Run Unit Tests passing

What need to know about GitHub Actions?

Refer to the file which is going to discussed in detail.

Step-by-step guide

`name`: This is just a human-readable name for the workflow. It can be seen as the workflow title in GitHub Actions.

```
name: Run Unit Tests
```

`on`: – When should this workflow run?

```
on:
  push:
    branches:
      - '**'
  pull_request:
```

- `push`: Triggers when someone pushes code to any branch ('**' means “all branches”).
- `pull_request`: Triggers when a pull request is created or updated (good for testing before merging code).

This ensures tests run automatically every time someone pushes or opens a PR.

`jobs`: – A job is a set of steps that run on a GitHub server

```
jobs:
  test:
    runs-on: ubuntu-latest
```

- The job is called `test`.
- `runs-on: ubuntu-latest`: The virtual machine (runner) GitHub will use. This runner comes with Python, Git, pip, and other dependencies.

`steps`: – Actions to perform inside this job

1. Checkout the repo

```
- name: Checkout the repository
  uses: actions/checkout@v3
```

This downloads GitHub repository into the runner so it can access your code.

2. Set up Python

```
- name: Set up Python
  uses: actions/setup-python@v4
  with:
      python-version: '3.10'
```

- This tells GitHub to install and use `Python 3.10` for all following steps.
- It can be changed to another version if needed.

3. Install dependencies

```
- name: Install dependencies
  run: |
      python -m pip install --upgrade pip
      pip install -r requirements.txt
```

- Upgrades pip
- Installs everything listed in requirements.txt — such as pytest
- Without this step, tests might fail because libraries like pytest will not be available.

4. Run the tests

```
- name: Run tests
  run: |
      export PYTHONPATH=.
      pytest
```

This does the actual testing:

- `export PYTHONPATH=.` tells Python to include current folder in the import path, so it can find imported modules/pakages.
- `pytest` runs all the test files (files like `test_*.py` inside the `test/` folder).

Conclusion

When someone pushes code or makes a PR:

1. GitHub creates a VM with Ubuntu + Python 3.10
2. It clones code
3. Installs dependencies
4. Runs tests using pytest
5. If tests pass: green checkmark ✅
6. If tests fail: red cross and logs showing what failed ❌

Code Coverage Analysis Tool (Codecov)

> How much of the code is being executed when tests run.

It is important to know:

- Did all functions tested?
- Did it reach to both `if` and `else` statement?
- Are **error cases** being tested?

It gives a quantitative measure of the "tested" code is.

Steps to implement

Install `pytest-cov`

Add it to `requirements.txt`

```
pytest
pytest-cov
```

Or install it manually:

```
pip install pytest-cov
```

Update GitHub Actions workflow to collect coverage data

```
- name: Run tests with coverage
  run: |
    export PYTHONPATH=.
    pytest --cov=my_package --cov-report=xml
```

Replace `my_package` with the actual folder where the source code lives. In this tutorial, `calculator.py` is in the current directory so using `--cov=.`

This creates a file called `coverage.xml` that Codecov will use.

Sign up at Codecov

1. Log in with your GitHub account
2. Find the repo
3. Authorize access