CONTRIBUTION GUIDE

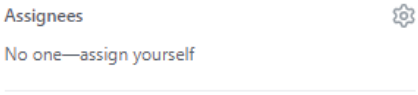
This guide lays out the whole workflow for writing rules.

1. SELECT A RULE (OR RULES) FROM THE ISSUES LIST AND ASSIGN YOURSELF

- Issue list is here: https://github.com/SocialFinanceDigitalLabs/quality-lac-data-beta-validator/issues
- To begin with, stick to 'simple-rule' tagged rules, e.g.

Error code 635 - There are entries for date of order and local authority code where previous permanence option was arranged but previous permanence code is Z1. simple-rule validation-rule

- Begin with one rule only – when you are used to the workflow you can choose two or three.
- Read the description and make sure the rule makes sense and DfE coding is clear and sensible (this is not always true!)
- Make sure to click "Assign yourself" on the rule so nobody else takes it.

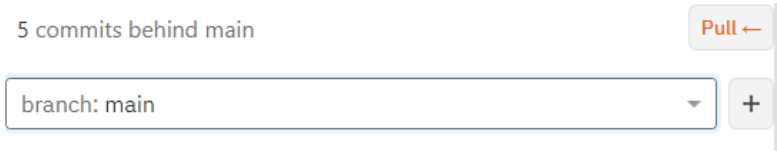


2. RUN ON REPL.IT (EITHER IN NEW OR EXISTING REPL)

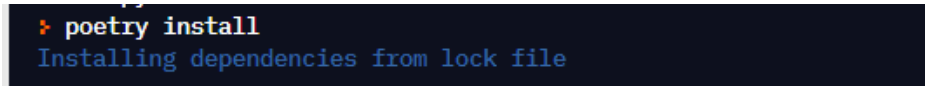
You can either click **Run on Repl.it** to create a new Repl to work in, or return to an existing Repl you have worked in before.

If you are not using a new Repl, before you start

1. Go to version control.
2. Switch branch to main branch.
3. Make sure you are up-to-date with main – if not, there should be a "Pull" button – click this.



Then run **poetry install** in the Console tab to install dependencies. You can continue working while this runs initially (will take 2-3 mins).



3. CREATE A BRANCH

- Under version control, make sure the main branch is selected.
- Press the + button to make a new branch from the main branch.
- Give the branch a unique name (good practice is "{issue number}-validate-{error code}")

4. ADD PLACEHOLDER RULE FUNCTION

First, add a placeholder function to the **validators.py** file.

- The rule function should be named “validate_{error code}”.
- It should begin with an ErrorDefinition, which copies information from the issue and describes the affected fields.
- It should also define an internal _validate function. This is a placeholder for now.
- It should then return the ErrorDefinition object, and the validate function.

For reference, the placeholder can look something like:

```
def validate_{error code goes here}():
    error = ErrorDefinition(
         code= # code goes here,
         description=# description goes here,
         affected_fields=[# list of affected fields goes here],
    )

    def _validate(dfs):
        return {}

    return error, _validate
```

5. ADD RULE TO CONFIGURATION

Then add the error to the configuration (**config.py**) under configured_errors.

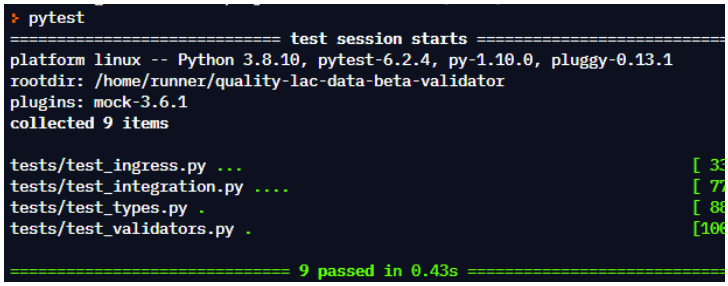
```
"""
List of all configured errors for validation.

These all should have return type (Error, Dict[str, DataFrame])
"""
configured_errors = [
    validate_101(),
```

6. RUN EXISTING TESTS

Run **pytest** to run all the existing tests. If you have written the placeholder correctly, these should pass.

If **pytest** command isn’t working, check you ran **poetry install** first.



7. WRITE TESTS FOR THE RULE

Now begin writing code to test the specific functionality that your rule should implement. These tests should fail initially. For implementing this, feel free to copy/paste from other tests.

8. WRITE CODE FOR THE RULE

Now do the hard work of writing the code that passes your test – using all the Python knowledge you've gained (plus judicious use of Google/stackoverflow/asking questions).

You can repeat steps 7 and 8 as appropriate to iterate until the function works.

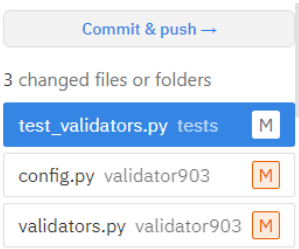
9. CHECK ALL TESTS PASS

Run **pytest** a final time to check everything passes – if it fails, continue to iterate until the issues are solved. If you get stuck, ask for help in Slack or by email!

10. COMMIT AND PUSH YOUR WORK

Go to version control on Repl.it and check your modifications. If these all make sense, write a message describing the work you've done and click commit and push.

As an initial check, you should have 3 modifications – the **validators.py**, **test_validators.py** and **config.py** file.



11. CREATE A PULL REQUEST

Go to GitHub and create a pull request from your branch to the main branch (if you have just pushed, there should be a button at the top of GitHub to make this easy).

Write what you did in the body of the pull request. Make sure to include a line reading "Closes #{issue number}" where you write the appropriate issue number.

12. AWAIT REVIEW AND CHECK FOR COMMENTS

You are done! I will review the work and may comment on areas that could be changed or improved. When I'm done with a review I'll let you know – then repeat steps 8, 9, and 10 – every time you commit and push your work, it will show up in the PR.