ICT GROUP Strypes

The Full Monty Python Training

June 2021

Mentors

Angel Pashev - Some Python experience, mainly on automations.

Ivan Dinev - 5 years of Python experience, mainly working on system tools - backup/restore/patch.

Stoyan Radev - 6 years of Python experience, mainly working on automation, system management and configuration, data science and machine learning.

Training

Web scraper - automatically gather info from selected websites (blogs):

1. Develop a scraper using a **T**est **D**riven **D**evelopment process.

1. Process the data for subsequent usage (storage/access/search).

1. Present the data through a simple frontend.

Training

3 weeks - the scraper is divided into 3 tasks - scraper, data process, frontend.

- The new task will be given out each week.
- Taks info will be sent each week, before the start of the task.

Interaction

1 weekly meeting - 1 hour per team.

- Teams present task, answer questions from mentors.
- Mentors answer/discuss technical questions (prepare in advance).
- Trainees send code whenever ready (or better yet, link to a git repo).

Project requirements

Create a web scraper :

1. Web scraping:
    - The scrapper must be able to collect blog posts from a predefined blog
    - The latest 20 blog posts must be collected into a chosen data structure
    - The collected blogs must be written in a file with a chosen format
2. Data processing:
    - The data must be read from the file in which it has been stored in phase one
    - The data must be formatted/reshaped/simplified/reduced
    - The formatted data must be stored in a new file
3. Web interface:
    - A web instance must be created using bottle/flask/django
    - The formatted data must be represented in the web instance
    - The format of the representation is not predefined
4. Overall requirements:
    - Step 1 and 2 must be written with a TDD approach using pytest
    - Overall coverage for these two stages >= 90%

Web scraping part

- The web scraping part must be implemented using OOP
- Mandatory packages are:
  - request - built-in package used for performing connections to a web instance
    Example:

```
from urllib.request import Request, urlopen
req = Request('https://usefull.blog.net')
webpage = urlopen(req).read()
```

Documentation: https://pypi.org/project/requests/

  - BeautifulSoup4 - non-built-in package for HTML/XML parsing
    installation: pip install beautifulsoup4
    Example:

```
from bs4 import BeautifulSoup
soup = BeautifulSoup(webpage, 'html.parser')
soup.find_all('a', href=True)
```

Documentation: https://www.crummy.com/software/BeautifulSoup/bs4/doc/

- The main blog page and sub-pages are containing links to the blog pages themselves
- Each blog post must be collected separately from its own page
- The file format in which the data will be stored is not predefined

Blogs to be scrapped

List of blogs (chose one) :

- https://blog.bozho.net/
- https://igicheva.wordpress.com/
- https://www.travelsmart.bg/
- https://pateshestvenik.com/
- https://az-moga.com/
- Have a favorite blog - suggest it!

Project structure

Example project structure :

```
blog-web-scraper/                          # The main directory of the project
        |--- main.py                       # The main executable of the project
        |--- module/                       # Directory containing all modules/libraries of the project
                |--- data_formatter.py     # Module for the data formatting
                |--- __init__.py           # Marks the module dir as a python module
                |--- web_scraper.py        # Module for web scrapping
        |--- README.md                     # Descriptive markdown file (documentation)
        |--- requirements.txt              # File that stores the requirements (non-built-in modules)
        |--- .coveragerc                   # Config file for pytest coverage
        |--- .gitignore                    # Git blacklist file
        |--- test/                         # Folder for all of the tests
                |--- unit_tests/           # Folder for all unit tests
                |--- conftest.py           # Pytest specific file
                |--- test_data_formatter.py  # Unit tests for the data formatter
                |--- test_web_scraper.py   # Unit tests for the web scrapper
```

Git repository containing the blueprint :

https://github.com/radevsto/blog-web-scrapper-blueprint

A bit about Pytest

- Installation:
  pip install pytest pytest-cov
- Pytest is a non-built-in python module that provides more functionality than the built-in unittest module.
- conftest.py - is a file in which is automatically available (like imported) in all test_* files.
- @pytest.fixture() - is a decorator that marks that given function can be directly used as a parameter of a test function.
- @pytest.mark.parametrize() - is used to run the same test multiple times with different input parameters
- Otherwise the same functionality and approach as from the unittest modules are available.
- .coveragerc file will be present in the project blueprint that we provide.
- Run the tests and check for coverage:
  python -m pytest --cov-config=.coveragerc --cov-report term-missing --cov-report html:coverage --cov-fail-under=90 --cov=. test/unit_tests
- Examples usage for pytest will be provided in the project blueprint