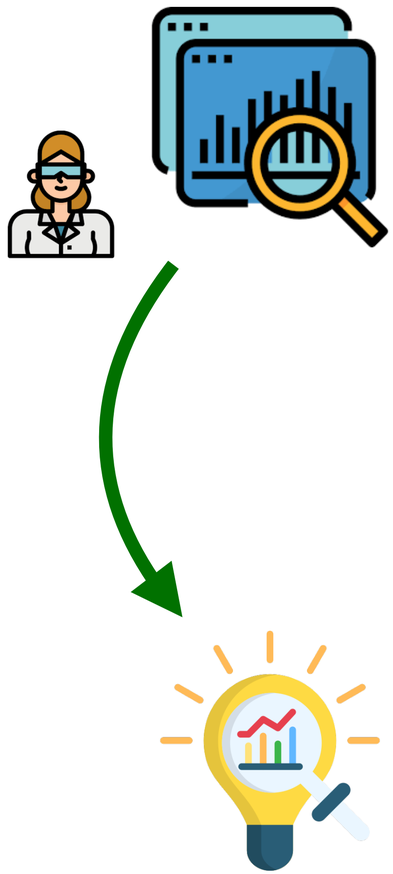
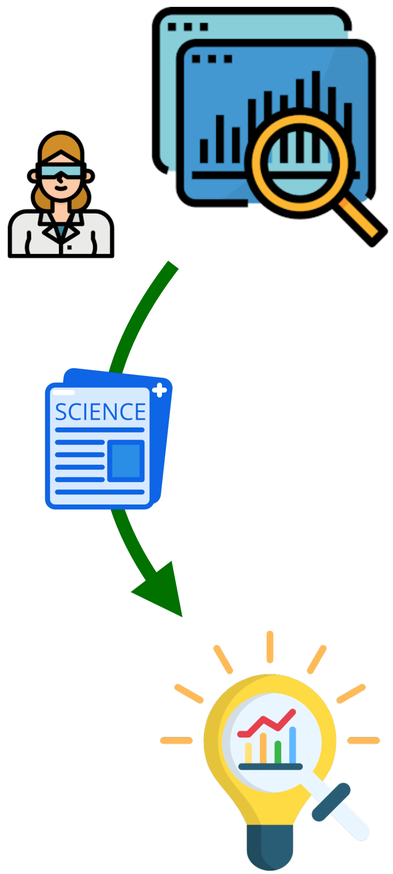
4th Online FAIR Training

Reproducible code: FAIR guidelines to software publication, visibility, and citation

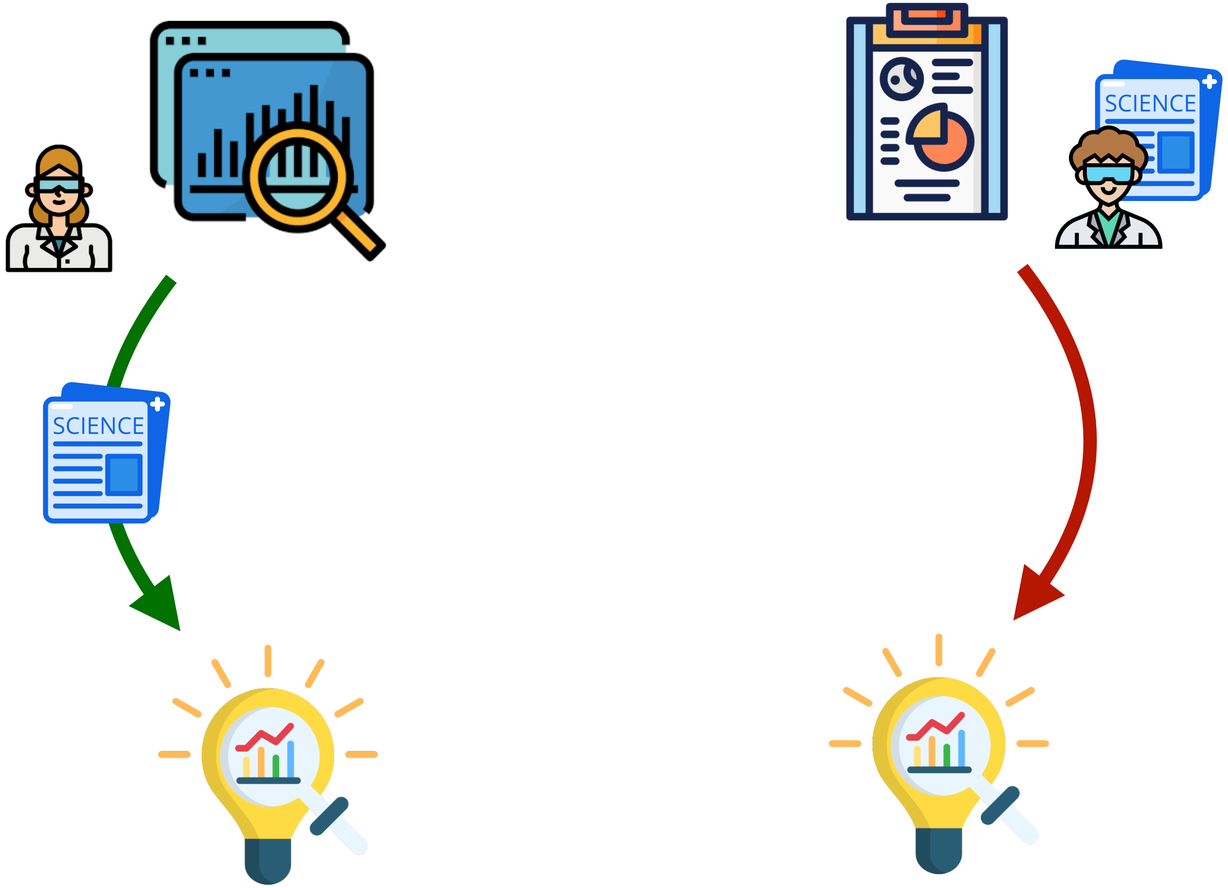
Stefano Rapisarda

26 November 2024



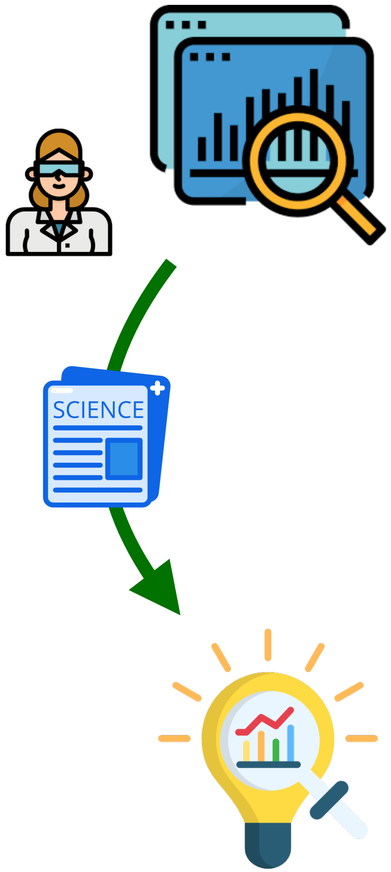

SCIENCE

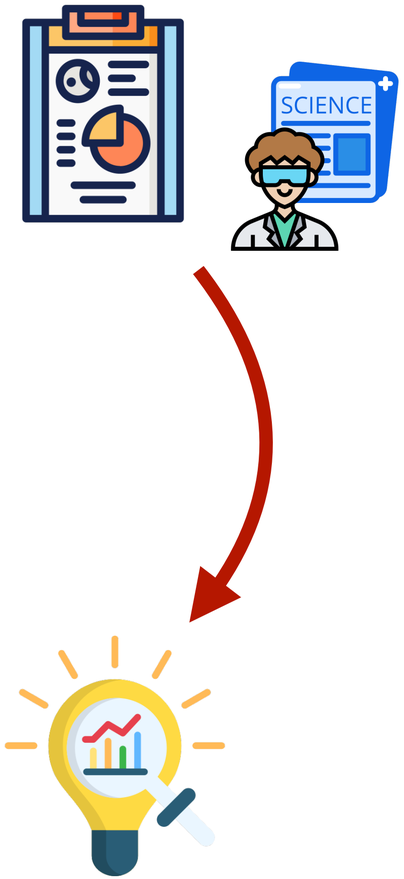
Replicability

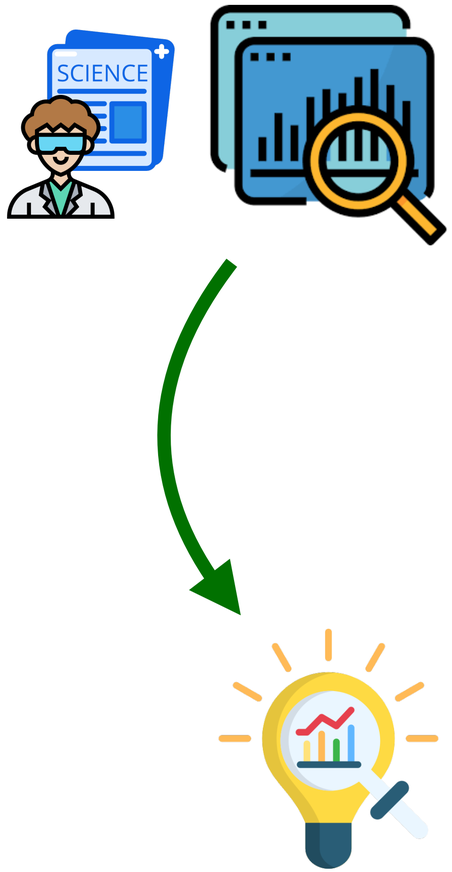


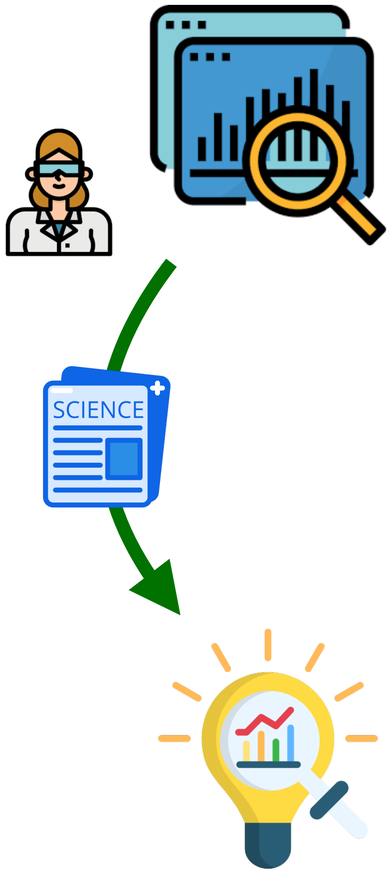
Replicability

Reproducibility

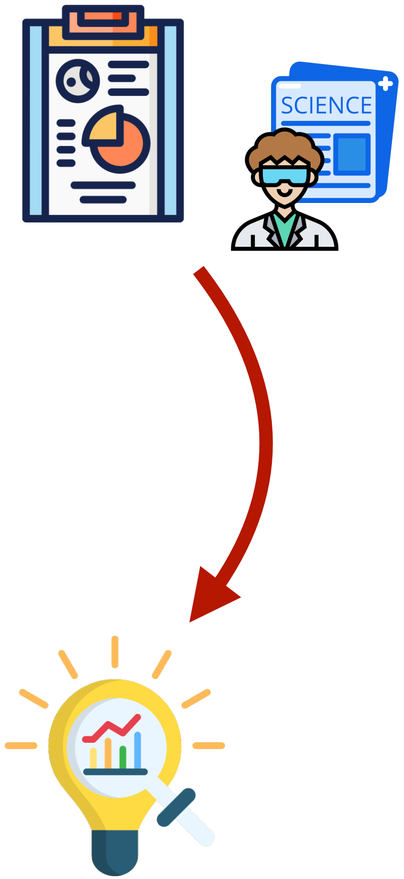




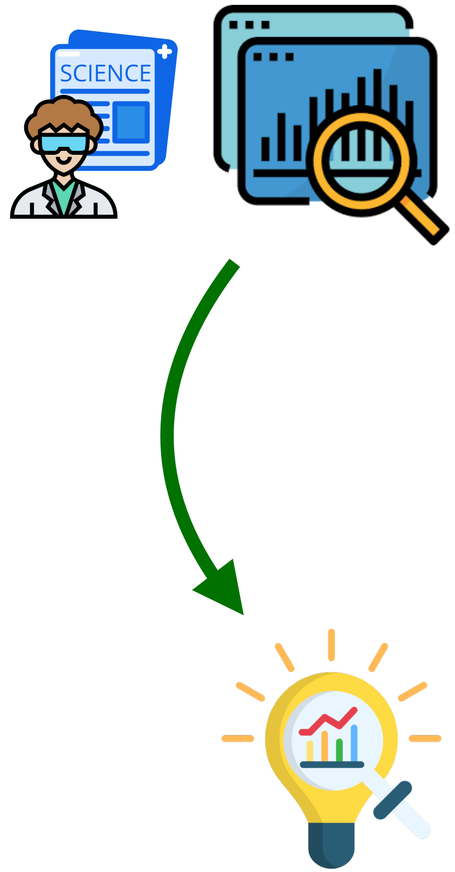




Replicability



Reproducibility



Archiving a repository

Create a GitHub repository

Archiving a repository

Create a GitHub repository

Login in Zenodo (with your GitHub credentials)

Archiving a repository

Create a GitHub repository

Login in Zenodo (with your GitHub credentials)

Select repository to preserve

Archiving a repository

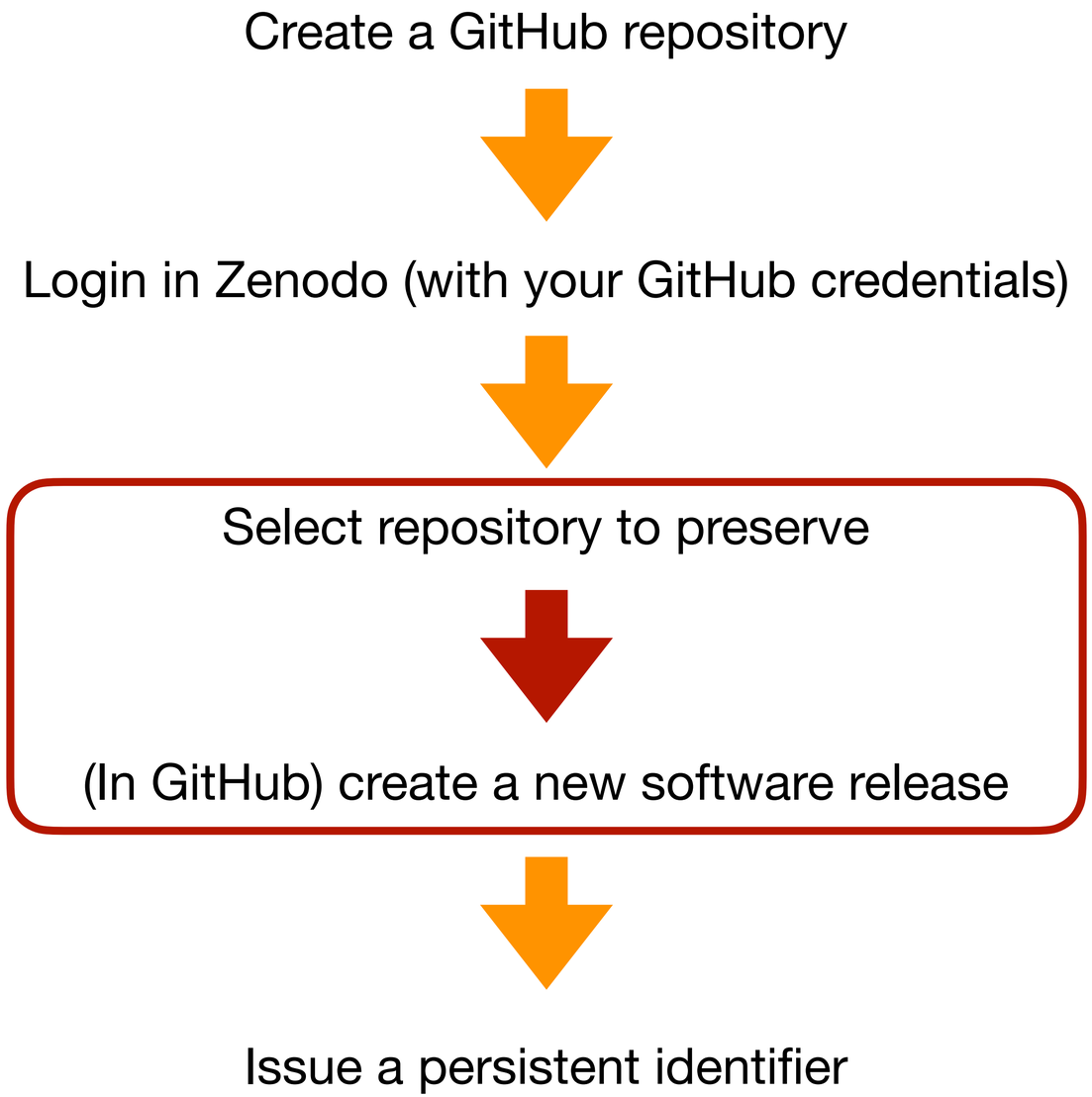
Create a GitHub repository

Login in Zenodo (with your GitHub credentials)

Select repository to preserve

(In GitHub) create a new software release

Archiving a repository



Hands On

Citing Software

Cite the publication

Citing Software

Cite the publication

Cite the DOI (get one with Zenodo or Figshare)

Citing Software

Cite the publication

Cite the DOI (get one with Zenodo or Figshare)

Cite the GitHub repository

Citing Software

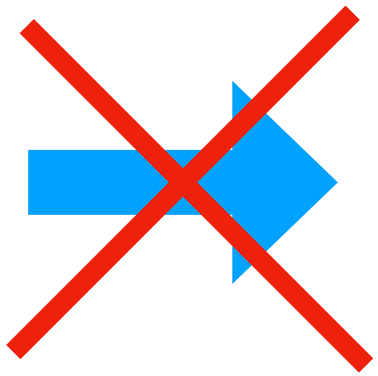
Cite the publication

Cite the DOI (get one with Zenodo or Figshare)

Cite the GitHub repository

IDEALLY ALL THREE

Public
And
Visible

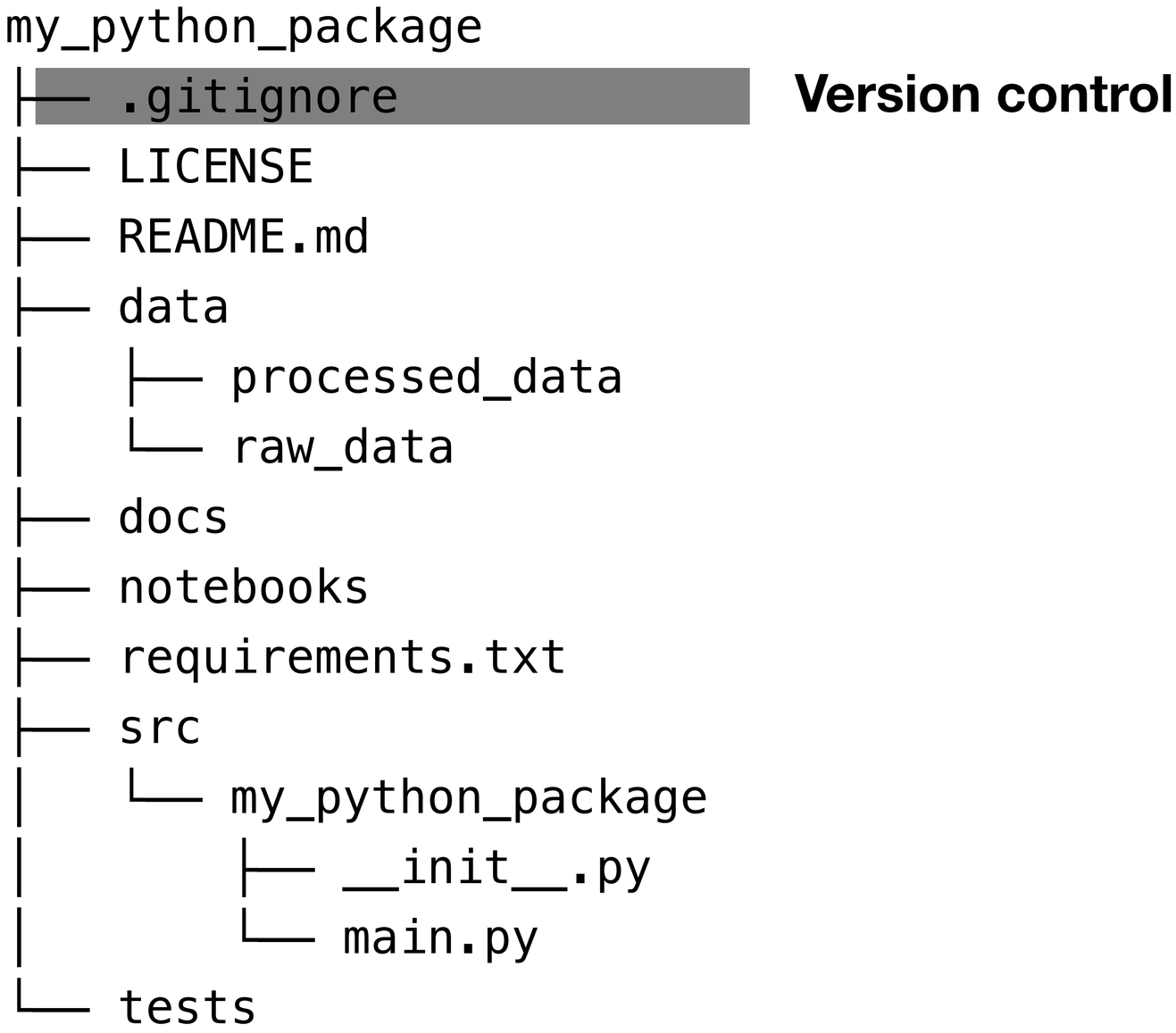


Reproducible
And
FAIR

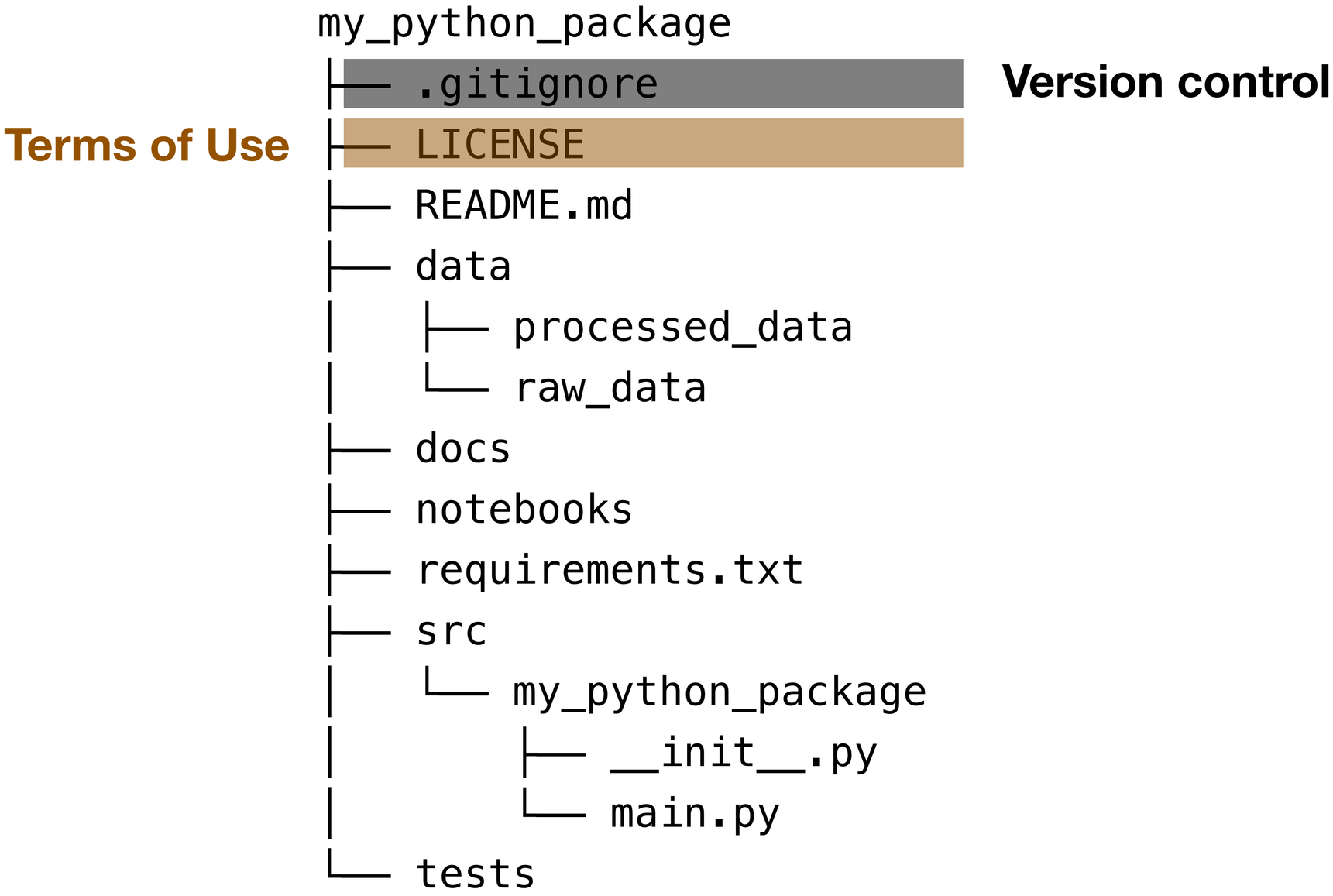
Project Structure (Python)

```
my_python_package
├── .gitignore
├── LICENSE
├── README.md
├── data
│   ├── processed_data
│   └── raw_data
├── docs
├── notebooks
├── requirements.txt
├── src
│   └── my_python_package
│       ├── __init__.py
│       └── main.py
└── tests
```

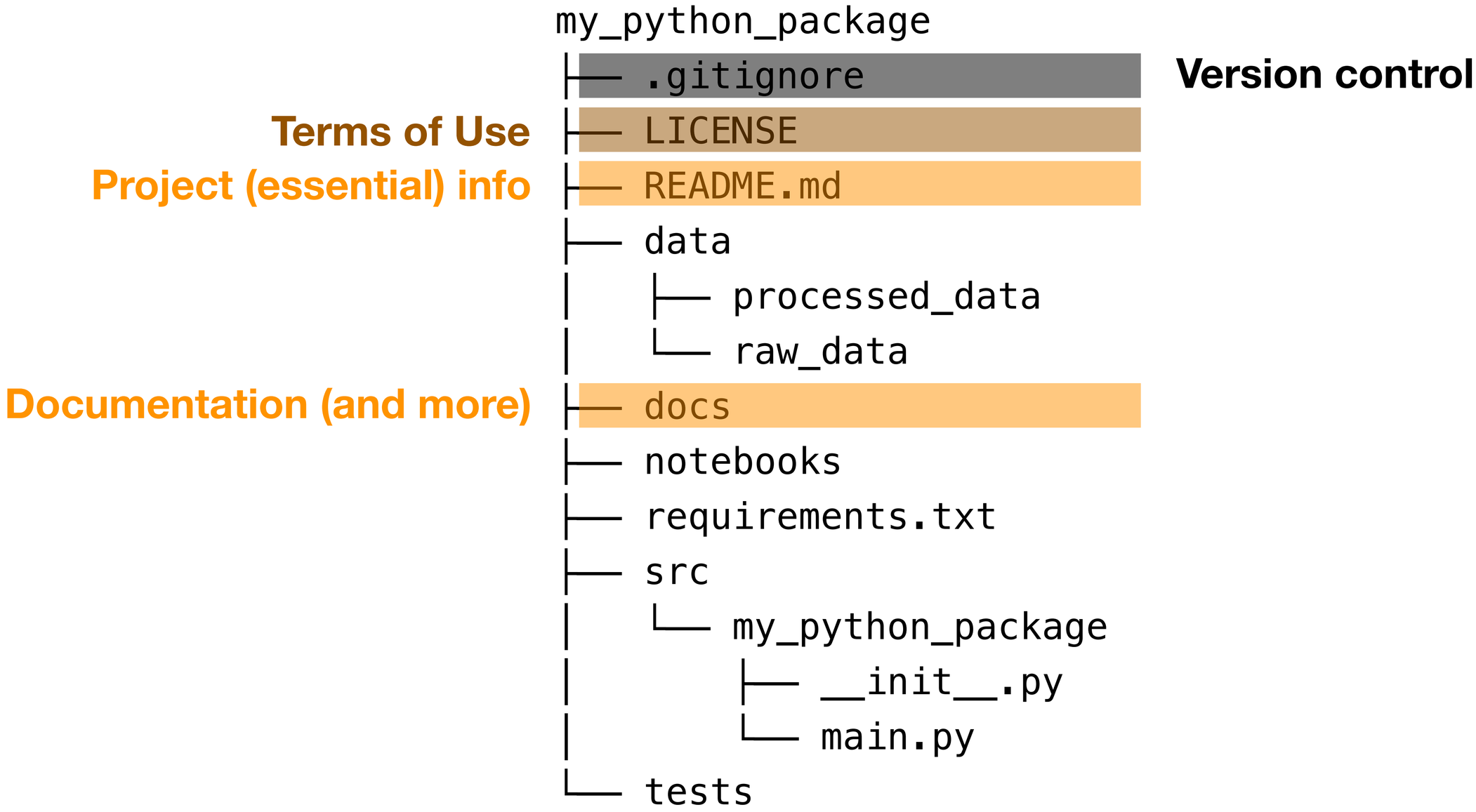
Project Structure (Python)



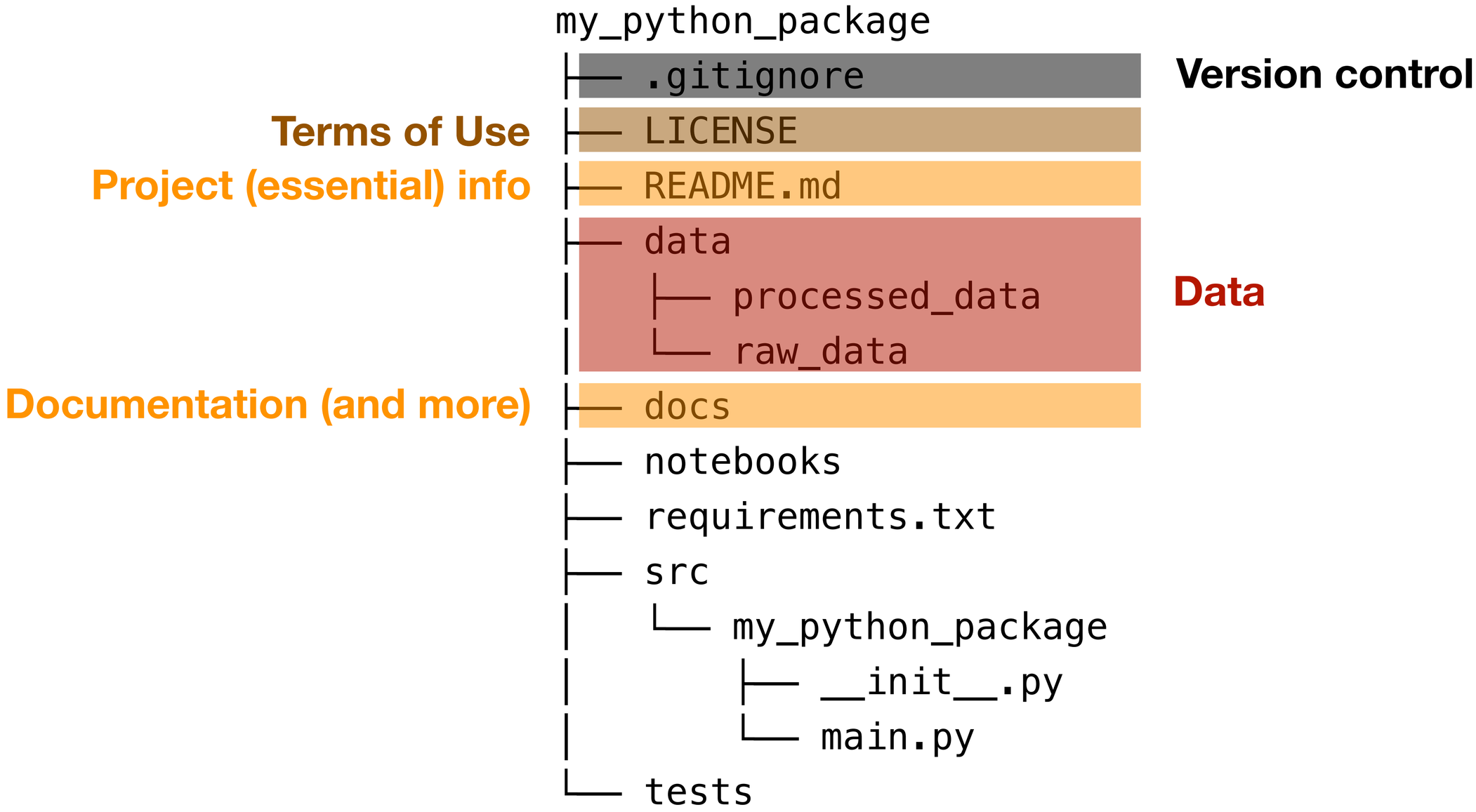
Project Structure (Python)



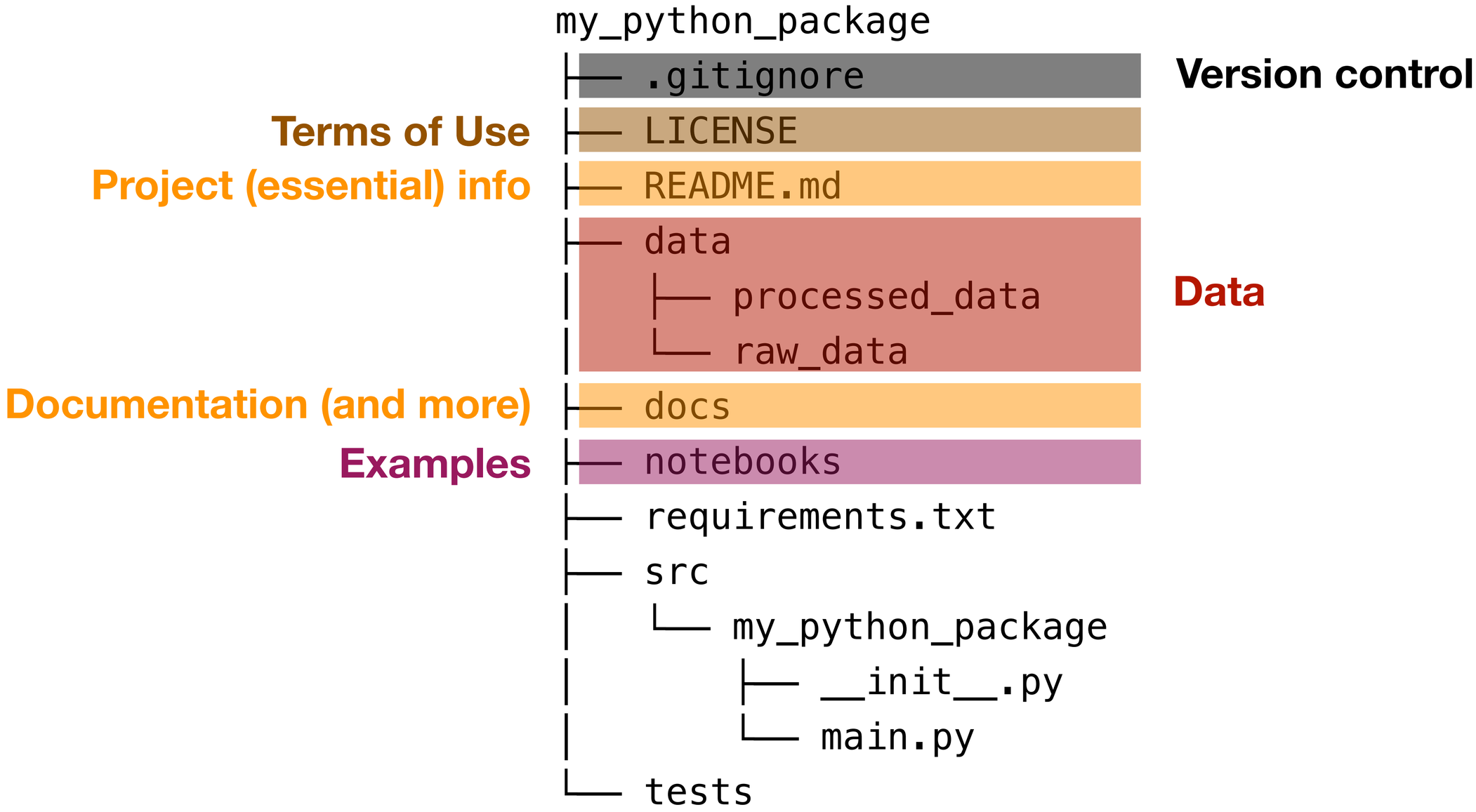
Project Structure (Python)



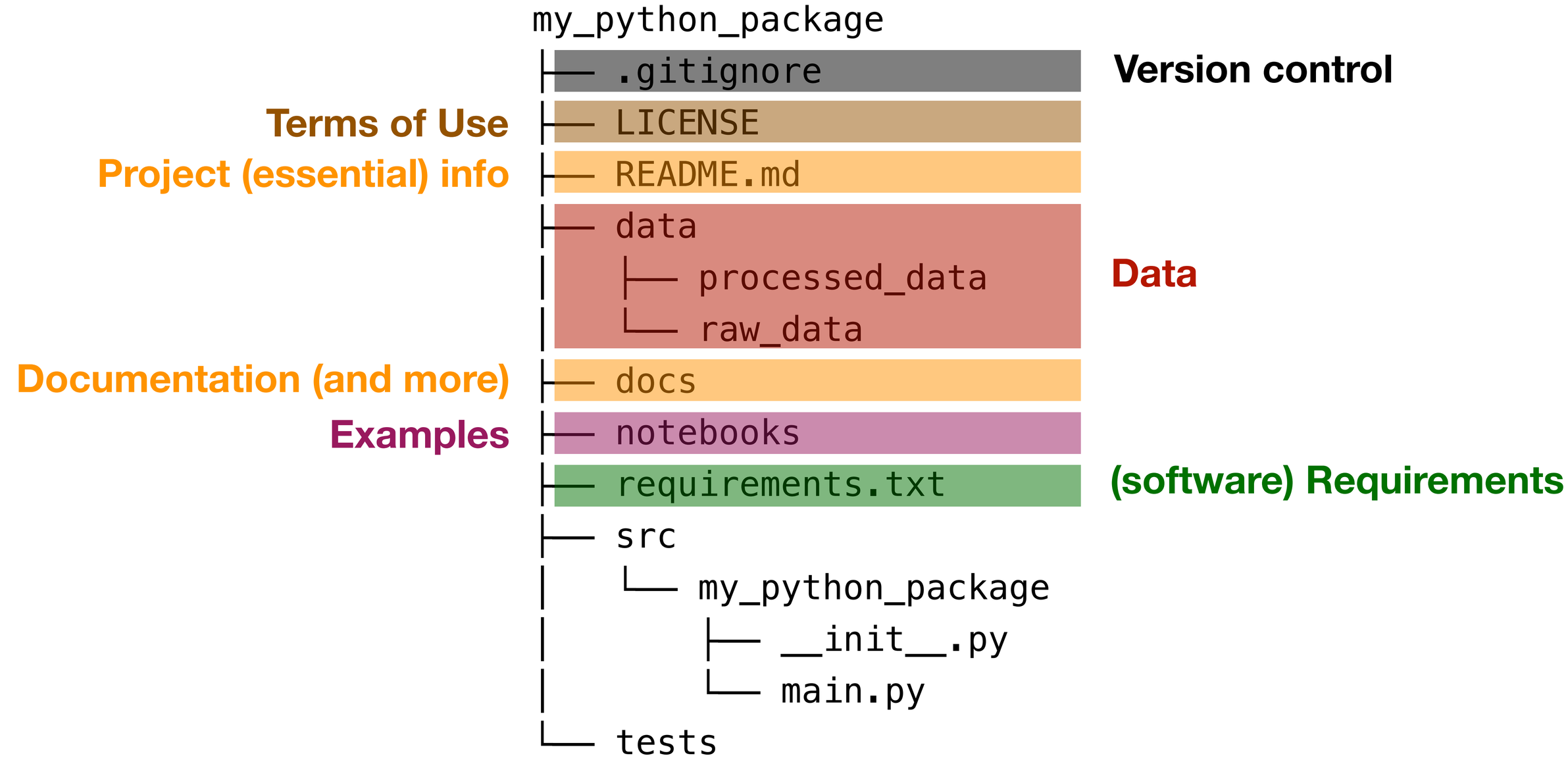
Project Structure (Python)



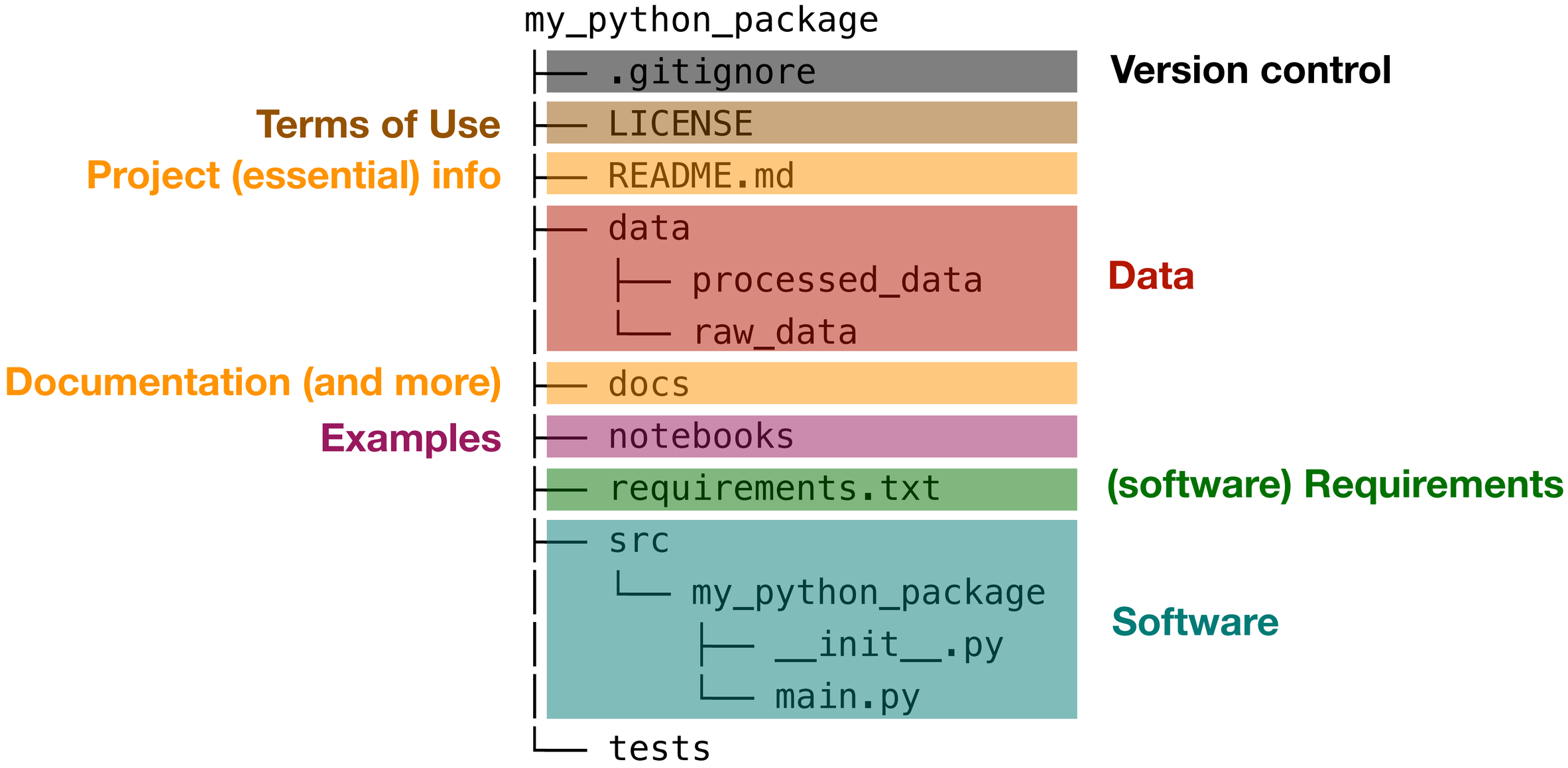
Project Structure (Python)



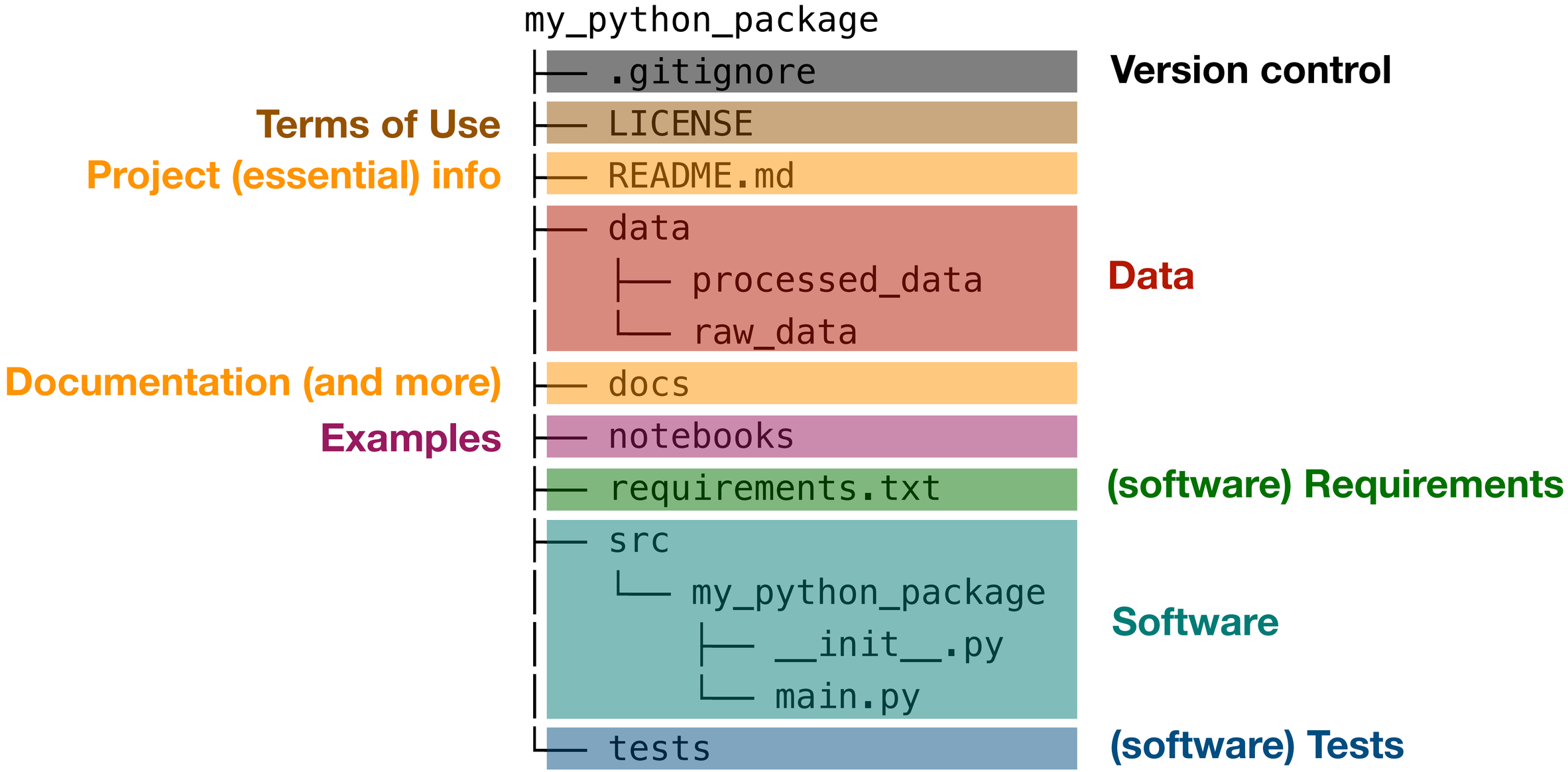
Project Structure (Python)



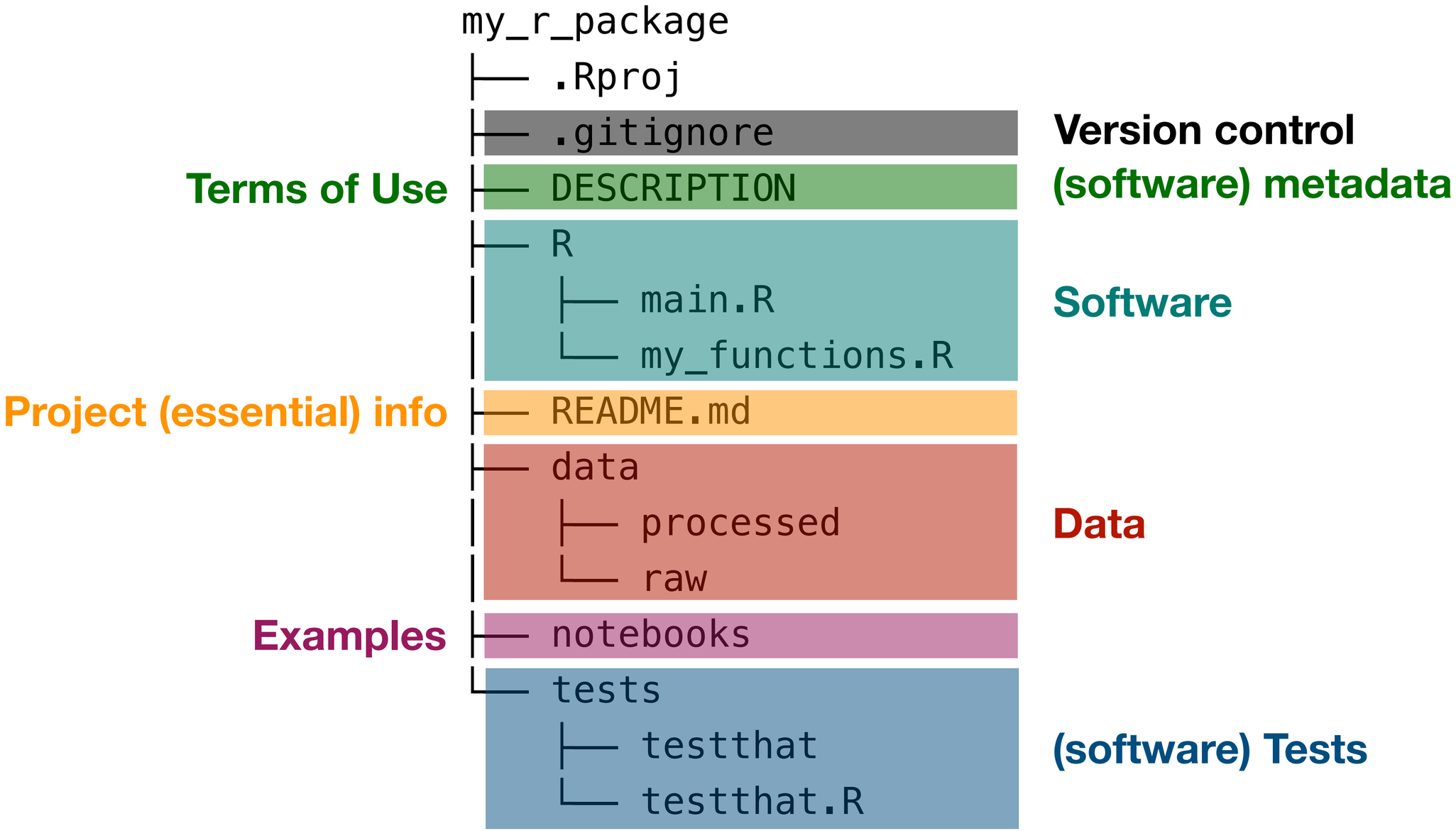
Project Structure (Python)



Project Structure (Python)



Project Structure (R)



Project Structure (Python)

Terms of Use

```
my_python_package
├── .gitignore
├── LICENSE
├── README.md
├── data
│   ├── processed_data
│   └── raw_data
├── docs
├── notebooks
├── requirements.txt
├── src
│   └── my_python_package
│       ├── __init__.py
│       └── main.py
└── tests
```

Choosing a License

Reflect on users

Choosing a License

Reflect on users

Choose a license (https://choosealicense.com/)

Choosing a License

Reflect on users

Choose a license (https://choosealicense.com/)

Consult your Department Research Support Office

Choosing a License

Reflect on users

Choose a license (https://choosealicense.com/)

Consult your Department Research Support Office

Provide the license information on your repository

Project Structure (Python)

Project (essential) info

```
my_python_package
├── .gitignore
├── LICENSE
├── README.md
├── data
│   ├── processed_data
│   └── raw_data
├── docs
├── notebooks
├── requirements.txt
├── src
│   └── my_python_package
│       ├── __init__.py
│       └── main.py
└── tests
```

Writing a README file

A README is often the first item a visitor will see when visiting a repository (DO NOT MAKE IT THE LAST!)

What is the project about

Project motivation

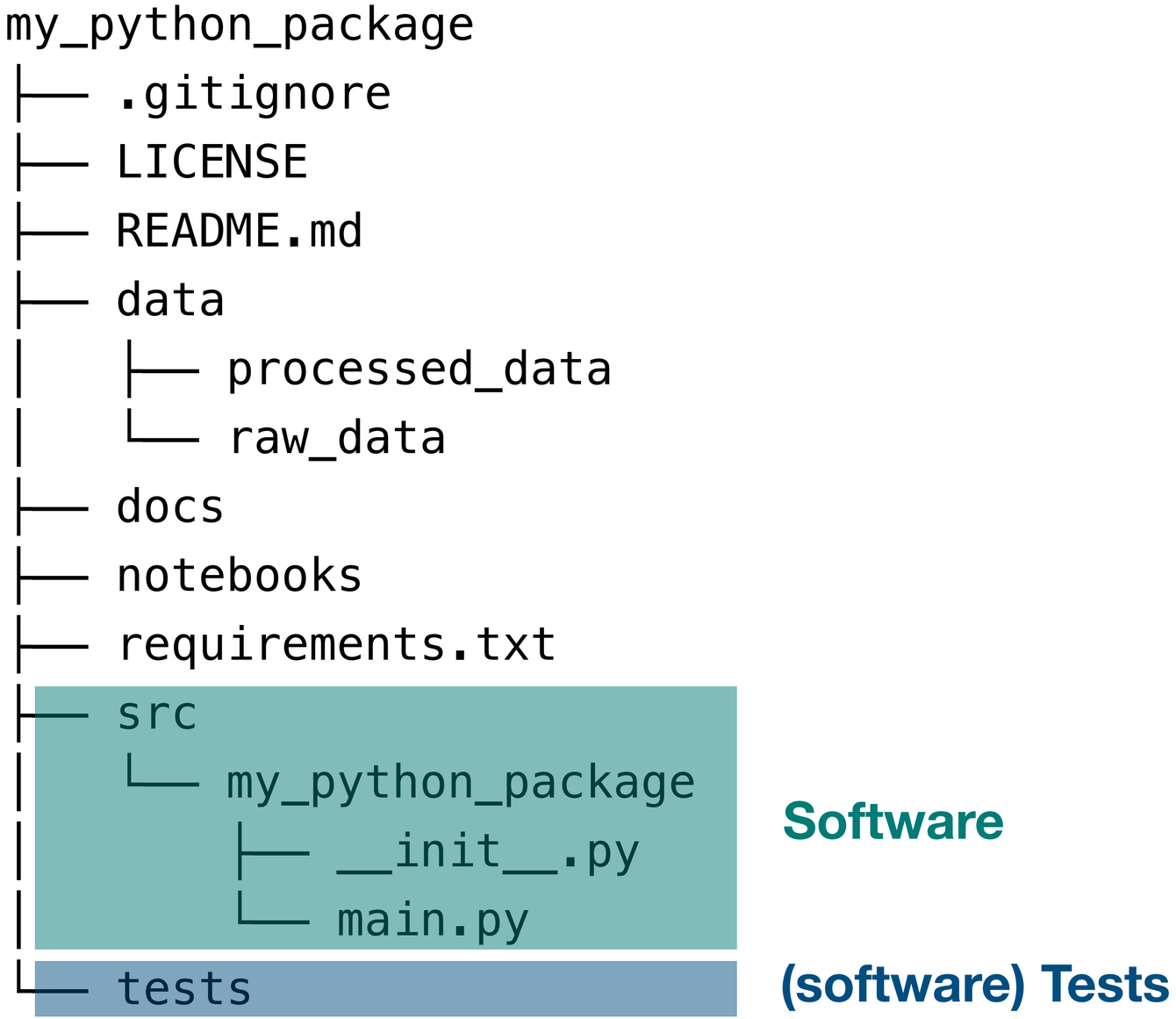
How can users get started

Where users can find help and support

Who maintains and distributes the project

[additional information]

Project Structure (Python)



FAIR (readable/clean/robust) software

Use white spaces and newlines strategically

Write good comments and docstrings

Use descriptive names for functions and variables

Adopt a consistent style

Assign to each function and class a single purpose

Test your code starting with Unit Test

Peer reviewed development

FAIR (readable/clean/robust) software

Use white spaces and newlines strategically

Write good comments and docstrings

Use descriptive names for functions and variables

Adopt a consistent style

Assign to each function and class a single purpose

Test your code starting with Unit Test

Peer reviewed development

FAIR (readable/clean/robust) software

Use white spaces and newlines strategically

Write good comments and docstrings

Use descriptive names for functions and variables

Adopt a consistent style

Assign to each function and class a single purpose

Test your code starting with Unit Test

Peer reviewed development

FAIR (readable/clean/robust) software

Use white spaces and newlines strategically

Write good comments and docstrings

Use descriptive names for functions and variables

Adopt a consistent style

Assign to each function and class a single purpose

Test your code starting with Unit Test

Peer reviewed development

FAIR (readable/clean/robust) software

Use white spaces and newlines strategically

Write good comments and docstrings

Use descriptive names for functions and variables

Adopt a consistent style

Assign to each function and class a single purpose

Test your code starting with Unit Test

Peer reviewed development

FAIR (readable/clean/robust) software

Use white spaces and newlines strategically

Write good comments and docstrings

Use descriptive names for functions and variables

Adopt a consistent style

Assign to each function and class a single purpose

Test your code starting with Unit Test

Peer reviewed development

FAIR (readable/clean/robust) software

Use white spaces and newlines strategically

Write good comments and docstrings

Use descriptive names for functions and variables

Adopt a consistent style

Assign to each function and class a single purpose

Test your code starting with Unit Test

Peer reviewed development

TAKE HOME (a.k.a. why all this??)

Reproducibility is a core principle of science

Publish your project to make it Findable

Define User terms

Provide the necessary information to make it Reusable and Interoperable

If a code runs, it does not mean it is good code