Cheatography

Clean ML code and PEP8 guidelines Cheat Sheet

by andreeas26 via cheatography.com/133480/cs/27273/

Optimizing names

Reveal an **intention**.

Key concepts names should communicate:
- **Why** it exists?
- **What** does it do?
- **How** is it used?

Differences between variables should be as close to the beginning of the name as possible.

Avoid noisy labels. Use mature optional typing system.

Make Siri say it. Avoid abbreviation and make your names self-explanatory.

Adding the datatype to a variable should be replaced with typing information. `features: DataFrame` vs. `featur es_df`

No magic numbers. Use constants.

Be consistent! Pick a single word per concept, and use it everywhere it fits in.

Use technical names for backend, and domain names as you get closer the customer.

Optimize functions

Small is beautiful. 3, 5, maybe 5 lines max!

FUNCTIONS SHOULD DO ONE THING. THEY SHOULD DO IT WELL. THEY SHOULD DO IT ONLY.

Stop at maximum 3 arguments!

Avoid boolean arguments. This points to the fact that the function does more than *one thing*.

Lengthy list of configuration arguments should be grouped in a configuration object that *share a concept*

Comman Query Separation (CQS)
- **Command:** a function is changing some external “state”.
- **Query:** a function is returning some “information”.

Avoid side effects in feature engineering pipelines.

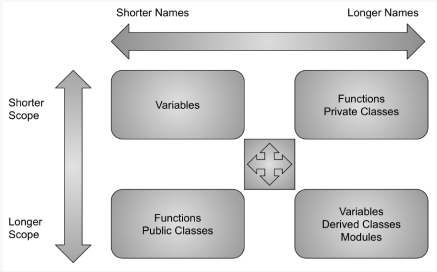
Make temporal couplings explicit.

DRY: Don't Repeat Yourself Eliminate Duplicates, Doubles, and Homologues.

Naming conventions

Function	Use verbs to represent actions. Use `is_` for functions that return a boolean.	`get_fe atures, fit, is_com pleted`
Variable	See scope length guidelines.	`x, var, my_var  iable`
Class	Use nouns to represent objects.	`Model, DataLoader`
Method	Lowercase word(s). Separate with unders-core.	`class_  me thod, method`
Constant	Uppercase single letter or word(s). Separate with unders-core.	`CONSTANT, MY_CON  STANT, MY_LON  NSTANT`
Module	Short lowercase word(s). Separate with unders-core.	module.py, my_module.py
Package	Short lowercase word(s). Do NOT separate with unders-core.	package, mypackage

The Scope Length Guidelines



How to avoid side effects in DataFrames

Copy any data coming in the function and return a fresh copy, after all the modifications.

Return only the transformed columns as a separated objects.

Append-only inside the functions.

By **andreeas26**

cheatography.com/andreeas26/

Published 25th March, 2021.
Last updated 26th March, 2021.

Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish Yours!
https://apollopad.com

Handling Exceptions

Promote the Happy Path with Exceptions.

Separate the Happy Path from the Outliers.

Don’t reuse unrelated Exceptions types.

Programming recommendations

Don’t compare Boolean values to True or False using the equivalence operator.

Use the fact that empty sequences are falsy in if statements.

Use `is not` rather than `not ... is` in `if` statements.

Don’t use `if x:` when you mean `if x is not None:`.

Use `.start swith()` and `.endsw ith()` instead of slicing strings.

Blank lines

Surround top-level functions and classes with two blank lines.

Surround method definitions inside classes with a single blank line.

Use blank lines sparingly inside functions to show clear steps.

Formulas always break before binary operations.

Whitespaces in Expressions and Statements

Assignment operators (`=, +=, -=,` and so forth).

Exception: when `=` is used to assign a default value to a function argument, do not surround it with spaces.

Comparisons (`==, !=, >, <. >=, <=`) and (`is, is not, in, not in`).

It is better to **only add whitespace around the operators with the lowest priority**, especially when performing mathematical manipulation.

Booleans (`and, not, or`).

Indentation

Spaces are the preferred indentation method.

Tabs should be used solely to remain consistent with code that is already indented with tabs.

Dictionaries should use the default formatting rules without indentation. Don't try to keep the values aligned.

Comments

Don’t Hide Bad Code Behind Comments. Let Code Explain Itself!

Limit the line length of comments and docstrings to 72 characters.

Use complete sentences, starting with a capital letter.

Make sure to update comments if you change your code.

Indent block comments to the same level as the code they describe.

Start each line with a # followed by a single space.

Separate paragraphs by a line containing a single #.

Write inline comments on the same line as the statement they refer to.

Separate inline comments by two or more spaces from the statement.

Don’t use them to explain the obvious.

Tips and tricks

Use an IDE or linters, programs that analyze code and flag errors (e.g pycodestyle, flake8)

You can use autoformatters that refactor your code to conform with PEP 8 automatically (e.g. black, autopep8, yapf).

Maximum Line Length

PEP 8 suggests lines should be limited to 79 characters.

Wrap long lines by using Python's implied line continuation inside parentheses, brackets and braces.

Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish Yours!
https://apollopad.com