

---

**MLOQ**

***Release 0.0.75***

**Guillem Duran, Vadim Markovtsev**

**Oct 16, 2022**



# WELCOME TO MLOQ

<b>1</b>	<b>About ML Ops Quickstart</b>	<b>3</b>
<b>2</b>	<b>Installation</b>	<b>5</b>
2.1	Install from pypi . . . . .	5
2.2	Install from source . . . . .	5
<b>3</b>	<b>Command line interface</b>	<b>7</b>
<b>4</b>	<b>Usage examples</b>	<b>9</b>
<b>5</b>	<b>mloq.yml config file</b>	<b>11</b>
<b>6</b>	<b>Repository files</b>	<b>13</b>
<b>7</b>	<b>Packaging</b>	<b>15</b>
<b>8</b>	<b>Code style</b>	<b>17</b>
<b>9</b>	<b>Requirements</b>	<b>19</b>
<b>10</b>	<b>Docker</b>	<b>21</b>
<b>11</b>	<b>Continuous integration using GitHub Actions</b>	<b>23</b>
<b>12</b>	<b>Testing</b>	<b>25</b>
<b>13</b>	<b>Project Makefile</b>	<b>27</b>
<b>14</b>	<b>License</b>	<b>29</b>
<b>15</b>	<b>Contributing</b>	<b>31</b>
<b>16</b>	<b>Contributing Guidelines</b>	<b>33</b>
16.1	Certificate of Origin . . . . .	33
16.2	Support Channels . . . . .	33
16.3	How to Contribute . . . . .	33
16.3.1	Format of the commit message . . . . .	34
16.4	Roadmap . . . . .	34
<b>17</b>	<b>API reference</b>	<b>35</b>
17.1	mloq . . . . .	35
17.1.1	Subpackages . . . . .	35

17.1.1.1	mloq.commands	35
17.1.1.1.1	Submodules	35
17.1.1.1.2	Package Contents	52
17.1.1.2	mloq.config	61
17.1.1.2.1	Submodules	61
17.1.2	Submodules	75
17.1.2.1	mloq.__main__	75
17.1.2.2	mloq._utils	75
17.1.2.2.1	Module Contents	75
17.1.2.3	mloq.cli	77
17.1.2.3.1	Module Contents	77
17.1.2.4	mloq.command	78
17.1.2.4.1	Module Contents	79
17.1.2.5	mloq.custom_click	81
17.1.2.5.1	Module Contents	81
17.1.2.6	mloq.failure	83
17.1.2.6.1	Module Contents	83
17.1.2.7	mloq.files	83
17.1.2.7.1	Module Contents	83
17.1.2.8	mloq.git	85
17.1.2.8.1	Module Contents	85
17.1.2.9	mloq.record	86
17.1.2.9.1	Module Contents	86
17.1.2.10	mloq.runner	88
17.1.2.10.1	Module Contents	88
17.1.2.11	mloq.templating	89
17.1.2.11.1	Module Contents	89
17.1.2.12	mloq.version	90
17.1.2.12.1	Module Contents	90
17.1.2.13	mloq.writer	90
17.1.2.13.1	Module Contents	90
17.1.3	Package Contents	91
<b>18</b>	<b>Indices and tables</b>	<b>93</b>
	<b>Python Module Index</b>	<b>95</b>
	<b>Index</b>	<b>97</b>

ML Ops Quickstart is a tool for initializing Machine Learning projects following ML Ops best practices.

Setting up new repositories is a time-consuming task that involves creating different files and configuring tools such as linters, docker containers and continuous integration pipelines. The goal of `mloq` is to simplify that process, so you can start writing code as fast as possible.

`mloq` generates customized templates for Python projects with focus on Machine Learning. An example of the generated templates can be found in [mloq-template](#).



## **ABOUT ML OPS QUICKSTART**

ML Ops Quickstart is a tool for initializing Machine Learning projects following ML Ops best practices.

Setting up new repositories is a time-consuming task that involves creating different files and configuring tools such as linters, docker containers, and continuous integration pipelines. The goal of `mloq` is to simplify that process, so you can start writing code as fast as possible.

`mloq` generates customized templates for Python projects with a focus on Machine Learning. An example of the generated templates can be found in [mloq-template](#).





## INSTALLATION

mloq is tested on Ubuntu 18.04+, and supports Python 3.6+.

### 2.1 Install from pypi

```
pip install mloq
```

### 2.2 Install from source

```
git clone https://github.com/FragileTech/ml-ops-quickstart.git
cd ml-ops-quickstart
pip install -e .
```



## COMMAND LINE INTERFACE

Options:

- `--file -f`: Name of the configuration file. If `file` is a directory, it will load the `mloq.yml` file present in it.
- `--overwrite -o`: Rewrite files that already exist in the target project.
- `--interactive -i`: Missing configuration data can be defined interactively from the CLI.



## USAGE EXAMPLES

Arguments:

- `OUTPUT_DIRECTORY`: Path to the target project.

To set up a new repository from scratch interactively in the current working directory:

```
mloq setup -i .
```

To load a `mloq.yml` configuration file from the current repository, and initialize the directory `example`, and overwrite all existing files with no interactivity:

```
mloq setup -f . -o example
```

```
Welcome to the MLOQ 0.0.16 interactive setup utility.

Please enter values for the following settings (just press Enter to accept a default value, if one is given in brackets).

The following values will occur in several places in the generated files.
> Select project name: my-new-project
> Short description of the project: This is a short description
> Github handle of the project owner: owner
> Owner contact email: owner@mail.com
> Author(s) of the project [owner]: Owner and Employee
> Github project url [https://github.com/owner/my-new-project]:

Please define the license of the project.
Open Source projects will include the corresponding LICENSE file and a DCO.md file
> Is the project Open Source? [Y/n]:
> Project license type (MIT, None) [MIT]:
> Copyright holder [owner]: Owner Name

What Python versions are supported? Please provide the values as a comma separated list.
> Supported python versions [3.6, 3.7, 3.8, 3.9]:
Please specify the requirements of the project as a comma separated list.
Available values:
  data-science: Common data science libraries such as numpy, pandas, sklearn...
  data-viz: Visualization libraries such as holoviews, bokeh, plotly, matplotlib...
  pytorch: Latest version of pytorch, torchvision and pytorch_lightning
  tensorflow:
> Project requirements [None]: pytorch

You can configure the continuous integration using Github Actions.
Available values:
  Python: Push workflow for pure Python projects.
  Dist: Push workflow for Python projects with compiled extensions.
  None: Do not set up the CI.
> Github Actions push workflow (python, dist, none) [python]:
> Default branch of the project [master]: main
A bot account will be used to automatically bump the version of your project.
> Bot's Github login to push commits in CI [Owner and Employee]: company-bot
> Bot account email [owner@mail.com]: bot@company.com
MLOQ will generate a Dockerfile for your project.
> Base docker image for the project's Docker container [fragiletech/ubuntu18.04-cuda-11.0-py39]:
You can optionally create an ML Flow MLproject file.
> Do you want to set up ML Flow? [y/N]:
Do you want to generate a mloq.yml file? [y/N]: y
Do you want to override existing files? [y/N]:
```



## MLOQ.YML CONFIG FILE

This yaml file contains all the information used by mloq to set up a new project. All values are strings except **python\_versions** and **requirements**, which are lists of strings. **null** values are interpreted as missing values.

```
# This yaml file contains all the information used by mloq to set up a new project.
# All values in template are strings and booleans,
# except "python_versions" and "requirements" that are lists of strings.
# "null" values are interpreted as non-defined values.
# -----

# project_config values are necessary to define the files that will be written, and the
↳ tools
# that will be configured.
project_config:
  open_source: null # boolean. If True, set up and Open Source project
  docker: null # boolean If True, set up a Docker image for the project
  ci: null # Name of the GitHub Actions CI workflow that will be configured.
  mlflow: null # boolean. If True configure a MLproject file compatible with ML Flow
↳ projects.
  requirements: null # List containing the pre-defined requirements of the project.

# template contains all the values that will be written in the generated files.
# They are loaded as a dictionary and passed to jinja2 to fill in the templates.
template:
  project_name: null # Name of the new Python project
  default_branch: null # Name of the default branch. Used in the CI push workflow.
  owner: null # Github handle of the project owner
  author: null # Person(s) or entity listed as the project author in setup.py
  email: null # Owner contact email
  copyright_holder: null # Owner of the project copyright.
  project_url: null # Project download url. Defaults to https://github.com/{owner}/
↳ {project_name}
  bot_name: null # GitHub login of the account used to push when bumping the project
↳ version
  bot_email: null # Bot account email
  license: null # Currently only proprietary and MIT license is supported
  description: null # Short description of the project
  python_versions: null # Supported Python versions
  docker_image: null # Your project Docker container will inherit from this image.
```

mloq offers a set of features focused on generating automatically all the required files to start your Machine Learning project.





## **REPOSITORY FILES**

Set up the following common repository files personalized for your project with the values defined in `mloq.yml`:

- `README.md`
- `DCO.md`
- `CONTRIBUTING.md`
- `CODE_OF_CONDUCT.md`
- `LICENSE`
- `.gitignore`



## PACKAGING

Automatic configuration of `pyproject.toml` and `setup.py` to distribute your project as a Python package.



## CODE STYLE

All the necessary configuration for the following tools is defined in `pyproject.toml`.

- `black`: Automatic code formatter.
- `isort`: Rearrange your imports automatically.
- `flakehell`: Linter tool build on top of `flake8`, `pylint` and `pycodestyle`



## **REQUIREMENTS**

`mloq` creates three different requirements files in the root directory of the project. Each file contains pinned dependencies.

- `requirements-lint.txt`: Contains the dependencies for running style check analysis and automatic formatting of the code.
- `requirements-test.txt`:

Dependencies for running `pytest`, `hypothesis`, and test coverage.

- `requirements.txt`: Contains different pre-configured dependencies that can be defined in `mloq.yml`. The available pre-configured dependencies are:
  - `data-science`: Dependencies of common data science libraries.
  - `data-visualization`: Common visualization libraries.
  - Last version of `pytorch` and `tensorflow`





**DOCKER**

A Dockerfile that builds a container on top of the FragileTech [Docker Hub](#) images:

- If *tensorflow* or *pytorch* are selected as requirements, the container has CUDA 11.0 installed.
- Installs all the packages listed in `requirements.txt`.
- Installs `requirements-test.txt` and `requirements-lint.txt` dependencies.
- Install a `jupyter notebook` server with a configurable password on port 8080.
- Installs the project with `pip install -e ..`



## CONTINUOUS INTEGRATION USING GITHUB ACTIONS

Set up automatically a continuous integration (CI) pipeline using GitHub actions with the following jobs:



Automatic build and tests:

- **Style Check:** Run `flake8` and `black --check` to ensure a consistent code style.
- **Pytest:** Test the project using `pytest` on all supported Python versions and output a code coverage report.
- **Test-docker:** Build the project's Docker container and run the tests inside it.
- **Build-pypi:** Build the project and upload it to [Test PyPI](#) with a version tag unique to each commit.
- **Test-pypi:** Install the project from Test PyPI and run the tests using `pytest`.
- **Bump-version:** Automatically bump the project version and create a tag in the repository every time the default branch is updated.

Deploy each new version:

- **Push-docker-container:** Upload the project's Docker container to [Docker Hub](#).
- **Release-package:** Upload to [PyPI](#) the source of the project and the corresponding wheels.



## TESTING

The last versions of `pytest`, `hypothesis`, and `pytest-cov` can be found in `requirements-test.txt`.

The folder structure for the library and tests is created.



## PROJECT MAKEFILE

A Makefile will be created in the root directory of the project. It contains the following commands:

- `make style`: Run `isort` and `black` to automatically arrange the imports and format the project.
- `make check`: Run `flakehell` and check black style. If it raises any error the CI will fail.
- `make test`: Clear the tests cache and run `pytest`.
- `make pipenv-install`: Install the project in a new Pipenv environment and create a new `Pipfile` and `Pipfile.lock`.
- `make pipenv-test`: Run `pytest` inside the project's Pipenv.
- `make docker-build`: Build the project's Docker container.
- `make docker-test`: Run `pytest` inside the projects docker container.
- `make docker-shell`: Mount the current project as a docker volume and open a terminal in the project's container.
- `make docker-notebook`: Mount the current project as a docker volume and open a jupyter notebook in the project's container. It exposes the notebook server on the port `8080`.





---

CHAPTER  
**FOURTEEN**

---

**LICENSE**

ML Ops Quickstart is released under the [MIT](#) license.



## CONTRIBUTING

Contributions are very welcome! Please check the contributing guidelines before opening a pull request.



## CONTRIBUTING GUIDELINES

mloq is *MIT licensed* and accepts contributions via GitHub pull requests. This document outlines some of the conventions on development workflow, commit message formatting, contact points, and other resources to make it easier to get your contribution accepted.

### 16.1 Certificate of Origin

By contributing to this project you agree to the Developer Certificate of Origin (DCO). This document was created by the Linux Kernel community and is a simple statement that you, as a contributor, have the legal right to make the contribution.

In order to show your agreement with the DCO you should include at the end of commit message, the following line: Signed-off-by: John Doe <john.doe@example.com>, using your real name.

This can be done easily using the `-s` flag on the `git commit`.

### 16.2 Support Channels

The official support channels, for both users and contributors, are:

- GitHub [issues](#)\*

\*Before opening a new issue or submitting a new pull request, it's helpful to search the project - it's likely that another user has already reported the issue you're facing, or it's a known issue that we're already aware of.

### 16.3 How to Contribute

Pull Requests (PRs) are the main and exclusive way to contribute to the official project. In order for a PR to be accepted it needs to pass a list of requirements:

- The CI style check passes (run locally with `make check`).
- Code Coverage does not decrease.
- All the tests pass.
- Python code is formatted according to
- If the PR is a bug fix, it has to include a new unit test that fails before the patch is merged.
- If the PR is a new feature, it has to come with a suite of unit tests, that tests the new functionality.

- In any case, all the PRs have to pass the personal evaluation of at least one of the maintainers.

### 16.3.1 Format of the commit message

The commit summary must start with a capital letter and with a verb in present tense. No dot in the end.

Add a feature  
Remove unused code  
Fix a bug

Every commit details should describe what was changed, under which context and, if applicable, the GitHub issue it relates to.

## 16.4 Roadmap

- [ ] Improve documentation and test coverage.
- [ ] Configure `sphinx` to build the docs automatically.
- [ ] Implement checks for additional best practices.
- [ ] Improve command-line interface and logging.
- [ ] Add new customization options.

## API REFERENCE

This page holds mloq API documentation, which might be helpful for final users or developers to create their own mloq-based utilities. Among the different sub-packages and modules, we might differentiate two big categories: core utilities and high-level ones.

- **Core API:** These routines are located within the *mloq.api* that show the different features of the library can be accessed programmatically.

### 17.1 mloq

Package for initializing ML projects following ML Ops best practices.

#### 17.1.1 Subpackages

##### 17.1.1.1 `mloq.commands`

Contains all the defined mloq Commands.

##### 17.1.1.1.1 Submodules

###### `mloq.commands.ci`

Mloq ci command implementation.

#### Module Contents

##### Classes

---

*CiCMD*

Implement the functionality of the ci Command.

---

## Attributes

---

*CI\_ASSETS\_PATH*

---

---

*push\_python\_wkf*

---

---

*WORKFLOW\_FILES*

---

`mloq.commands.ci.CI_ASSETS_PATH`

`mloq.commands.ci.push_python_wkf`

`mloq.commands.ci.WORKFLOW_FILES`

**class** `mloq.commands.ci.CiCMD`(*record*, *interactive=False*, *\*\*kwargs*)

Bases: *mloq.command.Command*

Implement the functionality of the ci Command.

### Parameters

- **record** (*mloq.writer.CMDRecord*) –
- **interactive** (*bool*) –

**cmd\_name** :str = ci

**ubuntu\_version**

**disable**

**docker**

**project\_name**

**default\_branch**

**docker\_org**

**bot\_name**

**bot\_email**

**ci\_python\_version**

**python\_versions**

**ci\_extra**

**vendor**

**open\_source**

**author**

**owner**

**email**



**project\_url****files****property directories**

Tuple containing Paths objects representing the directories created by the command.

**Return type** Tuple[pathlib.Path]

**interactive\_config()**

Generate the configuration of the project interactively.

**Return type** omegaconf.DictConfig

**record\_files()**

Register the files that will be generated by mloq.

**Return type** None

**mloq.commands.docker**

Mloq docker command implementation.

## Module Contents

### Classes

<i>DockerCMD</i>	Implement the functionality of the docker Command.
------------------	--

### Attributes

<i>DOCKER_ASSETS_PATH</i>
<i>dockerfile</i>
<i>makefile_docker</i>
<i>DOCKER_FILES</i>

mloq.commands.docker.DOCKER\_ASSETS\_PATH

mloq.commands.docker.dockerfile

mloq.commands.docker.makefile\_docker

mloq.commands.docker.DOCKER\_FILES

```
class mloq.commands.docker.DockerCMD(record, interactive=False, **kwargs)
    Bases: mloq.command.Command
    Implement the functionality of the docker Command.

    Parameters
        • record (mloq.writer.CMDRecord) –
        • interactive (bool) –

    cmd_name = docker
    files
    disable
    cuda
    cuda_image_type
    cuda_version
    ubuntu_version
    project_name
    docker_org
    python_version
    base_image
    test
    lint
    jupyter
    jupyter_password
    requirements
    extra
    makefile

    static require_cuda_from_requirements(project_config=None)
        Return True if any of the project dependencies require CUDA.
        Parameters project_config (Optional[omegaconf.DictConfig]) –
        Return type bool

    requires_cuda()
        Return True if the Docker container requires CUDA.
        Return type bool

    get_base_image()
        Return the name of the base image for the project Docker container.
```

**parse\_config()**

Update the configuration dictionary from the data entered by the user.

**Return type** omegaconf.DictConfig

**interactive\_config()**

Generate the configuration of the project interactively.

**Return type** omegaconf.DictConfig

**record\_files()**

Register the files that will be generated by mloq.

**Return type** None

**mloq.commands.docs**

Mloq docs command implementation.

**Module Contents****Classes**


---

*DocsCMD*


---

 Implement the functionality of the docs Command.
 

---

**Attributes**


---

*DOCS\_ASSETS\_PATH*


---

*makefile\_docs*


---

*make\_bat\_docs*


---

*docs\_req*


---

*conf\_py*


---

*index\_md*


---

*deploy\_docs*


---

*DOCS\_FILES*


---

mloq.commands.docs.DOCS\_ASSETS\_PATH

mloq.commands.docs.makefile\_docs

mloq.commands.docs.make\_bat\_docs

`mloq.commands.docs.docs_req`

`mloq.commands.docs.conf_py`

`mloq.commands.docs.index_md`

`mloq.commands.docs.deploy_docs`

`mloq.commands.docs.DOCS_FILES`

**class** `mloq.commands.docs.DocsCMD(record, interactive=False, **kwargs)`

Bases: [`mloq.command.Command`](#)

Implement the functionality of the docs Command.

**Parameters**

- **record** (`mloq.writer.CMDRecord`) –
- **interactive** (`bool`) –

`cmd_name` = docs

`disable`

`project_name`

`description`

`author`

`copyright_year`

`copyright_holder`

`deploy_docs`

`default_branch`

`project_url`

`files`

**property directories**

Tuple containing Paths objects representing the directories created by the command.

**Return type** `Tuple[pathlib.Path]`

**interactive\_config()**

Generate the configuration of the project interactively.

**Return type** `omegaconf.DictConfig`

**record\_files()**

Register the files that will be generated by mloq.

**Return type** `None`

## mloq.commands.globals

Mloq globals command implementation.

## Module Contents

### Classes

---

#### *GlobalsCMD*

Implement the functionality of the globals Command.

---

**class** mloq.commands.globals.GlobalsCMD(*record*, *interactive=False*, *\*\*kwargs*)

Bases: *mloq.command.Command*

Implement the functionality of the globals Command.

#### Parameters

- **record** (*mloq.writer.CMDRecord*) –
- **interactive** (*bool*) –

**cmd\_name** = globals

**project\_name**

**description**

**author**

**owner**

**email**

**open\_source**

**project\_url**

**default\_branch**

**license**

**use\_poetry**

**main\_python\_version**

**interactive\_config()**

Generate the configuration of the project interactively.

**Return type** omegaconf.DictConfig

**parse\_config()**

Generate the configuration of the project via a configuration file.

**Return type** omegaconf.DictConfig

## `mloq.commands.license`

Mloq license command implementation.

### Module Contents

#### Classes

---

<i>LicenseCMD</i>	Implement the functionality of the license Command.
-------------------	---

---

#### Attributes

---

<i>TEMPLATES_PATH</i>
<i>dco</i>
<i>mit_license</i>
<i>apache_license</i>
<i>gpl_license</i>
<i>LICENSES</i>
<i>LICENSE_FILES</i>

---

`mloq.commands.license.TEMPLATES_PATH`

`mloq.commands.license.dco`

`mloq.commands.license.mit_license`

`mloq.commands.license.apache_license`

`mloq.commands.license.gpl_license`

`mloq.commands.license.LICENSES`

`mloq.commands.license.LICENSE_FILES`

**class** `mloq.commands.license.LicenseCMD`(*record*, *interactive=False*, *\*\*kwargs*)

Bases: *mloq.command.Command*

Implement the functionality of the license Command.

#### Parameters

- **record** (*mloq.writer.CMDRecord*) –
- **interactive** (*bool*) –

`cmd_name = license``files``LICENSES``disable``license``copyright_year``copyright_holder``project_name``project_url``email``interactive_config()`

Generate the configuration of the project interactively.

**Return type** omegaconf.DictConfig

`record_files()`

Register the files that will be generated by mloq.

**Return type** None

`mloq.commands.lint`

Mloq lint command implementation.

## Module Contents

### Classes

---

*LintCMD*

---

Implement the functionality of the lint Command.

---

### Attributes

---

*lint\_req*

---

---

*LINT\_FILES*

---

`mloq.commands.lint.lint_req``mloq.commands.lint.LINT_FILES`

```
class mloq.commands.lint.LintCMD(record, interactive=False, **kwargs)
```

Bases: [mloq.command.Command](#)

Implement the functionality of the lint Command.

**Parameters**

- **record** ([mloq.writer.CMDRecord](#)) –
- **interactive** (*bool*) –

**cmd\_name** = lint

**files**

**disable**

**black**

**isort**

**linters**

**docstring\_checks**

**pyproject\_extra**

**project\_name**

**makefile**

**poetry\_requirements**

**ignore\_files**

**interactive\_config()**

Generate the configuration of the project interactively.

**Return type** [omegaconf.DictConfig](#)

**record\_files()**

Register the files that will be generated by mloq.

**Return type** None

[mloq.commands.package](#)

Mloq package command implementation.

## Module Contents

### Classes

---

[PackageCMD](#)

Implement the functionality of the package Command.

---



## Attributes

---

*PACKAGE\_ASSETS\_PATH*

---

*PYTHON\_VERSIONS*

---

*DEFAULT\_PYTHON\_VERSIONS*

---

*setup\_py*

---

*PACKAGE\_FILES*

---

`mloq.commands.package.PACKAGE_ASSETS_PATH`

`mloq.commands.package.PYTHON_VERSIONS = ['3.6', '3.7', '3.8', '3.9', '3.10']`

`mloq.commands.package.DEFAULT_PYTHON_VERSIONS = ['3.7', '3.8', '3.9', '3.10']`

`mloq.commands.package.setup_py`

`mloq.commands.package.PACKAGE_FILES`

**class** `mloq.commands.package.PackageCMD`(*record*, *interactive=False*, *\*\*kwargs*)

Bases: *mloq.command.Command*

Implement the functionality of the package Command.

### Parameters

- **record** (*mloq.writer.CMDRecord*) –
- **interactive** (*bool*) –

**cmd\_name** = `package`

**files**

**LICENSE\_CLASSIFIERS**

**disable**

**pyproject\_extra**

**project\_name**

**license**

**license\_classifier**

**description**

**default\_branch**

**project\_url**

**owner**

**author**

**email**

**main\_python\_version**

**python\_versions**

**use\_poetry**

**parse\_config()**

Update the configuration DictConfig with the Command parameters.

**Return type** omegaconf.DictConfig

**interactive\_config()**

Generate the configuration of the project interactively.

**Return type** omegaconf.DictConfig

**record\_files()**

Register the files that will be generated by mloq.

**Return type** None

**mloq.commands.project**

Mloq project command implementation.

## Module Contents

### Classes

---

*ProjectCMD*

Implement the functionality of the project Command.

---

## Attributes

---

*PROJECT\_ASSETS\_PATH*

---

---

*readme*

---

---

*gitignore*

---

---

*pre\_commit\_hook*

---

---

*init*

---

---

*main*

---

---

*test\_main*

---

---

*test\_req*

---

---

*version*

---

---

*code\_of\_conduct*

---

---

*codecov*

---

---

*contributing*

---

---

*PROJECT\_FILES*

---

---

`mloq.commands.project.PROJECT_ASSETS_PATH``mloq.commands.project.readme``mloq.commands.project.gitignore``mloq.commands.project.pre_commit_hook``mloq.commands.project.init``mloq.commands.project.main``mloq.commands.project.test_main``mloq.commands.project.test_req``mloq.commands.project.version``mloq.commands.project.code_of_conduct``mloq.commands.project.codecov``mloq.commands.project.contributing``mloq.commands.project.PROJECT_FILES`

```
class mloq.commands.project.ProjectCMD(record, interactive=False, **kwargs)
```

Bases: [mloq.command.Command](#)

Implement the functionality of the project Command.

**Parameters**

- **record** (*mloq.writer.CMDRecord*) –
- **interactive** (*bool*) –

**cmd\_name** = `project`

**files**

**disable**

**project\_name**

**owner**

**description**

**project\_url**

**license**

**tests**

**property directories**

Tuple containing Paths objects representing the directories created by the command.

**Return type** `Tuple[pathlib.Path]`

**record\_files()**

Register the files that will be generated by mloq.

**Return type** `None`

[mloq.commands.requirements](#)

Mloq requirements command implementation.

## Module Contents

### Classes

---

[RequirementsCMD](#)

Implement the functionality of the requirements Command.

---

## Attributes

---

*REQUIREMENTS\_PATH*

---

*requirements*

---

*data\_science\_req*

---

*data\_viz\_req*

---

*pytorch\_req*

---

*tensorflow\_req*

---

*lint\_req*

---

*test\_req*

---

*dogfood\_req*

---

*docs\_req*

---

*REQUIREMENTS\_FILES*

---

*REQUIREMENT\_CHOICES*

---

```
mloq.commands.requirements.REQUIREMENTS_PATH
mloq.commands.requirements.requirements
mloq.commands.requirements.data_science_req
mloq.commands.requirements.data_viz_req
mloq.commands.requirements.pytorch_req
mloq.commands.requirements.tensorflow_req
mloq.commands.requirements.lint_req
mloq.commands.requirements.test_req
mloq.commands.requirements.dogfood_req
mloq.commands.requirements.docs_req
mloq.commands.requirements.REQUIREMENTS_FILES
mloq.commands.requirements.REQUIREMENT_CHOICES = ['data-science', 'data-viz', 'torch',
'tensorflow', 'none', 'dogfood', 'None']

class mloq.commands.requirements.RequirementsCMD(record, interactive=False)
    Bases: mloq.command.Command
    Implement the functionality of the requirements Command.
```

**Parameters**

- **record** (`mloq.record.CMDRecord`) –
- **interactive** (*bool*) –

**cmd\_name** = requirements

**files**

**disable**

**requirements**

**REQUIREMENTS\_ALIASES**

**\_\_del\_\_()**

Remove the temporary directory when the instance is deleted.

**Return type** None

**classmethod get\_aliased\_requirements\_file(option)**

Get requirement file from aliased name.

**Parameters** **option** (*str*) –

**Return type** *mloq.files.File*

**classmethod read\_requirements\_file(option)**

Return the content of the target requirements file form an aliased name.

**Parameters** **option** (*str*) –

**Return type** str

**classmethod compose\_requirements(options)**

Return the content requirements.txt file with pinned dependencies.

The returned string contains the combined dependencies for the different options sorted alphabetically.

**Parameters** **options** (*Iterable[str]*) – Iterable containing the aliased names of the target dependencies for the project.

**Returns** str containing the pinned versions of all the selected requirements.

**Return type** str

**interactive\_config()**

Generate the configuration of the project interactively.

**Return type** omegaconf.DictConfig

**static requirements\_is\_empty(options)**

Return True if no requirements are specified for the project.

**Parameters** **options** (*Union[List[str], str]*) –

**Return type** bool

**record\_files()**

Register the files that will be generated by mloq.

**Return type** None

## mloq.commands.setup

Mloq setup command implementation.

### Module Contents

#### Classes

---

<i>SetupCMD</i>	Implement the functionality of the setup Command.
-----------------	---

---

#### Functions

---

*\_sub\_commands()*

---

#### Attributes

---

*SUB\_COMMANDS*

---

mloq.commands.setup.\_sub\_commands()

mloq.commands.setup.SUB\_COMMANDS

**class** mloq.commands.setup.SetupCMD(*record*, *interactive=False*)

Bases: *mloq.command.CommandMixin*

Implement the functionality of the setup Command.

##### Parameters

- **record** (*mloq.record.CMDRecord*) –
- **interactive** (*bool*) –

**cmd\_name** = setup

**files**

**SUB\_COMMAND\_CLASSES**

**property config**

List of all the commands that will be executed when running mloq setup.

**Return type** omegaconf.DictConfig

**property sub\_commands**

List of all the commands that will be executed when running mloq setup.

**Return type** List[*mloq.command.Command*]

**property directories**

Tuple containing Paths objects representing the directories created by the command.

**Return type** Tuple[pathlib.Path]

**interactive\_config()**

Generate the configuration of the project interactively.

**Return type** omegaconf.DictConfig

**parse\_config()**

Update the configuration DictConfig with the Command parameters.

**Return type** omegaconf.DictConfig

**run\_side\_effects()**

Apply additional configuration methods.

**Return type** None

**record\_files()**

Register the files that will be generated by mloq.

**Return type** None

### 17.1.1.1.2 Package Contents

#### Classes

<i>CiCMD</i>	Implement the functionality of the ci Command.
<i>DockerCMD</i>	Implement the functionality of the docker Command.
<i>DocsCMD</i>	Implement the functionality of the docs Command.
<i>GlobalsCMD</i>	Implement the functionality of the globals Command.
<i>LicenseCMD</i>	Implement the functionality of the license Command.
<i>LintCMD</i>	Implement the functionality of the lint Command.
<i>PackageCMD</i>	Implement the functionality of the package Command.
<i>ProjectCMD</i>	Implement the functionality of the project Command.
<i>RequirementsCMD</i>	Implement the functionality of the requirements Command.
<i>SetupCMD</i>	Implement the functionality of the setup Command.

```
class mloq.commands.CiCMD(record, interactive=False, **kwargs)
```

Bases: *mloq.command.Command*

Implement the functionality of the ci Command.

**Parameters**

- **record** (*mloq.writer.CMDRecord*) –
- **interactive** (*bool*) –

**cmd\_name** :str = ci

**ubuntu\_version**



`disable``docker``project_name``default_branch``docker_org``bot_name``bot_email``ci_python_version``python_versions``ci_extra``vendor``open_source``author``owner``email``project_url``files``property directories`

Tuple containing Paths objects representing the directories created by the command.

**Return type** Tuple[pathlib.Path]

`interactive_config()`

Generate the configuration of the project interactively.

**Return type** omegaconf.DictConfig

`record_files()`

Register the files that will be generated by mloq.

**Return type** None

**class** mloq.commands.**DockerCMD**(*record*, *interactive=False*, *\*\*kwargs*)

Bases: [mloq.command.Command](#)

Implement the functionality of the docker Command.

**Parameters**

- **record** (*mloq.writer.CMDRecord*) –
- **interactive** (*bool*) –

**cmd\_name** = `docker`

`files`

`disable`

`cuda`

`cuda_image_type`

`cuda_version`

`ubuntu_version`

`project_name`

`docker_org`

`python_version`

`base_image`

`test`

`lint`

`jupyter`

`jupyter_password`

`requirements`

`extra`

`makefile`

**static** `require_cuda_from_requirements(project_config=None)`

Return True if any of the project dependencies require CUDA.

**Parameters** `project_config` (*Optional*[*omegaconf.DictConfig*]) –

**Return type** bool

**requires\_cuda()**

Return True if the Docker container requires CUDA.

**Return type** bool

**get\_base\_image()**

Return the name of the base image for the project Docker container.

**parse\_config()**

Update the configuration dictionary from the data entered by the user.

**Return type** *omegaconf.DictConfig*

**interactive\_config()**

Generate the configuration of the project interactively.

**Return type** *omegaconf.DictConfig*

**record\_files()**

Register the files that will be generated by mloq.

**Return type** None

**class** mloq.commands.DocsCMD(*record*, *interactive=False*, *\*\*kwargs*)

Bases: [mloq.command.Command](#)

Implement the functionality of the docs Command.

**Parameters**

- **record** (*mloq.writer.CMDRecord*) –
- **interactive** (*bool*) –

**cmd\_name** = docs

**disable**

**project\_name**

**description**

**author**

**copyright\_year**

**copyright\_holder**

**deploy\_docs**

**default\_branch**

**project\_url**

**files**

**property directories**

Tuple containing Paths objects representing the directories created by the command.

**Return type** Tuple[pathlib.Path]

**interactive\_config()**

Generate the configuration of the project interactively.

**Return type** omegaconf.DictConfig

**record\_files()**

Register the files that will be generated by mloq.

**Return type** None

**class** mloq.commands.GlobalsCMD(*record*, *interactive=False*, *\*\*kwargs*)

Bases: [mloq.command.Command](#)

Implement the functionality of the globals Command.

**Parameters**

- **record** (*mloq.writer.CMDRecord*) –
- **interactive** (*bool*) –

`cmd_name = globals`

`project_name`

`description`

`author`

`owner`

`email`

`open_source`

`project_url`

`default_branch`

`license`

`use_poetry`

`main_python_version`

`interactive_config()`

Generate the configuration of the project interactively.

**Return type** `omegaconf.DictConfig`

`parse_config()`

Generate the configuration of the project via a configuration file.

**Return type** `omegaconf.DictConfig`

**class** `mloq.commands.LicenseCMD`(*record*, *interactive=False*, *\*\*kwargs*)

Bases: [`mloq.command.Command`](#)

Implement the functionality of the license Command.

**Parameters**

- **record** (`mloq.writer.CMDRecord`) –
- **interactive** (`bool`) –

`cmd_name = license`

`files`

`LICENSES`

`disable`

`license`

`copyright_year`

`copyright_holder`

`project_name`

`project_url`

**email**

**interactive\_config()**

Generate the configuration of the project interactively.

**Return type** omegaconf.DictConfig

**record\_files()**

Register the files that will be generated by mloq.

**Return type** None

**class** mloq.commands.**LintCMD**(*record*, *interactive=False*, *\*\*kwargs*)

Bases: [mloq.command.Command](#)

Implement the functionality of the lint Command.

**Parameters**

- **record** (*mloq.writer.CMDRecord*) –
- **interactive** (*bool*) –

**cmd\_name** = lint

**files**

**disable**

**black**

**isort**

**linters**

**docstring\_checks**

**pyproject\_extra**

**project\_name**

**makefile**

**poetry\_requirements**

**ignore\_files**

**interactive\_config()**

Generate the configuration of the project interactively.

**Return type** omegaconf.DictConfig

**record\_files()**

Register the files that will be generated by mloq.

**Return type** None

**class** mloq.commands.**PackageCMD**(*record*, *interactive=False*, *\*\*kwargs*)

Bases: [mloq.command.Command](#)

Implement the functionality of the package Command.

**Parameters**

- **record** (*mloq.writer.CMDRecord*) –
- **interactive** (*bool*) –

**cmd\_name** = package

**files**

**LICENSE\_CLASSIFIERS**

**disable**

**pyproject\_extra**

**project\_name**

**license**

**license\_classifier**

**description**

**default\_branch**

**project\_url**

**owner**

**author**

**email**

**main\_python\_version**

**python\_versions**

**use\_poetry**

**parse\_config()**

Update the configuration DictConfig with the Command parameters.

**Return type** omegaconf.DictConfig

**interactive\_config()**

Generate the configuration of the project interactively.

**Return type** omegaconf.DictConfig

**record\_files()**

Register the files that will be generated by mloq.

**Return type** None

**class** mloq.commands.**ProjectCMD**(*record*, *interactive=False*, *\*\*kwargs*)

Bases: [mloq.command.Command](#)

Implement the functionality of the project Command.

**Parameters**

- **record** (*mloq.writer.CMDRecord*) –
- **interactive** (*bool*) –

`cmd_name = project`

`files`

`disable`

`project_name`

`owner`

`description`

`project_url`

`license`

`tests`

**property directories**

Tuple containing Paths objects representing the directories created by the command.

**Return type** `Tuple[pathlib.Path]`

**record\_files()**

Register the files that will be generated by mloq.

**Return type** `None`

**class** `mloq.commands.RequirementsCMD(record, interactive=False)`

Bases: `mloq.command.Command`

Implement the functionality of the requirements Command.

**Parameters**

- **record** (`mloq.record.CMDRecord`) –
- **interactive** (`bool`) –

`cmd_name = requirements`

`files`

`disable`

`requirements`

**REQUIREMENTS\_ALIASES**

**`__del__()`**

Remove the temporary directory when the instance is deleted.

**Return type** `None`

**classmethod** `get_aliased_requirements_file(option)`

Get requirement file from aliased name.

**Parameters** `option (str)` –

**Return type** `mloq.files.File`

**classmethod** `read_requirements_file(option)`

Return the content of the target requirements file form an aliased name.

**Parameters** `option (str)` –

**Return type** `str`

**classmethod** `compose_requirements(options)`

Return the content requirements.txt file with pinned dependencies.

The returned string contains the combined dependencies for the different options sorted alphabetically.

**Parameters** `options (Iterable[str])` – Iterable containing the aliased names of the target dependencies for the project.

**Returns** `str` containing the pinned versions of all the selected requirements.

**Return type** `str`

**interactive\_config()**

Generate the configuration of the project interactively.

**Return type** `omegaconf.DictConfig`

**static** `requirements_is_empty(options)`

Return True if no requirements are specified for the project.

**Parameters** `options (Union[List[str], str])` –

**Return type** `bool`

**record\_files()**

Register the files that will be generated by mloq.

**Return type** `None`

**class** `mloq.commands.SetupCMD(record, interactive=False)`

Bases: `mloq.command.CommandMixin`

Implement the functionality of the setup Command.

**Parameters**

- **record** (`mloq.record.CMDRecord`) –
- **interactive** (`bool`) –

`cmd_name = setup`

`files`

`SUB_COMMAND_CLASSES`

**property** `config`

List of all the commands that will be executed when running mloq setup.

**Return type** `omegaconf.DictConfig`

**property** `sub_commands`

List of all the commands that will be executed when running mloq setup.

**Return type** `List[mloq.command.Command]`



**property directories**

Tuple containing Paths objects representing the directories created by the command.

**Return type** Tuple[pathlib.Path]

**interactive\_config()**

Generate the configuration of the project interactively.

**Return type** omegaconf.DictConfig

**parse\_config()**

Update the configuration DictConfig with the Command parameters.

**Return type** omegaconf.DictConfig

**run\_side\_effects()**

Apply additional configuration methods.

**Return type** None

**record\_files()**

Register the files that will be generated by mloq.

**Return type** None

**17.1.1.2 mloq.config****17.1.1.2.1 Submodules****mloq.config.configuration**

This module defines the Configurable class and associated logic.

The Configurable class extends the param.Parameterizable class to keep track of the class parameters using an omegaconf.DictConfig.

**Module Contents****Classes**

<i>Dataclass</i>	Type hinting to defined a dataclass as a typing Protocol.
<i>DictConfig</i>	param.Parameter that defines a DictConfig object.
<i>OmegaConfInterface</i>	Common functionality to work with configurations.
<i>BaseConfig</i>	Manages getters and setters to access the target's configuration values.
<i>Config</i>	Config handles the <i>.conf</i> attribute of a Configurable class.
<i>Configurable</i>	A Configurable class is an extension of param.Parameterized that allows to handle parameters with missing values and omegaconf interpolation strings.

## Functions

<code>param_to_dataclass_dict(obj)</code>	Create a dictionary that can be used to initialize a dataclass containing the parameters of the target <code>param.Parameterized</code> class.
<code>param_to_dataclass(obj)</code>	Create a dataclass equivalent to the target <code>param.Parameterized</code> target.
<code>param_to_omegaconf(obj)</code>	Transform a <code>param.Parameterized</code> class into an OmegaConf structured configuration.
<code>is_interpolation(s)</code>	Return True if the provided string is an OmegaConf interpolation string.
<code>to_param_type(obj, config, key)</code>	Transform the provided attribute of the target <code>param.Parameterized</code> object into the appropriate type so it can be stored in a configuration file.
<code>to_config(config, **kwargs)</code>	Transform the provided object into an <code>omegaconf.DictConfig</code> .
<code>resolve_as_dict(obj, config, **kwargs)</code>	Transform the provided object into a dictionary resolving all its interpolations.
<code>safe_select(cfg, key[, default])</code>	Access safely the target value of the provided <code>cfg DictConfig</code> .
<code>as_resolved_dict(cfg)</code>	Return a dictionary containing the resolved values for the provided <code>DictConfig</code> .

## Attributes

<code>DClassField</code>
<code>ConfigValue</code>
<code>ConfigurationDict</code>
<code>PythonType</code>
<code>ParamType</code>
<code>DataClassDict</code>
<code>PARAM_TO_TYPE</code>
<code>CONF_ATTRS</code>

`mloq.config.configuration.DClassField`

**class** `mloq.config.configuration.Dataclass`

Bases: `typing_extensions.Protocol`

Type hinting to defined a dataclass as a typing Protocol.

**`__dataclass_fields__`** :Dict

mloq.config.configuration.**ConfigValue**

mloq.config.configuration.**ConfigurationDict**

mloq.config.configuration.**PythonType**

mloq.config.configuration.**ParamType**

mloq.config.configuration.**DataClassDict**

**class** mloq.config.configuration.**DictConfig**(*default=None, doc=None, instantiate=True, per\_instance=True, \*\*kwargs*)

Bases: mloq.config.param\_patch.param.ClassSelector

param.Parameter that defines a DictConfig object.

#### Parameters

- **default** (*Optional[omegaconf.DictConfig]*) –
- **doc** (*Optional[str]*) –
- **instantiate** (*bool*) –
- **per\_instance** (*bool*) –

mloq.config.configuration.**PARAM\_TO\_TYPE**

mloq.config.configuration.**param\_to\_dataclass\_dict**(*obj*)

Create a dictionary that can be used to initialize a dataclass containing the parameters of the target param.Parameterized class.

**Parameters** *obj* (*Union[mloq.config.param\_patch.param.Parameterized, Any]*) – Class or instance of a param.Parameterized class.

**Returns** dict containing the fields required to define a dataclass with the obj parameters.

**Return type** Dict[str, Tuple[type, DClassField]]

mloq.config.configuration.**param\_to\_dataclass**(*obj*)

Create a dataclass equivalent to the target param.Parameterized target.

**Parameters** *obj* (*Union[mloq.config.param\_patch.param.Parameterized, Any]*) –

**Return type** type

mloq.config.configuration.**param\_to\_omegaconf**(*obj*)

Transform a param.Parameterized class into an OmegaConf structured configuration.

**Parameters** *obj* (*Union[mloq.config.param\_patch.param.Parameterized, Any]*) –

**Return type** omegaconf.DictConfig

mloq.config.configuration.**is\_interpolation**(*s*)

Return True if the provided string is an OmegaConf interpolation string.

**Parameters** *s* (*str*) –

**Return type** bool

mloq.config.configuration.**to\_param\_type**(*obj, config, key*)

Transform the provided attribute of the target param.Parameterized object into the appropriate type so it can be stored in a configuration file.

**Parameters**

- **obj** (*mloq.config.param\_patch.param.Parameterized*) –
- **config** (*DictConfig*) –
- **key** (*str*) –

**Return type** Any

`mloq.config.configuration.to_config(config, **kwargs)`

Transform the provided object into an omegaconf.DictConfig.

**Parameters** **config** (*Union[omegaconf.DictConfig, ConfigurationDict, Dataclass, mloq.config.param\_patch.param.Parameterized, None]*) –

**Return type** omegaconf.DictConfig

`mloq.config.configuration.resolve_as_dict(obj, config, **kwargs)`

Transform the provided object into a dictionary resolving all its interpolations.

**Parameters** **config** (*Union[omegaconf.DictConfig, ConfigurationDict, Dataclass, mloq.config.param\_patch.param.Parameterized]*) –

**Return type** ConfigurationDict

`mloq.config.configuration.safe_select(cfg, key, default=None)`

Access safely the target value of the provided cfg DictConfig.

Return MISSING if the value cannot be resolved or it's missing.

**Parameters**

- **cfg** (*DictConfig*) –
- **key** (*str*) –
- **default** (*Any*) –

**Return type** Any

`mloq.config.configuration.as_resolved_dict(cfg)`

Return a dictionary containing the resolved values for the provided DictConfig.

**Parameters** **cfg** (*DictConfig*) –

**Return type** ConfigurationDict

**class** `mloq.config.configuration.OmegaConfInterface(target, allow_missing=False)`

Common functionality to work with configurations.

**Parameters**

- **target** (*Configurable*) –
- **allow\_missing** (*bool*) –

**property** **config**

Return a DictConfig containing the target configuration.

**Return type** omegaconf.DictConfig

**property** **interpolations**

Return a dictionary containing the interpolations of the target configuration.

**Return type** ConfigurationDict

**property missing**

Return a list containing the names of the configuration that are MISSING.

**Return type** List[Union[str, int, enum.Enum, float, bool]]

**\_resolve\_inplace**(key=None)

Resolve and update the target attribute if it's an interpolation string.

**Parameters** **key** (Optional[str]) –

**Return type** None

**resolve**(key=None, inplace=False)

Resolve the target attribute if it is an interpolation string.

**Parameters**

- **key** (Optional[str]) – Name of the target's attribute to be resolved.
- **inplace** (bool) – If True, update the configuration value replacing the interpolation string with the resolved value.

**Returns** Resolved value of the target attribute.

**Return type** Union[omegaconf.Container, ConfigValue, None]

**is\_missing**(key)

Return True if the key target's attribute is Missing, otherwise return False.

**Parameters** **key** (str) –

**Return type** bool

**is\_interpolation**(key)

Check if the key target's attribute is an interpolation string.

**Parameters** **key** (str) –

**Return type** bool

**select**(key, default=None)

Select the key target's attribute.

Return MISSING if key corresponds to a missing value, or an interpolation that resolves to a missing value.

**Return type** Any

**class** mloq.config.configuration.**BaseConfig**(target, config=None, cfg\_node=None, allow\_missing=False, \*\*kwargs)

Bases: [OmegaConfInterface](#)

Manages getters and setters to access the target's configuration values.

**Parameters**

- **target** ([Configurable](#)) –
- **config** (Optional[Union[ConfigurationDict, omegaconf.DictConfig]]) –
- **cfg\_node** (Optional[str]) –
- **allow\_missing** (bool) –

**\_\_getitem\_\_**(*item*)

Access the target config value.

**Parameters** **item** (*str*) –

**Return type** Any

**\_\_setitem\_\_**(*key*, *value*)

Set the target config value.

**Parameters** **key** (*str*) –

**Return type** Any

**to\_container**(*resolve=False*, *\*\*kwargs*)

Return a container containing the target's configuration.

**Parameters** **resolve** (*bool*) –

**Return type** omegaconf.Container

**static** **\_resolve\_node**(*kwargs*, *config=None*, *cfg\_node=None*)

Return a DictConfig containing the resolved configuration values defined in kwargs.

**Parameters**

- **kwargs** (*dict*) –
- **config** (*Optional[omegaconf.DictConfig]*) –
- **cfg\_node** (*Optional[str]*) –

**Return type** omegaconf.DictConfig

**\_setup\_config**(*config=None*, *cfg\_node=None*, *\*\*kwargs*)

Initialize and validate the structured config of target.

**Parameters**

- **config** (*Optional[Union[ConfigurationDict, omegaconf.DictConfig]]*) –
- **cfg\_node** (*Optional[str]*) –

**class** mloq.config.configuration.**Config**(*target*, *config=None*, *cfg\_node=None*, *allow\_missing=False*, *\*\*kwargs*)

Bases: [BaseConfig](#)

Config handles the *.conf* attribute of a Configurable class.

It is analogous to *.param* for param.Parameterized classes. This class implements all the logic to access and update the config attribute of a Configurable class, which returns a DictConfig instance that is automatically update when the parameters of the class change.

**Parameters**

- **target** ([Configurable](#)) –
- **config** (*Optional[Union[ConfigurationDict, omegaconf.DictConfig]]*) –
- **cfg\_node** (*Optional[str]*) –
- **allow\_missing** (*bool*) –

**property params**

Return the param.Parameter dictionary of the target configurable.

**Return type** Dict[str, mloq.config.param\_patch.param.Parameter]

**resolve**(key=None, inplace=False)

Resolve the key attribute of the target Configurable.

**Parameters**

- **key** (Optional[str]) –
- **inplace** (bool) –

**Return type** Union[omegaconf.Container, ConfigValue, None]

**to\_param\_type**(key)

Transform the value of the key target's parameter to a DictConfig compatible type.

**Return type** Any

**dataclass\_dict**(ignore=None)

Return a dictionary to create a dataclass with the target's parameters.

**Parameters** **ignore** (Optional[Union[list, set, tuple, str]]) –

**Return type** DataClassDict

**to\_dataclass**()

Return a dataclass describing the parameter values of the target Configurable.

**Return type** type

**to\_dictconfig**()

Return a structured DictConfig containing the parameters of the target Configurable.

**Return type** DictConfig

**sync**()

Ensure the parameter values of the target class have the right type.

**\_setup\_config**(config=None, cfg\_node=None, \*\*kwargs)

Initialize and validate the structured config of target.

**Parameters**

- **config** (Optional[Union[ConfigurationDict, DictConfig]]) –
- **cfg\_node** (Optional[str]) –

mloq.config.configuration.CONF\_ATTRS

**class** mloq.config.configuration.**Configurable**(config=None, throw\_on\_missing=True, cfg\_node=None, \*\*kwargs)

Bases: mloq.config.param\_patch.param.Parameterized

A Configurable class is an extension of param.Parameterized that allows to handle parameters with missing values and omegaconf interpolation strings.

It add a config attribute containing an omegaconf.DictConfig that contains the values of the class param.Parameters.

It also provides a *conf* attribute that allows to access omegaconf functionality for managing configurations in a similar fashion as the *param* attribute allows to access param.Parameter functionality.

**Parameters**

- **config** (*Optional[Union[ConfigurationDict, DictConfig]]*) –
- **throw\_on\_missing** (*bool*) –
- **cfg\_node** (*Optional[str]*) –

**config****property conf**

Access the Config instance that tracks and manages the values in the class config.

**Return type** *Config***\_\_setattr\_\_** (*key, value*)

Update the config values when setting a parameter.

**\_\_getattr\_\_** (*item*)

Add support for MISSING values when accessing the parameter values.

**mloq.config.custom\_click**

This is mostly a copy paste from <https://github.com/pallets/click/blob/2fc486c880eda9fdb746ed8baa49416acab9ea6d/src/click/termui.py>

Modified to allow prompt input that has a different color than the prompt text, while keeping the color of the default prompt values the same as the prompt text color.

**Module Contents****Functions**

<i>hidden_prompt_func</i> (prompt)	Input hidden text from the user.
<i>_build_prompt</i> (text, suffix[, show_default, default, ...])	
<i>_format_default</i> (default)	
<i>prompt</i> (text[, default, hide_input, ...])	Prompts a user for input. This is a convenience function that can be used to prompt a user for input later.
<i>confirm</i> (text[, default, abort, prompt_suffix, ...])	Prompts for confirmation (yes/no question).

**Attributes**

<i>visible_prompt_func</i>
<i>_ansi_reset_all</i>

mloq.config.custom\_click.**visible\_prompt\_func**



```

mloq.config.custom_click._ansi_reset_all = [0m
mloq.config.custom_click.hidden_prompt_func(prompt)
    Input hidden text from the user.
mloq.config.custom_click._build_prompt(text, suffix, show_default=False, default=None,
                                       show_choices=True, type=None)

mloq.config.custom_click._format_default(default)
mloq.config.custom_click.prompt(text, default=None, hide_input=False, confirmation_prompt=False,
                                type=None, value_proc=None, prompt_suffix=': ', show_default=True,
                                err=False, show_choices=True)

```

Prompts a user for input. This is a convenience function that can be used to prompt a user for input later.

If the user aborts the input by sending a interrupt signal, this function will catch it and raise a **Abort** exception.

New in version 7.0: Added the `show_choices` parameter.

New in version 6.0: Added unicode support for `cmd.exe` on Windows.

New in version 4.0: Added the `err` parameter.

#### Parameters

- **text** – the text to show for the prompt.
- **default** – the default value to use if no input happens. If this is not given it will prompt until it's aborted.
- **hide\_input** – if this is set to true then the input value will be hidden.
- **confirmation\_prompt** – asks for confirmation for the value.
- **type** – the type to use to check the value against.
- **value\_proc** – if this parameter is provided it's a function that is invoked instead of the type conversion to convert a value.
- **prompt\_suffix** – a suffix that should be added to the prompt.
- **show\_default** – shows or hides the default value in the prompt.
- **err** – if set to true the file defaults to `stderr` instead of `stdout`, the same as with `echo`.
- **show\_choices** – Show or hide choices if the passed type is a `Choice`. For example if type is a `Choice` of either day or week, `show_choices` is true and text is "Group by" then the prompt will be "Group by (day, week): ".

**Returns** None

**Return type** None

```

mloq.config.custom_click.confirm(text, default=False, abort=False, prompt_suffix=': ', show_default=True,
                                err=False)

```

Prompts for confirmation (yes/no question).

If the user aborts the input by sending a interrupt signal this function will catch it and raise a **Abort** exception.

New in version 4.0: Added the `err` parameter.

#### Parameters

- **text** – the question to ask.
- **default** – the default for the prompt.

- **abort** – if this is set to *True* a negative answer aborts the exception by raising **Abort**.
- **prompt\_suffix** – a suffix that should be added to the prompt.
- **show\_default** – shows or hides the default value in the prompt.
- **err** – if set to true the file defaults to `stderr` instead of `stdout`, the same as with `echo`.

**Returns** User’s decision.

**Return type** `bool`

## `mloq.config.param_patch`

Patch param to allow omegaconf Missing values and interpolation strings.

## Module Contents

### Classes

---

<code>__ParamPatcher</code>	Patch the param package to handle missing values and interpolation strings.
-----------------------------	---

---

### Functions

---

<code>__init__patched(self[, default])</code>	Handle missing values and interpolations trings when initializing a <code>param.Parameter</code> .
<code>_create_param__(item)</code>	Patch the target <code>param.Parameter</code> to support missing values and interpolations strings.

---

### Attributes

---

<code>__DEFAULT_MARK</code>
<code>PATCHED_PARAMETERS</code>
<code>param</code>

---

`mloq.config.param_patch.__DEFAULT_MARK = __DEFAULT_MARK__`

`mloq.config.param_patch.__init__patched(self, default=__DEFAULT_MARK, **kwargs)`  
Handle missing values and interpolations trings when initializing a `param.Parameter`.

`mloq.config.param_patch._create_param__(item)`  
Patch the target `param.Parameter` to support missing values and interpolations strings.

`mloq.config.param_patch.PATCHED_PARAMETERS`

```

class mloq.config.param_patch.__ParamPatcher
    Patch the param package to handle missing values and interpolation strings.

    PATCHED_PARAMETERS

    __getattr__(item)
        Patch the parameters included in PATCHED_PARAMETERS.

    abstract __setattr__(key, value)
        Read only monkeypatching.

mloq.config.param_patch.param

```

## mloq.config.prompt

This file contains the logic defining all the parameters needed to set up a project with mloq.

## Module Contents

### Classes

<i>PromptParam</i>	Defines a configuration parameter.
<i>MultiChoicePrompt</i>	Define a configuration parameter that can take multiple values from a pre-defined set of values.
<i>StringPrompt</i>	Define a configuration parameter that can take a string value.
<i>IntPrompt</i>	Define a configuration parameter that can take an integer value.
<i>FloatPrompt</i>	Define a configuration parameter that can take a floating point value.
<i>BooleanPrompt</i>	Defines a boolean configuration parameter.
<i>Prompt</i>	Manage all the functionality needed to display a cli prompt.
<i>Promptable</i>	Configurable class that allows to define the parameter values interactively using CLI prompts.

### Attributes

<i>Choices</i>
<i>PARAM_TO_PROMPT</i>

mloq.config.prompt.**Choices**

```
class mloq.config.prompt.PromptParam(name, target, **kwargs)
```

Defines a configuration parameter.

**It allows to parse a configuration value from different sources in the following order:**

1. Environment variable named as MLOQ\_PARAM\_NAME
2. Values defined in mloq.yaml
3. Interactive prompt from CLI (Optional)

**Parameters**

- **name** (*str*) –
- **target** (`mloq.config.configuration.Configurable`) –

**property param**

Get the param.Parameter object corresponding to the current configuration parameter.

**Return type** param.Parameter

**property value**

Return the value of the configuration parameter.

**Return type** Any

**property config**

Return the value of the parameter as defined in its config DictConfig.

**Return type** Any

**\_\_call\_\_** (*interactive=False, default=None, \*\*kwargs*)

Return the value of the parameter parsing it from the different input sources available.

**Parameters**

- **interactive** (*bool*) – Prompt the user to input the value from CLI if it's not defined in config or as an environment variable.
- **default** (*Optional[Any]*) – Default value displayed in the interactive mode.
- **\*\*kwargs** – Passed to click.prompt in interactive mode. Overrides the values defined in `__init__`

**Returns** Value of the parameter.

**\_prompt** (*value, \*\*kwargs*)

Prompt user for value.

**class** `mloq.config.prompt.MultiChoicePrompt` (*name, target, choices=None, \*\*kwargs*)

Bases: `PromptParam`

Define a configuration parameter that can take multiple values from a pre-defined set of values.

**It allows to parse a configuration value from different sources in the following order:**

1. Environment variable named as MLOQ\_PARAM\_NAME
2. Values defined in mloq.yaml
3. Interactive prompt from CLI (Optional)

**Parameters**

- **name** (*str*) –
- **target** (`mloq.config.configuration.Configurable`) –
- **choices** (*Optional[Choices]*) –

**`_prompt`**(*value*, *\*\*kwargs*)

Transform the parsed string from the CLI into a list of selected values.

**Return type** List[str]

**static** **`_parse_string`**(*value*)

**Return type** List[str]

**class** `mloq.config.prompt.StringPrompt`(*name*, *target*, *\*\*kwargs*)

Bases: `PromptParam`

Define a configuration parameter that can take a string value.

**It allows to parse a configuration value from different sources in the following order:**

1. Environment variable named as MLOQ\_PARAM\_NAME
2. Values defined in mloq.yaml
3. Interactive prompt from CLI (Optional)

#### Parameters

- **`name`** (*str*) –
- **`target`** (`mloq.config.configuration.Configurable`) –

**class** `mloq.config.prompt.IntPrompt`(*name*, *target*, *\*\*kwargs*)

Bases: `PromptParam`

Define a configuration parameter that can take an integer value.

**It allows to parse a configuration value from different sources in the following order:**

1. Environment variable named as MLOQ\_PARAM\_NAME
2. Values defined in mloq.yaml
3. Interactive prompt from CLI (Optional)

#### Parameters

- **`name`** (*str*) –
- **`target`** (`mloq.config.configuration.Configurable`) –

**class** `mloq.config.prompt.FloatPrompt`(*name*, *target*, *\*\*kwargs*)

Bases: `PromptParam`

Define a configuration parameter that can take a floating point value.

**It allows to parse a configuration value from different sources in the following order:**

1. Environment variable named as MLOQ\_PARAM\_NAME
2. Values defined in mloq.yaml
3. Interactive prompt from CLI (Optional)

#### Parameters

- **`name`** (*str*) –
- **`target`** (`mloq.config.configuration.Configurable`) –

**class** mloq.config.prompt.**BooleanPrompt**(*name*, *target*, *\*\*kwargs*)

Bases: [PromptParam](#)

Defines a boolean configuration parameter.

It allows to parse a configuration value from different sources in the following order:

1. Environment variable named as MLOQ\_PARAM\_NAME
2. Values defined in mloq.yaml
3. Interactive prompt from CLI (Optional)

**Parameters**

- **name** (*str*) –
- **target** ([mloq.config.configuration.Configurable](#)) –

**\_prompt**(*value*, *\*\*kwargs*)

Prompt user for value.

mloq.config.prompt.**PARAM\_TO\_PROMPT**

**class** mloq.config.prompt.**Prompt**(*target*)

Manage all the functionality needed to display a cli prompt.

It allows to interactively define the values of the different parameters of a class.

**Parameters** **target** ([Promptable](#)) –

**\_\_call\_\_**(*key*, *inplace=False*, *\*\*kwargs*)

Display the a prompt to interactively define the parameter values of target.

**Parameters**

- **key** (*str*) –
- **inplace** (*bool*) –

**Return type** Any

**\_init\_prompts**()

Initialize the prompts corresponding to the target Promptable parameters.

**Return type** None

**prompt**(*key*, *inplace=False*, *\*\*kwargs*)

Display the a prompt to interactively define the parameter values of target.

**Parameters**

- **key** (*str*) –
- **inplace** (*bool*) –

**Return type** Any

**prompt\_all**(*inplace=False*, *\*\*kwargs*)

Prompt all the target's parameters.

Return a dictionary containing the provided values.

**Parameters** **inplace** (*bool*) –

**Return type** Dict[str, Any]

**class** `mloq.config.prompt.Promptable(**kwargs)`

Bases: `mloq.config.configuration.Configurable`

Configurable class that allows to define the parameter values interactively using CLI prompts.

It contains a prompt attribute in charge of managing the prompting functionality for the param.Parameters defined.

## 17.1.2 Submodules

### 17.1.2.1 `mloq.__main__`

Command line interface for mloq.

### 17.1.2.2 `mloq._utils`

This module contains some utilities that are not currently used.

#### 17.1.2.2.1 Module Contents

##### Functions

<code>dir_trees_are_equal(dir1, dir2)</code>	Compare two directories recursively. Files in each directory are assumed to be equal if their names and contents are equal.
<code>files_are_equal(path1, path2)</code>	Compare the content of two files.
<code>get_generated_files(path)</code>	List all the files generated in the last mloq run.
<code>get_generated_directories(path)</code>	List all the directories generated in the last mloq run.
<code>check_directories_exist(paths)</code>	Check if the provided paths exist.
<code>get_docker_python_version(template)</code>	Return the highest python version defined for the project.
<code>write_config_setup(config, path[, safe])</code>	Write setup config in a yaml file.
<code>load_empty_config_setup()</code>	Return a dictionary containing all the MLOQ setup config values set to None.

`mloq._utils.dir_trees_are_equal(dir1, dir2)`

Compare two directories recursively. Files in each directory are assumed to be equal if their names and contents are equal.

@param dir1: First directory path @param dir2: Second directory path

@return: **True if the directory trees are the same and** there were no errors while accessing the directories or files, False otherwise.

##### Parameters

- **dir1** (`Union[str, pathlib.Path]`) –
- **dir2** (`Union[str, pathlib.Path]`) –

**Return type** bool

`mloq._utils.files_are_equal(path1, path2)`

Compare the content of two files.

Compare two incoming files. They are assumed equal if their contents are the same.

**Parameters**

- **path1** (*Union[str, pathlib.Path]*) – Path containing the first file to be compared.
- **path2** (*Union[str, pathlib.Path]*) – Path containing the second file to be compared.

**Returns**

**It returns True if the two given files are equal and no errors** have arisen during the process.  
False otherwise.

**Return type** bool

`mloq._utils.get_generated_files(path)`

List all the files generated in the last mloq run.

**Parameters** **path** (*Union[str, pathlib.Path]*) – path to WHAT\_MLOQ\_GENERATED.md file.

**Returns** List of Path containing the names of the files generated by mloq.

**Return type** List[pathlib.Path]

`mloq._utils.get_generated_directories(path)`

List all the directories generated in the last mloq run.

**Parameters** **path** (*Union[str, pathlib.Path]*) – path to WHAT\_MLOQ\_GENERATED.md file.

**Returns** List of Path containing the names of the directories generated by mloq.

**Return type** List[pathlib.Path]

`mloq._utils.check_directories_exist(paths)`

Check if the provided paths exist.

**Parameters** **paths** (*List[Union[str, pathlib.Path]]*) – List of paths that will be checked

**Returns** True if all the provided paths exist. False otherwise.

**Return type** bool

`mloq._utils.get_docker_python_version(template)`

Return the highest python version defined for the project.

**Parameters** **template** (*omegaconf.DictConfig*) –

**Return type** str

`mloq._utils.write_config_setup(config, path, safe=False)`

Write setup config in a yaml file.

**Parameters**

- **config** (*omegaconf.DictConfig*) –
- **path** (*Union[pathlib.Path, str]*) –
- **safe** (*bool*) –



`mloq._utils.load_empty_config_setup()`

Return a dictionary containing all the MLOQ setup config values set to None.

**Return type** `omegaconf.DictConfig`

### 17.1.2.3 `mloq.cli`

Command line interface for mloq.

#### 17.1.2.3.1 Module Contents

##### Classes

<i><code>MloqCLI</code></i>	Load the commands available at runtime from the files present in the command module.
-----------------------------	--

##### Functions

<i><code>mloq_click_command(func)</code></i> <i><code>cli()</code></i>	Wrap a command function to interface with click.
<i><code>welcome_message([extra, string])</code></i>	Welcome message to be displayed during interactive setup.

##### Attributes

<i><code>overwrite_opt</code></i>
<i><code>config_file_opt</code></i>
<i><code>only_config_opt</code></i>
<i><code>output_directory_arg</code></i>
<i><code>interactive_opt</code></i>
<i><code>hydra_args</code></i>

`mloq.cli.overwrite_opt`

`mloq.cli.config_file_opt`

`mloq.cli.only_config_opt`

`mloq.cli.output_directory_arg`

`mloq.cli.interactive_opt`

`mloq.cli.hydra_args`

`mloq.cli.mloq_click_command(func)`

Wrap a command function to interface with click.

**class** `mloq.cli.MloqCLI`(*name=None, invoke\_without\_command=False, no\_args\_is\_help=None, subcommand\_metavar=None, chain=False, result\_callback=None, \*\*attrs*)

Bases: `click.MultiCommand`

Load the commands available at runtime from the files present in the command module.

#### Parameters

- **name** (*Optional[str]*) –
- **invoke\_without\_command** (*bool*) –
- **no\_args\_is\_help** (*Optional[bool]*) –
- **subcommand\_metavar** (*Optional[str]*) –
- **chain** (*bool*) –
- **result\_callback** (*Optional[Callable[Ellipsis, Any]]*) –
- **attrs** (*Any*) –

**command\_folder**

**list\_commands**(*ctx*)

List the names of the mloq commands available.

**get\_command**(*ctx, name*)

Create the command callable corresponding to the provided command name.

**Return type** `Callable`

`mloq.cli.cli()`

`mloq.cli.welcome_message(extra=False, string=None)`

Welcome message to be displayed during interactive setup.

#### Parameters

- **extra** (*bool*) –
- **string** (*Optional[str]*) –

### 17.1.2.4 `mloq.command`

This module defines the base `Command` class used for defining mloq commands.

### 17.1.2.4.1 Module Contents

#### Classes

<i>CommandMixin</i>	Class containing the interface for defining an MLOQ Command.
<i>Command</i>	Define blueprints for generating custom mloq commands.

**class** mloq.command.**CommandMixin**(*record*, *interactive=False*, \*args, \*\*kwargs)

Class containing the interface for defining an MLOQ Command.

#### Parameters

- **record** (*mloq.writer.CMDRecord*) –
- **interactive** (*bool*) –

**files** :tuple

**cmd\_name** = command

**property record**

Return a CMDRecord that keeps track of the files and directories the command creates.

**Return type** mloq.writer.CMDRecord

**property directories**

Tuple containing Paths objects representing the directories the Command creates.

Override this property if your command creates any directories.

**Returns** Tuple of Path objects representing the path to the directories that the *Command* will create.

**Return type** Tuple[pathlib.Path]

**parse\_config()**

Update the configuration dictionary from the data entered by the user.

Given the basic configuration skeleton (contained in mloq.yaml), this method updates the values of those parameters (included in CONFIG object) that are related to the selected command. Incoming values are introduced either interactively or via a custom user's mloq.yaml file.

**Returns** It returns an updated version of the 'config' attribute of the 'record' instance.

**Return type** omegaconf.DictConfig

**abstract interactive\_config()**

Pass user's configuration interactively.

**Return type** omegaconf.DictConfig

**record\_files()**

Register the files that will be generated by mloq.

**Return type** None

**record\_directories()**

Register the directories that will be generated by mloq.

**Return type** None

**configure()**

Save the updated version of the 'config' attribute.

After parsing the new configuration values introduced by the user, this method registers and saves this updated configuration within the '\_config' attribute of the 'record' instance.

**Return type** None

**run\_side\_effects()**

Apply additional configuration methods.

**Return type** None

**run()**

Record the files and directories generated by mloq according to the user's configuration.

This method updates the configuration dictionary with the values introduced by the user. Once the parameters have been revised, the files and directories that will be generated by mloq are registered within the 'record' instance.

**Returns**

**It returns an updated version of the CMDRecord instance, where** the files and directories that will be generated by mloq are recorded within the 'record' instance.

**Return type** mloq.writer.CMDRecord

**class** mloq.command.**Command**(*record*, *interactive=False*, *\*\*kwargs*)

Bases: [\*CommandMixin\*](#), [\*mloq.config.prompt.Promptable\*](#)

Define blueprints for generating custom mloq commands.

Base class used for defining new mloq commands. It establishes the fundamental methods for defining and updating the configuration values used to create the necessary files for the user's project, while registering the latter for later use.

This class is initialized from a CMDRecord instance, object where the user's configuration, as well as the files and directories that will be generated, are stored.

**class Attributes:** *name*: Name of the command. *files*: Tuple containing the templates used for creating the necessary files of your project. *CONFIG*: NamedTuple containing the keys and values of your configuration RELATIVE to the command.

**Parameters**

- **record** (*mloq.writer.CMDRecord*) –
- **interactive** (*bool*) –

**interactive\_config()**

Pass user's configuration interactively.

**Return type** omegaconf.DictConfig

**\_config\_from\_record()**

**Return type** omegaconf.DictConfig

### 17.1.2.5 mloq.custom\_click

This is mostly a copy paste from <https://github.com/pallets/click/blob/2fc486c880eda9fdb746ed8baa49416acab9ea6d/src/click/termui.py>

Modified to allow prompt input that has a different color than the prompt text, while keeping the color of the default prompt values the same as the prompt text color.

#### 17.1.2.5.1 Module Contents

##### Functions

<code>hidden_prompt_func(prompt)</code>	Input hidden text from the user.
<code>_build_prompt(text, suffix[, show_default, default, ...])</code>	
<code>_format_default(default)</code>	
<code>prompt(text[, default, hide_input, ...])</code>	Prompts a user for input. This is a convenience function that can be used to prompt a user for input later.
<code>confirm(text[, default, abort, prompt_suffix, ...])</code>	Prompts for confirmation (yes/no question).

##### Attributes

<code>visible_prompt_func</code>
<code>_ansi_reset_all</code>

`mloq.custom_click.visible_prompt_func`

`mloq.custom_click._ansi_reset_all = [0m`

`mloq.custom_click.hidden_prompt_func(prompt)`

Input hidden text from the user.

`mloq.custom_click._build_prompt(text, suffix, show_default=False, default=None, show_choices=True, type=None)`

`mloq.custom_click._format_default(default)`

`mloq.custom_click.prompt(text, default=None, hide_input=False, confirmation_prompt=False, type=None, value_proc=None, prompt_suffix=': ', show_default=True, err=False, show_choices=True)`

Prompts a user for input. This is a convenience function that can be used to prompt a user for input later.

If the user aborts the input by sending a interrupt signal, this function will catch it and raise a `Abort` exception.

New in version 7.0: Added the `show_choices` parameter.

New in version 6.0: Added unicode support for `cmd.exe` on Windows.

New in version 4.0: Added the `err` parameter.

**Parameters**

- **text** – the text to show for the prompt.
- **default** – the default value to use if no input happens. If this is not given it will prompt until it's aborted.
- **hide\_input** – if this is set to true then the input value will be hidden.
- **confirmation\_prompt** – asks for confirmation for the value.
- **type** – the type to use to check the value against.
- **value\_proc** – if this parameter is provided it's a function that is invoked instead of the type conversion to convert a value.
- **prompt\_suffix** – a suffix that should be added to the prompt.
- **show\_default** – shows or hides the default value in the prompt.
- **err** – if set to true the file defaults to `stderr` instead of `stdout`, the same as with `echo`.
- **show\_choices** – Show or hide choices if the passed type is a `Choice`. For example if type is a `Choice` of either day or week, `show_choices` is true and text is "Group by" then the prompt will be "Group by (day, week): ".

**Returns** None**Return type** None

```
mloq.custom_click.confirm(text, default=False, abort=False, prompt_suffix=' ', show_default=True, err=False)
```

Prompts for confirmation (yes/no question).

If the user aborts the input by sending an interrupt signal this function will catch it and raise a `Abort` exception.

New in version 4.0: Added the `err` parameter.

**Parameters**

- **text** – the question to ask.
- **default** – the default for the prompt.
- **abort** – if this is set to `True` a negative answer aborts the exception by raising `Abort`.
- **prompt\_suffix** – a suffix that should be added to the prompt.
- **show\_default** – shows or hides the default value in the prompt.
- **err** – if set to true the file defaults to `stderr` instead of `stdout`, the same as with `echo`.

**Returns** User's decision.**Return type** bool

17.1.2.6 mloq.failure

Define Failure - the exception to raise when we break.

17.1.2.6.1 Module Contents

**exception** mloq.failure.Failure

Bases: Exception

Raised when a project setup critical error happens.

**\_\_str\_\_**()

Delegate **\_\_str\_\_** to the underlying cause if it exists.

**Return type** str

**exception** mloq.failure.MissingConfigValue

Bases: Failure

Raised when a parameter is not defined in the mloq DictConfig.

17.1.2.7 mloq.files

This module defines all the different assets accessible from mloq.

17.1.2.7.1 Module Contents

Classes

<i>File</i>	Generates project files.
-------------	--------------------------

Functions

<i>file</i> (name, path[, description, dst, is_static])	Define a new asset as a File namedtuple.
<i>read_file</i> (file)	Return and string with the content of the provided file.

## Attributes

---

*ASSETS\_PATH*

---

*SHARED\_ASSETS\_PATH*

---

*MLOQ\_ASSETS\_PATH*

---

*REQUIREMENTS\_PATH*

---

*mloq.yml*

---

*what\_mloq\_generated*

---

*requirements*

---

*dogfood\_req*

---

*makefile*

---

*pyproject.toml*

---

### **class** mloq.files.File

Bases: NamedTuple

Generates project files.

This class defines templating files, which will be rendered according to the user's configuration. Besides, File instances have additional attributes used for specifying the destination of the generated file.

**Attributes of this class:** **name:** Name of the templating file. **src:** Location of the templating file. **dst:** Name of the file generated from the templating file. **description:** Short description of the current file. **is\_static:** Boolean value. If True, the templating file does not

admit render parameters.

**name** :str

**src** :pathlib.Path

**dst** :pathlib.Path

**description** :str

**is\_static** :bool

mloq.files.**file**(name, path, description=None, dst=None, is\_static=False)

Define a new asset as a File namedtuple.

#### **Parameters**

- **name** (str) –
- **path** (Union[pathlib.Path, str]) –
- **description** (Optional[str]) –



- **dst** (*Optional[Union[pathlib.Path, str]]*) –
- **is\_static** (*bool*) –

**Return type** *File*

`mloq.files.ASSETS_PATH`

`mloq.files.SHARED_ASSETS_PATH`

`mloq.files.MLOQ_ASSETS_PATH`

`mloq.files.REQUIREMENTS_PATH`

`mloq.files.mloq_yaml`

`mloq.files.what_mloq_generated`

`mloq.files.requirements`

`mloq.files.dogfood_req`

`mloq.files.makefile`

`mloq.files.pyproject_toml`

`mloq.files.read_file(file)`

Return and string with the content of the provided file.

**Parameters** **file** (*File*) –

**Return type** *str*

#### 17.1.2.8 mloq.git

Setup Git repository for the project.

##### 17.1.2.8.1 Module Contents

#### Functions

---

<code>setup_git(path, config)</code>	Initialize a Git repository over the generated files.
<code>_git_cmd(git_dir, *parts)</code>	

---

`mloq.git.setup_git(path, config)`

Initialize a Git repository over the generated files.

**Parameters**

- **path** (*Union[pathlib.Path, str]*) –
- **config** (*omegaconf.DictConfig*) –

**Return type** *None*

```
mloq.git._git_cmd(git_dir, *parts)
```

**Parameters**

- **git\_dir** (*str*) –
- **parts** (*str*) –

**Return type** None

### 17.1.2.9 mloq.record

This module contains the classes that keep track of the internal state of the application when running a Command.

#### 17.1.2.9.1 Module Contents

**Classes**

<i>Ledger</i>	Keep track of the generated files.
<i>CMDRecord</i>	Keep track of files and directories that will be created by mloq.

```
class mloq.record.Ledger
```

Keep track of the generated files.

**property files**

Return the list of generated file names.

**Return type** List[Tuple[str, str]]

```
register(file, description=None)
```

Append another generated file to the book.

**Parameters**

- **file** (*Union*[*mloq.files.File*, *str*, *pathlib.Path*]) –
- **description** (*Optional*[*str*]) –

**Return type** None

```
class mloq.record.CMDRecord(config=None, files=None, directories=None)
```

Keep track of files and directories that will be created by mloq.

The *CMDRecord* acts as a single source of truth for storing the files and directories generated by *mloq* as well as the necessary configuration to generate them.

This class registers three separate data sources:

- **config**: An *omegaconf.DictConfig* that contains the configuration of all the commands executed by *mloq*.
- **files**: A *dict* containing the content and location of the files generated by mloq. It is indexed by *Path* objects that indicate where the file will be created, and its values are instances of *mloq.files.File*.
- **directories**: List of *Path* instances pointing to the different directories that *mloq* will create.

This class is initialized from a configuration dictionary. The dictionary can be either an *omegaconf.DictConfig* or an empty dictionary.

**Parameters**

- **config** (*Optional*[*omegaconf.DictConfig*]) –
- **files** (*Optional*[*Dict*[*pathlib.Path*, *mloq.files.File*]]) –
- **directories** (*Optional*[*List*[*pathlib.Path*]]) –

**property config**

Store the configuration parameters that govern the project's structure.

It contains one configuration entry per each `Command` that will be run, and each entry will only be modified by the `Command` it represents. Each `Command` instance is responsible for updating its corresponding *config* values.

**Returns** *omegaconf.DictConfig* that contains the configuration of all the commands executed by *mloq*.

**Return type** *omegaconf.DictConfig*

**property files**

Return the dictionary of files used by *mloq* to generate the project configuration.

*files* is a *dict* containing the content and location of the files generated by *mloq*. It is indexed by *Path* objects that indicate where the file will be created, and its values are instances of *mloq.files.File*.

Each different `Command` is responsible for registering the files it generates to the *files* dictionary.

**Return type** *Dict*[*pathlib.Path*, *mloq.files.File*]

**property directories**

Contain the folders that will be created by *mloq* for storing the project's files.

**Return type** *List*[*pathlib.Path*]

**update\_config(config)**

Update the configuration attribute according to the values entered by the user.

**Parameters** **config** (*omegaconf.DictConfig*) –

**Return type** *None*

**register\_file(file, path, description=None)**

Append a new file to the 'files' container.

Keys are *Path* strings describing the location where the file will be created. Values are *File* objects containing the information of the file that will be generated.

**Parameters**

- **file** (*mloq.files.File*) –
- **path** (*Union*[*pathlib.Path*, *str*]) –
- **description** (*Optional*[*str*]) –

**Return type** *None*

**register\_directory(path)**

Append a new directory path to the 'directories' container.

**Parameters** **path** (*Union*[*pathlib.Path*, *str*]) –

**Return type** *None*

### 17.1.2.10 mloq.runner

This module defines the pipeline for running a mloq command, such as config loading, template writing and interfacing with click.

#### 17.1.2.10.1 Module Contents

##### Functions

<code>load_config(config_file, hydra_args)</code>	Load the necessary configuration for running mloq from a mloq.yaml file.
<code>write_record(record, path[, overwrite, only_config])</code>	Write the contents of the provided record to the target path.
<code>run_command(cmd_cls[, use_click])</code>	Run the given Command class.

`mloq.runner.load_config(config_file, hydra_args)`

Load the necessary configuration for running mloq from a mloq.yaml file.

If no path to mloq.yaml is provided, it returns a template to be filled in using the interactive mode.

##### Parameters

- **config\_file** (`Union[pathlib.Path, str]`) – Path to the target mloq.yaml file.
- **hydra\_args** (`str`) – Arguments passed to hydra for composing the project configuration.

**Returns** DictConfig containing the project configuration.

**Return type** omegaconf.DictConfig

`mloq.runner.write_record(record, path, overwrite=False, only_config=False)`

Write the contents of the provided record to the target path.

The writing process is performed by :class: *Writer*, class that fills in rendered templates according to the given configuration.

##### Parameters

- **record** (`mloq.record.CMDRecord`) – CMDRecord containing all the data to be written.
- **path** (`Union[pathlib.Path, str]`) – Target directory to write the data.
- **overwrite** (`bool`) – If True overwrite existing files.
- **only\_config** (`bool`) – Do not write any file except mloq.yaml

**Returns** None.

**Return type** None

`mloq.runner.run_command(cmd_cls, use_click=True)`

Run the given Command class.

##### Parameters

- **cmd\_cls** – Command to be executed.
- **use\_click** (`bool`) – If True run the function as a “cli” command.

**Returns** A function that will run the target class as a mloq command.

**Return type** Callable

### 17.1.2.11 mloq.templating

This module defines common functionality for rendering and writing File templates.

#### 17.1.2.11.1 Module Contents

##### Functions

<code>render_template(file, kwargs)</code>	Render a jinja template with the provided parameter dict.
<code>write_template(file, config, path, ledger[, overwrite])</code>	Create new file containing the rendered template found in <code>source_path</code> .

##### Attributes

<code>jinja_env</code>	
------------------------	--

`mloq.templating.jinja_env`

`mloq.templating.render_template(file, kwargs)`

Render a jinja template with the provided parameter dict.

##### Parameters

- **file** (`mloq.files.File`) – File object representing the jinja template that will be rendered.
- **kwargs** (`Mapping[str, Any]`) – Dictionary containing the parameters key and corresponding values that will be used to render the template.

**Returns** String containing the rendered template.

**Return type** str

`mloq.templating.write_template(file, config, path, ledger, overwrite=False)`

Create new file containing the rendered template found in `source_path`.

##### Parameters

- **file** (`mloq.files.File`) – File object representing the jinja template that will be rendered.
- **config** (`omegaconf.DictConfig`) – OmegaConf dictionary containing the parameters key and corresponding values that will be used to render the templates.
- **path** (`Union[pathlib.Path, str]`) – Absolute path to the folder where the file will be written.
- **ledger** (`mloq.record.Ledger`) – Book keeper to keep track of the generated files.
- **overwrite** (`bool`) – If False, copy the file if it does not already exists in the target path. If True, overwrite the target file if it is already present.

**Returns** None.

#### 17.1.2.12 `mloq.version`

Current version of the project. Do not modify manually.

##### 17.1.2.12.1 Module Contents

```
mloq.version.__version__ = 0.0.75
```

#### 17.1.2.13 `mloq.writer`

The writer module defines the `Writer` class, which is in charge of creating the files and directories specified in the `CMDRecord`.

##### 17.1.2.13.1 Module Contents

#### Classes

---

<i>Writer</i>	Write all the files specified on the provided <code>CMDRecord</code> .
---------------	--

---

```
class mloq.writer.Writer(path, record, overwrite=False)
```

Write all the files specified on the provided `CMDRecord`.

This class fills in rendered templates according to the provided configuration and generates the resulting file on the specified folder.

#### Attributes of this class:

**path:** Path string describing the destination folder where the file will be created.

**ledger:** Instance of the `Ledger` class. It contains a dictionary summarizing the files that will be generated by `mloq`.

**record:** Instance of the `CMDRecord` class. It keeps track of all files and directories that will be created from the user's configuration.

**overwrite:** Boolean value. If `True`, existing files will be rewritten by `mloq` application.

#### Parameters

- **path** (*Union[pathlib.Path, str]*) –
- **record** (`mloq.record.CMDRecord`) –
- **overwrite** (*bool*) –

#### property path

Path string describing the location where the files will be generated.

**Return type** `pathlib.Path`

#### property ledger

Keep track of the generated files.

**Return type** `mloq.record.Ledger`

**property record**

Register the files and directories generated by the mloq application.

**Return type** *mloq.record.CMDRecord*

**create\_directories()**

Create the folders registered inside the attribute 'record.directories'.

**Return type** None

**write\_templates()**

Generate the files recorded in the attribute 'record.files' on the specified path.

**Return type** None

**dump\_ledger()**

Write the summary of the generated files.

This method collects the elements stored in ledger to create a markdown document that summarizes all the files generated by the MLOQ application.

**Return type** None

**write\_template(file, path, config)**

Create new file containing the rendered template configuration.

**Parameters**

- **file** (*mloq.files.File*) – File object representing the jinja template that will be rendered.
- **path** (*pathlib.Path*) – Target folder where the generated files will be written.
- **config** (*omegaconf.DictConfig*) – DictConfig containing the selected project configuration.

**Returns** None.

**Return type** None

**run()**

Generate all files and directories registered inside the record instance.

**Return type** None

### 17.1.3 Package Contents

**mloq.\_logger**





## INDICES AND TABLES

genindex modindex search



## PYTHON MODULE INDEX

### m

- mloq, 35
- mloq.\_\_main\_\_, 75
- mloq.\_utils, 75
- mloq.cli, 77
- mloq.command, 78
- mloq.commands, 35
  - mloq.commands.ci, 35
  - mloq.commands.docker, 37
  - mloq.commands.docs, 39
  - mloq.commands.globals, 41
  - mloq.commands.license, 42
  - mloq.commands.lint, 43
  - mloq.commands.package, 44
  - mloq.commands.project, 46
  - mloq.commands.requirements, 48
  - mloq.commands.setup, 51
- mloq.config, 61
  - mloq.config.configuration, 61
  - mloq.config.custom\_click, 68
  - mloq.config.param\_patch, 70
  - mloq.config.prompt, 71
- mloq.custom\_click, 81
- mloq.failure, 83
- mloq.files, 83
- mloq.git, 85
- mloq.record, 86
- mloq.runner, 88
- mloq.templating, 89
- mloq.version, 90
- mloq.writer, 90



## Symbols

\_\_DEFAULT\_MARK (in module *mloq.config.param\_patch*), 70  
 \_\_ParamPatcher (class in *mloq.config.param\_patch*), 70  
 \_\_call\_\_() (*mloq.config.prompt.Prompt* method), 74  
 \_\_call\_\_() (*mloq.config.prompt.PromptParam* method), 72  
 \_\_dataclass\_fields\_\_ (*mloq.config.configuration.Dataclass* attribute), 62  
 \_\_del\_\_() (*mloq.commands.RequirementsCMD* method), 59  
 \_\_del\_\_() (*mloq.commands.requirements.RequirementsCMD* method), 50  
 \_\_getattr\_\_() (*mloq.config.configuration.Configurable* method), 68  
 \_\_getattr\_\_() (*mloq.config.param\_patch.\_\_ParamPatcher* method), 71  
 \_\_getitem\_\_() (*mloq.config.configuration.BaseConfig* method), 65  
 \_\_init\_patched\_\_() (in module *mloq.config.param\_patch*), 70  
 \_\_setattr\_\_() (*mloq.config.configuration.Configurable* method), 68  
 \_\_setattr\_\_() (*mloq.config.param\_patch.\_\_ParamPatcher* method), 71  
 \_\_setitem\_\_() (*mloq.config.configuration.BaseConfig* method), 66  
 \_\_str\_\_() (*mloq.failure.Failure* method), 83  
 \_\_version\_\_ (in module *mloq.version*), 90  
 \_ansi\_reset\_all (in module *mloq.config.custom\_click*), 68  
 \_ansi\_reset\_all (in module *mloq.config.custom\_click*), 81  
 \_build\_prompt() (in module *mloq.config.custom\_click*), 69  
 \_build\_prompt() (in module *mloq.config.custom\_click*), 81  
 \_config\_from\_record() (*mloq.command.Command* method), 80  
 \_create\_param\_\_() (in module *mloq.config.param\_patch*), 70  
 \_format\_default() (in module *mloq.config.custom\_click*), 69  
 \_format\_default() (in module *mloq.config.custom\_click*), 81  
 \_git\_cmd() (in module *mloq.git*), 85  
 \_init\_prompts() (*mloq.config.prompt.Prompt* method), 74  
 \_logger (in module *mloq*), 91  
 \_parse\_string() (*mloq.config.prompt.MultiChoicePrompt* static method), 73  
 \_prompt() (*mloq.config.prompt.BooleanPrompt* method), 74  
 \_prompt() (*mloq.config.prompt.MultiChoicePrompt* method), 73  
 \_prompt() (*mloq.config.prompt.PromptParam* method), 72  
 \_resolve\_inplace() (*mloq.config.configuration.OmegaConfInterface* method), 65  
 \_resolve\_node() (*mloq.config.configuration.BaseConfig* static method), 66  
 \_setup\_config() (*mloq.config.configuration.BaseConfig* method), 66  
 \_setup\_config() (*mloq.config.configuration.Config* method), 67  
 \_sub\_commands() (in module *mloq.commands.setup*), 51

## A

apache\_license (in module *mloq.commands.license*), 42  
 as\_resolved\_dict() (in module *mloq.config.configuration*), 64  
 ASSETS\_PATH (in module *mloq.files*), 85  
 author (*mloq.commands.ci.CiCMD* attribute), 36  
 author (*mloq.commands.CiCMD* attribute), 53  
 author (*mloq.commands.docs.DocsCMD* attribute), 40  
 author (*mloq.commands.DocsCMD* attribute), 55  
 author (*mloq.commands.globals.GlobalsCMD* attribute), 41  
 author (*mloq.commands.GlobalsCMD* attribute), 56  
 author (*mloq.commands.package.PackageCMD* attribute), 45  
 author (*mloq.commands.PackageCMD* attribute), 58

## B

`base_image` (*mloq.commands.docker.DockerCMD* attribute), 38

`base_image` (*mloq.commands.DockerCMD* attribute), 54

`BaseConfig` (class in *mloq.config.configuration*), 65

`black` (*mloq.commands.lint.LintCMD* attribute), 44

`black` (*mloq.commands.LintCMD* attribute), 57

`BooleanPrompt` (class in *mloq.config.prompt*), 73

`bot_email` (*mloq.commands.ci.CiCMD* attribute), 36

`bot_email` (*mloq.commands.CiCMD* attribute), 53

`bot_name` (*mloq.commands.ci.CiCMD* attribute), 36

`bot_name` (*mloq.commands.CiCMD* attribute), 53

## C

`check_directories_exist()` (in module *mloq.\_utils*), 76

`Choices` (in module *mloq.config.prompt*), 71

`CI_ASSETS_PATH` (in module *mloq.commands.ci*), 36

`ci_extra` (*mloq.commands.ci.CiCMD* attribute), 36

`ci_extra` (*mloq.commands.CiCMD* attribute), 53

`ci_python_version` (*mloq.commands.ci.CiCMD* attribute), 36

`ci_python_version` (*mloq.commands.CiCMD* attribute), 53

`CiCMD` (class in *mloq.commands*), 52

`CiCMD` (class in *mloq.commands.ci*), 36

`cli()` (in module *mloq.cli*), 78

`cmd_name` (*mloq.command.CommandMixin* attribute), 79

`cmd_name` (*mloq.commands.ci.CiCMD* attribute), 36

`cmd_name` (*mloq.commands.CiCMD* attribute), 52

`cmd_name` (*mloq.commands.docker.DockerCMD* attribute), 38

`cmd_name` (*mloq.commands.DockerCMD* attribute), 53

`cmd_name` (*mloq.commands.docs.DocsCMD* attribute), 40

`cmd_name` (*mloq.commands.DocsCMD* attribute), 55

`cmd_name` (*mloq.commands.globals.GlobalsCMD* attribute), 41

`cmd_name` (*mloq.commands.GlobalsCMD* attribute), 55

`cmd_name` (*mloq.commands.license.LicenseCMD* attribute), 42

`cmd_name` (*mloq.commands.LicenseCMD* attribute), 56

`cmd_name` (*mloq.commands.lint.LintCMD* attribute), 44

`cmd_name` (*mloq.commands.LintCMD* attribute), 57

`cmd_name` (*mloq.commands.package.PackageCMD* attribute), 45

`cmd_name` (*mloq.commands.PackageCMD* attribute), 58

`cmd_name` (*mloq.commands.project.ProjectCMD* attribute), 48

`cmd_name` (*mloq.commands.ProjectCMD* attribute), 58

`cmd_name` (*mloq.commands.requirements.RequirementsCMD* attribute), 50

`cmd_name` (*mloq.commands.RequirementsCMD* attribute), 59

`cmd_name` (*mloq.commands.setup.SetupCMD* attribute), 51

`cmd_name` (*mloq.commands.SetupCMD* attribute), 60

`CMDRecord` (class in *mloq.record*), 86

`code_of_conduct` (in module *mloq.commands.project*), 47

`codecov` (in module *mloq.commands.project*), 47

`Command` (class in *mloq.command*), 80

`command_folder` (*mloq.cli.MloqCLI* attribute), 78

`CommandMixin` (class in *mloq.command*), 79

`compose_requirements()`  
(*mloq.commands.requirements.RequirementsCMD* class method), 50

`compose_requirements()`  
(*mloq.commands.RequirementsCMD* class method), 60

`conf` (*mloq.config.configuration.Configurable* property), 68

`CONF_ATTRS` (in module *mloq.config.configuration*), 67

`conf_py` (in module *mloq.commands.docs*), 40

`Config` (class in *mloq.config.configuration*), 66

`config` (*mloq.commands.setup.SetupCMD* property), 51

`config` (*mloq.commands.SetupCMD* property), 60

`config` (*mloq.config.configuration.Configurable* attribute), 68

`config` (*mloq.config.configuration.OmegaConfInterface* property), 64

`config` (*mloq.config.prompt.PromptParam* property), 72

`config` (*mloq.record.CMDRecord* property), 87

`config_file_opt` (in module *mloq.cli*), 77

`Configurable` (class in *mloq.config.configuration*), 67

`ConfigurationDict` (in module *mloq.config.configuration*), 63

`configure()` (*mloq.command.CommandMixin* method), 80

`ConfigValue` (in module *mloq.config.configuration*), 62

`confirm()` (in module *mloq.config.custom\_click*), 69

`confirm()` (in module *mloq.custom\_click*), 82

`contributing` (in module *mloq.commands.project*), 47

`copyright_holder` (*mloq.commands.docs.DocsCMD* attribute), 40

`copyright_holder` (*mloq.commands.DocsCMD* attribute), 55

`copyright_holder` (*mloq.commands.license.LicenseCMD* attribute), 43

`copyright_holder` (*mloq.commands.LicenseCMD* attribute), 56

`copyright_year` (*mloq.commands.docs.DocsCMD* attribute), 40

`copyright_year` (*mloq.commands.DocsCMD* attribute), 55

`copyright_year` (*mloq.commands.license.LicenseCMD*

attribute), 43  
 copyright\_year (*mloq.commands.LicenseCMD* attribute), 56  
 create\_directories() (*mloq.writer.Writer* method), 91  
 cuda (*mloq.commands.docker.DockerCMD* attribute), 38  
 cuda (*mloq.commands.DockerCMD* attribute), 54  
 cuda\_image\_type (*mloq.commands.docker.DockerCMD* attribute), 38  
 cuda\_image\_type (*mloq.commands.DockerCMD* attribute), 54  
 cuda\_version (*mloq.commands.docker.DockerCMD* attribute), 38  
 cuda\_version (*mloq.commands.DockerCMD* attribute), 54

## D

data\_science\_req (in module *mloq.commands.requirements*), 49  
 data\_viz\_req (in module *mloq.commands.requirements*), 49  
 Dataclass (class in *mloq.config.configuration*), 62  
 dataclass\_dict() (*mloq.config.configuration.Config* method), 67  
 DataClassDict (in module *mloq.config.configuration*), 63  
 DClassField (in module *mloq.config.configuration*), 62  
 dco (in module *mloq.commands.license*), 42  
 default\_branch (*mloq.commands.ci.CiCMD* attribute), 36  
 default\_branch (*mloq.commands.CiCMD* attribute), 53  
 default\_branch (*mloq.commands.docs.DocsCMD* attribute), 40  
 default\_branch (*mloq.commands.DocsCMD* attribute), 55  
 default\_branch (*mloq.commands.globals.GlobalsCMD* attribute), 41  
 default\_branch (*mloq.commands.GlobalsCMD* attribute), 56  
 default\_branch (*mloq.commands.package.PackageCMD* attribute), 45  
 default\_branch (*mloq.commands.PackageCMD* attribute), 58  
 DEFAULT\_PYTHON\_VERSIONS (in module *mloq.commands.package*), 45  
 deploy\_docs (in module *mloq.commands.docs*), 40  
 deploy\_docs (*mloq.commands.docs.DocsCMD* attribute), 40  
 deploy\_docs (*mloq.commands.DocsCMD* attribute), 55  
 description (*mloq.commands.docs.DocsCMD* attribute), 40  
 description (*mloq.commands.DocsCMD* attribute), 55  
 description (*mloq.commands.globals.GlobalsCMD* attribute), 41  
 description (*mloq.commands.GlobalsCMD* attribute), 56  
 description (*mloq.commands.package.PackageCMD* attribute), 45  
 description (*mloq.commands.PackageCMD* attribute), 58  
 description (*mloq.commands.project.ProjectCMD* attribute), 48  
 description (*mloq.commands.ProjectCMD* attribute), 59  
 description (*mloq.files.File* attribute), 84  
 DictConfig (class in *mloq.config.configuration*), 63  
 dir\_trees\_are\_equal() (in module *mloq.\_utils*), 75  
 directories (*mloq.command.CommandMixin* property), 79  
 directories (*mloq.commands.ci.CiCMD* property), 37  
 directories (*mloq.commands.CiCMD* property), 53  
 directories (*mloq.commands.docs.DocsCMD* property), 40  
 directories (*mloq.commands.DocsCMD* property), 55  
 directories (*mloq.commands.project.ProjectCMD* property), 48  
 directories (*mloq.commands.ProjectCMD* property), 59  
 directories (*mloq.commands.setup.SetupCMD* property), 51  
 directories (*mloq.commands.SetupCMD* property), 60  
 directories (*mloq.record.CMDRecord* property), 87  
 disable (*mloq.commands.ci.CiCMD* attribute), 36  
 disable (*mloq.commands.CiCMD* attribute), 52  
 disable (*mloq.commands.docker.DockerCMD* attribute), 38  
 disable (*mloq.commands.DockerCMD* attribute), 54  
 disable (*mloq.commands.docs.DocsCMD* attribute), 40  
 disable (*mloq.commands.DocsCMD* attribute), 55  
 disable (*mloq.commands.license.LicenseCMD* attribute), 43  
 disable (*mloq.commands.LicenseCMD* attribute), 56  
 disable (*mloq.commands.lint.LintCMD* attribute), 44  
 disable (*mloq.commands.LintCMD* attribute), 57  
 disable (*mloq.commands.package.PackageCMD* attribute), 45  
 disable (*mloq.commands.PackageCMD* attribute), 58  
 disable (*mloq.commands.project.ProjectCMD* attribute), 48  
 disable (*mloq.commands.ProjectCMD* attribute), 59  
 disable (*mloq.commands.requirements.RequirementsCMD* attribute), 50  
 disable (*mloq.commands.RequirementsCMD* attribute), 59  
 docker (*mloq.commands.ci.CiCMD* attribute), 36  
 docker (*mloq.commands.CiCMD* attribute), 53

DOCKER\_ASSETS\_PATH (in module *mloq.commands.docker*), 37  
 DOCKER\_FILES (in module *mloq.commands.docker*), 37  
 docker\_org (*mloq.commands.ci.CiCMD* attribute), 36  
 docker\_org (*mloq.commands.CiCMD* attribute), 53  
 docker\_org (*mloq.commands.docker.DockerCMD* attribute), 38  
 docker\_org (*mloq.commands.DockerCMD* attribute), 54  
 DockerCMD (class in *mloq.commands*), 53  
 DockerCMD (class in *mloq.commands.docker*), 37  
 dockerfile (in module *mloq.commands.docker*), 37  
 DOCS\_ASSETS\_PATH (in module *mloq.commands.docs*), 39  
 DOCS\_FILES (in module *mloq.commands.docs*), 40  
 docs\_req (in module *mloq.commands.docs*), 39  
 docs\_req (in module *mloq.commands.requirements*), 49  
 DocsCMD (class in *mloq.commands*), 55  
 DocsCMD (class in *mloq.commands.docs*), 40  
 docstring\_checks (*mloq.commands.lint.LintCMD* attribute), 44  
 docstring\_checks (*mloq.commands.LintCMD* attribute), 57  
 dogfood\_req (in module *mloq.commands.requirements*), 49  
 dogfood\_req (in module *mloq.files*), 85  
 dst (*mloq.files.File* attribute), 84  
 dump\_ledger() (*mloq.writer.Writer* method), 91

## E

email (*mloq.commands.ci.CiCMD* attribute), 36  
 email (*mloq.commands.CiCMD* attribute), 53  
 email (*mloq.commands.globals.GlobalsCMD* attribute), 41  
 email (*mloq.commands.GlobalsCMD* attribute), 56  
 email (*mloq.commands.license.LicenseCMD* attribute), 43  
 email (*mloq.commands.LicenseCMD* attribute), 56  
 email (*mloq.commands.package.PackageCMD* attribute), 46  
 email (*mloq.commands.PackageCMD* attribute), 58  
 extra (*mloq.commands.docker.DockerCMD* attribute), 38  
 extra (*mloq.commands.DockerCMD* attribute), 54

## F

Failure, 83  
 File (class in *mloq.files*), 84  
 file() (in module *mloq.files*), 84  
 files (*mloq.command.CommandMixin* attribute), 79  
 files (*mloq.commands.ci.CiCMD* attribute), 37  
 files (*mloq.commands.CiCMD* attribute), 53  
 files (*mloq.commands.docker.DockerCMD* attribute), 38

files (*mloq.commands.DockerCMD* attribute), 53  
 files (*mloq.commands.docs.DocsCMD* attribute), 40  
 files (*mloq.commands.DocsCMD* attribute), 55  
 files (*mloq.commands.license.LicenseCMD* attribute), 43  
 files (*mloq.commands.LicenseCMD* attribute), 56  
 files (*mloq.commands.lint.LintCMD* attribute), 44  
 files (*mloq.commands.LintCMD* attribute), 57  
 files (*mloq.commands.package.PackageCMD* attribute), 45  
 files (*mloq.commands.PackageCMD* attribute), 58  
 files (*mloq.commands.project.ProjectCMD* attribute), 48  
 files (*mloq.commands.ProjectCMD* attribute), 59  
 files (*mloq.commands.requirements.RequirementsCMD* attribute), 50  
 files (*mloq.commands.RequirementsCMD* attribute), 59  
 files (*mloq.commands.setup.SetupCMD* attribute), 51  
 files (*mloq.commands.SetupCMD* attribute), 60  
 files (*mloq.record.CMDRecord* property), 87  
 files (*mloq.record.Ledger* property), 86  
 files\_are\_equal() (in module *mloq.\_utils*), 75  
 FloatPrompt (class in *mloq.config.prompt*), 73

## G

get\_aliased\_requirements\_file()  
     (*mloq.commands.requirements.RequirementsCMD* class method), 50  
 get\_aliased\_requirements\_file()  
     (*mloq.commands.RequirementsCMD* class method), 59  
 get\_base\_image() (*mloq.commands.docker.DockerCMD* method), 38  
 get\_base\_image() (*mloq.commands.DockerCMD* method), 54  
 get\_command() (*mloq.cli.MloqCLI* method), 78  
 get\_docker\_python\_version() (in module *mloq.\_utils*), 76  
 get\_generated\_directories() (in module *mloq.\_utils*), 76  
 get\_generated\_files() (in module *mloq.\_utils*), 76  
 gitignore (in module *mloq.commands.project*), 47  
 GlobalsCMD (class in *mloq.commands*), 55  
 GlobalsCMD (class in *mloq.commands.globals*), 41  
 gpl\_license (in module *mloq.commands.license*), 42

## H

hidden\_prompt\_func() (in module *mloq.config.custom\_click*), 69  
 hidden\_prompt\_func() (in module *mloq.custom\_click*), 81  
 hydra\_args (in module *mloq.cli*), 78



## I

[ignore\\_files](#) ([mloq.commands.lint.LintCMD](#) attribute), 44  
[ignore\\_files](#) ([mloq.commands.LintCMD](#) attribute), 57  
[index\\_md](#) (in module [mloq.commands.docs](#)), 40  
[init](#) (in module [mloq.commands.project](#)), 47  
[interactive\\_config\(\)](#) ([mloq.command.Command](#) method), 80  
[interactive\\_config\(\)](#) ([mloq.command.CommandMixin](#) method), 79  
[interactive\\_config\(\)](#) ([mloq.commands.ci.CiCMD](#) method), 37  
[interactive\\_config\(\)](#) ([mloq.commands.CiCMD](#) method), 53  
[interactive\\_config\(\)](#) ([mloq.commands.docker.DockerCMD](#) method), 39  
[interactive\\_config\(\)](#) ([mloq.commands.DockerCMD](#) method), 54  
[interactive\\_config\(\)](#) ([mloq.commands.docs.DocsCMD](#) method), 40  
[interactive\\_config\(\)](#) ([mloq.commands.DocsCMD](#) method), 55  
[interactive\\_config\(\)](#) ([mloq.commands.globals.GlobalsCMD](#) method), 41  
[interactive\\_config\(\)](#) ([mloq.commands.GlobalsCMD](#) method), 56  
[interactive\\_config\(\)](#) ([mloq.commands.license.LicenseCMD](#) method), 43  
[interactive\\_config\(\)](#) ([mloq.commands.LicenseCMD](#) method), 57  
[interactive\\_config\(\)](#) ([mloq.commands.lint.LintCMD](#) method), 44  
[interactive\\_config\(\)](#) ([mloq.commands.LintCMD](#) method), 57  
[interactive\\_config\(\)](#) ([mloq.commands.package.PackageCMD](#) method), 46  
[interactive\\_config\(\)](#) ([mloq.commands.PackageCMD](#) method), 58  
[interactive\\_config\(\)](#) ([mloq.commands.requirements.RequirementsCMD](#) method), 50  
[interactive\\_config\(\)](#) ([mloq.commands.RequirementsCMD](#) method), 60

[interactive\\_config\(\)](#) ([mloq.commands.setup.SetupCMD](#) method), 52  
[interactive\\_config\(\)](#) ([mloq.commands.SetupCMD](#) method), 61  
[interactive\\_opt](#) (in module [mloq.cli](#)), 77  
[interpolations](#) ([mloq.config.configuration.OmegaConfInterface](#) property), 64  
[IntPrompt](#) (class in [mloq.config.prompt](#)), 73  
[is\\_interpolation\(\)](#) (in module [mloq.config.configuration](#)), 63  
[is\\_interpolation\(\)](#) ([mloq.config.configuration.OmegaConfInterface](#) method), 65  
[is\\_missing\(\)](#) ([mloq.config.configuration.OmegaConfInterface](#) method), 65  
[is\\_static](#) ([mloq.files.File](#) attribute), 84  
[isort](#) ([mloq.commands.lint.LintCMD](#) attribute), 44  
[isort](#) ([mloq.commands.LintCMD](#) attribute), 57

## J

[jinja\\_env](#) (in module [mloq.templating](#)), 89  
[jupyter](#) ([mloq.commands.docker.DockerCMD](#) attribute), 38  
[jupyter](#) ([mloq.commands.DockerCMD](#) attribute), 54  
[jupyter\\_password](#) ([mloq.commands.docker.DockerCMD](#) attribute), 38  
[jupyter\\_password](#) ([mloq.commands.DockerCMD](#) attribute), 54

## L

[Ledger](#) (class in [mloq.record](#)), 86  
[ledger](#) ([mloq.writer.Writer](#) property), 90  
[license](#) ([mloq.commands.globals.GlobalsCMD](#) attribute), 41  
[license](#) ([mloq.commands.GlobalsCMD](#) attribute), 56  
[license](#) ([mloq.commands.license.LicenseCMD](#) attribute), 43  
[license](#) ([mloq.commands.LicenseCMD](#) attribute), 56  
[license](#) ([mloq.commands.package.PackageCMD](#) attribute), 45  
[license](#) ([mloq.commands.PackageCMD](#) attribute), 58  
[license](#) ([mloq.commands.project.ProjectCMD](#) attribute), 48  
[license](#) ([mloq.commands.ProjectCMD](#) attribute), 59  
[license\\_classifier](#) ([mloq.commands.package.PackageCMD](#) attribute), 45  
[license\\_classifier](#) ([mloq.commands.PackageCMD](#) attribute), 58  
[LICENSE\\_CLASSIFIERS](#) ([mloq.commands.package.PackageCMD](#) attribute), 45  
[LICENSE\\_CLASSIFIERS](#) ([mloq.commands.PackageCMD](#) attribute), 58  
[LICENSE\\_FILES](#) (in module [mloq.commands.license](#)), 42  
[LicenseCMD](#) (class in [mloq.commands](#)), 56

LicenseCMD (class in *mloq.commands.license*), 42  
LICENSES (in module *mloq.commands.license*), 42  
LICENSES (*mloq.commands.license.LicenseCMD* attribute), 43  
LICENSES (*mloq.commands.LicenseCMD* attribute), 56  
lint (*mloq.commands.docker.DockerCMD* attribute), 38  
lint (*mloq.commands.DockerCMD* attribute), 54  
LINT\_FILES (in module *mloq.commands.lint*), 43  
lint\_req (in module *mloq.commands.lint*), 43  
lint\_req (in module *mloq.commands.requirements*), 49  
LintCMD (class in *mloq.commands*), 57  
LintCMD (class in *mloq.commands.lint*), 43  
linters (*mloq.commands.lint.LintCMD* attribute), 44  
linters (*mloq.commands.LintCMD* attribute), 57  
list\_commands() (*mloq.cli.MloqCLI* method), 78  
load\_config() (in module *mloq.runner*), 88  
load\_empty\_config\_setup() (in module *mloq.\_utils*), 76

## M

main (in module *mloq.commands.project*), 47  
main\_python\_version  
    (*mloq.commands.globals.GlobalsCMD* attribute), 41  
main\_python\_version (*mloq.commands.GlobalsCMD* attribute), 56  
main\_python\_version  
    (*mloq.commands.package.PackageCMD* attribute), 46  
main\_python\_version (*mloq.commands.PackageCMD* attribute), 58  
make\_bat\_docs (in module *mloq.commands.docs*), 39  
makefile (in module *mloq.files*), 85  
makefile (*mloq.commands.docker.DockerCMD* attribute), 38  
makefile (*mloq.commands.DockerCMD* attribute), 54  
makefile (*mloq.commands.lint.LintCMD* attribute), 44  
makefile (*mloq.commands.LintCMD* attribute), 57  
makefile\_docker (in module *mloq.commands.docker*), 37  
makefile\_docs (in module *mloq.commands.docs*), 39  
missing (*mloq.config.configuration.OmegaConfInterface* property), 64  
MissingConfigValue, 83  
mit\_license (in module *mloq.commands.license*), 42  
mloq  
    module, 35  
mloq.\_\_main\_\_  
    module, 75  
mloq.\_utils  
    module, 75  
mloq.cli  
    module, 77  
mloq.command  
    module, 78  
mloq.commands  
    module, 35  
mloq.commands.ci  
    module, 35  
mloq.commands.docker  
    module, 37  
mloq.commands.docs  
    module, 39  
mloq.commands.globals  
    module, 41  
mloq.commands.license  
    module, 42  
mloq.commands.lint  
    module, 43  
mloq.commands.package  
    module, 44  
mloq.commands.project  
    module, 46  
mloq.commands.requirements  
    module, 48  
mloq.commands.setup  
    module, 51  
mloq.config  
    module, 61  
mloq.config.configuration  
    module, 61  
mloq.config.custom\_click  
    module, 68  
mloq.config.param\_patch  
    module, 70  
mloq.config.prompt  
    module, 71  
mloq.custom\_click  
    module, 81  
mloq.failure  
    module, 83  
mloq.files  
    module, 83  
mloq.git  
    module, 85  
mloq.record  
    module, 86  
mloq.runner  
    module, 88  
mloq.templating  
    module, 89  
mloq.version  
    module, 90  
mloq.writer  
    module, 90  
MLOQ\_ASSETS\_PATH (in module *mloq.files*), 85  
mloq\_click\_command() (in module *mloq.cli*), 78  
mloq\_yaml (in module *mloq.files*), 85

MloqCLI (class in mloq.cli), 78

module

- mloq, 35
- mloq.\_\_main\_\_, 75
- mloq.\_utils, 75
- mloq.cli, 77
- mloq.command, 78
- mloq.commands, 35
- mloq.commands.ci, 35
- mloq.commands.docker, 37
- mloq.commands.docs, 39
- mloq.commands.globals, 41
- mloq.commands.license, 42
- mloq.commands.lint, 43
- mloq.commands.package, 44
- mloq.commands.project, 46
- mloq.commands.requirements, 48
- mloq.commands.setup, 51
- mloq.config, 61
- mloq.config.configuration, 61
- mloq.config.custom\_click, 68
- mloq.config.param\_patch, 70
- mloq.config.prompt, 71
- mloq.custom\_click, 81
- mloq.failure, 83
- mloq.files, 83
- mloq.git, 85
- mloq.record, 86
- mloq.runner, 88
- mloq.templating, 89
- mloq.version, 90
- mloq.writer, 90

MultiChoicePrompt (class in mloq.config.prompt), 72

## N

name (mloq.files.File attribute), 84

## O

OmegaConfInterface (class in mloq.config.configuration), 64

only\_config\_opt (in module mloq.cli), 77

open\_source (mloq.commands.ci.CiCMD attribute), 36

open\_source (mloq.commands.CiCMD attribute), 53

open\_source (mloq.commands.globals.GlobalsCMD attribute), 41

open\_source (mloq.commands.GlobalsCMD attribute), 56

output\_directory\_arg (in module mloq.cli), 77

overwrite\_opt (in module mloq.cli), 77

owner (mloq.commands.ci.CiCMD attribute), 36

owner (mloq.commands.CiCMD attribute), 53

owner (mloq.commands.globals.GlobalsCMD attribute), 41

owner (mloq.commands.GlobalsCMD attribute), 56

owner (mloq.commands.package.PackageCMD attribute), 45

owner (mloq.commands.PackageCMD attribute), 58

owner (mloq.commands.project.ProjectCMD attribute), 48

owner (mloq.commands.ProjectCMD attribute), 59

## P

PACKAGE\_ASSETS\_PATH (in module mloq.commands.package), 45

PACKAGE\_FILES (in module mloq.commands.package), 45

PackageCMD (class in mloq.commands), 57

PackageCMD (class in mloq.commands.package), 45

param (in module mloq.config.param\_patch), 71

param (mloq.config.prompt.PromptParam property), 72

param\_to\_dataclass() (in module mloq.config.configuration), 63

param\_to\_dataclass\_dict() (in module mloq.config.configuration), 63

param\_to\_omegaconf() (in module mloq.config.configuration), 63

PARAM\_TO\_PROMPT (in module mloq.config.prompt), 74

PARAM\_TO\_TYPE (in module mloq.config.configuration), 63

params (mloq.config.configuration.Config property), 66

ParamType (in module mloq.config.configuration), 63

parse\_config() (mloq.command.CommandMixin method), 79

parse\_config() (mloq.commands.docker.DockerCMD method), 38

parse\_config() (mloq.commands.DockerCMD method), 54

parse\_config() (mloq.commands.globals.GlobalsCMD method), 41

parse\_config() (mloq.commands.GlobalsCMD method), 56

parse\_config() (mloq.commands.package.PackageCMD method), 46

parse\_config() (mloq.commands.PackageCMD method), 58

parse\_config() (mloq.commands.setup.SetupCMD method), 52

parse\_config() (mloq.commands.SetupCMD method), 61

PATCHED\_PARAMETERS (in module mloq.config.param\_patch), 70

PATCHED\_PARAMETERS (mloq.config.param\_patch.\_\_ParamPatcher attribute), 71

path (mloq.writer.Writer property), 90

poetry\_requirements (mloq.commands.lint.LintCMD attribute), 44

poetry\_requirements (mloq.commands.LintCMD attribute), 57

[pre\\_commit\\_hook](#) (in module *mloq.commands.project*), [47](#)  
[PROJECT\\_ASSETS\\_PATH](#) (in module *mloq.commands.project*), [47](#)  
[PROJECT\\_FILES](#) (in module *mloq.commands.project*), [47](#)  
[project\\_name](#) (*mloq.commands.ci.CiCMD* attribute), [36](#)  
[project\\_name](#) (*mloq.commands.CiCMD* attribute), [53](#)  
[project\\_name](#) (*mloq.commands.docker.DockerCMD* attribute), [38](#)  
[project\\_name](#) (*mloq.commands.DockerCMD* attribute), [54](#)  
[project\\_name](#) (*mloq.commands.docs.DocsCMD* attribute), [40](#)  
[project\\_name](#) (*mloq.commands.DocsCMD* attribute), [55](#)  
[project\\_name](#) (*mloq.commands.globals.GlobalsCMD* attribute), [41](#)  
[project\\_name](#) (*mloq.commands.GlobalsCMD* attribute), [56](#)  
[project\\_name](#) (*mloq.commands.license.LicenseCMD* attribute), [43](#)  
[project\\_name](#) (*mloq.commands.LicenseCMD* attribute), [56](#)  
[project\\_name](#) (*mloq.commands.lint.LintCMD* attribute), [44](#)  
[project\\_name](#) (*mloq.commands.LintCMD* attribute), [57](#)  
[project\\_name](#) (*mloq.commands.package.PackageCMD* attribute), [45](#)  
[project\\_name](#) (*mloq.commands.PackageCMD* attribute), [58](#)  
[project\\_name](#) (*mloq.commands.project.ProjectCMD* attribute), [48](#)  
[project\\_name](#) (*mloq.commands.ProjectCMD* attribute), [59](#)  
[project\\_url](#) (*mloq.commands.ci.CiCMD* attribute), [36](#)  
[project\\_url](#) (*mloq.commands.CiCMD* attribute), [53](#)  
[project\\_url](#) (*mloq.commands.docs.DocsCMD* attribute), [40](#)  
[project\\_url](#) (*mloq.commands.DocsCMD* attribute), [55](#)  
[project\\_url](#) (*mloq.commands.globals.GlobalsCMD* attribute), [41](#)  
[project\\_url](#) (*mloq.commands.GlobalsCMD* attribute), [56](#)  
[project\\_url](#) (*mloq.commands.license.LicenseCMD* attribute), [43](#)  
[project\\_url](#) (*mloq.commands.LicenseCMD* attribute), [56](#)  
[project\\_url](#) (*mloq.commands.package.PackageCMD* attribute), [45](#)  
[project\\_url](#) (*mloq.commands.PackageCMD* attribute), [58](#)  
[project\\_url](#) (*mloq.commands.project.ProjectCMD* attribute), [48](#)  
[project\\_url](#) (*mloq.commands.ProjectCMD* attribute), [59](#)  
[ProjectCMD](#) (class in *mloq.commands*), [58](#)  
[ProjectCMD](#) (class in *mloq.commands.project*), [47](#)  
[Prompt](#) (class in *mloq.config.prompt*), [74](#)  
[prompt\(\)](#) (in module *mloq.config.custom\_click*), [69](#)  
[prompt\(\)](#) (in module *mloq.custom\_click*), [81](#)  
[prompt\(\)](#) (*mloq.config.prompt.Prompt* method), [74](#)  
[prompt\\_all\(\)](#) (*mloq.config.prompt.Prompt* method), [74](#)  
[Promptable](#) (class in *mloq.config.prompt*), [75](#)  
[PromptParam](#) (class in *mloq.config.prompt*), [71](#)  
[push\\_python\\_wkf](#) (in module *mloq.commands.ci*), [36](#)  
[pyproject\\_extra](#) (*mloq.commands.lint.LintCMD* attribute), [44](#)  
[pyproject\\_extra](#) (*mloq.commands.LintCMD* attribute), [57](#)  
[pyproject\\_extra](#) (*mloq.commands.package.PackageCMD* attribute), [45](#)  
[pyproject\\_extra](#) (*mloq.commands.PackageCMD* attribute), [58](#)  
[pyproject\\_toml](#) (in module *mloq.files*), [85](#)  
[python\\_version](#) (*mloq.commands.docker.DockerCMD* attribute), [38](#)  
[python\\_version](#) (*mloq.commands.DockerCMD* attribute), [54](#)  
[PYTHON\\_VERSIONS](#) (in module *mloq.commands.package*), [45](#)  
[python\\_versions](#) (*mloq.commands.ci.CiCMD* attribute), [36](#)  
[python\\_versions](#) (*mloq.commands.CiCMD* attribute), [53](#)  
[python\\_versions](#) (*mloq.commands.package.PackageCMD* attribute), [46](#)  
[python\\_versions](#) (*mloq.commands.PackageCMD* attribute), [58](#)  
[PythonType](#) (in module *mloq.config.configuration*), [63](#)  
[pytorch\\_req](#) (in module *mloq.commands.requirements*), [49](#)

## R

[read\\_file\(\)](#) (in module *mloq.files*), [85](#)  
[read\\_requirements\\_file\(\)](#) (*mloq.commands.requirements.RequirementsCMD* class method), [50](#)  
[read\\_requirements\\_file\(\)](#) (*mloq.commands.RequirementsCMD* class method), [59](#)  
[readme](#) (in module *mloq.commands.project*), [47](#)  
[record](#) (*mloq.command.CommandMixin* property), [79](#)  
[record](#) (*mloq.writer.Writer* property), [90](#)  
[record\\_directories\(\)](#) (*mloq.command.CommandMixin* method), [79](#)

`record_files()` (*mloq.command.CommandMixin* method), 79  
`record_files()` (*mloq.commands.ci.CiCMD* method), 37  
`record_files()` (*mloq.commands.CiCMD* method), 53  
`record_files()` (*mloq.commands.docker.DockerCMD* method), 39  
`record_files()` (*mloq.commands.DockerCMD* method), 54  
`record_files()` (*mloq.commands.docs.DocsCMD* method), 40  
`record_files()` (*mloq.commands.DocsCMD* method), 55  
`record_files()` (*mloq.commands.license.LicenseCMD* method), 43  
`record_files()` (*mloq.commands.LicenseCMD* method), 57  
`record_files()` (*mloq.commands.lint.LintCMD* method), 44  
`record_files()` (*mloq.commands.LintCMD* method), 57  
`record_files()` (*mloq.commands.package.PackageCMD* method), 46  
`record_files()` (*mloq.commands.PackageCMD* method), 58  
`record_files()` (*mloq.commands.project.ProjectCMD* method), 48  
`record_files()` (*mloq.commands.ProjectCMD* method), 59  
`record_files()` (*mloq.commands.requirements.RequirementsCMD* method), 38  
`record_files()` (*mloq.commands.RequirementsCMD* method), 50  
`record_files()` (*mloq.commands.RequirementsCMD* method), 60  
`record_files()` (*mloq.commands.setup.SetupCMD* method), 52  
`record_files()` (*mloq.commands.SetupCMD* method), 61  
`register()` (*mloq.record.Ledger* method), 86  
`register_directory()` (*mloq.record.CMDRecord* method), 87  
`register_file()` (*mloq.record.CMDRecord* method), 87  
`render_template()` (in module *mloq.templating*), 89  
`require_cuda_from_requirements()` (*mloq.commands.docker.DockerCMD* static method), 38  
`require_cuda_from_requirements()` (*mloq.commands.DockerCMD* static method), 54  
`REQUIREMENT_CHOICES` (in module *mloq.commands.requirements*), 49  
`requirements` (in module *mloq.commands.requirements*), 49  
`requirements` (in module *mloq.files*), 85  
`requirements` (*mloq.commands.docker.DockerCMD* attribute), 38  
`requirements` (*mloq.commands.DockerCMD* attribute), 54  
`requirements` (*mloq.commands.requirements.RequirementsCMD* attribute), 50  
`requirements` (*mloq.commands.RequirementsCMD* attribute), 59  
`REQUIREMENTS_ALIASES` (*mloq.commands.requirements.RequirementsCMD* attribute), 50  
`REQUIREMENTS_ALIASES` (*mloq.commands.RequirementsCMD* attribute), 59  
`REQUIREMENTS_FILES` (in module *mloq.commands.requirements*), 49  
`requirements_is_empty()` (*mloq.commands.requirements.RequirementsCMD* static method), 50  
`requirements_is_empty()` (*mloq.commands.RequirementsCMD* static method), 60  
`REQUIREMENTS_PATH` (in module *mloq.commands.requirements*), 49  
`REQUIREMENTS_PATH` (in module *mloq.files*), 85  
`RequirementsCMD` (class in *mloq.commands*), 59  
`RequirementsCMD` (class in *mloq.commands.requirements*), 49  
`requires_cuda()` (*mloq.commands.docker.DockerCMD* method), 54  
`requires_cuda()` (*mloq.commands.DockerCMD* method), 54  
`resolve()` (*mloq.config.configuration.Config* method), 67  
`resolve()` (*mloq.config.configuration.OmegaConfInterface* method), 65  
`resolve_as_dict()` (in module *mloq.config.configuration*), 64  
`run()` (*mloq.command.CommandMixin* method), 80  
`run()` (*mloq.writer.Writer* method), 91  
`run_command()` (in module *mloq.runner*), 88  
`run_side_effects()` (*mloq.command.CommandMixin* method), 80  
`run_side_effects()` (*mloq.commands.setup.SetupCMD* method), 52  
`run_side_effects()` (*mloq.commands.SetupCMD* method), 61

## S

`safe_select()` (in module *mloq.config.configuration*), 64  
`select()` (*mloq.config.configuration.OmegaConfInterface* method), 65  
`setup_git()` (in module *mloq.git*), 85



`setup_py` (in module `mloq.commands.package`), 45  
`SetupCMD` (class in `mloq.commands`), 60  
`SetupCMD` (class in `mloq.commands.setup`), 51  
`SHARED_ASSETS_PATH` (in module `mloq.files`), 85  
`src` (`mloq.files.File` attribute), 84  
`StringPrompt` (class in `mloq.config.prompt`), 73  
`SUB_COMMAND_CLASSES`  
    (`mloq.commands.setup.SetupCMD` attribute), 51  
`SUB_COMMAND_CLASSES` (`mloq.commands.SetupCMD` attribute), 60  
`SUB_COMMANDS` (in module `mloq.commands.setup`), 51  
`sub_commands` (`mloq.commands.setup.SetupCMD` property), 51  
`sub_commands` (`mloq.commands.SetupCMD` property), 60  
`sync()` (`mloq.config.configuration.Config` method), 67

## T

`TEMPLATES_PATH` (in module `mloq.commands.license`), 42  
`tensorflow_req` (in module `mloq.commands.requirements`), 49  
`test` (`mloq.commands.docker.DockerCMD` attribute), 38  
`test` (`mloq.commands.DockerCMD` attribute), 54  
`test_main` (in module `mloq.commands.project`), 47  
`test_req` (in module `mloq.commands.project`), 47  
`test_req` (in module `mloq.commands.requirements`), 49  
`tests` (`mloq.commands.project.ProjectCMD` attribute), 48  
`tests` (`mloq.commands.ProjectCMD` attribute), 59  
`to_config()` (in module `mloq.config.configuration`), 64  
`to_container()` (`mloq.config.configuration.BaseConfig` method), 66  
`to_dataclass()` (`mloq.config.configuration.Config` method), 67  
`to_dictconfig()` (`mloq.config.configuration.Config` method), 67  
`to_param_type()` (in module `mloq.config.configuration`), 63  
`to_param_type()` (`mloq.config.configuration.Config` method), 67

## U

`ubuntu_version` (`mloq.commands.ci.CiCMD` attribute), 36  
`ubuntu_version` (`mloq.commands.CiCMD` attribute), 52  
`ubuntu_version` (`mloq.commands.docker.DockerCMD` attribute), 38  
`ubuntu_version` (`mloq.commands.DockerCMD` attribute), 54  
`update_config()` (`mloq.record.CMDRecord` method), 87

`use_poetry` (`mloq.commands.globals.GlobalsCMD` attribute), 41  
`use_poetry` (`mloq.commands.GlobalsCMD` attribute), 56  
`use_poetry` (`mloq.commands.package.PackageCMD` attribute), 46  
`use_poetry` (`mloq.commands.PackageCMD` attribute), 58

## V

`value` (`mloq.config.prompt.PromptParam` property), 72  
`vendor` (`mloq.commands.ci.CiCMD` attribute), 36  
`vendor` (`mloq.commands.CiCMD` attribute), 53  
`version` (in module `mloq.commands.project`), 47  
`visible_prompt_func` (in module `mloq.config.custom_click`), 68  
`visible_prompt_func` (in module `mloq.custom_click`), 81

## W

`welcome_message()` (in module `mloq.cli`), 78  
`what_mloq_generated` (in module `mloq.files`), 85  
`WORKFLOW_FILES` (in module `mloq.commands.ci`), 36  
`write_config_setup()` (in module `mloq._utils`), 76  
`write_record()` (in module `mloq.runner`), 88  
`write_template()` (in module `mloq.templating`), 89  
`write_template()` (`mloq.writer.Writer` method), 91  
`write_templates()` (`mloq.writer.Writer` method), 91  
`Writer` (class in `mloq.writer`), 90