Regex Locator – Adi Zavalkovsky

Regex Locator – A project aimed at running unit tests on a script (main.py) whose objective is to look for regex in files.
The tests were developed using 'pytest'.
The tests are ran in a docker container.

Usage

'make' – runs two targets found in the Makefile, responsible for building the container's image and running a container. *Should be ran first time.

'make run' – updates code files in the container and re-runs the tests. *Relieves the need to rebuild the image.

Regex Locator (main.py) is a script, which looks for a regex in files using python.

Project Structure

```
Regex_locator-main
|   Dockerfile
|   Makefile
|   README.md
|   requirements.txt
|
|
\---py_code
    |   main.py
    |   run_tests.py
    |   __init__.py
    |   test.log
    |
    \---tests
        |   test_cli_tests.py
        |   test_file_amount.py
        |   test_file_size.py
        |   __init__.py
        |
        \---test_files
                basic.txt
                __init__.py
```

Dockerfile

The project's image is built by a standard Dockerfile.
It's parent image is 'python:3.7.1' available on DockerHub's public repo.
The Dockerfile copies the project's requirements (python modules) file, and install the module using RUN.
The Dockerfile also specifies the creation of a project folder, and creates a mountpoint using VOLUME.

Makefile

The project's container is ran using a Makefile.
The Makefile has two targets -
build – responsible for building the container off the Dockerfile.
run – responsible for removing old containers, and run a new container, while mounting the folder containing the python code to the container.

requirements.txt

Python modules required for running the script and testing.
Modules used –
colorama – Highlighting matches found by the sciprt.
lorem-text – Generating ran
pytest – Testing environment.

py_code/main.py

The project's main script, responsible for the desired functionality – scanning files for a regex.

There are two high level functions - run(), main(), and one Class - Locator.
When main.py is ran through a cli, run() is the first function to run -
it's responsibility is to take in user input and pass it to main(),
which is responsible for initializing an instance of Locator(), pass the user's arguments and run a scan.
Locator() prints the scan's results to the screen.

The distinction between run() and main() is for testing purposes.
While run() is used for collecting user input, main() is responsible for using that input.
When testing the script - most tests run on main().

py_code/run_tests.py

Runs tests and logs them to test.log

py_code/test.log

Log file. Updated after each test.

Test Files

File	Test Name	Description
test_cli_tests.py	test_normal()	Make sure no error is thrown on valid input
	Test_missing_exp()	Make sure an error is thrown when not passing a regex
	test_optional_args()	Make sure no error is thrown when optional arguments aren't passed
	test_missing_file()	Make sure an error is thrown when passing a nonexistent file
	test_no_file()	Make sure an error is thrown when no file is passed
test_file_amount.py	test_file_amount	Determine whether the amount of files passed to the Locator() class affect the accuaracy of the classes' scan. Multiple tests are ran, each with a different amount of files.
test_file_size.py	test_file_size	Determine whether different file sizes affect the accuracy of the Locator()'s scan. Multiple tests are ran, each with a different amount of files.
test_file_name_length.py	test_file_name_length	Determine whether different file name length affect the accuracy of the classes' scan. Multiple tests are ran, each with a different amount of files.