How to start a new Python project

Create a folder to organize your coding projects

If you don't have a folder where you keep your coding projects, it's a good idea to create one to help you keep all of your coding projects organized.

I recommend creating a folder named `dev` in your user home folder, and a `projects` folder within the `dev` folder. You can do that with the following commands:

```
cd
mkdir -p ~/dev/projects
cd ~/dev/projects
```

`cd` changes to your user home directory
`~` is a shortcut referring to your user home directory
`mkdir -p ~/dev/projects` creates folders, and `-p` creates all folders in the chain that don't already exist
`cd ~/dev/projects` changes to your projects folder

Create a new project

1. `cd` into your projects folder
2. create a new folder for your project and `cd` into it. For example:

```
mkdir my-project
cd my-project
```

3. run `git init` to initialize git

4. add files:

    - .gitignore

    ```
    .pytest_cache
    .venv
    *.db
    *.log
    ```

    - README.md - instructions on how to use the project

- LICENSE - could be anything - I usually use the Open Source MIT license (https://opensource.org/licenses/MIT)
- .editorconfig

```
# This file is for unifying the coding style for different editors and IDEs
# editorconfig.org

root = true

[*]
end_of_line = lf
charset = utf-8
insert_final_newline = true
trim_trailing_whitespace = true
indent_style = space
indent_size = 4

[Makefile]
indent_style = tab

[{*.md,*.json}]
max_line_length = null
```

- .flake8

```
[flake8]
max-line-length = 88
per-file-ignores = __init__.py:F401
ignore = E501 # exceeded max line length, instance has no member
```

- .coveragerc

```
[run]
branch=True

[report]
# Regexes for lines to exclude from consideration
exclude_lines =
    # Have to re-enable the standard pragma
    pragma: no cover
```

```
    # Don't complain about missing debug-only code:
    def __repr__
    if self\.debug

    # Don't complain if tests don't hit defensive assertion code:
    raise AssertionError
    raise NotImplementedError

    # Don't complain if non-runnable code isn't run:
    if 0:
    if __name__ == .__main__.:

ignore_errors = True

omit = .venv/*,tests/*,coverage/*,data/*,docs/*,rest-client/*

[html]
directory = coverage
```

- Pipfile

```
[[source]]
url = "https://pypi.org/simple"
verify_ssl = true
name = "pypi"

[packages]

[dev-packages]
black = "*"
flake8 = "*"
isort = "*"
pytest = "*"
pytest-cov = "*"
python-dotenv = "*"
rope = "*"

[pipenv]
allow_prereleases = true
```

```
[requires]
python_version = "3.9"
```

- conftest.py - the file is needed for unit tests, but it doesn't have to have anything in it. Probably a bug. Might be different by the time you read this.
- app.py - your application entry point

```
def get_greeting():
    return "hello world"

if __name__ == "__main__":
    greeting = get_greeting()
    print(greeting)
```

- test_app.py

```
from app import get_greeting

def test_get_greeting():
    greeting = get_greeting()
    assert greeting == "hello world"
```

Then open a terminal and initialize your project's virtual environment with

`pipenv shell`

Tell VS Code to use the python interpreter from the .venv in your project folder by clicking on Python in left side of the the status bar at the bottom of the screen and follow the prompts from VS Code.

Install your project dependencies using

`pipenv install`
`pipenv install --dev`

Then run the app with

`python app.py`

To test the app run:

`pytest`

To deactivate the virtual environment press CTRL+C to stop your application (if it is still running), and then type:

```
exit
```