

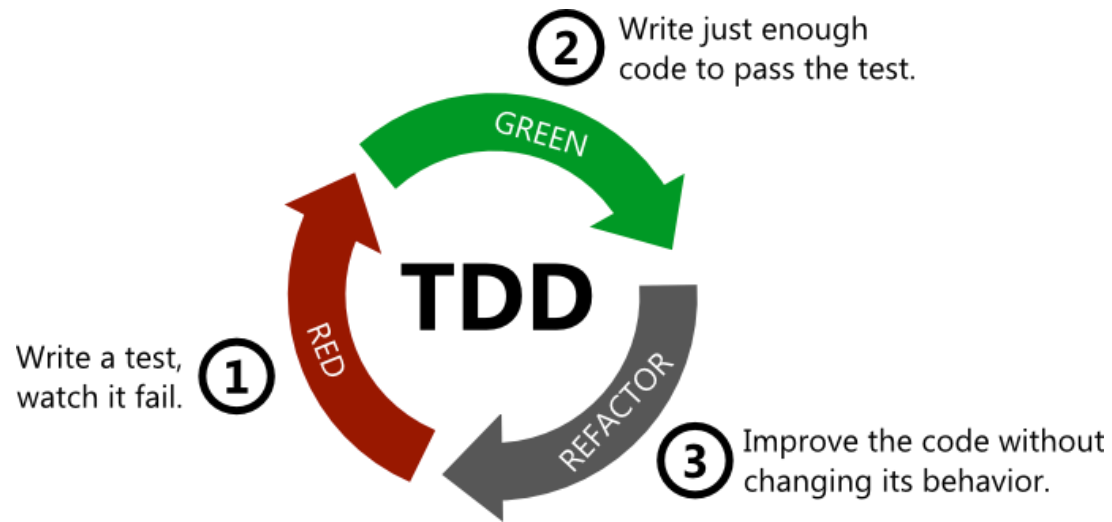
Python을 이용한 테스트 및 패키지 배포

시스템소프트웨어 연구실
이건희

목차

1. TDD 개발방법
2. Python Unit-test
3. Travis-CI를 통한 테스트
4. Coveralls를 이용한 Code Coverage 측정
5. Pypi에 파이썬 패키지 배포하기

TDD 개발방법



• Test-Driven-Development

1. 테스트를 코드를 짜되 실패하게끔.
2. 이 실패하는 코드를 성공시킬 것.
3. 그리고 리팩토링.

TDD 개발방법- 1. 실패하는 코드를 짜라

```
import unittest
from khleepkg03.app import Caculator

calc = Caculator(True, True, True, True, True)
dict = calc.getStatus()

class TestPackage(unittest.TestCase):
    def test_module_adder(self):
        # Member Value: Dict["add"]
        if self.assertTrue(dict["add"]):
            pass
```

- 테스트 코드
- 실행되지 않는 코드
- 실행이 되게 만들어야 함

TDD 개발방법- 2. 코드가 돌아가게 해라

```
class Caculator:
    def __init__(self, isAdd, isSub, isMul, isDiv, isRem):
        self.isAdd = isAdd
        self.isSub = isSub
        self.isMul = isMul
        self.isDiv = isDiv
        self.isRem = isRem
```

```
def getStatus(self):
    return {
        "add": self.isAdd,
        "sub": self.isSub,
        "mul": self.isMul,
        "div": self.isDiv,
        "rem": self.isRem
    }
```

- 실행되는 코드를 만들기
- 테스트 코드로부터 생성
- 테스트 코드가 실행되어야 함

TDD 개발방법- 3. 리팩토링

- 효율적으로 돌아가는 소스코드로 바꾸기
- 초기에 설정한 기능에서 변경되선 안됨
- 에러 처리, 최적화 등등

TDD 개발방법의 장점

1. 보다 탄탄한 코드 구조
2. 재설계 시간의 단축
3. 디버깅 시간의 단축
4. 테스트 문서 대체 가능
5. 추가 기능의 용이함

TDD 개발방법의 단점

1. 코드 생산성이 떨어진다.
2. 시간이 다소 오래걸린다.
3. 도구/규칙에 집착할 수록 접근성이 떨어질 수 있다.

Python의 유닛 단위 테스트

- 기본적으로 **unittest**라는 모듈이 주어짐
- pytest 모듈과 함께 사용
- 테스트 클래스에 unittest 클래스를 상속할 것
- 테스트할 메서드 이름 앞에 항상 test를 붙일 것

Python unittest 모듈을 이용한 테스트

```
import unittest
from khleepkg03.app import Caculator

class TestPackage(unittest.TestCase):
    def test_module(self):
        calc = Caculator(isAdd=True,
                          isSub=False,
                          isMul=False,
                          isDiv=False,
                          isRem=False)

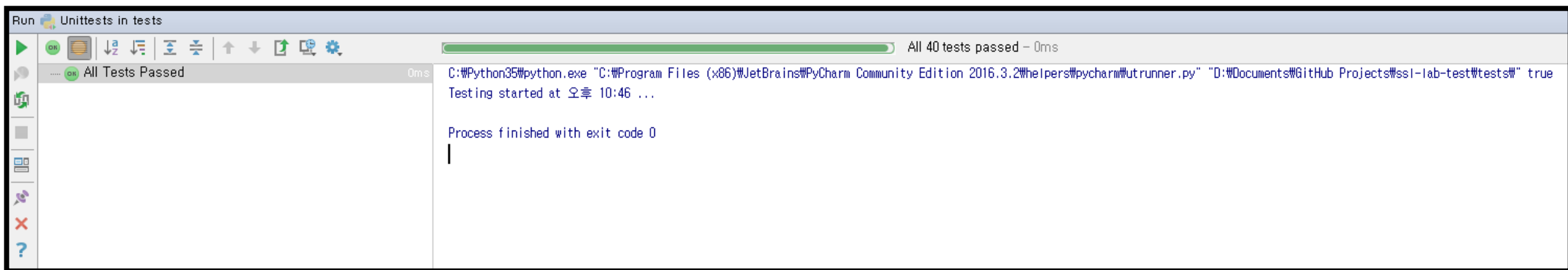
        dict = calc.getStatus()
        if self.assertTrue(dict["add"]): pass
        if self.assertFalse(dict["sub"]): pass
        if self.assertFalse(dict["mul"]): pass
        if self.assertFalse(dict["div"]): pass
        if self.assertFalse(dict["rem"]): pass

    def test_adder_1(self):
        calc = Caculator(isAdd=True,
                          isSub=False,
                          isMul=False,
                          isDiv=False,
                          isRem=False)

        self.assertEqual(calc.adder(1, 2), 3)
```

- import unittest
- unittest의 **TestCase** 클래스 상속
- 테스트할 메서드 이름 앞에 **test** 붙이기
- 내장 함수로 assertFalse 등이 있음

Pycharm을 이용한 pytest 실행



▲ 5개의 모듈 및 40개의 테스트 메서드 실행 결과

Github로 소스코드 커밋하기

- 일부 제외할 파일이 있으면 **.gitignore** 파일에 명시
- 테스트 코드는 **tests** 디렉토리에 모아두기
- 패키지명에 맞춰서 메인 소스코드 디렉토리 이름 지정
- 라이선스 – **LICENSE**, 사용설명 등 – **README.md**
- 그 외의 문서 등

Github로 소스코드 커밋하기

```
$ git status
```

```
$ git add *
```

```
$ git commit -m "<커밋 내용>"
```

```
$ git push origin <커밋할 브랜치>
```

만약 커밋할 브랜치와 싱크가 안맞으면 git pull 명령어 사용해야 함

Github를 이용한 테스트/문서화

- 테스트 자동화 도구 : Travis-CI, Jenkins 등
- **Code Coverage** : Coveralls, (python의 경우 coverage모듈) 등
- 문서화 : codedocs 등



Travis CI 를 통한 테스트 자동화


- **CI**(Continuous Integration) 툴의 일종
- Repository 내의 **.travis.yml**라는 파일을 통해 스크립트 실행
- Travis-ci에 테스트를 할 파일을 실행하게 할 것
- 정상적으로 완료되면 **build passing**이라 알려줌

Travis-CI를 사용하는 방법

- Travis-CI에 **Legacy Services Integration** 허용하기
- **.travis.yml** 스크립트 파일 작성 및 커밋 넣기
- 테스트 결과를 기다린다.

Legacy Services Integration 허용

MY ACCOUNT

 K.H Lee


Sync account

ORGANIZATIONS

You are not currently a member of any organization.

MISSING AN ORGANIZATION?

Review and add your authorized organizations.

 K.H Lee












@KeonHeeLee

Repositories Settings

We're only showing your public repositories. You can find your private projects on [travis-ci.com](#).

Legacy Services Integration

Filter repositories

 practice-nodejs	<input type="checkbox"/>	Settings
 practice-swift	<input type="checkbox"/>	Settings
 pratic-travis-ci	<input checked="" type="checkbox"/>	Settings
 proxy-server	<input type="checkbox"/>	Settings
 pymc3	<input type="checkbox"/>	Settings
 PyMySQL	<input type="checkbox"/>	Settings
 React-practice-todoList	<input type="checkbox"/>	Settings
 simple_naver_talktalk	<input type="checkbox"/>	Settings
 simple-chatting-system	<input type="checkbox"/>	Settings
 simple-pocket-mon-game	<input type="checkbox"/>	Settings
 ssl-lab-test	<input checked="" type="checkbox"/>	Settings

.travis.yml 스크립트 작성

```
language: python
python:
  - "3.4"
  - "3.5"
  - "3.6"

install:
  - pip install -r requirements.txt
  - pip install coveralls

script:
  - coverage run --source=khleepkg03 setup.py test

after_success:
  - coveralls

branches:
  only:
    - master
```

1. 사용할 언어 및 버전 작성

2. 의존성 있는 패키지 설치

3. 실행할 스크립트 작성

4. 기타사항 작성

Coveralls를 이용한 Code Coverage

- python기준 **Coverage.py**와 **nosetests**가 있음
- Travis-CI를 사용할 경우, 스크립트에 이를 명시
- 파이썬 패키지 단위로 테스트 후 Coveralls에 제출하는 방식

Coveralls 연동시키기

- spec_helper.rb 파일 작성
- **.travis.yml** 에 coverage 테스트 결과를 coveralls로 제출
- 결과를 기다린다.

```
2 lines (2 sloc) | 35 Bytes
```

```
1  require 'coveralls'
2  Coveralls.wear!
```

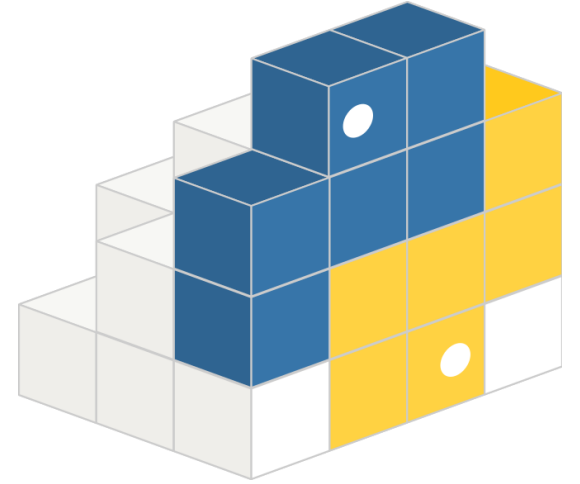
Travis-CI 및 Coveralls 적용결과

SSL-Lab-seminar-test



- Travis-CI 결과 : build passing
- Coveralls 결과 : coverage 21%

Pypi로 패키지 배포하기



- **Python Package Index**
- 여기에 패키지를 등록함으로써, pip으로 패키지 사용 가능
- **twine** 모듈을 이용한 빌드 및 배포가 가능

setup.py 파일 작성

```
from setuptools import setup, find_packages

version = "0.0.1"

with open("README.md", "r") as fh:
    long_description = fh.read()

setup(
    name = "khleepkg03",
    version = version,
    description = "SSL-Seminar-test repository:: Topic - Package Test and Deployment",
    long_description= long_description,
    long_description_content_type="text/markdown",
    author = "Keon-Hee Lee",
    author_email = "beta1360@naver.com",
    url = "https://github.com/KeonHeeLee/ssl-lab-test",
    install_requires= [],
    packages = find_packages(exclude=['test*']),
    keywords = ['Test', 'TDD', 'Deployment', 'Seminar'],
    python_requires = '>=3',
    zip_safe = False,
    license = 'MIT',
    classifiers=[
        'Programming Language :: Python :: 3',
        'Programming Language :: Python :: 3.2',
        'Programming Language :: Python :: 3.3',
        'Programming Language :: Python :: 3.4',
        'Programming Language :: Python :: 3.5',
        'Programming Language :: Python :: 3.6',
        'Programming Language :: Python :: 3.7',
        'License :: OSI Approved :: MIT License'
    ]
)
```

- 패키지 설치를 위한 소스코드
- 이 코드에 따라 패키지 배포
- 패키지를 묶을 때 사용

기타 파일 작성

```
[metadata]
license = "MIT"
license_file = LICENSE
author = Keon-Hee Lee
author_email = beta1360@naver.com
```

setup.cfg

```
include LICENSE
include README.md
```

MANIFEST.in

```
[distutils]
index-servers=pypi

[pypi]
repository = https://pypi.python.org/pypi
username = the_gigi
```

~/.pypirc

Pypi로 패키지 배포

```
$ python3 setup.py sdist
```

```
# 패키지 빌드하기 (build, dist 디렉토리 및 .whl파일 생성)
```

```
$ pip3 install twine
```

```
# twine 모듈이 설치가 안되었다면 설치해줄 것.
```

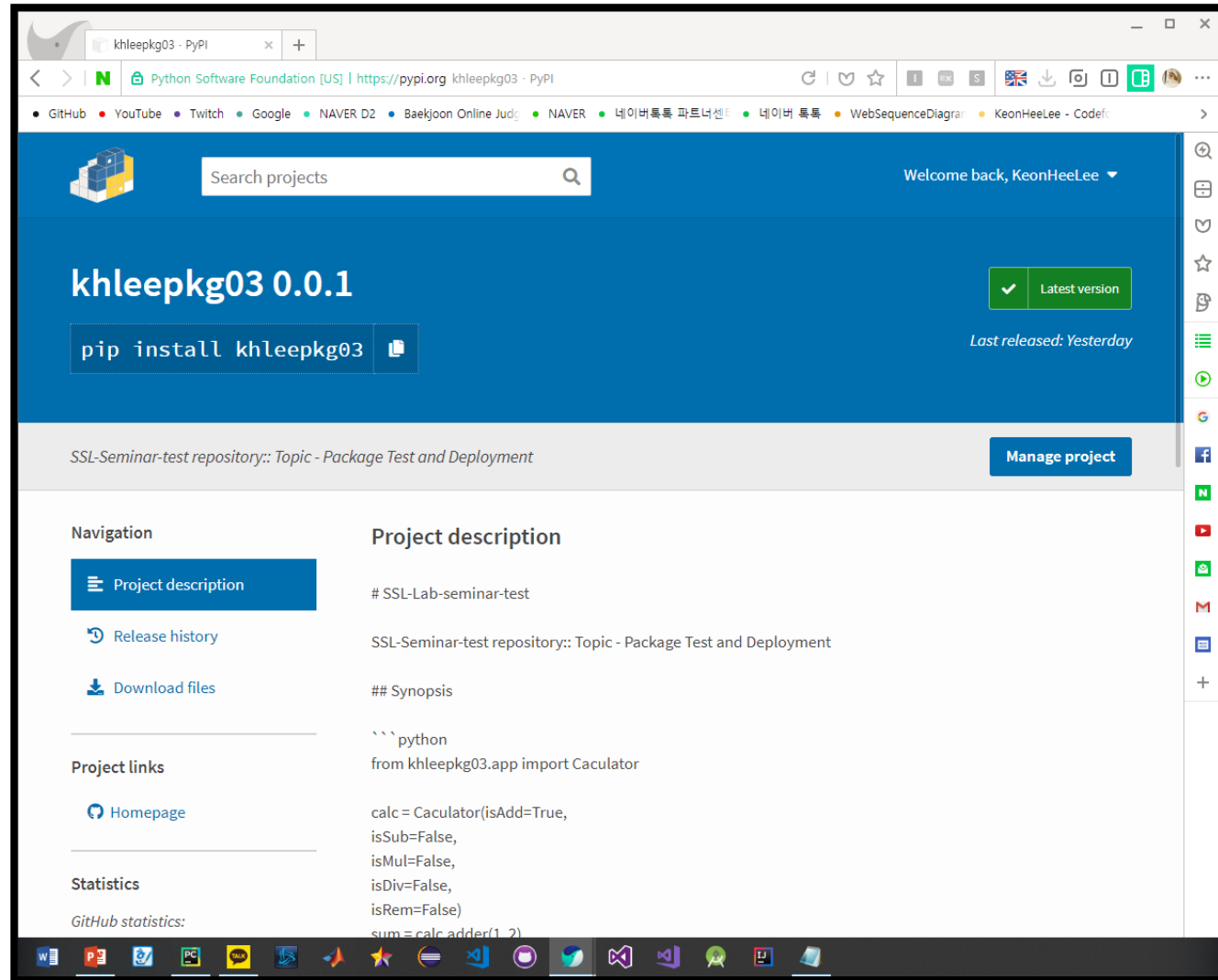
```
$ twine register dist/*
```

```
# twine으로 ".pypirc"에 등록한 유저정보로 dist/*내에 있는 파일 등록
```

```
$ twine upload dist/*
```

```
# pypi에 패키지 업로드하기
```

Pypi에 패키지 등록된 것을 확인



배포한 패키지를 사용해보자

```
C:\#Python35#Scripts>pip install khleepkg03
Collecting khleepkg03
  Using cached https://files.pythonhosted.org/packages/3d/63/06bc9de86e1f2a4663914c62f688500f4307fc1edc69b3f3c9d37ea04527/khleepkg03-0.0.1-py3-none-any.whl
Installing collected packages: khleepkg03
Successfully installed khleepkg03-0.0.1
You are using pip version 10.0.1, however version 18.0 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.
```

```
1  from khleepkg03.app import Caculator
2
3  calc = Caculator(True, True, True, True, True)
4  print("add = %d" %calc.adder(1,2))
5
6  div, rem = calc.division(4, 3)
7  print("div= %d, rem= %d" %(div, rem))
```

Run test

```
C:\#Python35#python.exe C:/Python/testPkg/test.py
add = 3
div= 1, rem= 1
Process finished with exit code 0
```

- pip을 이용한 설치
- khleepkg03 내의 app모듈 사용

참고 자료

- Pypi using guide
<https://packaging.python.org/>
- Travis-CI documentation
<https://docs.travis-ci.com/>
- Coveralls Document
<https://docs.coveralls.io/>
- TDD 개발론 참고자료
<https://nesoy.github.io/articles/2017-01/TDD>
- 실습에 사용한 Repository
<https://github.com/KeonHeeLee/ssl-lab-test>
- unittest framework document
<https://docs.python.org/3/library/unittest.html>
- twine documentation
<https://twine.readthedocs.io/en/latest/>