Best practices

Suggestions on how to make a mess less messy

(by Frank for lab meeting on 2022-03-08)

Suggested “rules”

Code for your colleagues, not for the computer

1. Good names
2. Short code blocks
3. Singular use
4. Good documentation

Infrastructure

1. Shared code ownership (anyone can edit everything)
2. four-eyes principle (reviews)
3. Be nice

Why

Increasing difficult to create

1. Code that works once
2. Code that works tomorrow
3. Code you can read tomorrow
4. Code that works on other machines
5. Code others can read
6. ...

1735. Bug-free perfect code :-)

(depending on complexity, this can add a few extra minutes...hours)

Increasingly difficult over project lifetime

1. Bug-free perfect code :-)
2. Code others can read and edit
3. Code that works for one [person|computer]

(depending on complexity, this adds a few extra minutes...weeks)

Good Names

- Intention-revealing
- Pronounceable
- Use `snake_case`, except `ClassNames` in CamelCase (Python)

Good:

- `column_name`
- `def get_all_ol_neurons()`
- `def find_shortest_path(neuron1, neuron2)`

Bad:

- `C`
- `def myNeurons(nc)`
- `def pathfinder(n1, n2)`

Short code blocks

- The shorter the better!
- No executable code longer than 1 screen
- Functions should be less than 20 lines

Single use

- Functions have one purpose
- One level of abstraction per function
- No side effects
- Create complexity by applying different functions

Good:

```
l1_criteria = NC(type=”L1”)
l1_neurons =
get_all_ol_neurons(l1_criteria)
l1_completion_stats =
calc_synapse_completion(l1_neurons)
```

Worse?:

```
l1_completion_stats =
calc_synapse_completion(“L1”)
```

Good documentation

- Explain intention
- Avoid redundancy
- Use names instead of comments where possible
- Use numpydoc [1] style (Python) for docstrings

Good:

```
l1_neurons =
fetch_neurons_by_type(type=”L1”)
```

Bad:

```
n = fetch(“L1”) # retrieve a list of L1
neurons and store in n
```

[1] https://numpydoc.readthedocs.io/en/latest/format.html

Technical solutions

- Static code analysis tools (linter)
- `pylint` <python-script.py>
  Generates score < 10.0
- Ask for review
  (example https://github.com/reiserlab/optic-lobe-connectome/pull/52)

Alternative linters: `pyflakes`, `pycodestyle`, `flake8`

Sources

- Robert C. Martin *Clean Code: A Handbook of Agile Software Craftsmanship* https://www.goodreads.com/book/show/3735293-clean-code
- Google Python Style Guide: https://google.github.io/styleguide/pyguide.html
- Andrew Hunt and David Thomas *The Pragmatic Programmer* https://www.goodreads.com/book/show/4099.The_Pragmatic_Programmer
- Python Enhancement Proposals (PEP) https://www.python.org/dev/peps/