Hints for developing a Python project

Lucas Borges Ferreira

2023-11-16

Goals of a good Python project

- Readable
- Reusable
- Reliable

How to make my code more readable, reusable and reliable?

Naming conventions

- Use meaningful variable names
- Follow variable naming conventions
  - `snake_case` for variable and function names
  - `PascalCase` for class names

```
1  my_variable = 'something'
2
3  def my_function():
4      # do something
5
6  class MyClass:
7      def my_method(self):
8          # do something
```

Comments

- Use comments to explain non-obvious code

Unuseful comment:

```
1  # Change values below 0 to 0
2  data[data < 0] = 0
```

Useful comment:

```
1  # Values below 0 are removed since they are not physically possible
2  data[data < 0] = 0
```

> **General rule**: code says what, comments say why

Docstrings

- Use docstrings to describe functions and classes
  - Docstring can be seen when you call the function using an IDE (VS Code, PyCharm, etc)
  - Fast way to write docstrings: AutoDocstring (VS Code extension) and GitHub Copilot

```python
def clip_values(data, vmin=0, vmax=1):
    """Clips data to a minimum and maximum value.

    Args:
        data (np.ndarray): data to be processed
        vmin (float): minimum value. Defaults to 0.
        vmax (float): maximum value. Defaults to 1.

    Returns:
        np.ndarray: clipped data
    """
```

```
12      data = data.clip(vmin, vmax)
13      return data
```

Type hints

- Use type hints to make your code easier to understand and have a better IDE assistance

> Type hint is a formal solution to indicate the type of a variable

```python
def print_student_age(name: str, age: int):
    print(f"{name} is {age} years old")

def filter_data(data: np.ndarray, vmin: float) -> np.ndarray:
    return data[data > vmin]
```

Write good code

- Make your code as simple as possible
- Avoid duplicated code
- Functions are a great way to avoid duplicated code
- Classes provide a powerful way to create a well-structured code
- Object-oriented programming (OOP) and design patterns can help a lot in creating complex code in a more structured way

Some software development concepts

- Cohesion: how well the code inside a class or function is related
- Copling: how much a class or function is related to other classes or functions
- High cohesion and low coupling are desirable
- Single responsibility principle: a class or function should have only one responsibility

Scientific programming tips

- We often run several experiments with different parameters
- To keep your code organized, separate the code that creates your algorithm from the code that runs the experiments

Practical example

- Task description: create a code to read a geotiff file, compute vegetation indices, stack them, plot them, and export the result as a geotiff file
- Your code should be:
  - Readable
  - Reusable
  - Reliable

Other tips

- When to use classes or functions?
- Organizing your code as a package is very useful to make it more reusable
- Use a version control system (e.g. Git) to keep track of your code changes
- Autoformat your code using a tool like Black
- Code should also be reproducible - Save your dependencies in a requirements file

Thank you!