# Contributing guidelines

## Before contributing

Welcome to [TheAlgorithms/Python](https://github.com/TheAlgorithms/Python)! Before sending your pull requests, make sure that you \*\*read the whole guidelines\*\*. If you have any doubt on the contributing guide, please feel free to [state it clearly in an issue](https://github.com/TheAlgorithms/Python/issues/new) or ask the community in [Gitter](https://gitter.im/TheAlgorithms).

## Contributing

### Contributor

We are very happy that you consider implementing algorithms and data structure for others! This repository is referenced and used by learners from all over the globe. Being one of our contributors, you agree and confirm that:

- You did your work - no plagiarism allowed

- Any plagiarized work will not be merged.

- Your work will be distributed under [MIT License](License) once your pull request is merged

- You submitted work fulfils or mostly fulfils our styles and standards

\*\*New implementation\*\* is welcome! For example, new solutions for a problem, different representations for a graph data structure or algorithm designs with different complexity.

\*\*Improving comments\*\* and \*\*writing proper tests\*\* are also highly welcome.

### Contribution

We appreciate any contribution, from fixing a grammar mistake in a comment to implementing complex algorithms. Please read this section if you are contributing your work.

Your contribution will be tested by our [automated testing on Travis CI](https://travis-ci.org/TheAlgorithms/Python/pull\_requests) to save time and mental energy. After you have submitted your pull request, you should see the Travis tests start to run at the bottom of your submission page. If those tests fail, then click on the \_\_\_details\_\_\_ button try to read through the Travis output to understand the failure. If you do not understand, please leave a comment on your submission page and a community member will try to help.

Please help us keep our issue list small by adding fixes: #{$ISSUE\_NO} to the commit message of pull requests that resolve open issues. GitHub will use this tag to auto close the issue when the PR is merged.

#### What is an Algorithm?

An Algorithm is one or more functions (or classes) that:

\* take one or more inputs,

\* perform some internal calculations or data manipulations,

\* return one or more outputs,

\* have minimal side effects (Ex. print(), plot(), read(), write()).

Algorithms should be packaged in a way that would make it easy for readers to put them into larger programs.

Algorithms should:

\* have intuitive class and function names that make their purpose clear to readers

\* use Python naming conventions and intuitive variable names to ease comprehension

\* be flexible to take different input values

\* have Python type hints for their input parameters and return values

\* raise Python exceptions (ValueError, etc.) on erroneous input values

\* have docstrings with clear explanations and/or URLs to source materials

\* contain doctests that test both valid and erroneous input values

\* return all calculation results instead of printing or plotting them

Algorithms in this repo should not be how-to examples for existing Python packages. Instead, they should perform internal calculations or manipulations to convert input values into different output values. Those calculations or manipulations can use data types, classes, or functions of existing Python packages but each algorithm in this repo should add unique value.

#### Coding Style

We want your work to be readable by others; therefore, we encourage you to note the following:

- Please write in Python 3.7+. \_\_print()\_\_ is a function in Python 3 so \_\_print "Hello"\_\_ will \_not\_ work but \_\_print("Hello")\_\_ will.

- Please focus hard on naming of functions, classes, and variables. Help your reader by using \_\_descriptive names\_\_ that can help you to remove redundant comments.

- Single letter variable names are \_old school\_ so please avoid them unless their life only spans a few lines.

- Expand acronyms because \_\_gcd()\_\_ is hard to understand but \_\_greatest\_common\_divisor()\_\_ is not.

- Please follow the [Python Naming Conventions](https://pep8.org/#prescriptive-naming-conventions) so variable\_names and function\_names should be lower\_case, CONSTANTS in UPPERCASE, ClassNames should be CamelCase, etc.

- We encourage the use of Python [f-strings](https://realpython.com/python-f-strings/#f-strings-a-new-and-improved-way-to-format-strings-in-python) where the make the code easier to read.

- Please consider running [\_\_psf/black\_\_](https://github.com/python/black) on your Python file(s) before submitting your pull request. This is not yet a requirement but it does make your code more readable and automatically aligns it with much of [PEP 8](https://www.python.org/dev/peps/pep-0008/). There are other code formatters (autopep8, yapf) but the \_\_black\_\_ formatter is now hosted by the Python Software Foundation. To use it,

```bash

pip3 install black # only required the first time

black .

```

- All submissions will need to pass the test \_\_flake8 . --ignore=E203,W503 --max-line-length=88\_\_ before they will be accepted so if possible, try this test locally on your Python file(s) before submitting your pull request.

```bash

pip3 install flake8 # only required the first time

flake8 . --ignore=E203,W503 --max-line-length=88 --show-source

```

- Original code submission require docstrings or comments to describe your work.

- More on docstrings and comments:

If you used a Wikipedia article or some other source material to create your algorithm, please add the URL in a docstring or comment to help your reader.

The following are considered to be bad and may be requested to be improved:

```python

x = x + 2 # increased by 2

```

This is too trivial. Comments are expected to be explanatory. For comments, you can write them above, on or below a line of code, as long as you are consistent within the same piece of code.

We encourage you to put docstrings inside your functions but please pay attention to indentation of docstrings. The following is a good example:

```python

def sum\_ab(a, b):

"""

Return the sum of two integers a and b.

"""

return a + b

```

- Write tests (especially [\_\_doctests\_\_](https://docs.python.org/3/library/doctest.html)) to illustrate and verify your work. We highly encourage the use of \_doctests on all functions\_.

```python

def sum\_ab(a, b):

"""

Return the sum of two integers a and b

>>> sum\_ab(2, 2)

4

>>> sum\_ab(-2, 3)

1

>>> sum\_ab(4.9, 5.1)

10.0

"""

return a + b

```

These doctests will be run by pytest as part of our automated testing so please try to run your doctests locally and make sure that they are found and pass:

```bash

python3 -m doctest -v my\_submission.py

```

The use of the Python builtin \_\_input()\_\_ function is \*\*not\*\* encouraged:

```python

input('Enter your input:')

# Or even worse...

input = eval(input("Enter your input: "))

```

However, if your code uses \_\_input()\_\_ then we encourage you to gracefully deal with leading and trailing whitespace in user input by adding \_\_.strip()\_\_ as in:

```python

starting\_value = int(input("Please enter a starting value: ").strip())

```

The use of [Python type hints](https://docs.python.org/3/library/typing.html) is encouraged for function parameters and return values. Our automated testing will run [mypy](http://mypy-lang.org) so run that locally before making your submission.

```python

def sum\_ab(a: int, b: int) -> int:

return a + b

```

- [\_\_List comprehensions and generators\_\_](https://docs.python.org/3/tutorial/datastructures.html#list-comprehensions) are preferred over the use of `lambda`, `map`, `filter`, `reduce` but the important thing is to demonstrate the power of Python in code that is easy to read and maintain.

- Avoid importing external libraries for basic algorithms. Only use those libraries for complicated algorithms.

- If you need a third party module that is not in the file \_\_requirements.txt\_\_, please add it to that file as part of your submission.

#### Other Standard While Submitting Your Work

- File extension for code should be `.py`. Jupyter notebook files are acceptable in machine learning algorithms.

- Strictly use snake\_case (underscore\_separated) in your file\_name, as it will be easy to parse in future using scripts.

- Please avoid creating new directories if at all possible. Try to fit your work into the existing directory structure.

- If possible, follow the standard \*within\* the folder you are submitting to.

- If you have modified/added code work, make sure the code compiles before submitting.

- If you have modified/added documentation work, ensure your language is concise and contains no grammar errors.

- Do not update the README.md or DIRECTORY.md file which will be periodically autogenerated by our Travis CI processes.

- Add a corresponding explanation to [Algorithms-Explanation](https://github.com/TheAlgorithms/Algorithms-Explanation) (Optional but recommended).

- All submissions will be tested with [\_\_mypy\_\_](http://www.mypy-lang.org) so we encourage to add [\_\_Python type hints\_\_](https://docs.python.org/3/library/typing.html) where it makes sense to do so.

- Most importantly,

- \*\*Be consistent in the use of these guidelines when submitting.\*\*

- \*\*Join\*\* [Gitter](https://gitter.im/TheAlgorithms) \*\*now!\*\*

- Happy coding!

Writer [@poyea](https://github.com/poyea), Jun 2019.