

METplus Coding Standards/Developer Best Practices

Minna Win, Dan Adriaansen

Background

METplus is a work in progress, spanning multiple agencies and numerous contributors. In an effort to maintain code consistency and maintainability, some developer best practices/coding standards are needed.

```
204         if valid_hh == '00' or valid_hh == '12':
205             track_dict['lead_group'] = '0'
206         elif valid_hh == '06' or valid_hh == '18':
207             track_dict['lead_group'] = '6'
208         else:
209             # To gracefully handle any hours other
210             # than 0, 6, 12, or 18
211             track_dict['lead_group'] = ''
212
213         all_tracks_list.append(track_dict)
214
215         # For future work, support for MSLP when
216         # generating regional plots-
217         # implementation goes here...
218         # Finishing up, do any cleaning up,
219         # logging, etc.
220         self.logger.info("INFO: All criteria met, " +
221                          "saving track data init " +
222                          track_dict['init_time'] +
223                          " lead " +
224                          track_dict['fcst_lead_hh'] +
225                          " lon " +
226                          str(track_dict['lon']) +
227                          " lat " +
228                          str(track_dict['lat']))
```

Coding Standards

- NCEP Coding Standards
- NCO WCOSS Implementation Standards for directory structure and script naming conventions:
http://www.nco.ncep.noaa.gov/idsb/implemntation_standards/)
- pep8 for code style
- Doxygen and Python docstrings for documentation

Coding Standards

- **NCEP Coding Standards**

 - Python section of NCEP standards

 - Based on Google Python Coding standards

 - Language rules

 - Version 2016a found on NCAR/METplus GitHub wiki

 - https://github.com/NCAR/METplus/doc/NCEP_Coding_Standards.pdf

- NCO WCOSS Implementation Standards for directory structure and script naming conventions:

 - http://www.nco.ncep.noaa.gov/idsb/implementation_standards/)

- pep8 for code style

- Doxygen and Python docstrings for documentation

Coding Standards

- NCEP Coding Standards
- **NCO WCOSS Implementation Standards** for directory structure and script naming conventions:
http://www.nco.ncep.noaa.gov/idsb/implemntation_standards/)
- pep8 for code style
- Doxygen and Python docstrings for documentation

Coding Standards

- NCEP Coding Standards
- NCO WCOSS Implementation Standards for directory structure and script naming conventions:
http://www.nco.ncep.noaa.gov/idsb/implemntation_standards/)
- **pep8 for code style**
Addresses code consistency, readability, maintainability
Defines layout such as whitespace, line length, naming conventions
Interested in the details? Go to
<https://www.python.org/dev/peps/pep-0008>
- Doxygen and Python docstrings for documentation

METplus Coding Style

use *pythoncodestyle* to check if your code adheres to pep8:

The screenshot shows the PyPI page for the `pycodestyle` package. The header is blue with a search bar and navigation links (Help, Donate, Log in, Register). The main content area is white with a blue header for the package name `pycodestyle 2.4.0`. A green badge indicates it is the "Latest version" and another note says "Last released: Apr 10, 2018". Below this, there is a command to install the package: `pip install pycodestyle`. The package description states it is a "Python style guide checker". The page is divided into sections: Navigation (Project description, Release history, Download files), Project links (Homepage), Statistics (Libraries.io, Google BigQuery), Meta (License: MIT License, Author: Johann C. Rocholl, Maintainer: Ian Lee), and Maintainers (graffatcolmingov, IanLee1521). The Project description section includes a "Project description" heading, a list of badges (build passing, docs passing, wheel yes, chat on glitter), a paragraph describing the tool, a "Note" about the package's name change from `pep8` to `pycodestyle`, a "Features" list, an "Installation" section with commands, and an "Example usage and output" section.

Navigation

- Project description
- Release history
- Download files

Project links

- Homepage

Statistics

View statistics for this project via [Libraries.io](#), or by using [Google BigQuery](#)

Meta

License: MIT License (Expat license)

Author: [Johann C. Rocholl](#)

Maintainer: [Ian Lee](#)

`pycodestyle`, `pep8`, `PEP 8`, `PEP-8`, `PEP8`

Maintainers

- [graffatcolmingov](#)
- [IanLee1521](#)

Project description

pycodestyle (formerly called pep8) - Python style guide checker

build passing docs passing wheel yes chat on glitter

pycodestyle is a tool to check your Python code against some of the style conventions in [PEP 8](#).

Note

This package used to be called `pep8` but was renamed to `pycodestyle` to reduce confusion. Further discussion can be found [in the issue where Guido requested this change](#), or in the lightning talk at PyCon 2016 by [@IanLee1521](#): [slides video](#).

Features

- Plugin architecture: Adding new checks is easy.
- Parseable output: Jump to error location in your editor.
- Small: Just one Python file, requires only `stdlib`. You can use just the `pycodestyle.py` file for this purpose.
- Comes with a comprehensive test suite.

Installation

You can install, upgrade, and uninstall `pycodestyle.py` with these commands:

```
$ pip install pycodestyle
$ pip install --upgrade pycodestyle
$ pip uninstall pycodestyle
```

There's also a package for Debian/Ubuntu, but it's not always the latest version.

Example usage and output

```
$ pycodestyle --first-optional
```

METplus Documentation

Doxygen:

- Is a documentation generator
- Can be created by commenting the source code (with some additional syntax)
- Is relatively easy to keep up to date
- Creates HTML files which you can view in any browser

← → ↻ ⓘ Not Secure | www.doxygen.nl



Doxygen

Donate

€ (EUR)

Home Downloads Documentation Extensions Support

Doxygen

About

- ▶ Downloads
- Changelog
- ▶ Documentation
- ▶ Get Involved
- Wish list
- ▶ Examples
- ▶ Links
- Extensions
- Support
- Report bugs
- Donate

Doxygen is the de facto standard tool for generating documentation from annotated C++ sources, but it also supports other popular programming languages such as C, Objective-C, C#, PHP, Java, Python, IDL (Corba, Microsoft, and UNO/OpenOffice flavors), Fortran, VHDL, Tcl, and to some extent D.

Doxygen can help you in three ways:

1. It can generate an on-line documentation browser (in HTML) and/or an off-line reference manual (in \LaTeX) from a set of documented source files. There is also support for generating output in RTF (MS-Word), PostScript, hyperlinked PDF, compressed HTML, and Unix man pages. The documentation is extracted directly from the sources, which makes it much easier to keep the documentation consistent with the source code.
2. You can [configure](#) doxygen to extract the code structure from undocumented source files. This is very useful to quickly find your way in large source distributions. Doxygen can also visualize the relations between the various elements by means of include dependency graphs, inheritance diagrams, and collaboration diagrams, which are all generated automatically.
3. You can also use doxygen for creating normal documentation (as I did for the doxygen user manual and web-site).

Doxygen is developed under Mac OS X and Linux, but is set-up to be highly portable. As a result, it runs on most other Unix flavors as well. Furthermore, executables for Windows are available.



GitHub Repository

METplus code is available on a public repository:

<https://github.com/NCAR/METplus>

Based on *workflows* (widely accepted practice by the private sector)

This is how we make releases

Branches you'll see at any given time:

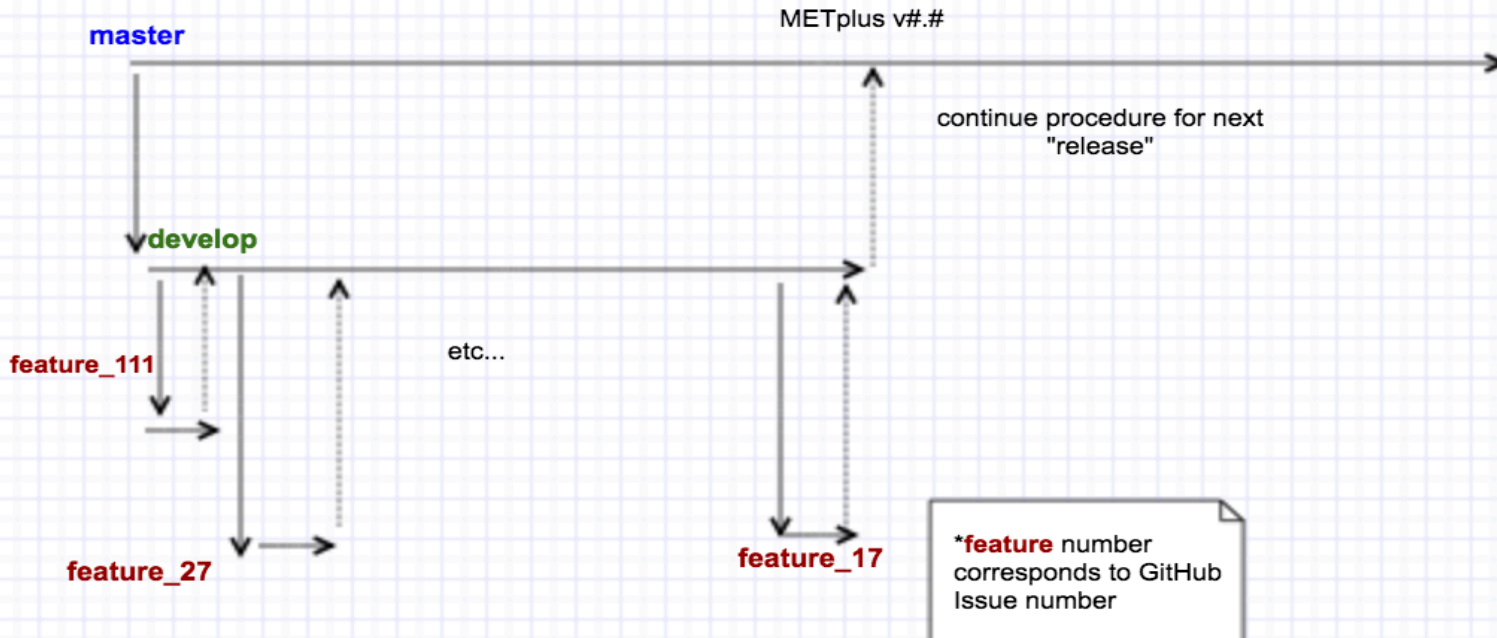
master

develop

feature_### (associated with GitHub Issue #)

GitHub Repository cont'd

Time (increasing from left to right)



LEGEND

→ "branch" /source code
*longer line indicates longer period of time

↑ merge to "parent"
↓ copy from source

Getting the source

<https://github.com/NCAR/METplus>

*No GitHub account is necessary (repository is public)

Clone the repository (get the source) three ways:

1. Command line

- Best if planning on contributing source

2a. Your browser - downloading the tarball

- Good for users - Source and docs only

2b. Your browser - release link

- Good for users - Source, docs, and sample data

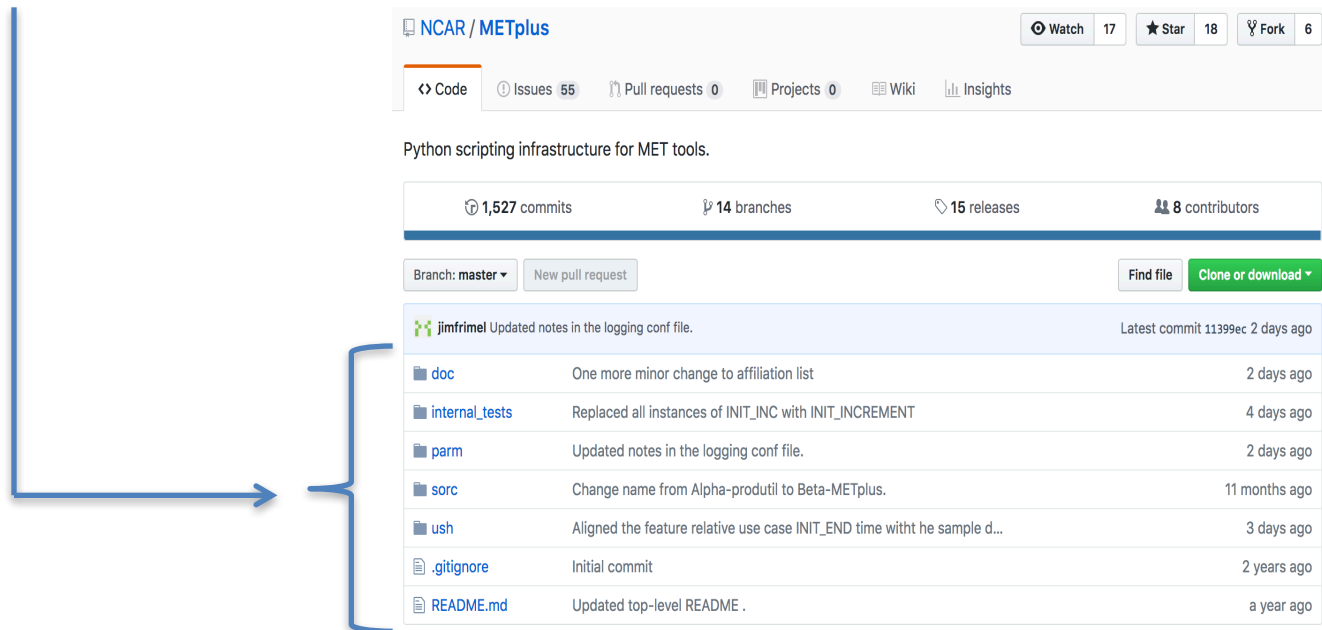
FIRST: Create the directory where you want to store the repository

Cloning from the command line

1. Change to the directory you created
2. Run the following from your terminal:

```
git clone https://github.com/NCAR/METplus
```

The entire repository resides under the *METplus* directory



NCAR / METplus

Python scripting infrastructure for MET tools.

1,527 commits 14 branches 15 releases 8 contributors

Branch: master New pull request Find file Clone or download

File/Folder	Description	Time
doc	One more minor change to affiliation list	2 days ago
internal_tests	Replaced all instances of INIT_INC with INIT_INCREMENT	4 days ago
parm	Updated notes in the logging conf file.	2 days ago
sorc	Change name from Alpha-productil to Beta-METplus.	11 months ago
ush	Aligned the feature relative use case INIT_END time with the sample d...	3 days ago
.gitignore	Initial commit	2 years ago
README.md	Updated top-level README .	a year ago

Cloning from browser (2b)

NCAR / METplus

<> Code Issues 55 Pull requests 0 Projects 0 Wiki Insights

Releases Tags

Latest release

v2.0
11399ec

METplus-2.0
jimfrimel released this 2 days ago

Assets 6

- METplus_Users_Guide.pdf
- sample_data-cyclone_track_feature.tgz
- sample_data-grid_to_grid.tgz
- sample_data-qpf.tar.gz
- Source code (zip)
- Source code (tar.gz)

Note other options-
User guide and
sample data available
at the release page.

Step 3

Cloning- what do you get?

Time (increasing from left
to right)

master

METplus v#.#



METplus Common Installations

Theia:

`/contrib/met/METplus/METplus-2.0`

Tide:

`/global/noscrub/Julie.Prestopnik/METplus/METplus-2.0`

Configuration files

master_metplus.py

METplus
config
scripts

metplus_final.conf

```
63 MET_BUILD_BASE = /path/to
64 MET_BASE = {MET_BUILD_BASE}/share/met
65
66 ## Output directories
67 LOG_DIR = {OUTPUT_BASE}/logs
68 TMP_DIR = {OUTPUT_BASE}/tmp
69
70 # DIRECTORIES
71 # [dir]
72 # Input data
73 [exe] # This is the executable
74 # NON_EXECUTABLE # This is a non-executable
75 WGRIB2 # WGRIB2 executable
76 CUT_SHORT # CUT_SHORT executable
77 TR_EXTRACT # TR_EXTRACT executable
78 RM_EXTRACT # RM_EXTRACT executable
79 NCAP2 # NCAP2 executable
80 CONVERT # CONVERT executable
81 NCDUMP # NCDUMP executable
82 EGREP # EGREP executable
83
84 [config]
85 EXPT=METplus ;; Experiment name, used for finding installation location
86
87 # Options are processes, times
88 LOOP_METHOD = processes
89
90 # Processes to run in master script (master_met_plus.py)
91 PROCESS_LIST = Usage
92
93 # NOTE: "TOTAL" is a REQUIRED cnt statistic used by the series analysis
94 STAT_LIST = TOTAL, FBAR, OBAR, ME, MAE, RMSE, BCSE, E50, E1QR, MAD
95
96 # Init time
97 INIT_TIME_FMT = %Y%m%d
98 INIT_BEG = 20141214
99 INIT_END = 20141216
100 INIT_INC = 21600
101 #21600 sec (6hours) The increment in seconds in integer format
102
103 # LOGGING
104 LOG_LEVEL = DEBUG ;; Levels: DEBUG, INFO, WARNING, ERROR, CRITICAL
105 LOG_FILENAME = {LOG_DIR}/master_met_plus.log ;; NOTE: current YYYYMMDD
```

⋮

Input

MET Tool
1

Output
1

METplus
Script 1

METplus
Script 2

Output
2

METplus
Script 3

MET Tool
2

Output
3

⋮

From .conf
to running MET