

# Git Thermite Description

## Description of Git Thermite

Git Thermite is a tool to visualize GitHub Pull requests. A pull request in GitHub is a formal request made to the maintainers of an open source project hosted in GitHub to integrate a set of git commits. A pull request can be accepted or rejected by the maintainers of the open source project. Determining whether a pull request deserves to be included in the application mainstream branch is often perceived as a difficult and tedious task since it involves comparing source code. The objective of Git Thermite is to ease this activity by offering a visual tool to project maintainers.

# Pull Request Overview

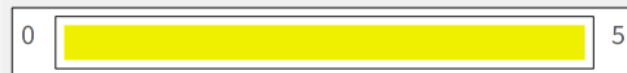
The figure above displays our visualization for a pull request in Python or Pharo. It contains a title bar (Part A) with the name of the pull request, the type of the pull request if it can be inferred, and a square indicating about the possibility of doing an automatic merge. The second section (Part B) displays the different charts with metrics about the modifications done in the pull request. By clicking on the bars of the different charts it is possible obtain a visualization of the elements that are represented by the bar. The last section (Part C) contains a number of navigation buttons to move around into the different visualizations of the pull request provided by Thermite.

A

DEP: Deprecate `np.ma.MaskedArray.mini` Deprecate

Metrics

Files



Lines of Code



Classes



Methods/Functions



B

Legend

■ Unchanged

■ Additions

■ Deletions

■ Modified

■ Moved/Renamed

C

Visualizations

File Changes

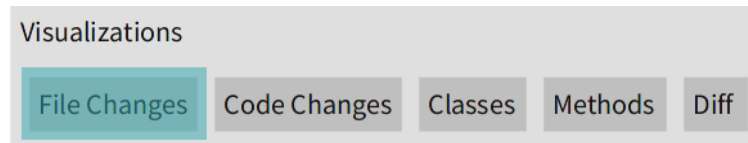
Code Changes

Classes

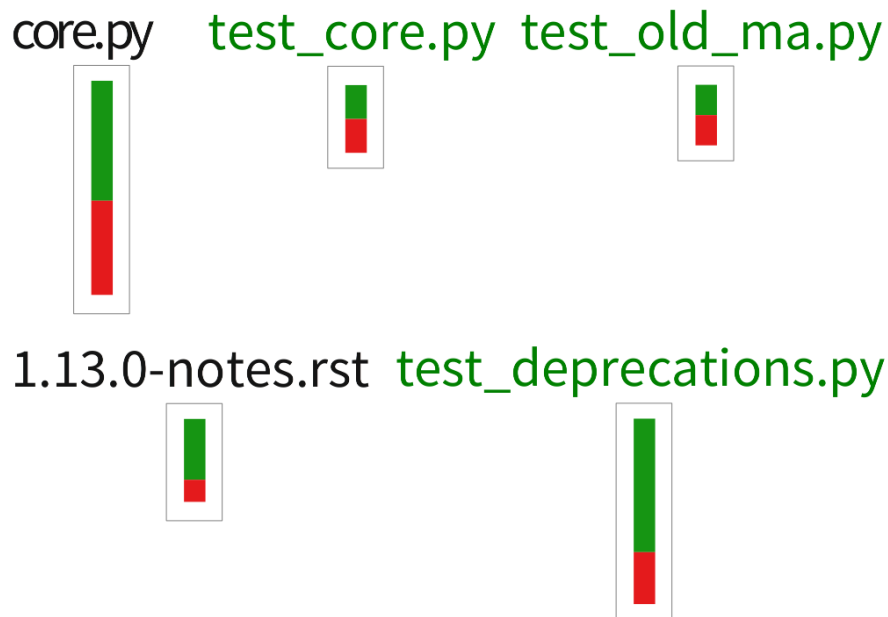
Methods

Diff

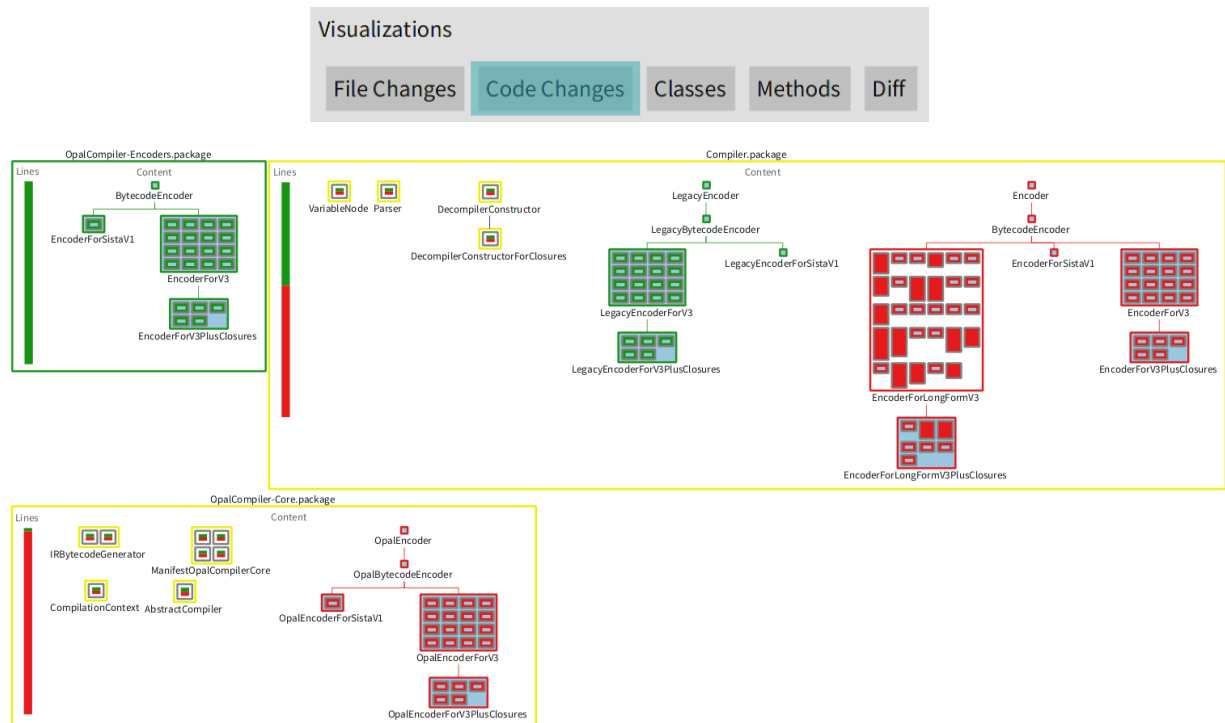
## File Changes



The file changes visualization displays changes made to text files in the pull request. In this visualization, each text file is represented by a box with an embedded chart that display the number of lines that are unchanged in gray, added in green and removed in red. If the file name contains the word 'test', then the filename is green to indicates the possibility of having unit test. By hovering the mouse, it is possible to raise a popup with a detailed description of the changes. Clicking on a file opens the text diff with the changes to the clicked file.



# Code Changes



This is the section with the main visualization. This visualization represents all of the structural changes that are done during a pull request or a commit. The structure of the code is represented by the use of embedding for representing the *contains a* relationships between packages, files, classes and methods. The type of change that was made to a container element (package/file/class) is represented by using the border color with the same legend that is used for the metrics in the pull request overview. In the special case of files (Python) and packages (Pharo), there is a chart with the number of lines added/removed/unchanged on the left side of the element.

Methods and function are represented in two different ways: a solid rectangle if the method was completely added/removed/moved or is unchanged; and as a rectangle with an embedded chart with metrics for the number of lines that are unchanged, added or deleted.

If the options for *Adding unchanged lines/methods/classes* are not selected when building the visualization, these elements are completely omitted from this visualization and all the other ones.

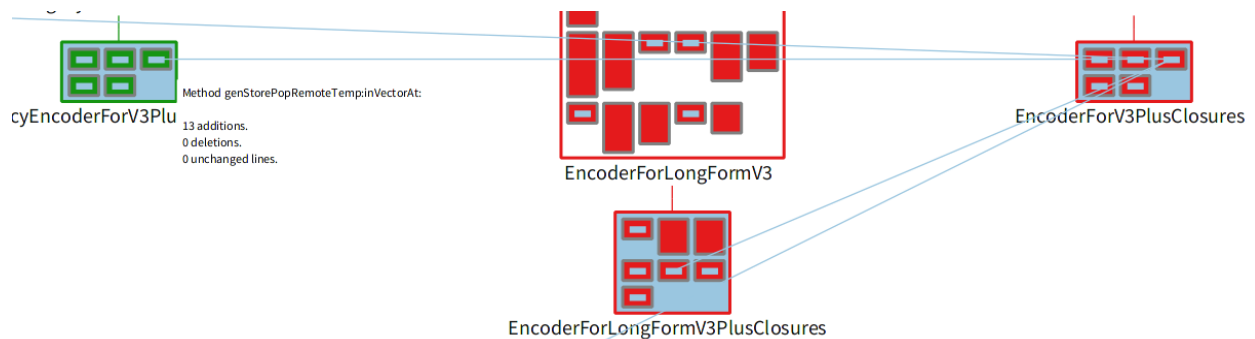
By highlighting an element with the mouse, it is possible to obtain a tooltip with a detailed description of the element, along with its metrics.

By clicking on an element it is possible to obtain the textual diff of the element. The textual diff of an element uses the green color for added lines, and the red color for lines removed.

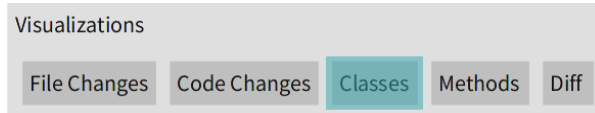
The screenshot shows a web interface with a file tree on the left and a diff view on the right. The file tree includes files like 'doc/release/1130-notes.rst', 'numpy/ma/tests/test\_old\_ma.py', 'numpy/ma/tests/test\_deprecations.py', 'numpy/ma/core.py', and 'numpy/ma/tests/test\_core.py'. The diff view shows the code for 'test\_testMinMax2' with green lines for additions and red lines for deletions.

```
def test_testMinMax2(self):
    # Test of minimum, maximum.
    assert_(eq(minimum([1, 2, 3], [4, 0, 9]), [1, 0, 3]))
    assert_(eq(maximum([1, 2, 3], [4, 0, 9]), [4, 2, 9]))
    x = arange(5)
    y = arange(5) - 2
    x[3] = masked
    y[0] = masked
    assert_(eq(minimum(x, y), where(less(x, y), x, y)))
    assert_(eq(maximum(x, y), where(greater(x, y), x, y)))
    assert_(minimum.reduce(x) == 0)
    assert_(maximum.reduce(x) == 4)
    assert_(minimum(x) == 0)
    assert_(maximum(x) == 4)
```

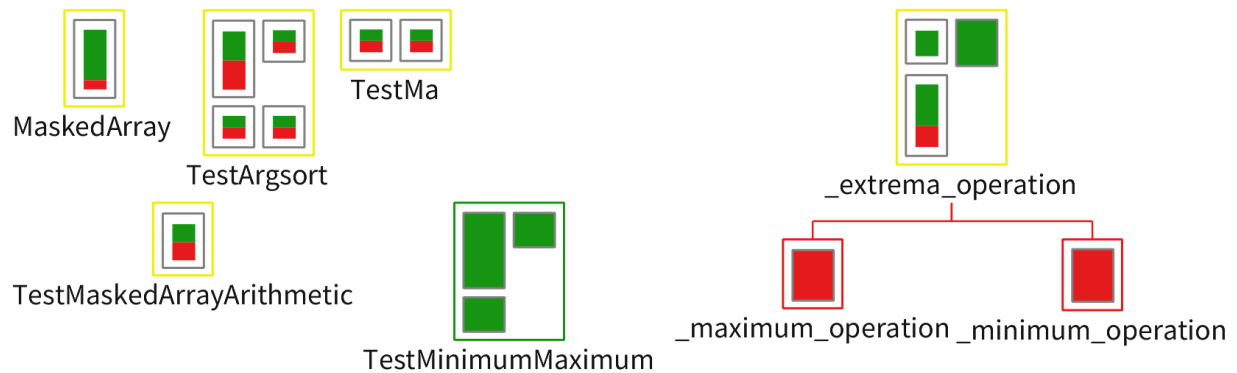
Moved and/or renamed elements are represented with a blue background. Edges connecting the similar elements are displayed when the mouse is highlighting an element that is moved and/or renamed.



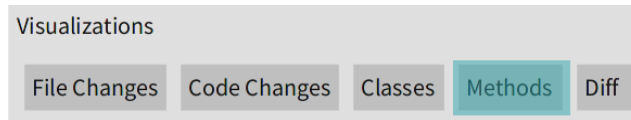
# Classes



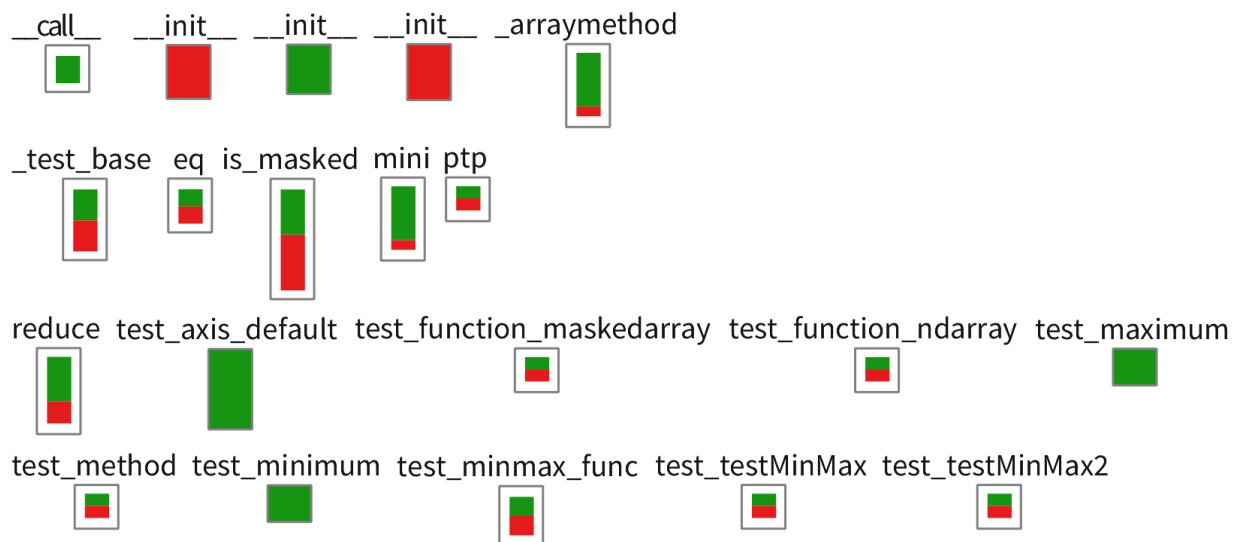
This contains a visualization of the source code changes with only the classes. This visualization does not show packages or the files, which can help on displaying the class hierarchies. By clicking on the bars present in the chart of number of classes, it is possible to obtain this same visualization for a subset of the classes (e.g: only added classes, or only the removed classes, etc)



# Methods



This contains a visualization with only the changed methods, without the classes and packages.



# Diff

## Visualizations

[File Changes](#)[Code Changes](#)[Classes](#)[Methods](#)[Diff](#)

This provides a global textual diff with for the whole changes in the pull request.

```
diff --git a/doc/release/1.13.0-notes.rst b/doc/release/1.13.0-notes.rst
index 2f32ddb28..3deca6a8c 100644
--- a/doc/release/1.13.0-notes.rst
+++ b/doc/release/1.13.0-notes.rst
@@ -28,10 +28,17 @@ Deprecations
* Use of the C-API ``NPY_CHAR`` type number deprecated since version 1.7 will
  now raise deprecation warnings at runtime. Extensions built with older f2py
  versions need to be recompiled to remove the warning.
- * ``np.ma.argsort`` should be called with an explicit ``axis`` argument when
-   applied to arrays with more than 2 dimensions, as the default value of
-   this argument (``None``) is inconsistent with the rest of numpy (``-1``).
+ * ``np.ma.argsort``, ``np.ma.minimum.reduce``, and ``np.ma.maximum.reduce``
+   should be called with an explicit ``axis`` argument when applied to arrays with
+   more than 2 dimensions, as the default value of this argument (``None``) is
+   inconsistent with the rest of numpy (``-1``, ``0``, and ``0``, respectively).
+ * ``np.ma.MaskedArray.mini`` is deprecated, as it almost duplicates the
+   functionality of ``np.MaskedArray.min``. Exactly equivalent behaviour
+   can be obtained with ``np.ma.minimum.reduce``.
+ * The single-argument form of ``np.ma.minimum`` and ``np.ma.maximum`` is
+   deprecated. ``np.ma.minimum(x)`` should now be spelt
+   ``np.ma.minimum.reduce(x)`` which is consistent with how this would be done
+   with ``np.minimum``.

Build System Changes
=====

diff --git a/numpy/ma/core.py b/numpy/ma/core.py
index 20cc77bc4..bccb0bce1 100644
--- a/numpy/ma/core.py
+++ b/numpy/ma/core.py
@@ -5621,6 +5621,14 @@ class MaskedArray(ndarray):
    """
    Return the array minimum along the specified axis.

+ .. deprecated:: 1.13.0
+    This function is identical to both:
+
+    * ``self.min(keepdims=True, axis=axis).squeeze(axis=axis)``
+    * ``np.ma.minimum.reduce(self, axis=axis)``
+
+    Typically though, ``self.min(axis=axis)`` is sufficient.
+
+

```