FRI:DAY

FRIDAY code challnge

Street Address Automatic Parser

Jon Fernández Bedia

_

Table of contents

1. FRIDAY_code_challenge

Code challenge for FRIDAY enterprise

1.1 Requisites

- pipenv
- If you are using Visual Studio Code, you need to install Python extension

1.2 Start

`bash`
`make start`

1.2.1 Git Hooks

- **Pre-commit**: before each commit the `make run-tests` command will be executed.

Note: If you want to ignore the pre-commit hook, add the flag `--no-verify` to the commit command.

1.3 Install dependencies

`bash`
`make install-deps`

1.4 Add/remove dependency

```bash make install dep={{dependency}} ver={{dependency_version}} # Example: make install dep=requests ver=2.26.0
make install-dev dep={{dependency}} ver={{dependency_version}} # Example: make install-dev dep=requests ver=2.26.0

make uninstall dep={{dependency}} # Example: make uninstall dep=requests ```

## 1.5 Run

Runs main application, located in app/main.py . `bash`
`make run`

## 1.6 Run tests

Run all tests defined in app/services/tests/test.py. `bash`
`make run-tests`

## 1.7 Format code

Orders dependencies with isort and formats the code using the requirementes defined on pyproject.toml.

`bash`
`make format`
```

1.8 Check format

```bash
make check-format
```

1.9 Check types

```bash
make check-types
```

2. Modus operandi

1. Read statement
2. Free thinking:
    - First challenge looks easy, thinking about split() / strip().
    - Second challenge also easy, read until a numeric is found.
    - Third challenge looks like it needs another field such as postal code or an IA app to identify the country so as to know how to split streets or maybe an IA that learns how to split streets.
    - Do I need a country, street and number database to cover all cases?
3. Looks like regex is needed for the second challenge, did not think about german dictionary, is not going to be easy.
    - Nice link!
    - Nice link 2! Regex Example
    - Need to try to understand how does regex exactly works and adapt to my needs.
4. We found a good regex that works 7/9 of the times (all streets but Calle 39 No 1540) , we are going to try another kind of process to parse streets. `"^(\b\D+\b)?\s*(\b.*?\d.*?\b)\s*(\b\D+\b)?$"`
5. We achieve to parse the last street using: `(.*?)\s*(\d+(?:[/-]\d+)?)?$`
6. We found an state-of-the-art library for parsing multinational street addresses using deep learning, let's see how it works.
7. We read again the challenge statement and as we where blocked with 8/9 parsed streets, we decide to start using TDD and use as many python best practices as we can, incorporating the following concepts:
8. Start developing following TDD, this is, start using automated unit tests to drive the development of our code.
9. Add code formatting frameworks such as Black, Flake8 and Isort and automatizing code format checks whenever a commit was done.
10. Add Makefile so it simplifies code running, testing and formatting no matter what IDE is being used.
11. Add docs folder and organizing better documentation. Also adding testing and coverage reports into the documentation.

3. Faced problems

3.1 Multiple SSH keys on same machine

1. To clone and interact with another github account, just create a new ssh-key:

```
ssh-keygen -t rsa
```

1. Save it on **/home/jon/.ssh/** with a different name if one already exists.

```
ssh-add ~/.ssh/<new_private_key_file>
```

1. Copy public key and paste on Github>Settings>SSH and GPG keys
2. Add configuration on **/home/jon/.ssh/config** `Host <alias_name>`

    ```
    Hostname github.com
    User git
    IdentityFile ~/.ssh/<new_private_key_file>
    IdentitiesOnly yes
    ```

3. Finally clone repository with `git clone <alias_name>:<github_name>/<proyect_name>.git`

3.2 REGEX

We need to learn what was regex from the very beginning, we find some useful websites as that given a regex explain what was deifning each character.

3.3 Deepparse

We try a pre-trained machine learning classification method to see if we get better results than with the regex method, we need to understand how to extract data from the output returned by the model and which models where available using the library and which one was better for this purpose. As if we were not worried about cpu limitations we used 'best' model.

```
{bibliography}
    author = {Marouane Yassine and David Beauchemin},
    title  = {{Deepparse: A State-Of-The-Art Deep Learning Multinational Addresses Parser}},
    year   = {2020},
    note   = [url](https://deepparse.org)
```

3.4 Testing

We discovered helpful documentation about python testing. We about fixtures as we are not used to make tests. Although we knew about TDD methodology.

Also, we modify the output of tests from a boolean type to a comparison between actual processed streets and expected streets, as a boolean gives no information about which streets are well parsed and which not. We did not find out a good library for asserting json lists so we convert them into python lists first and so we could make use of unittest TestCase library.

3.5 Pipenv

Somehow, **pipenv run** commands was not working, giving us the following error: `Traceback (most recent call last):`

```
File "/usr/bin/pipenv", line 11, in <module>
  load_entry_point('pipenv==11.9.0', 'console_scripts', 'pipenv')()
File "/usr/lib/python3/dist-packages/pipenv/vendor/click/core.py", line 722, in __call__
  return self.main(*args, **kwargs)
File "/usr/lib/python3/dist-packages/pipenv/vendor/click/core.py", line 697, in main
  rv = self.invoke(ctx)
File "/usr/lib/python3/dist-packages/pipenv/vendor/click/core.py", line 1066, in invoke
  return _process_result(sub_ctx.command.invoke(sub_ctx))
File "/usr/lib/python3/dist-packages/pipenv/vendor/click/core.py", line 895, in invoke
  return ctx.invoke(self.callback, **ctx.params)
File "/usr/lib/python3/dist-packages/pipenv/vendor/click/core.py", line 535, in invoke
  return callback(*args, **kwargs)
File "/usr/lib/python3/dist-packages/pipenv/cli.py", line 602, in run
  core.do_run(command=command, args=args, three=three, python=python)
File "/usr/lib/python3/dist-packages/pipenv/core.py", line 2200, in do_run
  command = ' '.join(project.scripts[command])
File "/usr/lib/python3/dist-packages/pipenv/project.py", line 374, in scripts
  scripts[k] = shlex.split(v, posix=True)
File "/usr/lib/python3.8/shlex.py", line 311, in split
  return list(lex)
File "/usr/lib/python3.8/shlex.py", line 300, in __next__
  token = self.get_token()
File "/usr/lib/python3.8/shlex.py", line 109, in get_token
  raw = self.read_token()
File "/usr/lib/python3.8/shlex.py", line 140, in read_token
```

```
  nextchar = self.instream.read(1)
AttributeError: 'list' object has no attribute 'read'
```

We fix it reinstalling pipenv:

```
sudo apt-get remove pipenv
pip3 install pipenv
```

Flake8 Makes the following warning: `./app/services/middle_processor.py:14:39: W605 invalid escape sequence '\d'` It seems that Python 3 interprets string literals as Unicode strings, and therefore our \d is treated as an escaped Unicode character. As we are following Flake8 rules regarding W605 to avoid the warning we add --ignore flag when checking flake8 format:

`check-flake8 = "python -m flake8 --exclude=.venv/ --ignore=W605"` Even so, a warning is raised when we run tests with pytest.