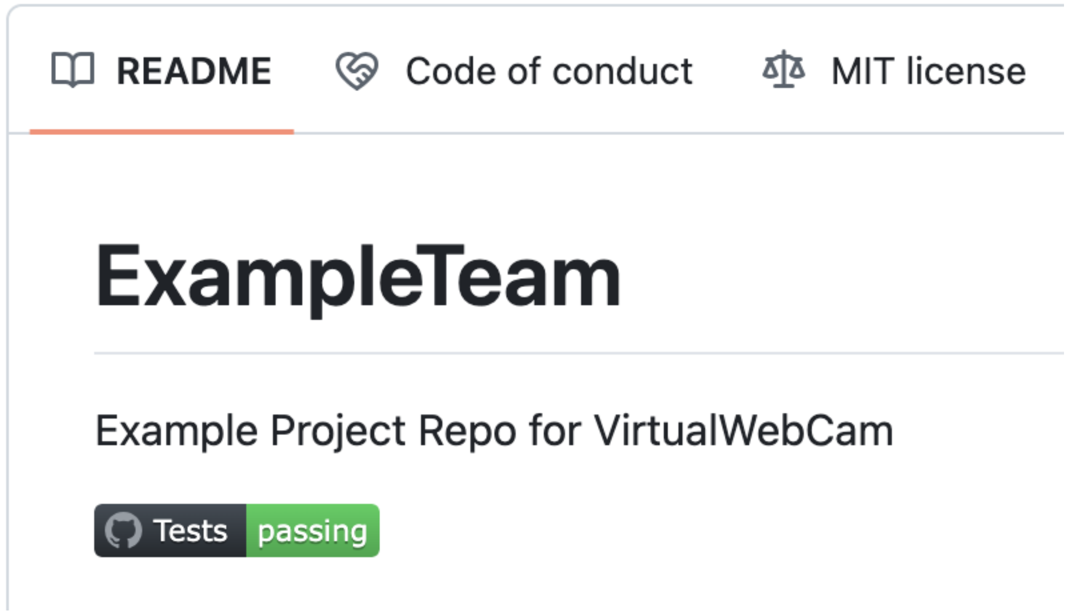
Introduction to Automated Code Testing and GitHub Actions



ExampleTeam

Example Project Repo for VirtualWebCam

This is a good tutorial on YouTube that shows how to setup python tests

This document largely follows this youtube video:

https://www.youtube.com/watch?v=DhUpxWjOhME

Creating a dedicated python environment

```
# create the blank environment.  In this case, I named it .venv
python3 -m venv .venv
```

```
# once you have the environment, you need to activate it by calling
the activate code located in the bin subfolder
source .venv/bin/activate
```

```
# now that the code is activated, any pip installs you make will be
added to this environment
```

```
# Let’s install some of the python code that we will use in this
course.  Using the “==#.#.#” notation specifies a specific version
of the library.

pip3 install opencv_python==4.11.0.86
pip3 install mediapipe==0.10.21
pip3 install numpy
pip3 install matplotlib
```

# Once you have the python install the way that works for your code, you can use the pip freeze command to create a requirements file listing all the libraries you have installed and their exact versions.

```
pip3 freeze requirements.txt
```

```
matplotlib==3.10.1
mediapipe==0.10.21
numpy==1.26.4
opencv-contrib-python==4.11.0.86
opencv-python==4.11.0.86
pillow==11.2.1
```

# Someone else can then install your exact setup using this requirements.txt file

```
pip3 install -r ./requirements.txt
```

Making your code a stand-alone installable python module

Step 1. Structuring the project

```
MyProject                          no __init__.py here
        /src                                        namespace
                /virtualwebcam                       yes __init__.py here
                        __init__.py
                        mycode.py
        /tests
                __init__.py
                test_mycode.py
```

The __init__.py file in a directory tells Python that this is structure of the library. This allows the functions in mycode.py to be called as:

virtualwebcam.mycode.<function>

The __init__.py file just needs to exist, but does not need to have anything in it.

It can have optional info on the version and/or import directives

This allows libraries to have organized structures.

For Python version 3.3 and later, the first level after source defines the namespace and does not include the __init__.py. All subsequent levels do include this.

Making your code a stand-alone installable python module

Step 2. pyproject.toml file

```
MyProject
        /src
                /virtualwebcam
                        __init__.py
                        mycode.py
        /tests
                __init__.py
                test_mycode.py

        /pyproject.toml
```

The pyproject.toml file is used to tell python how to run setup for your program. Here, we will tell python to use the legacy setuptools method, which will use a setup.py file that we will create next.

Later, we will add info for our pytest code into this pyproject.toml file

Contents of pyproject.toml

```
[build-system]
requires = ["setuptools>42.0", "wheel"]
build-backend = "setuptools.build_meta"
```

Making your code a stand-alone installable python module

Step 3. setup.py

```
MyProject
        /src
                /virtualwebcam
                        __init__.py
                        mycode.py
        /tests
                __init__.py
                test_mycode.py

        /pyproject.toml
        /setup.py
```

The pyproject.toml told Python that our code should be installed using the legacy setuptools. This will then look for the file setup.py. This file simply calls setup() from the setuptools library.

The setup( ) function call is now going to look for a setup.cfg file, that we need to create next

Contents of setup.py

```
from setuptools import setup

if __name__ == "__main__":
        setup()
```

Making your code a stand-alone installable python module

Step 4. setup.cfg

```
MyProject
        /src
                /virtualwebcam
                        __init__.py
                        mycode.py
        /tests
                __init__.py
                test_mycode.py

        /pyproject.toml
        /setup.py
        /setup.cfg
```

Note, the requirements.txt was used to setup the python env. The install requirements are what THIS package needs to allow installation,

```
[metadata]
name = virtualwebcam
description = example project for ECE 1390/2390
author = Dr Huppert
license = MIT
license_file = LICENSE.md

platforms = unix, linux, cygwin, osx, win32
classifiers =
        Programming Language :: Python :: 3.12

[options]
packages = virtualwebcam
install_requires =
        matplotlib>=3.0
        mediapipe>=0.10
        numpy>=1.26
        opencv-contrib-python>=4.11
        opencv-python>=4.11
        pillow>=11.2

package_dir =
        =src
```

Making your code a stand-alone installable python module

Step 5. install package step

pip install –e MyProject

```
MyProject
        /src
                /virtualwebcam
                        __init__.py
                        mycode.py
        /tests
                __init__.py
                test_mycode.py

        /pyproject.toml
        /setup.py
        /setup.cfg
```

Note: Using the –e (editable mode) defines the install to Python using links to the original code. This allows you to make edits into the library (e.g. mycode.py) and not have to reinstall the package again.

Adding Python Tests to your code

PYTEST Allows definitions of a set of custom tests for your code.

Step 6. Create tests for PyTest

```
MyProject
        /src
                /virtualwebcam
                        __init__.py
                        mycode.py
        /tests
                __init__.py
                test_mycode.py

        /pyproject.toml
        /setup.py
        /setup.cfg
```

Tests must begin with "test_"

```
import virtualweb
def test_some_test():
        assert (something==1)

def test_some_other_test():
        assert (something==2)
```

Adding Python Tests to your code

PYTEST Allows definitions of a set of custom tests for your code.

Step 6. Add Pytest to our pyproject.toml

```
MyProject
        /src
                /virtualwebcam
                        __init__.py
                        mycode.py
        /tests
                __init__.py
                test_mycode.py

        /pyproject.toml
        /setup.py
        /setup.cfg
```

Add lines:

```
[tool.pytest.ini_options]
testpaths = [
        "tests",
    ]
```

This will search the “tests” folder for any files beginning with “test_” and then run any sub-routines in those files beginning with “test_”

Adding Python Tests to your code

```
.venv/bin/pytest .
========================== test session starts ==========================
platform darwin -- Python 3.12.11, pytest-8.3.5, pluggy-1.6.0
rootdir: ExampleSemesterProjectTemplate
configfile: pyproject.toml
plugins: cov-6.1.1

collected 2 items
            tests/test_webcam.py .        [100%]

========================== 2 passed in 0.03s ==========================
```

Adding Python Tests to your code

MYPY MyPy is a function that checks your code for proper Python language and syntax use based on the PEP 484 standard.

Step 7. Add MyPY to our pyproject.toml

```
MyProject
        /src
                /virtualwebcam
                        __init__.py
                        mycode.py
        /tests
                __init__.py
                test_mycode.py

        /pyproject.toml
        /setup.py
        /setup.cfg
```

Add lines:

```
[tool.mypy]
mypy_path = "src"
check_untyped_defs = true
disallow_any_generics = true
ignore_missing_imports = true
no_implicit_optional = true
show_error_codes = true
strict_equality = true
warn_redundant_casts = true
warn_return_any = true
warn_unreachable = true
warn_unused_configs = true
```

Adding Python Tests to your code

MYPY

MyPy is a function that checks your code for proper Python language and syntax use based on the PEP 484 standard.

What are the PEP standards?

Basically, these are just community agreed upon conventions known as the Python Enhancement Proposals (PEP).

https://peps.python.org/

```
.venv/bin/mypy .
```

```
Success: no issues found in 5 source files
```

Adding Python Tests to your code

FLAKE8 Flake is a linter. Linters check for style violations. Examples are:

- Private attributes all begin with <>.__variable
- Classes all use CamelCase
- Constants all use UPPERCASE

Step 9. Add Flake options to setup.cfg

```
MyProject
        /src
                /virtualwebcam
                        __init__.py
                        mycode.py
        /tests
                __init__.py
                test_mycode.py

        /pyproject.toml
        /setup.py
        /setup.cfg
```

Add lines:

```
[flake8]
max-line-length = 160
exclude = .git,__pycache__,.venv
```

Our code and testing so far

```
git clone https://github.com/SSOE-ECE1390/ExampleSemesterProjectTemplate.git
cd ExampleSemesterProjectTemplate
python3.12 -m venv .venv
source .venv/bin/activate
pip install -r ./requirements_dev.txt
pip install -e .

.venv/bin/pytest .
.venv/bin/mypy .
.venv/bin/flake8 .
```

TOX: Testing your code in different environments

Creates virtual environments and runs
your tests in each environment

Step 10. Add tox.ini file

```
MyProject
        /src
                /virtualwebcam
                        __init__.py
                        mycode.py
        /tests
                __init__.py
                test_mycode.py

        /pyproject.toml
        /setup.py
        /setup.cfg
        /tox.ini
```

Known python environments:
py36,py37, ... py311,py312, py313

Custom python envs

```
[tox]
minversion = 3.12.0
envlist = py312, mypy, flake8
isolated_build = true

[gh-actions]
python = 3.12: py3, mypy, flake8

[testenv]
setenv = PYTHONPATH = {toxinidir}
deps = -r{toxinidir}/requirements_dev.txt
commands = pytest --basetemp={envtmpdir}

[testenv:flake8]
basepython = python3.12
deps = flake8
commands = flake8 src tests

[testenv:mypy]
basepython = python3.12
deps = -r{toxinidir}/requirements_dev.txt
commands = mypy src
```

>> tox

```
py312: install_deps> python -I -m pip install -r /Users/theodorehuppert/Desktop/Teaching2/ECE1390/2025/ExampleSemesterProjectTemplate/requirements_dev.txt
========================================================= test session starts
platform darwin -- Python 3.12.11, pytest-8.3.5, pluggy-1.6.0
cachedir: .tox/py312/.pytest_cache
rootdir: /Users/theodorehuppert/Desktop/Teaching2/ECE1390/2025/ExampleSemesterProjectTemplate
configfile: pyproject.toml
testpaths: tests
plugins: cov-6.1.1
collected 2 items
tests/test_webcam.py ..                                                    [100%]
========================================================= 2 passed in 0.02s
py312: OK ✓ in 1 minute 2.93 seconds


mypy: install_deps> python -I -m pip install -r /Users/theodorehuppert/Desktop/Teaching2/ECE1390/2025/ExampleSemesterProjectTemplate/requirements_dev.txt
mypy: commands[0]> mypy src
Success: no issues found in 2 source files
mypy: OK ✓ in 2 minutes 5.02 seconds


flake8: install_deps> python -I -m pip install flake8
flake8: commands[0]> flake8 src tests


  py312: OK (62.93=setup[61.08]+cmd[1.85] seconds)
  mypy: OK (125.02=setup[60.03]+cmd[64.99] seconds)
  flake8: OK (55.82=setup[54.52]+cmd[1.30] seconds)
  congratulations :) (244.00 seconds)
```

Running tests on GitHub Actions

Step 11. Add .github config files

```
MyProject
        .github/
                /workflows
                        /tests.yml
        /src
                /virtualwebcam
                        __init__.py
                        mycode.py
        /tests
                __init__.py
                test_mycode.py

        /pyproject.toml
        /setup.py
        /setup.cfg
        /tox.ini
```

```
name: Tests
on:
        - push
        - pull_request
jobs:
        test:
                runs-on: ${{ matrix.os }}
                strategy:
                        matrix:
                                os: [ubuntu-latest, windows-latest]
                                python-version: ['3.12’]

                steps:
                - uses: actions/checkout@v2
                - name: Set up Python ${{ matrix.python-version }}
                        uses: actions/setup-python@v2
                        with:
                        python-version: ${{ matrix.python-version }}
                - name: Install dependencies
                        run: |
                                python -m pip install --upgrade pip
                                pip install tox tox-gh-actions
                - name: Test with tox
                        run: tox
```

TOX: Testing your code in different environments

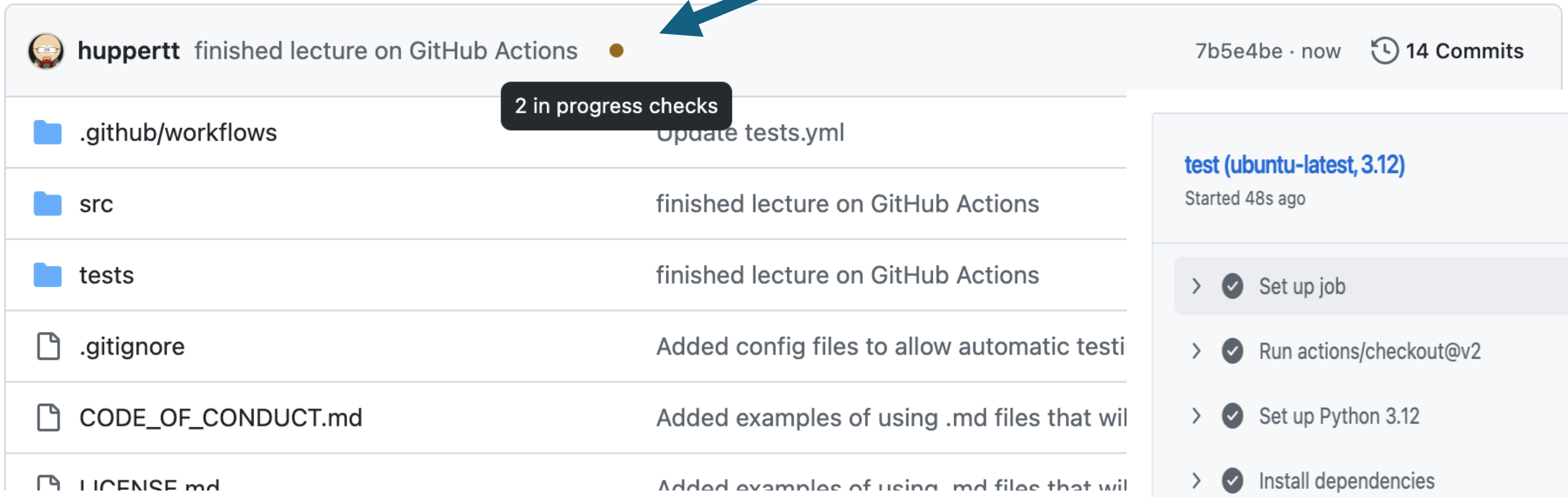
Creates virtual environments and runs your tests in each environment

Step 10. Add tox.ini file

```
MyProject
        /src
                /virtualwebcam
                        __init__.py
                        mycode.py
        /tests
                __init__.py
                test_mycode.py

        /pyproject.toml
        /setup.py
        /setup.cfg
        /tox.ini
```

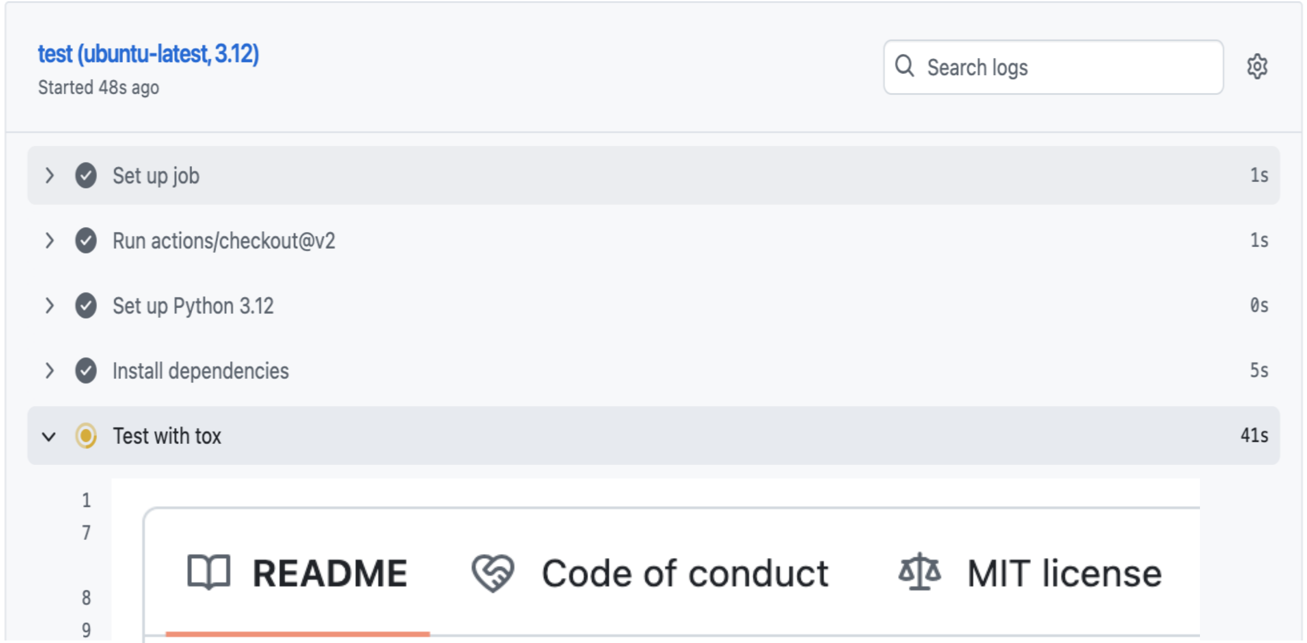
This "translates" between what GitHub actions calls the python envs and what tox needs

```
[tox]
minversion = 3.12.0
envlist = py312, mypy, flake8
isolated_build = true

[gh-actions]
python = 3.12: py3, mypy, flake8

[testenv]
setenv = PYTHONPATH = {toxinidir}
deps = -r{toxinidir}/requirements_dev.txt
commands = pytest --basetemp={envtmpdir}

[testenv:flake8]
basepython = python3.12
deps = flake8
commands = flake8 src tests

[testenv:mypy]
basepython = python3.12
deps = -r{toxinidir}/requirements_dev.txt
commands = mypy src
```

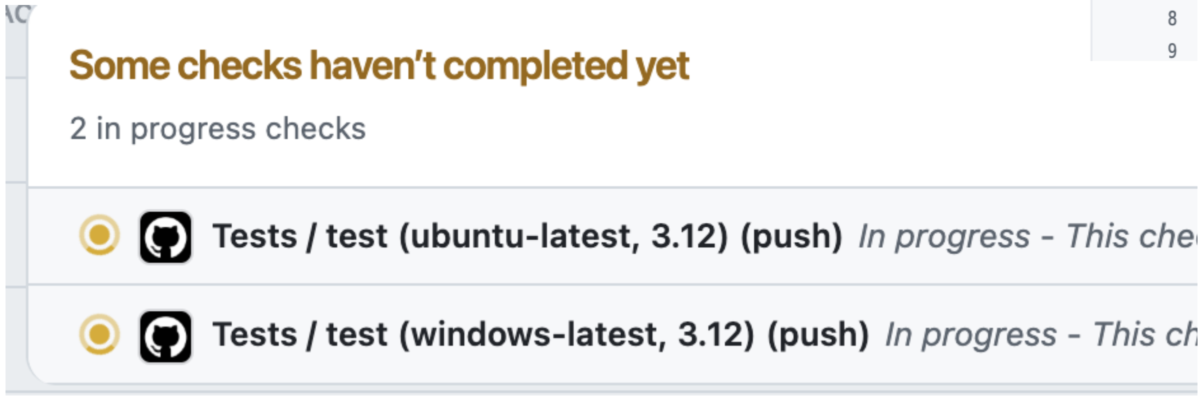

huppertt finished lecture on GitHub Actions ● 7b5e4be · now 14 Commits

2 in progress checks

.github/workflows	Update tests.yml
src	finished lecture on GitHub Actions
tests	finished lecture on GitHub Actions
.gitignore	Added config files to allow automatic testi
CODE_OF_CONDUCT.md	Added examples of using .md files that wil
LICENSE.md	Added examples of using .md files that wil


test (ubuntu-latest, 3.12)
Started 48s ago

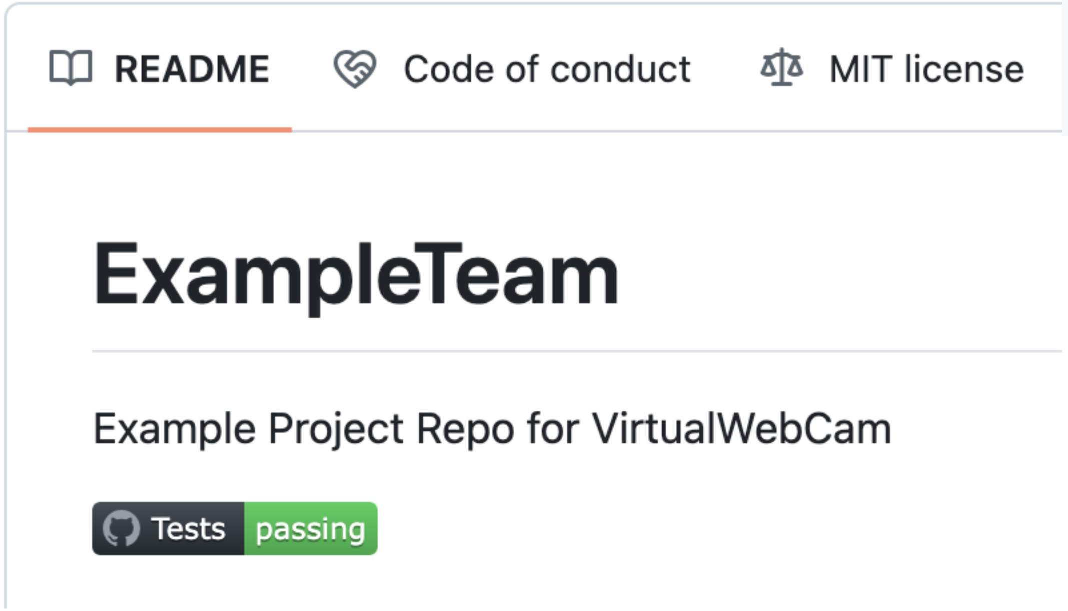
Set up job	1s
Run actions/checkout@v2	1s
Set up Python 3.12	0s
Install dependencies	5s
Test with tox	41s


Some checks haven't completed yet

2 in progress checks

Tests / test (ubuntu-latest, 3.12) (push) *In progress - This che*

Tests / test (windows-latest, 3.12) (push) *In progress - This ch*



ExampleTeam

Example Project Repo for VirtualWebCam