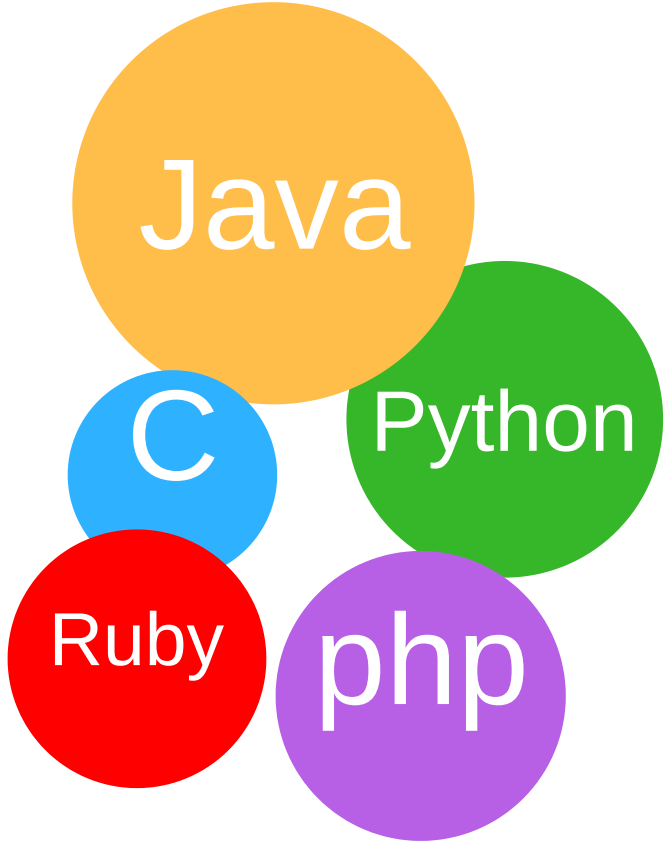
How do I complete the challenge?

So, you have signed up to the Lintol Coding Challenge but you are not sure how to get started or submit your entry?

What programming language do I use?

Although Python will be the focus, other programming languages are welcome. It will be easiest to get started in Python as we have the required tooling.

All code needs to be licensed, either MIT or Apache and publicly available by the time of submission.



What do I need to create?

Once you have selected your programming language of choice, there are a couple of things that will be required for you to create a successful submission for the challenge.

Each submission must contain a script file. This file should contain the majority of the code created. The script can reference other scripts, external libraries and resources. Any external libraries or resources should be publicly accessible.

This script file in the example project is called processor.py. The script file outputs a standard json format. As long as the script file outputs json using this same structure, it will be able to operate with the Lintol platform.

How do I submit my code?

You have created your challenge winning code and now you want to submit it for judging.

Your script file, along with any supporting files, must be submitted to a publicly accessible repository on Github (github.com). If you do not have a Github account you will have to sign up to Github and create one.

To let us know about your submission you will need to complete the submission form at **bit.ly/lintol-coding-challenge**. You will need to fill out form with your submission details.

* Please choose an open source license from **choosealicense.com** for your repository

How do I win?

Our judges will assess each valid submission against a set of criteria. Points will be awarded for how well the criteria has been met. The criteria will be comprised of the following areas:

Commenting

Documentation

Creativity and Innovation

Presentation

Collaboration

Security

LINTOL

Coding Challenge

Functionality

Does the code function correctly, robustly and produce the correct output in Github pages?

Commenting

Does the code have comments that are helpful in explaining the process and functionality?

Documentation

Is there documentation that explains what the script does, how it works and in why it will be useful? Is there documentation for any additional resources or references?

Security

Does the code address any potential security issues? Have any security issue mitigation methods been employed to increase the security of the script?

Presentation

Is the script and any supporting materials presented in clear, concise and aesthetically pleasing format? This will be best included in the readme file of the repository.

Creativity and Innovation

Does the script solve a validation issue in a new and creative way? Is the validation issue being solved an unrecognised issue and one that, if solved, will be beneficial to the open data community? Does the script use an innovative approach to solving a validation issue?

Collaboration

Has the script been created in a collaborative way or does it encourage additional feedback from the open data or development communities?

Getting Started

An example processor is available at https://gitlab.com/lintol/coding-challenge-2021-exemplar-1

For more detailed setup instructions please view the readme file in the example repository.

Set up a Python environment

- Install Python 3.7: apt install python3.7
- Install pip3: apt install python3-pip
- Install pipenv: pip3 install -user pipenv
- Clone a fork of the example processor

Navigate to the project directory and activate a new python environment using pipenv: python3 -m pipenv shell

Install dependencies for the project: pip3 install -r requirements.txt

You are now ready to run the example processor.

Run the processor.py file using python3 with the path to the example text file.

```
python3 processor.py ./sample_transcripts/out-example-2021-02-01-hansard-plenary.txt
```

Updating the example

This processor starts on line 168 then on line 170 and 171 it declares and initialises a CityFinderProcessor object

```
168    if __name__ == "__main__":
169        argv = sys.argv
170        processor = CityFinderProcessor()
171        processor.initialize()
172        workflow = processor.build_workflow(argv[1])
173        print(get(workflow, 'output'))
174
```

Line 172 will run the get_workflow method in the CityFinderProcessor class located on line 145 of the example. Line 173 then prints the output from that method.

The get_workflow method

The get_workflow method allows for the combining of multiple steps for a processor. The example has only one step enabled on line 149.

```
145     def get_workflow(self, filename, metadata={}):
146         workflow = {
147             'load-text': (load_text, filename),
148             'get-report': (self.make_report,),
149             'step-A': (city_finder, 'load-text', 'get-report'),
150             # 'step-B': (town_finder, 'load-text', 'get-report'),
151             # 'step-C': (country_finder, 'load-text', 'get-report'),
152             'output': (workflow_condense, 'step-A') #, 'step-B', 'step-C')
153         }
154         return workflow
```

The code on line 149 calls the city_finder method on line 29.

The city_finder method

The city_finder method begins on line 29 and is where most of processing logic lives.
The 'add' on line 84 adds an individual item to the summary for the paragraph being processed.

```
84                    content.add(
85                        note=f'Occurence of {city.title()}',
86                        start_offset=match.start(),
87                        end_offset=match.end(),
88                        level=logging.INFO,
89                        tags=['city']
```

The add_issue on line 102 then adds the summary content to the report

```
102                   rprt.add_issue(
103                       urgency,
104                       'city-cropped-up',
105                       f'Found {city}',
106                       line_number=line_number,
107                       character_number=0,
```

For more detailed instructions, check out the readme file for the example processor.

Not sure about something?

Just Ask!

The Lintol community is here to help

Hit us up

Rocket Chat: chat.openindustry.in

Channel: lintol-coding-challenge-2021-support-for-competitors

or

Github: github.com/lintol

Remember to submit your code

before the 8th March 2021

lintol.io/challenge/submission/

Good Luck !