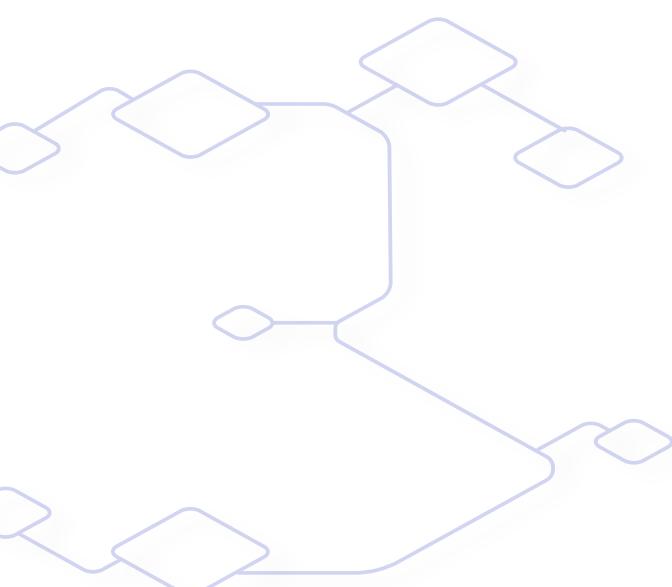
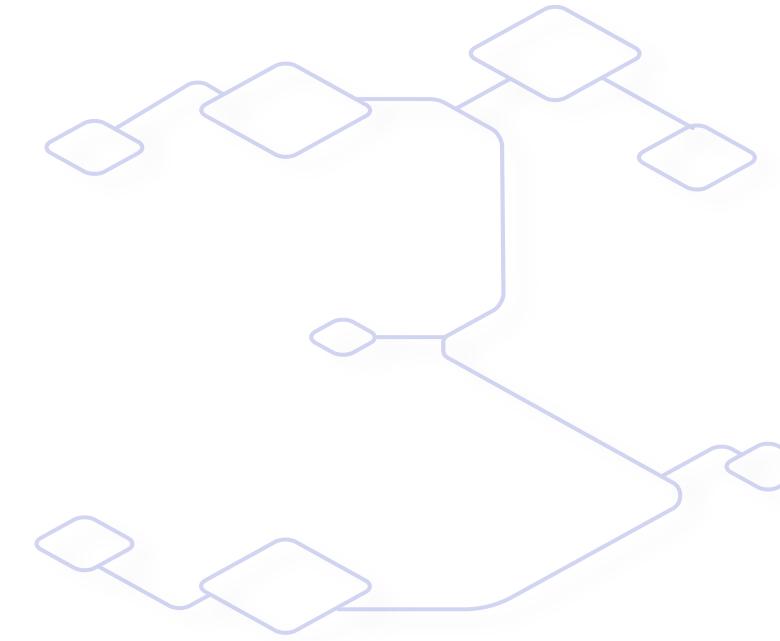


단위 테스트 도구

PYTEST

과제2 - DevOps를 구성하는 OSS



항공교통물류학부 물류전공	2019126043 박상천
소프트웨어학과	2020125068 최수빈
소프트웨어학과	2021125017 김지홍
소프트웨어학과	2021125076 김지현
소프트웨어학과	2023125032 안민서
소프트웨어학과	2023125043 이승주

오픈소스 SW 입문 - team G

DEVOPS를 구성하는 OSS

PYTEST 목차

CHAPTER
01

PYTEST 탄생 배경

CHAPTER
03

PYTEST 현재 상태

CHAPTER
05

PYTEST 핵심 내용

CHAPTER
02

PYTEST 커뮤니티

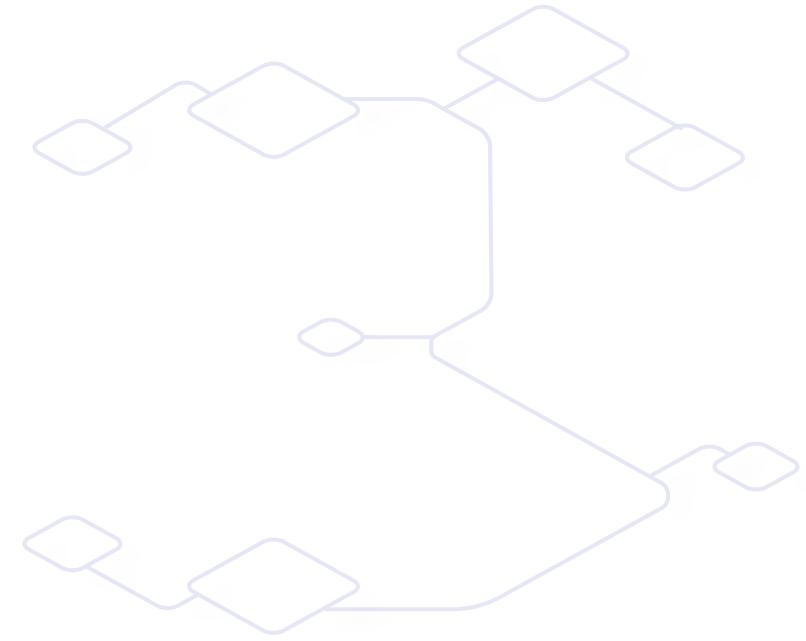
CHAPTER
04

PYTEST 개발 목적, 개발 상황

CHAPTER
06

PYTEST 향후 방향

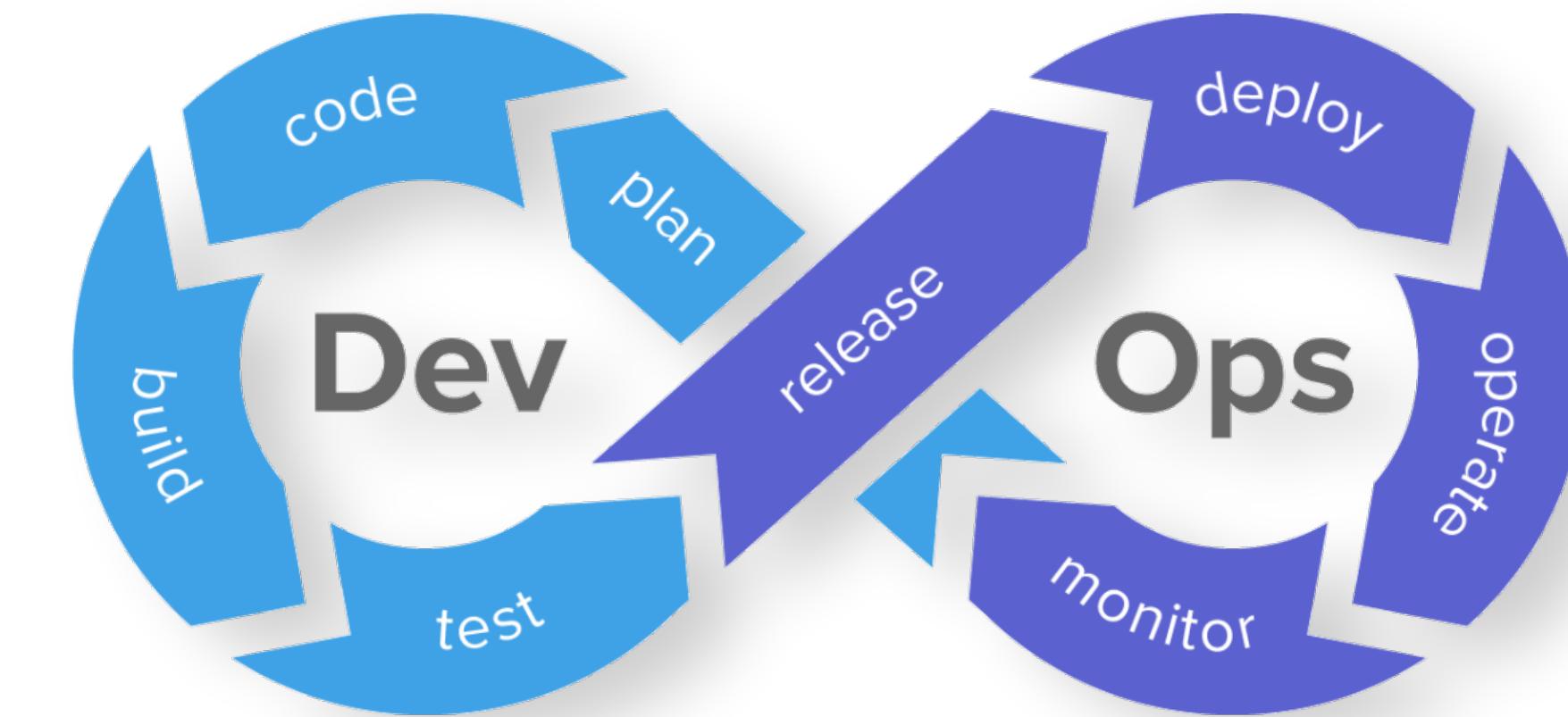
01 PYTEST 탄생 배경



- 1.1) DevOps
- 1.2) PyTest 타임라인
- 1.3) PyTest 탄생배경

1.1

DevOps



- 개발과 운영의 합성어 = Develop + Operation
- 소프트웨어 제공 프로세스를 자동화하고 개선하기 위한 방법론
 - 협업을 강화하고 신속한 배포를 가능하게 함
- PyTest는 DevOps 철학을 지원하는 도구

1.2 PYTEST 타임라인

2002~2003



PYPY, STD

- PyPy 프로젝트와 std 프로젝트가 시작
- 이는 추후 PyTest의 기본 원칙을 제시하는 "보충 표준 라이브러리"로 계획됨

2007



첫 번째 커밋

- PyTest 저장소에 첫 번째 커밋이 등록
- 파일 435개에 걸쳐 변경 이루어지고 삽입된 내용이 5만개가 넘음

보충 표준 라이브러리

2009



PYTEST

- 공식적인 첫 번째 릴리스 마침
 - 커뮤니티 기여를 받으며 빠르게 성장
- 플러그인 시스템이 도입되어 사용자들이 PyTest를 확장할 수 있게 됨

새로운 개발 참고

2020~



인기 있는 테스트 프레임워크

- 개선과 업데이트가 성능 최적화와 사용자 경험 개선 중심으로 지속적으로 이루어짐

개선과 업데이트

1.3

PYTEST 탄생 배경



01

TDD의 확산

02

UNITTEST의 한계

03

PYPY 프로젝트의
시작

04

테스트 프레임워크의
필요성

01 TDD의 확산

테스트 주도 개발의 확산

TDD

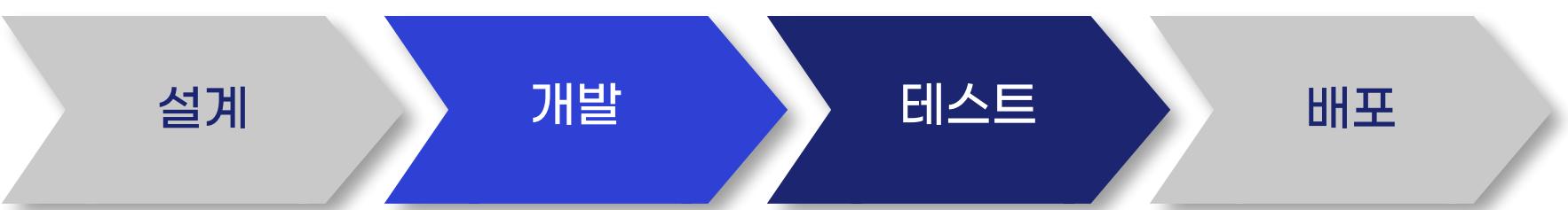
- 본격적인 개발에 들어가기 전 테스트 계획 및 코드를 작성하는 개발 방법론
- 매우 짧은 개발 사이클을 반복하는 프로세스

기존의 개발 프로세스 → 개발을 느리게 하는 잠재적 위험 존재

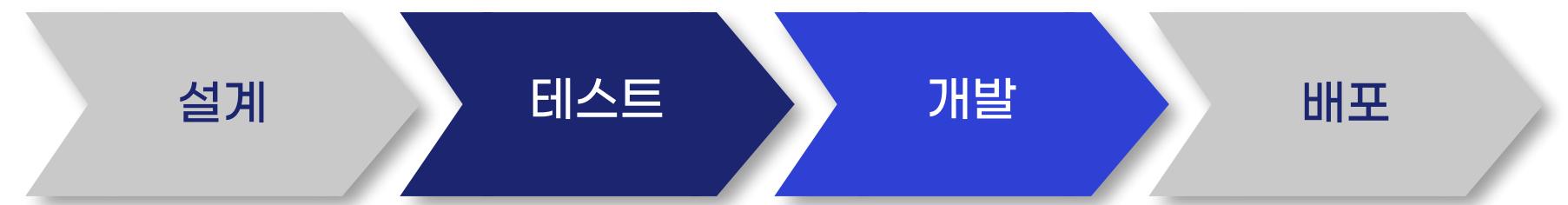
작은 부분의 기능 수정에도 모든 부분을 테스트해야 하므로 버그를 검출
하기 어려워짐

TDD 개발 프로세스는 테스트 코드를 작성한 뒤에 실제 코드를 작성하기
때문에 **효과적**

• 기존 개발 프로세스

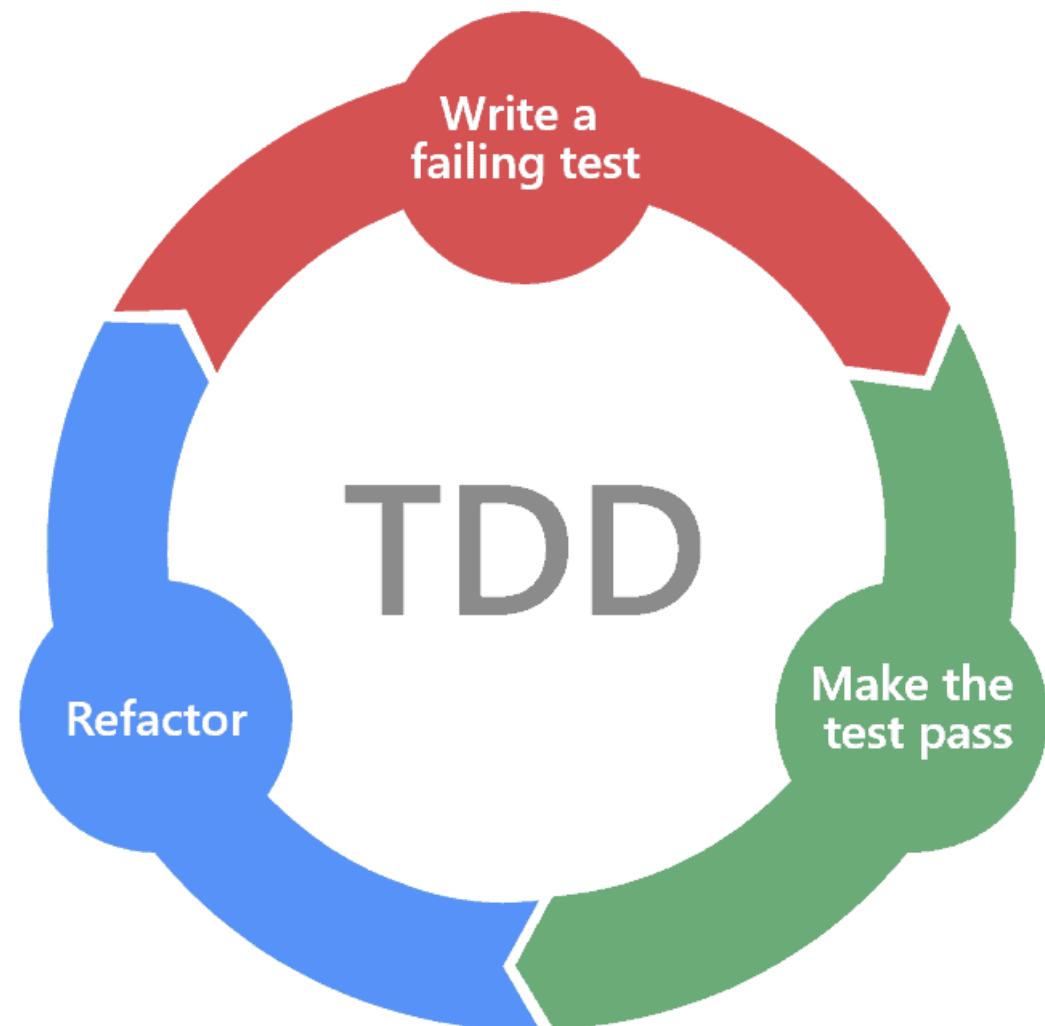


• TDD 개발 프로세스



01 TDD의 확산

테스트 주도 개발의 확산



[TDD 개발 주기]

RED 단계

실패하는 테스트 코드를 먼저 작성한다.

GREEN 단계

테스트 코드를 성공시키기 위한 실제 코드를 작성한다.

BLUE 단계

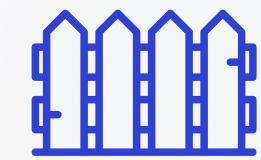
중복 코드 제거, 일반화 등의 리팩토링을 수행한다.

* RED 단계에서 실패하는 테스트 코드를 먼저 작성하는 이유

: 빠른 피드백이 가능해짐(문제 조기 발견 및 수정 가능)

→ 개발 속도 향상 및 현재 개발하려는 기능이 무엇인지, 개발자가 무엇을 구현해야 하는지 명확해짐

02



UNITTEST의 한계

Python의 기본 테스트 모듈인 unittest는 사용자 친화적이지 않은 구문과 제한된 유연성을 가짐

03



PYPY 프로젝트의 시작

PyPy 프로젝트는 Python 인터프리터를 다시 구현하는 프로젝트로 초기에 PyPy에 대한 테스트를 작성하기 위한 도구가 필요했음

04

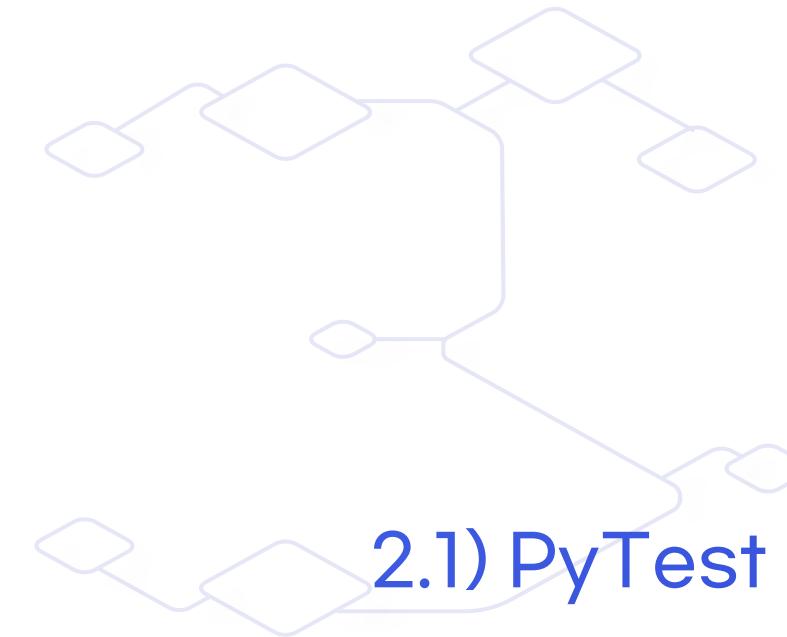


테스트 프레임워크의 필요성

개발자들은 테스트 프레임워크를 통해 테스트를 보다 쉽게 작성하고 관리할 수 있는 필요성을 느꼈음

* unittest = 단위테스트를 작성하고 실행하는데 사용되는 표준 라이브러리

02 PYTEST 커뮤니티



- 2.1) PyTest 공식 웹사이트
- 2.2) GitHub 레포지토리
- 2.3) Real Python
- 2.4) Lambda Test 사이트
- 2.5) The PyCharm Blog 사이트
- 2.6) Dev community
- 2.7) PyTest with Eric

2.1

HTTPS://DOCS.PYTEST.ORG/EN/8.2.X/

PYTEST 공식 웹사이트

- PyTest 프레임워크의 기본적인 소개
- 설치 방법
- 시작 가이드
- 유용한 문서 링크 제공
- 코드 테스트의 기본적인 정보 제공
- 리소스 접근성 향상

The screenshot shows the official Pytest website homepage. It features a large logo at the top left, followed by a navigation bar with links like Home, Get started, How-to guides, Reference guides, Explanation, Complete table of contents, and Library of examples. Below the navigation is a section titled "About pytest" which includes a brief description of what Pytest is and a "Contents" sidebar with links to Changelog, Contributing, Backwards Compatibility, Sponsor, pytest for Enterprise, License, and Contact Channels. At the bottom of this sidebar is a link to "pytest: helps you write better programs".

pytest: helps you write better programs

The `pytest` framework makes it easy to write small, readable tests, and can scale to support complex functional testing for applications and libraries.

`pytest` requires: Python 3.8+ or PyPy3.

PyPI package name: [pytest](#)

A quick example

```
# content of test_sample.py
def inc(x):
    return x + 1

def test_answer():
    assert inc(3) == 5
```

To execute it:

```
$ pytest
=====
platform linux -- Python 3.x.y, pytest-8.x.y, pluggy-1.x.y
rootdir: /home/sweet/project
collected 1 item

test_sample.py F [100%]

=====
 FAILURES =====
 test_answer
=====
 def test_answer():


```

Next Open Trainings and Events

- [Professional Testing with Python](#), via [Python Academy](#) (3 day in-depth training):
 - **June 11th to 13th 2024**, Remote
 - **March 4th to 6th 2025**, Leipzig, Germany / Remote
- [pytest development sprint](#), **June 17th – 22nd 2024**
- [pytest tips and tricks for a better test-suite](#), [Europython 2024](#), **July 8th – 14th 2024** (3h), Prague

Also see [previous talks and blogposts](#).



02 PYTEST 커뮤니티

2.2

HTTPS://GITHUB.COM/PYTEST-DEV/PYTEST

GITHUB 레포지토리

- 소스 코드, 이슈 트래커, 기여 가이드라인 등 다양한 정보 포함
- PyTest 업데이트 시 GitHub 또한 업데이트

02 PYTEST 커뮤니티



The screenshot shows the GitHub repository page for `pytest-dev / pytest`. The repository has 797 issues, 72 pull requests, and 4 projects. It is a public repository with 15,924 commits. Recent commits include:

- nicoddemus Attest package provenance (#12333) - 635fbe2 · 16 hours ago
- .github Attest package provenance (#12333) - 16 hours ago
- bench [ruff UP031] Fix to use format specifiers instead of perce... - 3 weeks ago
- changelog Attest package provenance (#12333) - 16 hours ago
- doc/en Spelling and minor changes (#12122) - 2 days ago
- extra [ruff UP031] Fix to use format specifiers instead of perce... - 3 weeks ago
- scripts [pre-commit.ci] pre-commit autoupdate (#12115) - 2 months ago
- src Merge pull request #12329 from pytest-dev/fix-package-... - 2 days ago
- testing Merge pull request #12329 from pytest-dev/fix-package-... - 2 days ago
- .coveragerc coverage: add assert_never to exclude_lines - 3 years ago
- .git-blame-ignore-revs [ruff] Add change to format in the git blame ignore file - 2 weeks ago
- .gitattributes Add codecov support to AppVeyor and remove covers... - 6 years ago
- .gitignore Parse args from file (#12085) - 2 months ago
- .pre-commit-config.yaml [pre-commit.ci] pre-commit autoupdate (#12292) - 2 weeks ago
- .readthedocs.yml Configure ReadTheDocs to fail on warnings (#11535) - 7 months ago

About

The pytest framework makes it easy to write small tests, yet scales to support complex functional testing

pytest.org

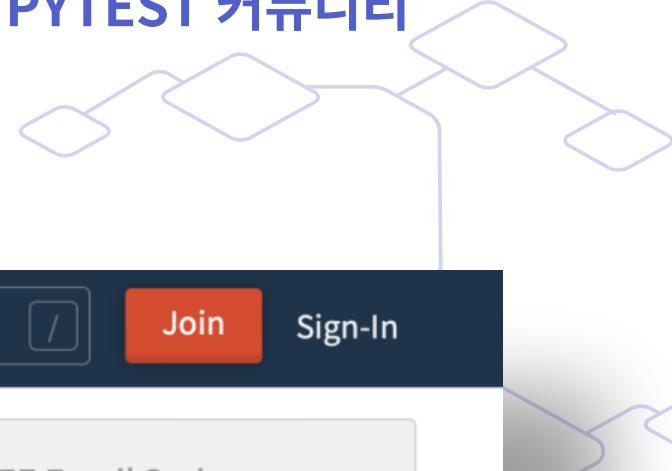
python testing unit-testing test
hacktoberfest

Readme
MIT license
Code of conduct
Cite this repository
Activity
Custom properties
11.5k stars
191 watching
2.5k forks
Report repository

Releases 67

8.2.0 Latest
3 weeks ago

+ 66 releases



2.3

[HTTPS://REALPYTHON.COM/PYTEST-PYTHON-TESTING/](https://realpython.com/pytest-python-testing/)

REAL PYTHON 사이트

- PyTest 프로그래밍에 대한 다양한 튜토리얼 & 가이드 제공
- Python 프로그래밍 전반적인 것을 다룬

Real Python

Start Here Learn Python More Search / Join Sign-In

— FREE Email Series —

Python Tricks 🎉

```

1 # How to merge two dicts
2 # in Python 3.5+
3
4 >>> x = {'a': 1, 'b': 2}
5 >>> y = {'b': 3, 'c': 4}
6
7 >>> z = {**x, **y}
8
9 >>> z
10 {'c': 4, 'a': 1, 'b': 3}

```

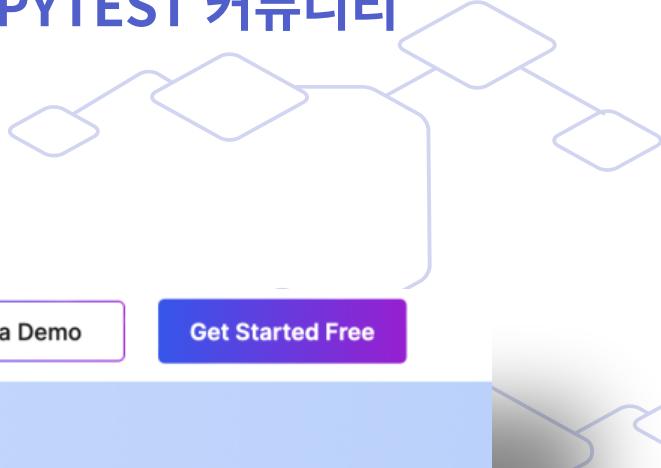
Email... Get Python Tricks »

No spam. Unsubscribe any time.

Browse Topics Guided Learning Paths

Basics Intermediate Advanced

api best-practices career community



2.4

[HTTPS://WWW.LAMBDATEST.COM/LEARNING-HUB/PYTEST-TUTORIAL](https://www.lambdatest.com/learning-hub/pytest-tutorial)

LAMBDA TEST 사이트

- 클라우드 기반 크로스 브라우저 테스팅 플랫폼 위주로 제공

LAMBDATEST Platform Enterprise Resources Developers Pricing Login Book a Demo Get Started Free

Next-Gen App & Browser Testing Cloud

Trusted by 2 Mn+ QAs & Devs to accelerate their release cycles

Start free with Google Start free with Email

Automation Home / Learning Hub / What Is pytest - June 10 2023

What Is pytest: Complete Guide With Best Practices

This pytest tutorial delves into what is pytest and explores its various features and best practices.

ON THIS PAGE

* 클라우드 기반 크로스 브라우저 테스팅 플랫폼

= 다양한 웹 브라우저와 운영체제 환경에서 웹 애플리케이션을 테스팅할 수 있는 온라인 도구, 즉 브라우저나 운영체제 설치가 불필요함



Follow: [X](#) [YouTube](#) [RSS](#)

2.5

[HTTPS://BLOG.JETBRAINS.COM/PYCHARM/2024/03/PYTEST-VS-UNITTEST/](https://blog.jetbrains.com/pycharm/2024/03/pytest-vs-unitest/)

THE PYCHARM BLOG

- PyCharm에 대한 블로그, 테스팅 프레임워크 사용법, 통합 방법, PyTest와 Unittest 및 분석 등의 정보 제공



The PyCharm Blog

The Python IDE for data science and web development

All Releases Tutorials Web Development Data Science Livestreams

Testing

Pytest vs. Unittest: Which Is Better?

Maha Taqi March 15, 2024

Python, being a versatile and widely used programming language, offers several testing frameworks to facilitate the testing process. Two prominent choices are pytest and unittest, both of which come with their own sets of features and advantages.

In this article, we'll be covering the following sections:

- Introduction to each Python testing framework.
- Syntax and coding examples to give you a starting point.
- Advantages and disadvantages.

2.6

[HTTPS://DEV.TO/BSHADMEHR/ADVANCED-TECHNIQUES-WITH-PYTEST-TEST-PARAMETERIZATION-MARKERS-AND-COVERAGE-REPORTING-13DL](https://dev.to/bshadmehr/advanced-techniques-with-pytest-test-parameterization-markers-and-coverage-reporting-13dl)

DEV COMMUNITY

- 개발자 커뮤니티, PyTest 를 포함한 다양한 개발 주제에 대한 튜토리얼과 경험담을 공유
- 개발자들이 효과적으로 테스트 진행할 수 있도록 함

The screenshot shows a Dev.to post by Bahman Shadmehr. The post title is "Advanced Techniques with Pytest: Test Parameterization, Markers, and Coverage Reporting". It includes a series of five articles under the heading "Testing in python with pytest (5 Part Series)". The articles are: 1. Mastering Pytest: An Introduction to Python Testing, 2. Pytest Unleashed: Installation, Writing Test Cases, and T..., 3. Supercharging Your Tests: Pytest Fixtures for Reusable S..., 4. Advanced Techniques with Pytest: Test Parameteriz..., and 5. Mastering Pytest: Tips, Best Practices, and Further Reso...". The post was posted on 2023년 6월 30일. On the right side, there is a sidebar for Bahman Shadmehr, showing his profile picture, name, a "Follow" button, his title as a Software Engineer, his work as a Software Developer, and the date he joined Dev.to (2023년 6월 27일). There are also links to more posts from him and an AWS promoted post.

Bahman Shadmehr
Posted on 2023년 6월 30일

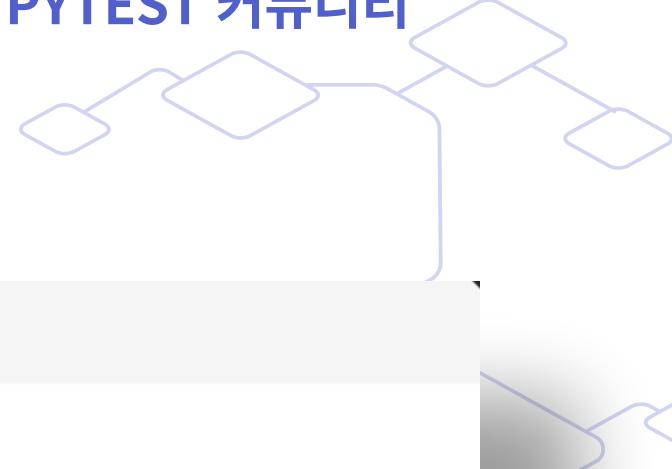
Advanced Techniques with Pytest: Test Parameterization, Markers, and Coverage Reporting

Testing in python with pytest (5 Part Series)

- 1 Mastering Pytest: An Introduction to Python Testing
- 2 Pytest Unleashed: Installation, Writing Test Cases, and T...
- 3 Supercharging Your Tests: Pytest Fixtures for Reusable S...
- 4 Advanced Techniques with Pytest: Test Parameteriz...
- 5 Mastering Pytest: Tips, Best Practices, and Further Reso...

Bahman Shadmehr
Follow
Software Engineer
WORK Software Developer
JOINED 2023년 6월 27일
More from Bahman Shadmehr
Populating Your Online Food Ordering Database with Comprehensive Mock Data #database #sql #programming #webdev
Crafting DDL Statements for an Online Food Ordering Service Database #sql #database #programming
Designing Database Tables for an Online Food Ordering Service #sql #database #programming
AWS PROMOTED

02 PYTEST 커뮤니티

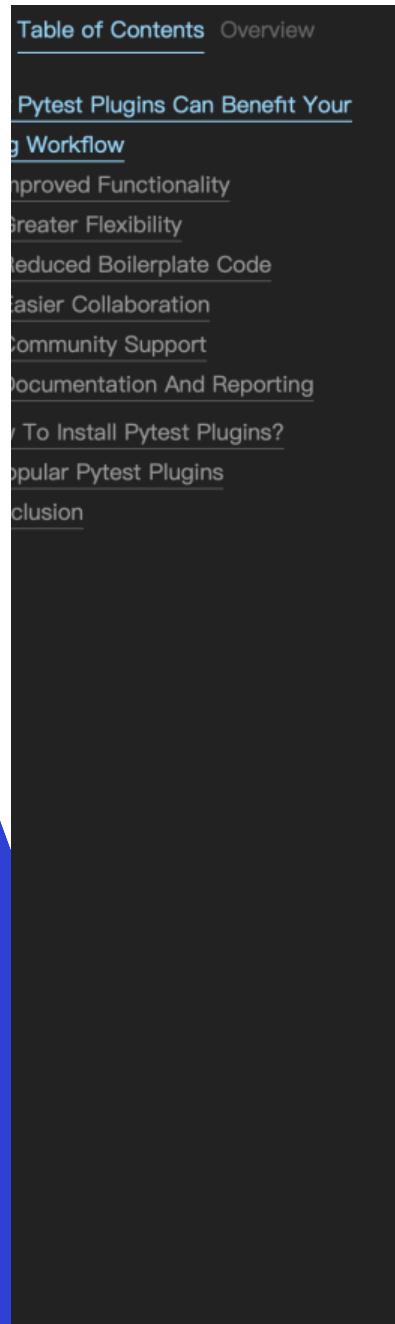


2.7

[HTTPS://PYTEST-WITH-ERIC.COM/PYTEST-BEST-PRACTICES/PYTEST-PLUGINS/](https://pytest-with-eric.com/pytest-best-practices/pytest-plugins/)

PYTEST WITH ERIC

- 인기 있는 플러그인들에 대한 설명과 사용 방법을 상세하게 안내하여 개발자들이 테스트 과정을 효율적으로 관리하고 실행할 수 있도록 함



Pytest With Eric

[Home](#) [Categories](#) [Search](#)

8 Useful Pytest Plugins To Make Your Python Unit Tests Easier, Faster and Prettier

Posted on 2023-02-24 | Edited on 2024-05-16 | In Beginner , Pytest Plugins

Word count in article: 2k | Reading time ≈ 7 mins.

Testing is a critical component of software development, and Pytest is one of the most popular testing frameworks out there.

It provides a powerful and flexible testing environment, allowing you to write concise, maintainable and easy-to-understand tests.

One of the beautiful things about Pytest is that it can be easily extended using plugins, which provide additional functionality and make it even easier to write and maintain your unit tests.

This is a great way to customize and enhance your testing environment, whether measuring code coverage, running tests in parallel, or mocking external services.

In this article, we'll explore some of the most popular and useful Pytest plugins, and discuss how they can help streamline your testing workflow and improve code quality.

Whether you're new to Pytest or a seasoned veteran, you'll find plugins that can help take your testing workflow to the next level.

03 PYTEST 현재 상태



2024년 4월 27일
PyTest 8.2 ver update

PYTEST 업데이트 내용

- 업데이트 내용들이 깃허브 사이트에 공지되며,
피드백을 통해 지속적인 수정을 이뤄나감



기능 삭제

매개변수를 요청할 때 폐기 예정 경고가 발생
`pytest_ignore_collect`, `pytest_collect_file`,
`pytest_pycollect_makemodule`, `pytest_report_header`,
`pytest_report_collectionfinish` 등

기능 추가

파일에서 명령줄 인자를 읽을 수 있는 기능이 추가됨
(예: pytest @tests.txt 파일에는 한 줄에 하나의 인자가 있어야 함)

개선 사항

pytest.importorskip 함수가 ModuleNotFoundError 대신 ImportError를
발생시킬 경우 경고 발생시킴

버그 수정

pytest 8.0.0에서 발생한 회귀 버그 수정
→ setup_method를 포함한 테스트 클래스와 @staticmethod 또는
@classmethod를 사용하는 테스트가 AttributeError로 인해 충돌하는 문제
가 해결됨

PYTEST 8.2.0 버전 소스 파일

- GitHub 속 PyTest 8.2.0 버전 업데이트 내용
- 버그 수정과 사소한 수정 내용, 자료들이 제공됨

POINT 01

소스 코드

POINT 02

업데이트 정보 제공

POINT 03

테스트 파일

POINT 04

소프트웨어 사용 방식

Bug Fixes

- [#12065](#): Fixed a regression in pytest 8.0.0 where test classes containing `setup_method` and tests using `@staticmethod` or `@classmethod` would crash with `AttributeError: 'NoneType' object has no attribute 'setup_method'`. Now the `request.instance <pytest.FixtureRequest.instance> {.interpreted-text role="attr"}` attribute used by fixtures using `@staticmethod` and `@classmethod` is no longer `None`, but a fresh instance of the class, like all other methods. Previously it was `None`, and all fixtures of such tests would share a single `self`.
- [#12135](#): Fixed issue where fixtures adding their finalizer multiple times to fixtures they request would result in unreliable and non-intuitive teardown ordering in some instances.
- [#12194](#): Fixed a bug with `--importmode=importlib` and `--doctest-modules` where child module names were added as attributes in parent modules.
- [#1489](#): Fixed some instances where teardown of higher-scoped fixtures was not happening in the reverse order they were initialized in.

Trivial/Internal Changes

- [#12069](#): `pluggy>=1.5.0` is now required.
- [#12167](#): `cache <cache> {.interpreted-text role="ref"}: create supporting files (CACHEDIR.TAG , .gitignore) in a temporary directory to provide atomic semantics.`

▼ Assets

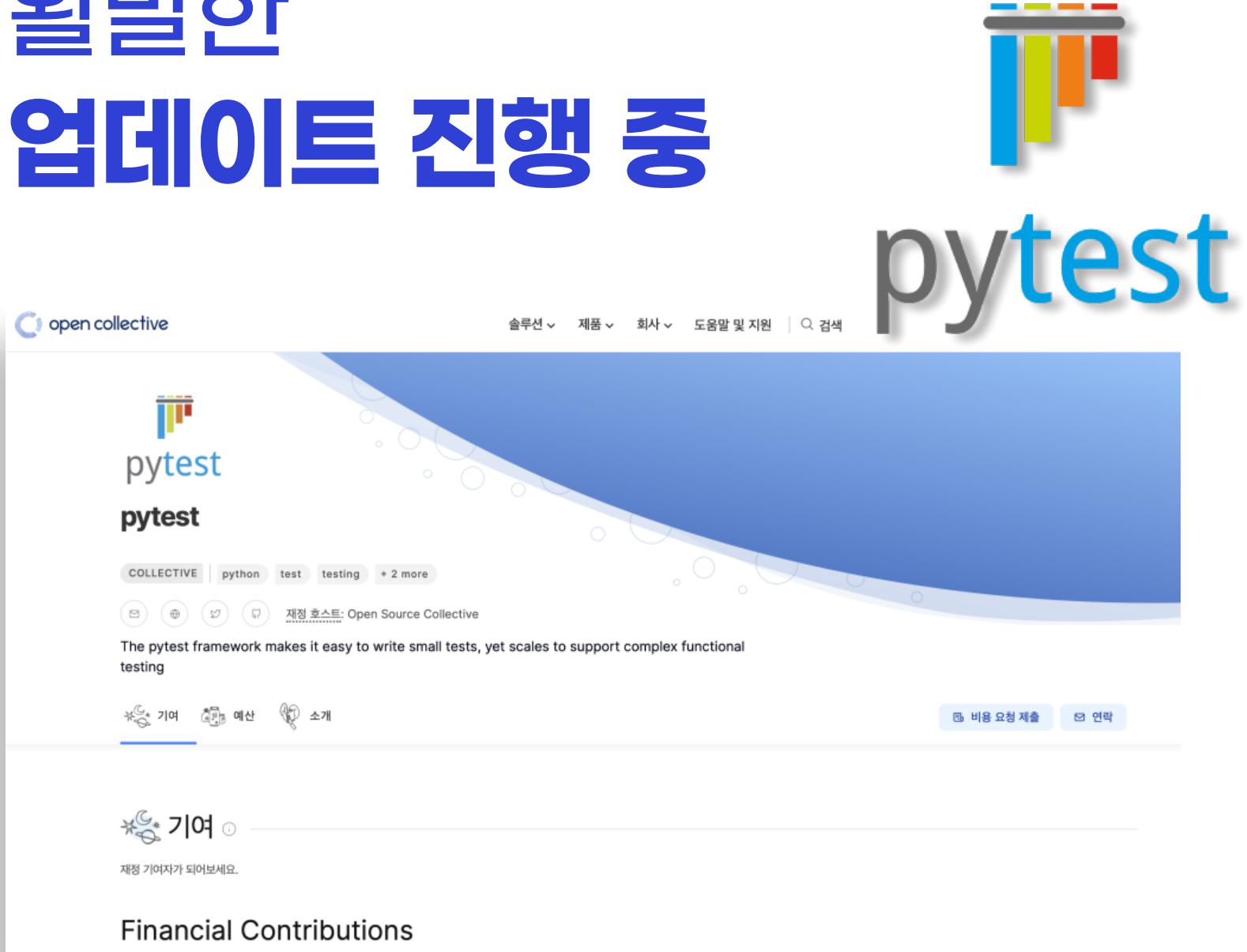
 pytest-8.2.0-py3-none-any.whl	331 KB
 pytest-8.2.0.tar.gz	1.36 MB
 Source code (zip)	
 Source code (tar.gz)	

PYTEST 8.2.0 버전 소스 파일 속 아카이브

- 소스 파일 속 아카이브 내용
- PyTest 버전 관리 팁, 공정한 커뮤니티 환경 조성을 위한 안내서, 테스트 커버리지 요구 사항과 보고서 형식을 정의하는 방법 등이 있음

이름	수정일
AUTHORS	2024년 4월 21
bench	2024년 4월 21
changelog	2024년 4월 21
CHANGELOG.rst	2024년 4월 21
CITATION	2024년 4월 21
CODE_OF_CONDUCT.md	2024년 4월 21
codecov.yml	2024년 4월 21
CONTRIBUTING.rst	2024년 4월 21
doc	2024년 4월 21
extra	2024년 4월 21
LICENSE	2024년 4월 21
OPENCOLLECTIVE.rst	2024년 4월 21
pyproject.toml	2024년 4월 21
README.rst	2024년 4월 21
RELEASING.rst	2024년 4월 21
scripts	2024년 4월 21
src	2024년 4월 21
testing	2024년 4월 21
TIDELIFT.rst	2024년 4월 21
tox.ini	2024년 4월 21

활발한 업데이트 진행 중



[PyTest 후원 사이트]

UPDATE

현재도 활발한 업데이트가 지속 중

NEW VER

새로운 버전이 나오거든 사용자들은
pip install -upgrade pytest
명령어를 사용해 업데이트를 할 수 있음

후원

후원 사이트를 통해
지속적인 후원을 받고 있음

개발 유지

후원을 통해 PyTest를 유지 및 개발

04 PYTEST 개발 목적, 개발 상황

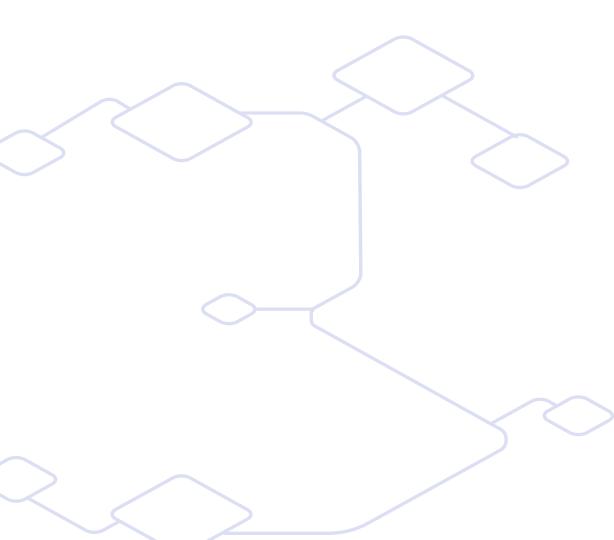


- 4.1) PyTest 개발 목적
- 4.2) PyTest 개발 상황

4.1

PYTEST 개발 목적

- python 개발자들 사이에서 있기있는 테스팅 도구로 만들고,
효율적이고 효과적인 테스팅 프로세스를 구축하는 데 중요한
역할을 함



01

간결한 테스트 코드 작성

02

테스트 주도 개발(TDD) 지원

03

다양한 테스트 시나리오 지원

04

다양한 플러그인 지원



4.2

PYTEST 개발 상황

- 개발 스프린트 개최



- 2016년 크라우드 펀딩 성공적
- pytest 3.0의 tox를 해킹하는 개발 스프린트를 가짐

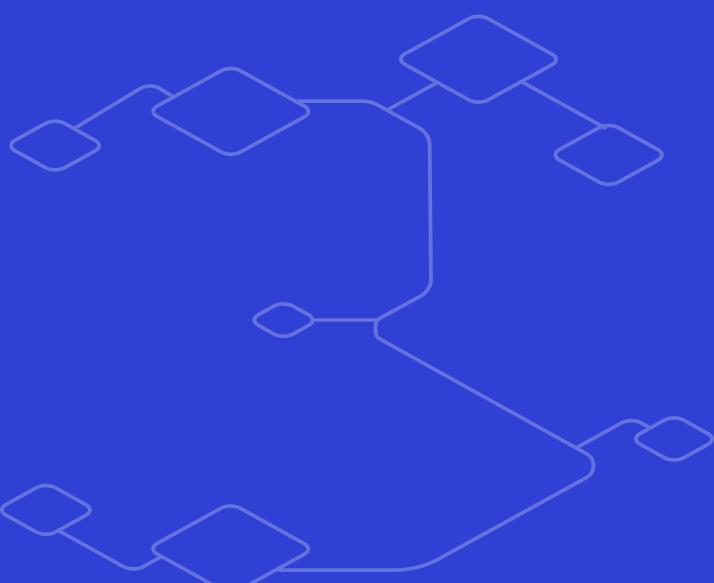


4.2

PYTEST 개발 상황

- 꾸준한 플러그인 업데이트

<https://pypi.org/project/pytest-xdist/>



XDIST v 3.6.1

COV v 5.0.0

MOCK v 3.14.0

Search projects

Help Sponsors Log in Register

pytest-xdist 3.6.1

pip install pytest-xdist

Released: Apr 29, 2024

pytest xdist plugin for distributed testing, most importantly across multiple CPUs

Navigation

- Project description
- Release history
- Download files

Project description

python v3.6.1 conda-forge v3.5.0 python 3.8 | 3.9 | 3.10 | 3.11 | 3.12 test passing code style black

The [pytest-xdist](#) plugin extends pytest with new test execution modes, the most used being distributing tests across multiple CPUs to speed up test execution:

```
pytest -n auto
```

With this call, pytest will spawn a number of workers processes equal to the number of available CPUs, and distribute the tests randomly across them.

4.2

PYTEST 개발 상황

- 대규모 오픈소스



01

 **pandas**

PANDAS

코드 테스트를 pytest로 진행
공식 문서에 pytest 사용법 설명

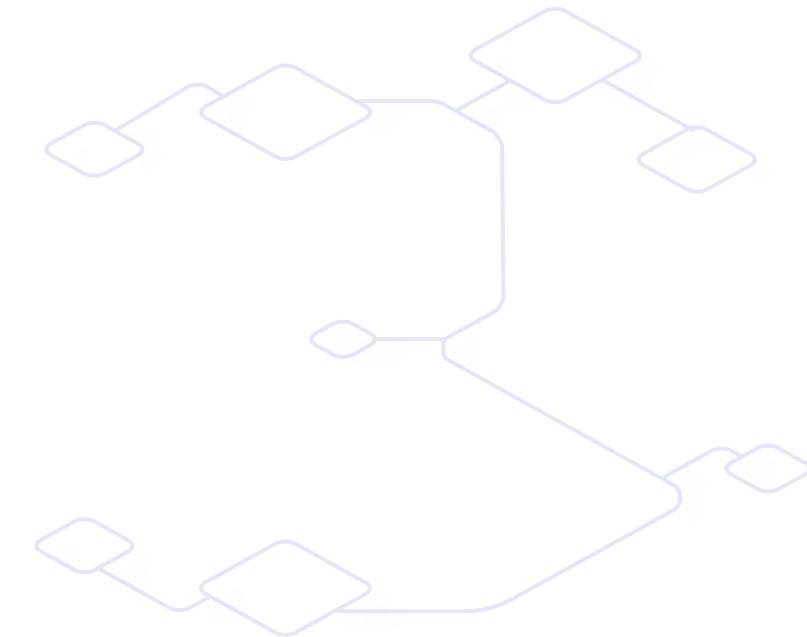
02

SQLAlchemy

SQLALCHEMY

코드 테스트를 pytest로 진행
코드 테스트의 중요성 강조

05 PYTEST 핵심 내용



- 5.0) PyTest 설치 방법
- 5.1) PyTest 유용한 이유 5가지
- 5.2) PyTest 핵심 Keyword

5.0

PYTEST 설치 방법

설치 방법 url - 공식 문서

<https://docs.pytest.org/en/7.2.x/getting-started.html#continue-reading>

01 사전 설치 점검

PyTest를 설치하기 전에, Python이 시스템에 설치되어 있어야 한다.

PyTest는 Python 3.8 버전 이상에서 최적으로 작동한다.

Python 설치 여부와 버전은 명령 프롬프트나 터미널에서 다음 명령어를 입력하여 확인할 수 있다.

```
> pip install -U pytest
```

Python 3 버전을 사용하는 경우, 다음 명령어를 입력하여 확인한다.

```
> Python3 --version
```

02 PyTest 설치

PyTest는 Python 패키지 관리 시스템인 pip을 통해 쉽게 설치할 수 있다. 설치 과정은 다음과 같다.

PyTest를 설치하기 위해 시스템의 명령 프롬프트나 터미널을 연다.

다음 명령어를 입력하여 PyTest를 설치한다.

```
> pip install pytest
```

이 명령어는 PyTest와 필요한 모든 의존성을 자동으로 설치한다.

```
> pip3 install pytest
```

설치가 성공적으로 완료되었는지 확인하기 위해, 다음 명령어를 입력하여 PyTest의 버전을 확인한다.

```
> pytest --version
```

(이 명령어는 설치된 PyTest의 버전을 표시함)

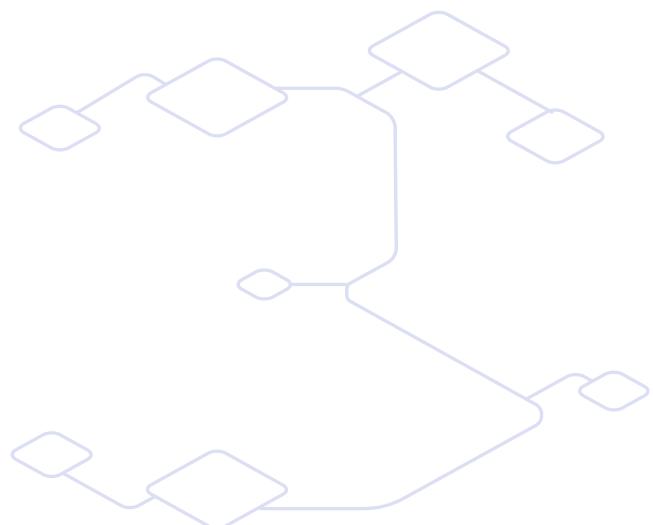
PyTest는 Python 기반의 프로젝트를 위한 강력한 테스트 도구이다.

다음의 설치 방법을 통해 사용자는 쉽게 PyTest를 설치하고, Python 프로젝트의 테스트를 시작할 수 있다.

05 PYTEST 핵심 내용

5.1

PYTEST 유용한 이유 5가지



5.1.1) LESS BOILERPLATE

5.1.2) 더 나은 OUTPUT

5.1.3) LESS TO LEARN

5.1.4) 상태 및 의존성 관리

5.1.5) PLUGIN - BASED 아키텍쳐



01 LESS BOILERPLATE

Boilerplate 코드를 줄여주고 테스트 작성을 더 쉽게 만들 수 있음

BOILERPLATE 란?

- 프로그래밍에서 자주 반복되는 코드 패턴이나 구조
- 특정 작업에 있어서 필수적이지만 해당 작업과 직접적인 관련이 없는 코드들

UNITTEST

Unittest 와 같은 프레임 워크에서 사용할 경우,
특정 메서드를 사용해야만 프로그래밍 가능

PYTEST

기본적으로 Python의 assert 키워드를 이용하여
간단하게 프로그래밍에 대한 테스트 작성 가능



02 더 나은 OUTPUT

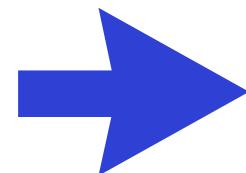
Pytest의 자세한 출력은 테스트 이해와 문제 해결에 유용하며, 코드의 버그를 신속하게 수정 가능

[실행 예시]

PYTEST

- 각 테스트의 실행 결과는 점(성공) 또는 F(실패), E(예외) 처럼 기호로 표시
- 실패한 테스트의 경우, `AssertionError`같은 예외 메시지, 실패한 이유 같이 출력
- 이처럼 어떤 부분에서 문제가 발생했는지 쉽게 파악 가능

```
plaintext
.
.
F
E
```



```
# 잘못된 테스트
def test_modulus():
    # 10을 3으로 나눈 나머지는 0이 되어야 한다고 잘못 기대함
    assert 10 % 3 == 0

=====
===== FAILURES =====
-----
----- test_modulus -----

def test_modulus():
>     assert 10 % 3 == 0
E     assert 1 == 0
E     + where 1 = 10 % 3

test_example.py:15: AssertionError
```

[코드 1] `test_modulus()` 함수 사용 예시

- 10을 3으로 나눈 나머지가 실패하는 테스트

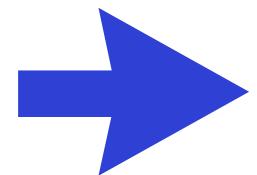


03 LESS TO LEARN

간단한 코드를 이용하여 테스트를 작성하기에 활용 용이

PYTEST

- pytest에서는 일반적인 Python 코드를 이용하여 테스트 작성이 가능
- 테스트를 하기 위해 새로운 문법이나 메서드를 배울 필요 없음
- 전반적으로 Pytest의 학습 unittest보다 훨씬 다루기 쉬움



```
# 예시 테스트 함수
def test_addition():
    assert 2 + 2 == 4

def test_subtraction():
    assert 5 - 3 == 2
```

[코드 2] assert문 사용 예시

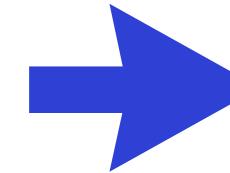


04 상태 및 의존성 관리

Pytest의 간단한 기능을 이용하여 필요한 작업을 효율적으로 처리 가능

PYTEST

- fixture 기능을 사용하면 작업들을 매번 반복해서 코딩할 필요 없음
- 테스트에서 사용할 데이터베이스 연결하고 파일을 읽어오는 작업 예시
 - **database_connection**
테스트 실행 전에 DB 연결을 준비하는 역할
 - **test_read_data_from_database**
실제로 데이터를 읽어오는 역할



```
import pytest

@pytest.fixture
def database_connection():
    connection = setup_database_connection()
    yield connection
    connection.close()

def test_read_data_from_database(database_connection):
    data = database_connection.read_data()
    assert len(data) > 0
```

[코드 3] @pytest.fixture 테스트 코드

- 데이터의 길이를 확인하여 데이터베이스 연결이 되었는지 확인하는 테스트



05 PLUGIN - BASED 아키텍쳐

플러그인들은 Pytest의 기본 기능을 보완, 새로운 기능을 추가하여 테스트 환경 개선

Plugin이란?

소프트웨어에 추가 기능 또는 기능을 확장하기 위한 소프트 웨어 모듈

- **테스트 결과 보고서 출력 기능**

- pytest의 플러그인을 활용하면 테스트 결과를 다양한 형식으로 출력 가능
- pytest-html 플러그인을 사용하면 HTML 형식의 보고서로 테스트 결과를 출력 가능

- **코드 커버리지 측정 기능**

- 테스트가 얼마나 완벽하게 코드를 커버하고 있는지를 평가 가능
- 코드 커버리지를 측정하여 테스트된 코드의 백분율을 확인할 수 있는 pytest-cov 플러그인을 제공

5.2

PYTEST 핵심 KEYWORD



5.2.1) FIXTURES

5.2.2) MARKER

5.2.3) PARAMETETERIZATION(매개변수화)

5.2.4) DURATION REPORTS

5.2.5) PYTEST PLUGINS



01 FIXTURES

테스트할 때 필요한 데이터를 미리 준비해 주기 때문에, 테스트를 더 쉽고 효율적으로 활용 가능

픽스처를 이용한 테스트

→ 반복 코드를 줄이고, 테스트를 관리하기 쉽게 만듦

[Before]

```
def test_addition():
    data = [1, 2, 3]
    assert sum(data) == 6

def test_subtraction():
    data = [4, 5, 6]
    assert max(data) == 6
```

[코드 4] fixture를 사용하지 않고 동일 데이터를 반복 정의하는 예제

- data라는 변수를 테스트를 진행하며,
매번 정의하고 각 테스트에서 이를 재사용 해야하는 경우

[After]

```
import pytest

@pytest.fixture
def sample_data():
    return [1, 2, 3, 4, 5, 6]

def test_addition(sample_data):
    assert sum(sample_data) == 21

def test_subtraction(sample_data):
    assert max(sample_data) == 6
```

[코드 5] sample_data라는 fixture를 사용한 예제

- sample_data 픽스처를 테스트 함수에서 사용
- 매번 정의할 필요 없이 pytest에서 준비 후 사용



02 MARKER

마커(marker)

→ 모든 테스트를 실행하는 것이 아니라 특정한 테스트만 실행하는 것이 좋기에 특정한 테스트만 실행하는데 사용

데이터베이스와 관련 기능을 테스트

- 특정 테스트가 데이터베이스에 접근해야 하는 경우



- @pytest.mark.database_access 마크를 지정
- 이를 통해 데이터베이스 관련 테스트만 실행 가능

마커 사용법

- pytest -m 마커이름

SKIP

특정 조건에 따라
테스트를 건너뜀

SKIPIF

특정 표현식이 참이면
테스트를 건너뜀

XFAIL

실패할 것으로 예상됨으
로 실패해도 통과함

PARAMTRIZE

여러 인수를 사용하여
테스트의 여러 변형을 만듦



03 PARAMETRIZATION

매개변수화란?

→ 테스트 코드를 작성할 때 비슷한 여러 테스트를 한꺼번에 작성하는 방법

- 거꾸로 읽어도 같은 단어나 문장을 확인하는 함수를 테스트 하는 경우 각각의 테스트를 따로 작성하는 예시
- 각 테스트는 다른 입력에 대해 회문 여부를 확인하는데, 테스트는 거의 비슷한 구조를 가지고 있어 **코드 중복**이 발생

```
def test_is_palindrome_empty_string():
    assert is_palindrome("")

def test_is_palindrome_single_character():
    assert is_palindrome("a")

def test_is_palindrome_mixed_casing():
    assert is_palindrome("Bob")

def test_is_palindrome_with_spaces():
    assert is_palindrome("Never odd or even")

def test_is_palindrome_with_punctuation():
    assert is_palindrome("Do geese see God?")

def test_is_palindrome_not_palindrome():
    assert not is_palindrome("abc")

def test_is_palindrome_not Quite():
    assert not is_palindrome("abab")
```

[코드6] 다른 입력에 대한 회문 여부 확인



03 PARAMETRIZATION

```
import pytest

@pytest.mark.parametrize("word, expected_result", [
    ("", True),
    ("a", True),
    ("Bob", True),
    ("Never odd or even", True),
    ("Do geese see God?", True),
    ("abc", False),
    ("abab", False),
])
def test_is_palindrome(word, expected_result):
    assert is_palindrome(word) == expected_result
```

- `@pytest.mark.parametrize()`를 사용 단일 테스트 함수 정의를 여러 입력 값과 예상 결과를 가진 테스트 케이스로 확장
- 더 적은 양의 코드로 동일한 작업을 수행 가능

[코드7] 매개변수화를 사용 테스트를 합치는 예제



04 DURATION REPORTS

오버헤드란?

- 테스트를 실행할 때마다 코드를 전환하는 데 시간이 걸리는 상황

테스트가 오래 걸리면 오버헤드가 더 많이 발생하게 되고, 오버헤드가 많이 발생할 수록 테스트 및 작업에 방해

```
import time; import pytest

def test_fast(): time.sleep(0.1); assert True

def test_medium(): time.sleep(0.5); assert True

def test_slow(): time.sleep(1); assert True
```

```
===== test session starts =====
platform linux -- Python 3.8.5, pytest-6.2.1, py-1.10.0, pluggy-0.13.1
collected 3 items

test_example.py ...

===== slowest durations =====
1.00s call    test_example.py::test_slow
0.50s call    test_example.py::test_medium
0.10s call    test_example.py::test_fast
===== 3 passed in 1.62s =====
```



PYTEST Duration Report

Pytest에서는 테스트의 실행 시간을 기록하고 가장 오래 걸리는 테스트를 시각적으로 보여주는 Duration Reports의 기능을 사용할 수 있음



05 PYTEST PLUGIN

Pytest 플러그인

- pytest의 기능을 확장하고 보완하는 데 사용되는 도구
- 유연성과 확장성을 높여주며, 사용자가 더욱 효율적으로 테스트를 작성하고 관리할 수 있도록 도와줌

1. Pytest-randomly

- 테스트 순서를 무작위로 바꿔주는 플러그인
- 코드가 커질수록 순서에 따라 영향을 줄 수 있기에 이러한 문제를 방지하는 플러그인

2. Pytest-cov

- 테스트가 코드를 얼마나 커버하는지 측정하는 코드 커버리지 플러그인

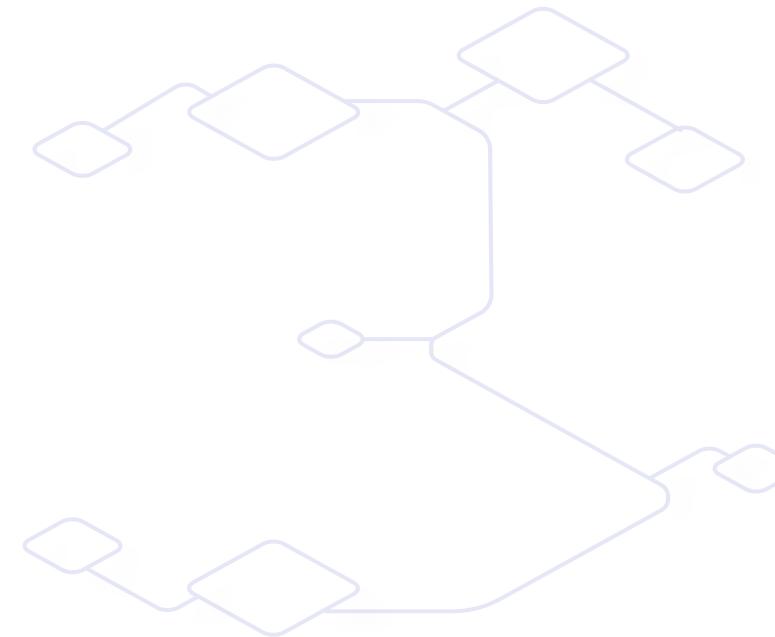
3. Pytest-django

- 데이터베이스와 상호작용을 쉽게 테스트하는 플러그인 작업을 줄이고 효율적인 테스트 수행 가능

4. Pytest-bdd

- BDD : 행동주도 개발애플리케이션 동작 중심 개발하는 방식
- 간단한 언어로 기능 구현을 결정하는 방법

06 PYTEST 향후 방향



- 6.1) 성능 향상
- 6.2) 확장성과 유연성 강화
- 6.3) 새로운 기능 추가
- 6.4) 비동기 및 병렬 프로그램

01 성능 향상

테스트 코드를 주기적으로 리뷰하고 리팩토링하여 성능을 개선

- 중복 제거: 중복된 코드를 제거하고, 공통된 부분을 함수나 fixture로 추출
- 불필요한 I/O 제거: 테스트 중 불필요한 파일 시스템 접근이나 네트워크 요청을 모킹하여 제거

* I/O = Input/Output



지속적인 코드 리팩토링

커뮤니티와 기여

- Pytest 커뮤니티 참여: pytest는 오픈 소스 프로젝트로, GitHub 저장소를 통해 활발히 개발되고 있음. 커뮤니티에 참여하여 새로운 기능 요청, 버그 리포트, 코드 기여 등을 통해 발전에 기여할 수 있음
- 플러그인 개발: pytest는 플러그인 아키텍처를 가지고 있어, 새로운 플러그인을 개발하거나 기존 플러그인을 개선함으로써 성능을 향상시킬 수 있음.

```
pip install pytest-benchmark
```

- Pytest-benchmark: 테스트 성능을 벤치마킹하고 분석할 수 있는 플러그인
- 성능 분석 도구: 테스트 실행 중 CPU, 메모리 사용량을 모니터링하여 병목 지점을 식별



02

유연성과 확장성



● 테스트 자동 생성

AI와 머신러닝을 활용하여 자동으로 테스트 케이스를 생성
& 코드 변경에 따른 영향을 예측하여 테스트 효율성 극대화

● 자동화된 테스트 최적화

머신러닝 모델을 사용하여 테스트 실행 순서 최적화 & 테스
트 실행 시간 최소화

- PyTest는 다양한 설정 옵션을 제공하여 사용자가 자신의 테스트 환경에 맞게 조정할 수 있도록 함
→ 테스트 실행, 리포팅, 어설션 등의 다양한 측면에서 제공될 예정
- 테스트 환경이나 조건에 따라 동적으로 테스트를 구성할 수 있는 기능이 강화될 것

03 새로운 기능 추가

PyTest의 사용자 편의성 개선을 위한 새로운 기능

1) 사용자 편의성 개선을 위한 새로운 어설션 유형

- 기존의 assert 문법을 보완하여 보다 직관적인 표현이 가능한 어설션들이 추가될 수 있음
→ 테스트 코드의 가독성을 높여줌
- 데이터 타입, 예외 처리, 조건부 테스트 등을 보다 쉽게 표현할 수 있는 어설션들이 추가

2) 픽스처 관리 방법 등 추가 예정

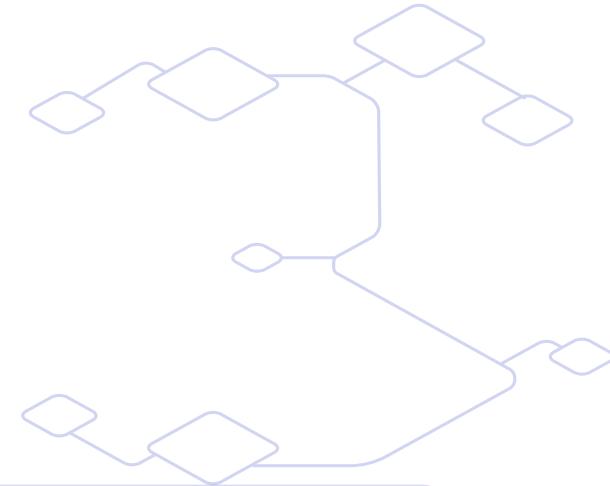
- 특정 조건에 따라 동적으로 픽스처를 생성할 수 있는 기능이 추가될 예정
→ 다양한 테스트 시나리오에 맞춰 픽스처를 유연하게 관리할 수 있게 해줌
- 픽스처의 적용 범위를 더 세밀하게 설정할 수 있는 기능이 추가
→ 모듈, 클래스, 함수 등 다양한 범위에 대해 픽스처를 적용할 수 있음
- 픽스처가 더 이상 필요하지 않을 때 자동으로 정리(cleanup) 작업을 수행하는 기능 강화

3) 효율적인 테스트 작성과 실행

- 테스트가 변경되었을 때 즉시 실행하여 피드백을 빠르게 제공하는 기능이 추가
→ 개발자들이 코드 변경 사항을 바로 확인할 수 있게 해줌



04 비동기 및 병렬 프로그램



```
pip install pytest-asyncio
```

- pytest-asyncio 플러그인 설치

```
import pytest
import asyncio

@pytest.mark.asyncio
async def test_async_function():
    result = await some_async_function()
    assert result == expected_value
```

- PyTest는 `async def`로 정의된 비동기 테스트 함수의 지원을 강화할 예정
→ 비동기 코드의 테스트를 보다 자연스럽고 직관적으로 작성할 수 있게 함
- 비동기 함수에서 사용할 수 있는 비동기 픽스처의 지원 확대
→ 비동기 코드 테스트를 위한 설정 및 정리 작업을 더 효율적으로 수행할 수 있게 함
- 병렬 테스트 실행 시 자원을 동적으로 할당하여 최적의 성능을 발휘할 수 있도록 지원하는 기능 추가
→ 테스트 실행 중 자원 사용의 효율성을 높여줍니다.

* 비동기 프로그래밍

: 작업이 시작되고 그 작업이 마무리 되기까지 기다리는 동안 프로그램이 다른 작업을 계속 수행할 수 있도록 하는 것. 특히 I/O 작업에서 유용함

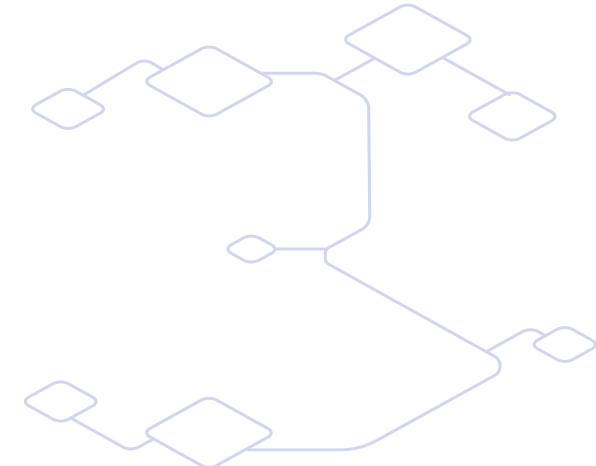
* 병렬 프로그래밍

: 여러 작업을 동시에 실행해서 성능을 향상시키는 방식. CPU 작업에서 효과적 (대규모 데이터 처리와 복잡한 계산에 유용함)

참고 문헌

- <https://docs.pytest.org/en/latest/contents.html>
- 파이테스트 공식문서 How-to guides
- <https://realpython.com/pytest-python-testing/#less-boilerplate>
- real python 사이트 Effective Python Testing With Pytest
- <https://www.datacamp.com/tutorial/pytest-tutorial-a-hands-on-guide-to-unit-testing>
- 데이터캠프 How to Use Pytest for Unit Testing
- <https://dzone.com/articles/10-awesome-features-of-pytest>
- dzon사이트 10 Awesome Features of Pytest
- https://www.tutorialspoint.com/pytest/pytest_introduction.htm
- Pytest Tutorial
- <https://www.lambdatest.com/learning-hub/pytest-tutorial>
- What Is pytest: Complete Guide With Best Practices
- <https://www.design-reuse.com/articles/53256/pytest-for-functional-test-automation-with-python.html>
- Pytest for Functional Test Automation with Python
- <https://github.com/pytest-dev/pytest/issues>
- 깃허브의 Bugs/Requests
- <https://docs.pytest.org/en/7.2.x/getting-started.html#continue-reading>
- 파이테스트 설치 방법

감사합니다.



부록

- **PYPY 프로젝트**

파이썬의 언어 구현 중 하나. 백준 사이트에서 코딩 문제를 풀 때 python으로 돌릴 수도 있고 pypy로 돌릴 수도 있다. pypy는 프로젝트 규모가 커짐에 따라 성능 개선의 필요성이 생겨 나타난 것이며 python용 동적 언어 컴파일러이다. python언어를 더 빠르게 실행할 수 있다.

- **클라우드 기반 크로스 브라우저 테스팅**

최대한 많은 종류의 웹 브라우저에서 정상적으로 작동하는 웹페이지를 만드는 방법 / 다양한 플랫폼과 웹 페이지에서 해당 페이지가 일관되게 기능하고 보이도록 만드는 웹 개발 기법

- **PYTEST를 기존의 UNITTEST 코드의 통합이 가능한가요?**

가능! Pytest는 Unittest 코드와 호환되며 기존 Unittest 테스트를 그대로 실행 가능하다. Pytest를 사용하여 코드를 점진적으로 Pytest로 이동시킬 수 있다.

- **PYTEST에서 코드를 그룹화할 수 있나요?**

가능! Pytest에서는 mark 기능을 사용하여 테스트를 그룹화할 수 있다. Pytest.mark.<marker_name>을 사용하여 테스트에 마커를 추가하고, 특정 마커를 가진 테스트만 실행할 수 있다.

- **테스트 커버리지 측정을 위해 PYTEST와 어떤 도구를 함께 사용할 수 있나요?**

Pytest는 coverage.py와 통합하여 테스트 커버리지를 측정할 수 있다. Pytest-cov 플러그인을 설치하고 --cov 옵션을 사용하여 커버리지 보고서를 생성할 수 있다.

- **PYTEST에서 외부 API를 모킹하는 방법이 무엇인가요?**

외부 API 호출을 모킹하려면 pytest-mock 또는 unittest.mock 모듈을 사용할 수 있다. pytest-mock은 PyTest와 잘 통합되어 있으며, 간편하게 외부 의존성을 모킹할 수 있다. mocker.patch를 사용하여 특정 함수나 메서드를 모킹하고, 원하는 반환 값을 설정 할 수 있다. 이를 통해 외부 API 호출을 실제로 수행하지 않고도 테스트할 수 있다.

부록

- **PYTEST 를 CI/CD 파이프라인에 통합하는 방법**

PyTest를 CI/CD 파이프라인에 통합하려면, 먼저 CI/CD 도구(예: Jenkins, GitLab CI/CD, GitHub Actions 등)에 PyTest를 설치하고 실행할 수 있는 환경을 설정해야 한다. 그런 다음, PyTest를 실행하는 스크립트를 작성하고, 테스트 결과를 리포팅할 수 있는 플러그인을 사용하여 테스트 결과를 시각화한다. 예를 들어, pytest-cov를 사용하여 코드 커버리지를 측정하고, 결과를 CI/CD 도구에 통합할 수 있다.

- **PYTEST에서 테스트 결과를 여러 형식으로 출력할 수 있다고 했는데 어떤 형식으로 출력이 가능한지**

플러그인을 통해서 HTML 형식으로 출력이 가능하다. 또한, JUnit XML 형식으로 테스트 결과를 출력할 수 있다. 이는 CI/CD 도구와의 통합에 유용하다. JSON 형식으로도 출력이 가능하다. 다양한 플러그인을 활용하여 pytest의 테스트 결과를 필요한 형식으로 출력하고, 이를 CI/CD 파이프라인이나 보고서 작성 등에 유용하게 활용할 수 있다.

- **UNITTEST**

프로그램이 잘 작동하는지 확인 하는 것은 동일하나, unittest는 ‘테스트 클래스’를 만들어야 하고, 클래스 안에서 각 부분을 테스트 하는 ‘테스트 메서드’ 또한 명시적으로 작성해야한다. pytest의 경우 특별한 메서드 없이 단순 함수 형식으로 테스트 가능하다.

- **PYTEST가 UNITTEST의 차이점**

PyTest와 unittest는 둘 다 python의 테스팅 프레임워크로 소프트웨어의 기능을 검증하고 코드의 품질을 유지하는 데 사용된다. PyTest는 매우 간결한 문법을 제공하며 테스트 함수를 작성할 때 클래스나 메서드로 감싸지 않아도 되며, 함수 이름만 "test_"로 시작하면 된다. unittest는 테스트는 반드시 클래스 내에 정의되어야 하며, 클래스는 `unittest.TestCase`를 상속받아야 한다. 각 테스트 메서드는 "test"로 시작해야 한다. PyTest는 매우 강력한 플러그인 시스템을 가지고 있어 기능 확장이 용이하고, unittest는 플러그인 시스템이 내장되어 있지 않아서 기능 확장이 제한적이다.

PyTest는 내장된 어서션 기능을 사용하며 unittest는 다양한 어서션 메서드를 제공하지만, assert문을 직접 사용할 수는 없다. 어서션 메서드를 사용해야 한다.

- **BDD VS TDD**

TDD는 코드의 동작을 검증하고 보증하기 위해 코드 기능과 로직에 집중되어 있고, BDD는 사용자 동작을 중심으로 기대한 동작을 수행하는지를 확인하는 데 중점을 둔다.