NICOLAUS COPERNICUS UNIVERSITY IN TORUŃ

RESEARCH UNIVERSITY EXCELLENCE INITIATIVE

How to structure a project

Iulia Emilia Brumboiu

iubr@umk.pl

3 November 2025

Today

Quick reminder of what we did last time.

How to structure a Python project:

- **Repository structure**
  - General structure
  - How to chose a license
- **Code structure**
  - General notes
  - Modules and classes

Last time:

- Intro to git: clone, add, commit, push, branch;
- You created a **git repository** for your project:
    - **README**
    - **pyproject.toml** (we'll use today and discuss in more details in the package and distribution lecture),
    - feature/iodata **branch**,
    - **Merge** request / pull request.

How to structure a Python project

Goal

Obtain a **clean** and **effective** code which is **easy to use and understand**.

An effective **project architecture** is obtained by having

- a **well-organized repository**,
- a **clear** and **well-ordered code**,
- **clear documentation**.

Guiding principle (Kenneth Reitz)[1]:

"Build tools for others that you want to be built for you."

[1] https://kennethreitz.org/essays/2013-01-how_i_develop_things_and_why

NICOLAUS COPERNICUS UNIVERSITY IN TORUŃ

Repository structure

- **README**
- **LICENSE**
- Files required for installation (e.g. **pyproject.toml**, or setup.py and requirements.txt)
- **The source code (packagename/)**
- Documentation (docs/)
- Tests (tests/)
- Examples (examples/)

Example repositories

- Simplemind
- VIAMD

[1] https://github.com/kennethreitz/simplemind
[2] https://github.com/scanberg/viamd

NICOLAUS COPERNICUS UNIVERSITY IN TORUŃ

The README file

- README.md – markdown file, uses a similar syntax as Jupyter markdown cells.
- Written using a lightweight markup language to add formatting elements:
  - Headers,
  - Lists,
  - Figures/images,
  - Equations,
  - Links.
- Should contain:
  - Short description of **what the code is for**,
  - **Installation instructions**,
  - Examples (or links to examples) of **how to run** the code,
  - Other useful information (DOI, how to cite, how to contribute, etc.).

NICOLAUS COPERNICUS UNIVERSITY IN TORUŃ

The LICENSE file

The licenses listed here apply to **open source software**. However, there are other types of software (proprietary, closed source, available source, etc.)

- You don’t need a license as long as your project is private.
- No license = no permissions, “all rights reserved”.
- For public projects, the license lets others know the conditions under which they can **use**, **modify** and **distribute** your code.
- Choose one and copy+paste the text into your LICENSE.txt or LICENSE.md file
- Most commonly used open source licenses:
  - GNU general public licenses (GPL)
  - MIT license
  - Apache license
  - BSD licenses

https://www.youtube.com/watch?v=UMIG4KnM8xw

The LICENSE file

Copyleft = the same rights preserved in derivative works (derivatives inherit license)

Permissive	**Weak copyleft**	**Strong copyleft**
MIT: license and copyright notice.	**LGPL**: disclose source, license and copyright notice, same license (library), state changes.	**GPL:** disclose source, license and copyright notice, same license, state changes.
BSD: license and copyright notice.	**MPL**: disclose source, license and copyright notice, same license (file).	
Apache: license and copyright notice, state changes.		

https://choosealicense.com/licenses/
https://choosealicense.com/appendix/
https://www.sonatype.com/blog/open-source-licenses-explained

NICOLAUS COPERNICUS UNIVERSITY IN TORUŃ

How to choose a license

- You can use an online tool like: https://choosealicense.com/

Repository structure

- **README**
- **LICENSE**
- Files required for installation (e.g. **pyproject.toml**, or setup.py and requirements.txt)
- **The source code (packagename/)**
- Documentation (docs/)
- Tests (tests/)
- Examples (examples/)

Code structure

- **Everything under packagename/**
- __init__.py
- Modules and classes

https://github.com/kennethreitz/kennethreitz.org

Code structure

- Everything under packagename/
- **__init__.py**
- Modules and classes

Any directory with an **__init__.py** file is considered **a Python package**.

__init__.py:

- gathers all package-wide definitions,
- should contain very little code (in some cases it can even be empty),
- controls what is made available when running **`import packagename`**.

https://docs.python-guide.org/writing/structure/

Code structure

- Everything under packagename/
- __init__.py
- **Modules** and classes

A module is a file containing **Python definitions and statements**.

- Imported using **import module_name**, where **module_name** is the name of the file without the **.py** suffix.
- If not part of a package, a module can be imported if it's in the same folder, or the folder path has been added to the **PYTHONPATH** environment variable.
- Routines of the module are accessed as **module_name.routine_name(...)**
- The script which you implemented last time to read/write NumPy arrays from/to a checkpoint file is a **module**.

https://docs.python.org/3/tutorial/modules.html

NICOLAUS COPERNICUS UNIVERSITY IN TORUŃ

Code structure

- Everything under packagename/
- __init__.py
- Moudules and **classes**

A class **defines** a **new type** of **Python object**.

- **Instance variables** define the object properties,
- **__init__** routine instantiates an object and assigns values to the object properties, or performs any necessary operations when an object is created.
- Within the class, the variable **self** refers to the current instance of the class and is used to access the **properties** and **methods** of the class.

https://docs.python.org/3/tutorial/classes.html
https://www.w3schools.com/python/python_oop.asp

Example: Solar system generator

Possible code structure:

```
solsysgen/
|— __init__.py
|— sun.py                 # Sun class
|— planet.py              # Planet class
|— solarsystem.py         # SolarSystem class
|— utils/                 # Utilities
    |— checkpoint.py      # Read/write from/to checkpoint files.
```

https://docs.python-guide.org/writing/structure/

Homework

- **Complete the merge/pull request** and delete the temporary feature/iodata branch from your repository.
- **Design** or **decide** which **project** you would like to carry out;
- Think about what modules and classes you will need and **sketch a code structure** (how modules relate to each other).
- **Update the README file** on your repository with a short description of your project and what your code is supposed to do. You can add equations and references/links to useful material.
- **Choose a license** for your project and add a **LICENSE** file to your repo.

Next time we will:

- Discuss your project ideas and the structure you plan for your code.

NICOLAUS COPERNICUS UNIVERSITY IN TORUŃ

Thank you for today!

Questions?