units, called tasks, are executed concurrently on a single or more worker servers using multiprocessing, Eventlet, or gevent. Tasks can execute asynchronously (in the background) or synchronously (wait until ready).

Celery is used in production systems to process millions of tasks a day.

Celery Projects Website Server Upgraded To Ubuntu 18.04 Bionic

on 17 Apr 2019, 8:13 p.m.

Celery 4.3 Stable Version Released

on 1 Apr 2019, 5:05 p.m.

[Read all news](#)

Something went wrong when trying to get package information from pypi

<http://www.celeryproject.org/>

An Elegant way to run periodic tasks in Python

An elegant way to run periodic tasks in python



sankalp jonna [Follow](#)
Oct 30, 2018 · 3 min read



At GreedyGame, we try to help Game Developers monetize without hampering the gameplay experience of the Gamer. We work towards creating a better ad ecosystem by bringing native ads to mobile games and creating Ads people ❤️



<https://bit.ly/2GqrCKf>

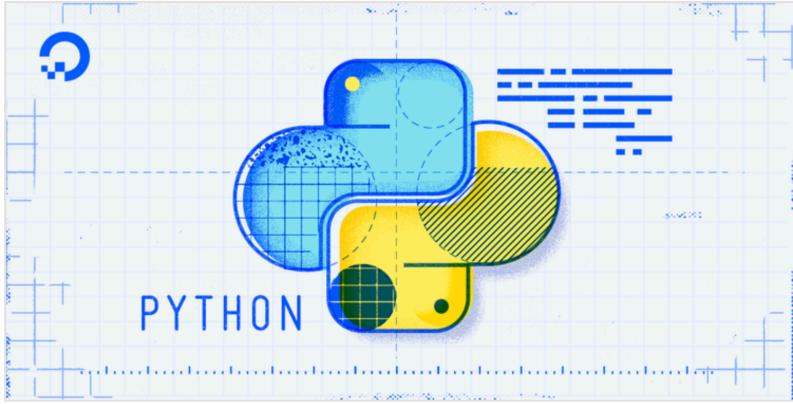
Python 모듈화

https://www.digitalocean.com/community/tutorials/how-to-write-modules-in-python-3

The screenshot shows a web browser displaying a DigitalOcean community tutorial. The title of the page is "How To Write Modules in Python 3". The page includes a sidebar with navigation links like "Community", "Tutorials", "Questions", and "Get Involved". A search bar and a "Sign Up" button are also present. The main content area features a large Python logo on a grid background, followed by the title "How To Write Modules in Python 3". Below the title, there's a snippet of code and some explanatory text. A "Related" sidebar on the right lists other tutorials.

Contents

- Writing and Importing Modules
- Accessing Modules from Another Directory
- Conclusion



How To Write Modules in Python 3

Posted February 3, 2017 | 282.7k | PYTHON | DEVELOPMENT

By Lisa Tagliaferri | [Become an author](#)

Introduction

Python **modules** are .py files that consist of Python code. Any Python file can be referenced as a module.

Some modules are available through the [Python Standard Library](#) and are therefore installed with your Python installation. Others can be installed with Python's package manager `pip`.

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics. Enter your email address and click "Sign Up".

<https://www.digitalocean.com/community/tutorials/how-to-write-modules-in-python-3>

Python UnitTesting

← → ⌂ https://docs.python.org/3/library/unittest.html

Python » English 3.8.1 Documentation » The Python Standard Library » Development Tools » Quick search Go | previous | next | modules | index

Table of Contents

- unittest — Unit testing framework
 - Basic example
 - Command-Line Interface
 - Command-line options
 - Test Discovery
 - Organizing test code
 - Re-using old test code
 - Skipping tests and expected failures
 - Distinguishing test iterations using subtests
 - Classes and functions
 - Test cases
 - Deprecated aliases
 - Grouping tests
 - Loading and running tests
 - `load_tests` Protocol
 - Class and Module Fixtures
 - `setUpClass` and `tearDownClass`
 - `setUpModule` and `tearDownModule`
 - Signal Handling

Previous topic
[doctest — Test interactive Python examples](#)

Next topic
[unittest.mock — mock object library](#)

This Page
Report a Bug
Show Source

unittest — Unit testing framework

Source code: [Lib/unittest/__init__.py](#)

(If you are already familiar with the basic concepts of testing, you might want to skip to [the list of assert methods](#).)

The `unittest` unit testing framework was originally inspired by JUnit and has a similar flavor as major unit testing frameworks in other languages. It supports test automation, sharing of setup and shutdown code for tests, aggregation of tests into collections, and independence of the tests from the reporting framework.

To achieve this, `unittest` supports some important concepts in an object-oriented way:

test fixture
A *test fixture* represents the preparation needed to perform one or more tests, and any associated cleanup actions. This may involve, for example, creating temporary or proxy databases, directories, or starting a server process.

test case
A *test case* is the individual unit of testing. It checks for a specific response to a particular set of inputs. `unittest` provides a base class, `TestCase`, which may be used to create new test cases.

test suite
A *test suite* is a collection of test cases, test suites, or both. It is used to aggregate tests that should be executed together.

test runner
A *test runner* is a component which orchestrates the execution of tests and provides the outcome to the user. The runner may use a graphical interface, a textual interface, or return a special value to indicate the results of executing the tests.

See also:

Module doctest
Another test-support module with a very different flavor.

Simple Smalltalk Testing: With Patterns
Kent Beck's original paper on testing frameworks using the pattern shared by `unittest`.

pytest
Third-party unittest framework with a lighter-weight syntax for writing tests. For example,

<https://docs.python.org/3/library/unittest.html>

See also:

Module `doctest`

Another test-support module with a very different flavor.

[Simple Smalltalk Testing: With Patterns](#)

Kent Beck's original paper on testing frameworks using the pattern shared by `unittest`.

[pytest](#)

Third-party `unittest` framework with a lighter-weight syntax for writing tests. For example,

```
assert func(10) == 42.
```

[The Python Testing Tools Taxonomy](#)

An extensive list of Python testing tools including functional testing frameworks and mock object libraries.

[Testing in Python Mailing List](#)

A special-interest-group for discussion of testing, and testing tools, in Python.

The script `Tools/unittestgui/unittestgui.py` in the Python source distribution is a GUI tool for test discovery and execution. This is intended largely for ease of use for those new to unit testing. For production environments it is recommended that tests be driven by a continuous integration system such as [Buildbot](#), [Jenkins](#) or [Hudson](#).

Tips

- 코드 주석처리 및 설명 기술
 - # 혹은 """ 활용
 - `requirements.txt` 파일에 사용한 모듈 잘 정리되었는지 확인
 - (+alpha) 앞서 언급한 `unittest` 테스팅이 잘 동작하는지, test coverage는 어떤지 체크해보기
- README.md 파일 기술
 - 누군가 코칭을 해줄수는 없음. 자신의 스타일에 알맞게 적어놓아야 함.
 - <https://github.com/matiassingers/awesome-readme> 참고
- GitLab Repository Issue / Slack / Google 잘 활용
 - 개발상의 이슈는 GitLab 레파지토리를 활용하여 정리해놓으면 편리함.
 - Slack을 통해서 개발관련 사항 및 궁금증 공유 (@ 사용)
 - 구글을 잘 활용하면 코드 짤때 모르는 부분을 손쉽게 해결할 수 있음.