Unit Testing – Exercise (*Course 67778, 2023*)

Contents

Optional: Introduction to Unit Tests

- בדיקות יחידה – ויקיפדיה (wikipedia.org)
- לבדיקות אינטגרציה (integration tests) על ההבדל בין בדיקות יחידה (unit tests) - אנשים ומחשבים
- Full pytest documentation — pytest documentation

Cloning the T-Tweak Project

You will need:

- Python installed. Any version large or equal to 3.8 should work.
- git installed, and access to clone from github.

Important: These instructions are written for Windows. T-Tweak will work on Linux as well, but you may need to adapt the commands to it.

Run these commands:

- `cd <YOUR_FOLDER>`
- `git clone https://github.com/sgershon/t-tweak.git`

This will create a directory called t-tweak with the t-tweak code.

Alternative: If you can't or don't want to clone or have any other git issue, you can download the project as a zip file from https://github.com/sgershon/t-tweak/archive/refs/heads/main.zip. Extract the content of `t-tweak-main` (*inside the zip*) into a folder called `t-tweak`.

Check you have a bunch of files:

- Make sure you are in the t-tweak folder (`cd t-tweak`)
- `dir`

Optional: We recommend running t-tweak from a virtual environment:

- Make sure you are in the t-tweak folder (`cd t-tweak`)
- `python -m venv tt_venv`
- `.\tt_venv\Scripts\activate.bat`

Install all the necessary python modules:

- `pip install -r requirements.txt`

Running Pytest on T-Tweak

- Make sure you are in the t-tweak folder (`cd t-tweak`)
- `pytest`

You should see the short results of pytest. To see more details, use:

- `pytest -v`

Calculating Code Coverage on T-Tweak

- Make sure you are in the t-tweak folder (`cd t-tweak`)
- `coverage run -m pytest`
- `coverage report`

To see pretty details:

- `coverage html`
- `.\htmlcov\index.html`

Optional: Running the server locally

You can run your own copy of the t-tweak API server with these commands:

- Make sure you are in the t-tweak folder (`cd t-tweak`)
- `uvicorn main:app`

And then open this link: http://127.0.0.1:8000/docs

How to understand T-Tweak

Important files on t-tweak's code:

- Most of the code is in **`main.py`**.
- The functions that represent external behavior (*good candidates for monkeypatching*) are at **`extra.py`**.
- The unit tests are in the **`tests`** directory, **`test_unit.py`**.

- The unit tests inside **`tests/test_unit.py`** include comment on what they are demonstrating.

Exercise: Adding your own Unit Tests

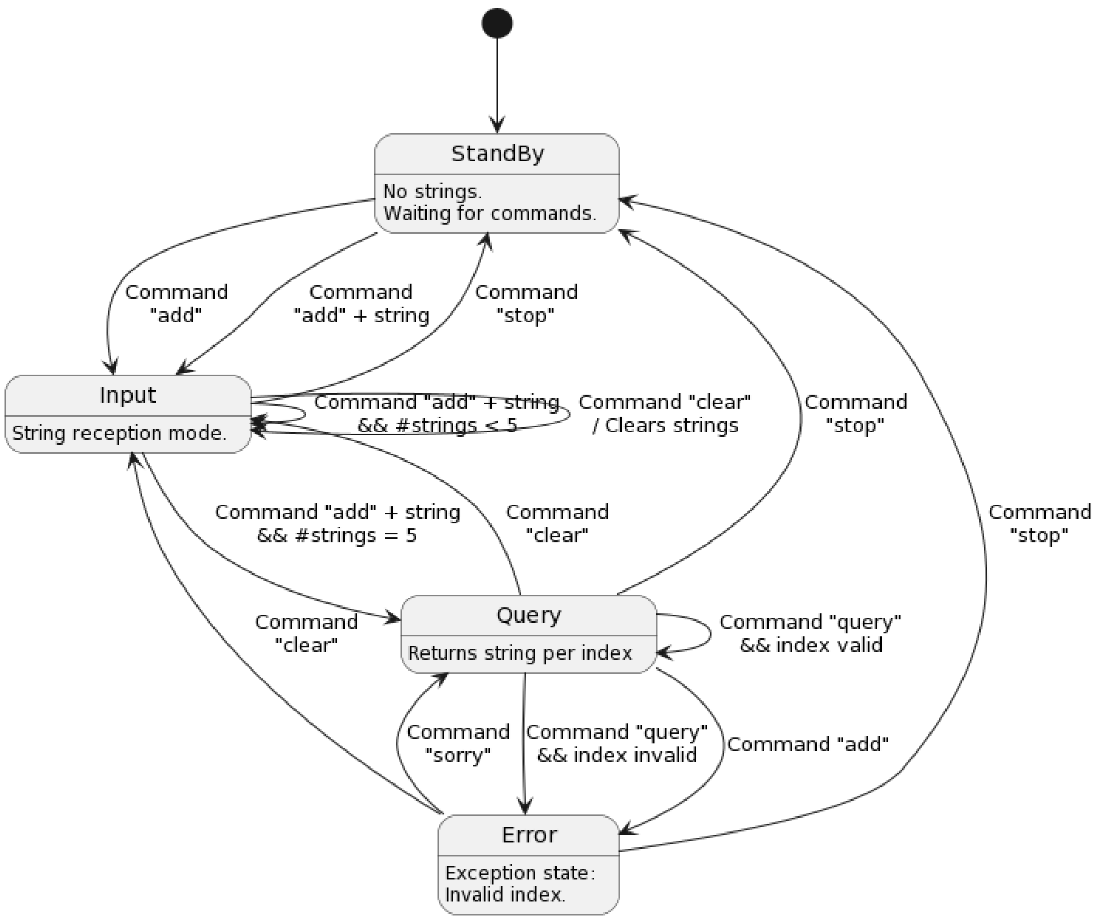
The tests in **`tests/test_unit.py`** are numbered.

- Tests 01 to 09 are already written and are provided as examples.
- Tests 10 to 14 are empty and need to be prepared by you. The requirements (*and implementation hints!*) for these tests are described in the python file itself.

- For Test 10, which tests the password, you will need to test all the equivalence partitions. Below are all the partitions, and the values that cover them:

Equivalence Class	**Test**
Length = 0	*no text, just an empty string*
0 < length <=2	`1`
Length > 20	`Too-L0ng-4-the-allowed-input-length`
Password is "root"	`root`
Password is "password"	`password`
Password is "admin"	`admin`
2 < length < 12, lowcase, upcase, digit	`G00dShort`
2 < length < 12, all same char	`gggggggg`
4 < length < 12, no uppercase	`gfs98ased`
4 < length < 12, no lowercase	`NOT1LOWCASE`
4 < length < 12, no digit	`noDIGIT`
6 < length < 12, lowcase only	`lowcaseonly`
6 < length < 12, upcase only	`UPCASEONLY`
6 < length < 12, digit only	`1234567`
6 < length < 12, special chars only	`%@#$^&*`
12 < length, lowcase, upcase, digit	`L0ng-And-G00d`
12 < length, all same char	`Aaaaaaaaaaaaaaaaa`
12 < length, no uppercase	`l0ngbutnouppercase`
12 < length, no lowercase	`LONG1BUTNOLOWCASE`
12 < length, no digit	`LongButNotOneDigit`
12 < length, lowcase only	`Longbutonlylowecase`
12 < length, upcase only	`LONGBUTONLYUPCASE`
12 < length, digit only	`12345678901234`
12 < length, special chars only	`%@#$%$%$%$%$%$%$`

- To help you with Test 14, which tests the storage function flow, we provide here a state-machine representing the code. Covering the State Machine test cases will help you reach 100% statement coverage:



Exercise Submission

Submit a python file with the unit_tests, using the name **`test_<your-ID>.py`** (*the word "test", an underscore, your ID, the* **`.py`** *extension*). Deadline for the submission is June 29th.

Exercise check

We will test the exercise by running "pytest" and "coverage", against source code that expose bugs and coverage relevant to the tests requested. We will use the same commands above:

- `pytest -v`
- `coverage run -m pytest`

We recommend you run these commands too in your system to check your answers before submitting, to make sure the file is properly formatted. In your test, the unit tests should pass, and coverage for storage section should be 100%.

Good luck 😊 !