### Python Code Review

First of all: great work :)

There are a few things I would like to point out:

1. Code layout

### Some of the code lines are too long according to PEP8 coding standard (see Code Layout > Maximum Line Length and Line Breaking at <https://realpython.com/python-pep8/>).

There are python code linter tools that check your code syntax and provide instructions on how to clean it. Flake8 is a great toolkit for checking your code base against coding style (PEP8), programming errors (like “library imported but unused” and “Undefined name”) and to check cyclomatic complexity.

- Install using pip:

pip install flake8

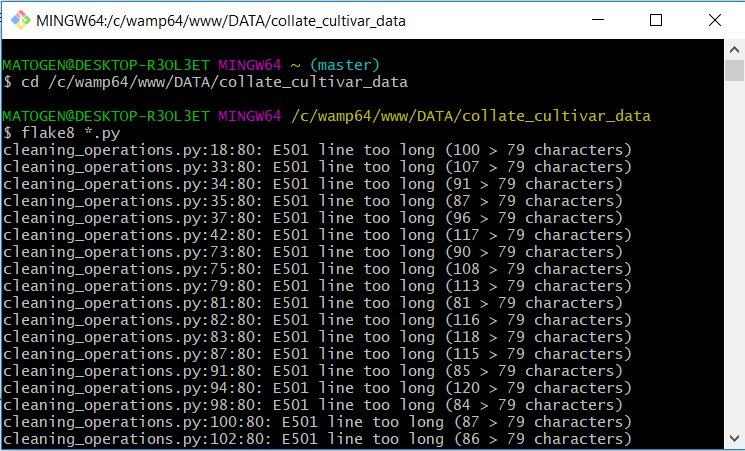
- Run inside project root directory:

flake8 path/to/your\_code/main.py #check particular file

- Or give it all the python files in the project root directory:

flake8 \*.py #check all python files

- Output:



- The output is formatted as:

file path : line number : column number : error code : short description

Error code prefix:

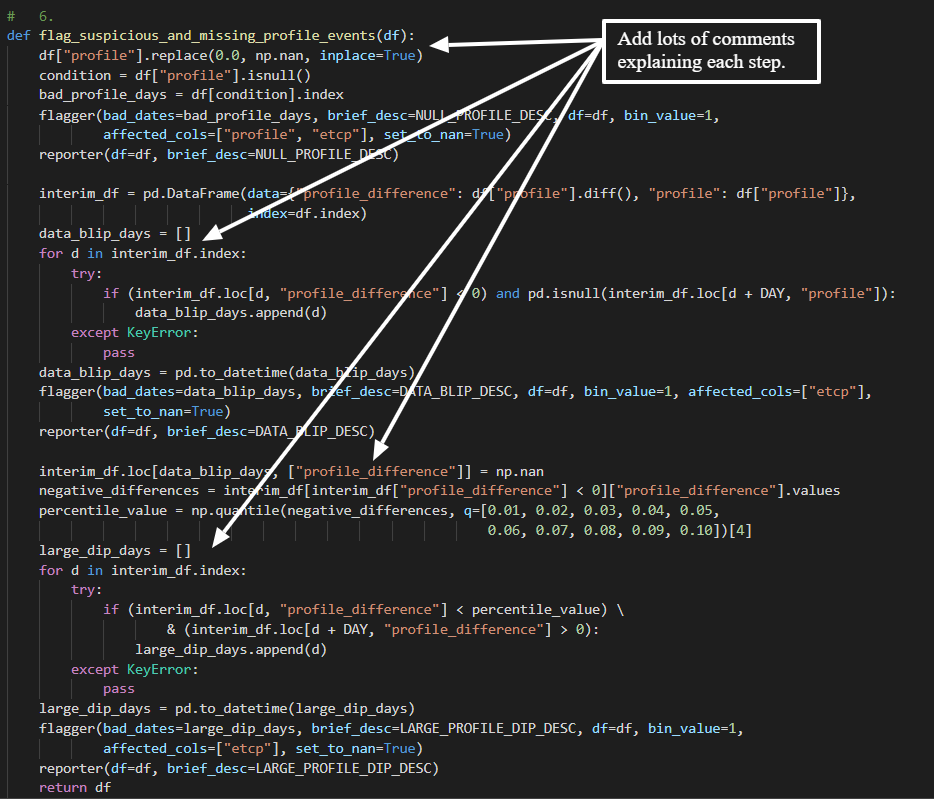
* **E\*\*\*/W\*\*\*:** pep8 errors and warnings
* **F\*\*\*:** PyFlakes codes
* **C9\*\*:** McCabe complexity plugin mccabe
* **N8\*\*:** Naming Conventions plugin pep8-naming

Full list of error codes are available here:

* [PEP8 error codes](http://pep8.readthedocs.io/en/latest/intro.html" \l "error-codes" \t "https://simpleisbetterthancomplex.com/packages/2016/08/05/_blank): <https://pep8.readthedocs.io/en/latest/intro.html#error-codes>
* [PEP8 Naming error codes](https://github.com/PyCQA/pep8-naming" \t "https://simpleisbetterthancomplex.com/packages/2016/08/05/_blank): <https://github.com/PyCQA/pep8-naming>

1. Comments

Add more comments within the code:



1. Remove duplicate functions

Functions that is used in multiple files can be placed in a separate file (for example: functions.py) and then called whenever needed. This will prevent duplicate code and make maintaining the code (helper functions) easier.

Example of duplicate functions found:

- find\_nearest\_index(model\_array, raw\_value)

- gaussian(x, amp=1, mean=0, sigma=10)

- weighted\_moving\_average(x, y, step\_size=1.0, width=10, x\_lims=None)

- get\_r\_squared(x\_raw, y\_raw, x\_fit, y\_fit)

- get\_n\_local\_extrema(y\_fit)

- get\_prized\_index(n\_bumps\_list)

Example:

- File named functions.py with functions:

def b():

# Something

return 1

def c():

# Something

return 2

- Import these functions in a different file (file.py):

Import functions as f

- Then call any function inside file.py using:

f.b()