

# How to Use

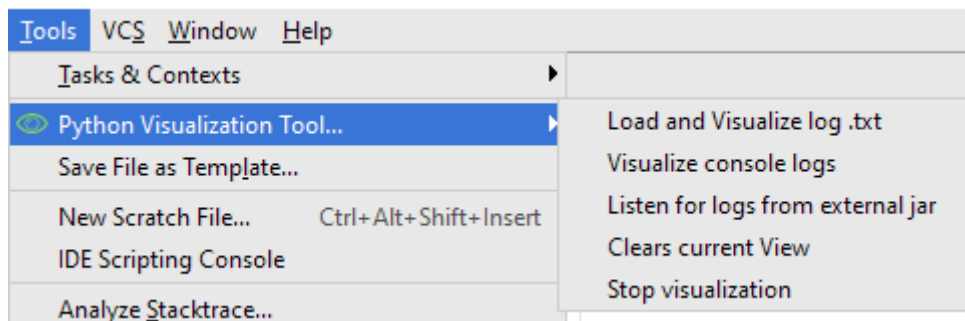
[Github link](#)

This plugin comes with two main features:

- Visualization of lines using stack trace from pytest or python runs
- Creating a dependency graph from the project

## Visualization of lines using stack trace

To get the options for this feature, just navigate to the tool window as shown below



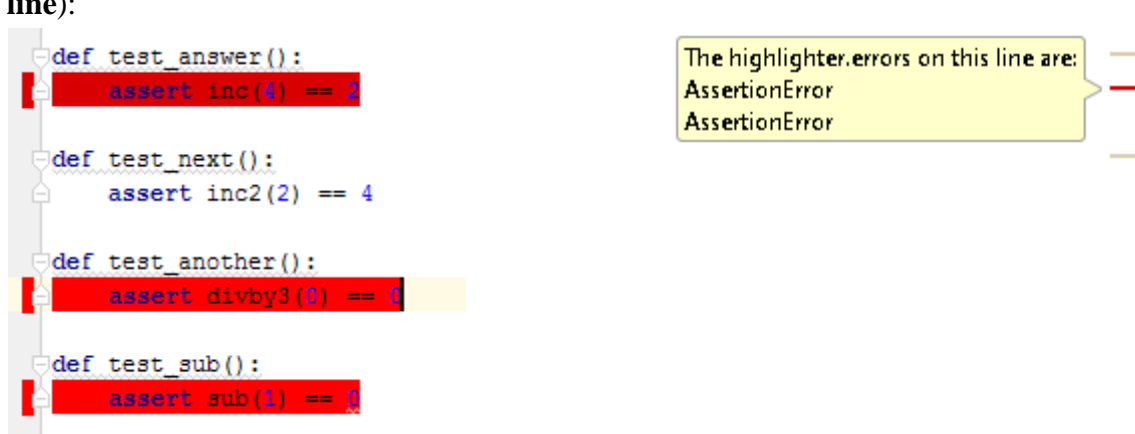
To get the stack trace, three options which **can** be combined are proposed:

1. Load a txt file with stack trace
2. Visualization of console logs from PyCharm
3. Listen on localhost, port **9876** for UDP packets of 1024-bit size -> this way you can redirect any stack trace from console to the PyCharm IDE for visualization

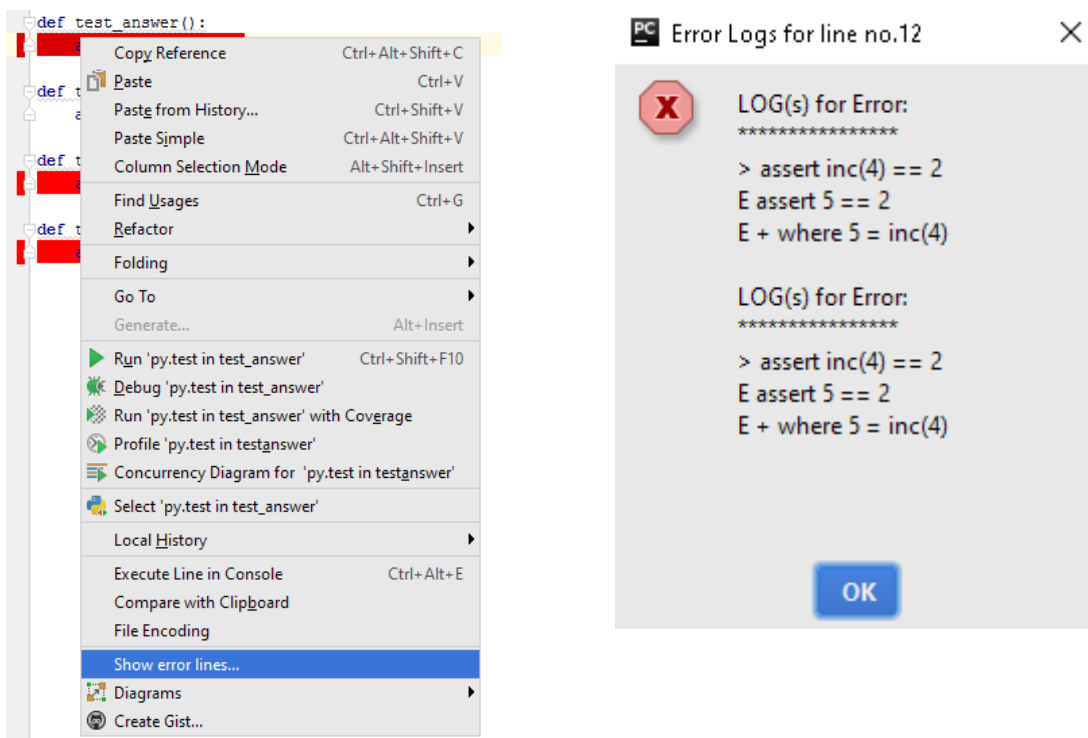
Other actions:

- **Clears current view** – Clears all highlighting done by the plugin for the current tab, after switching between tabs the visualization is redrawn
- **Stop visualization** – Clears all highlighting done by the plugin for all the tabs and clears all visualization history

Here is a screenshot of the visualization (The darker the colour -> the more errors on that line):



Right clicking on a specific line, this option appears (show error lines) which produces:



### Notes:

- There is a known bug that sometimes the first run of console doesn't visualize the lines. Then just repeat the same run and all should be fine from that point on.
- Visualization from Listen for logs and Console logs are done every time the user switches (or opens) editor tabs as the implementation is right now limited to this approach.

- The python and pytest error finding from stack trace is based on the default formatting of errors for both pytest and python exceptions.
- A feature to have line from now successfully passing test removed from highlighting is in the doing

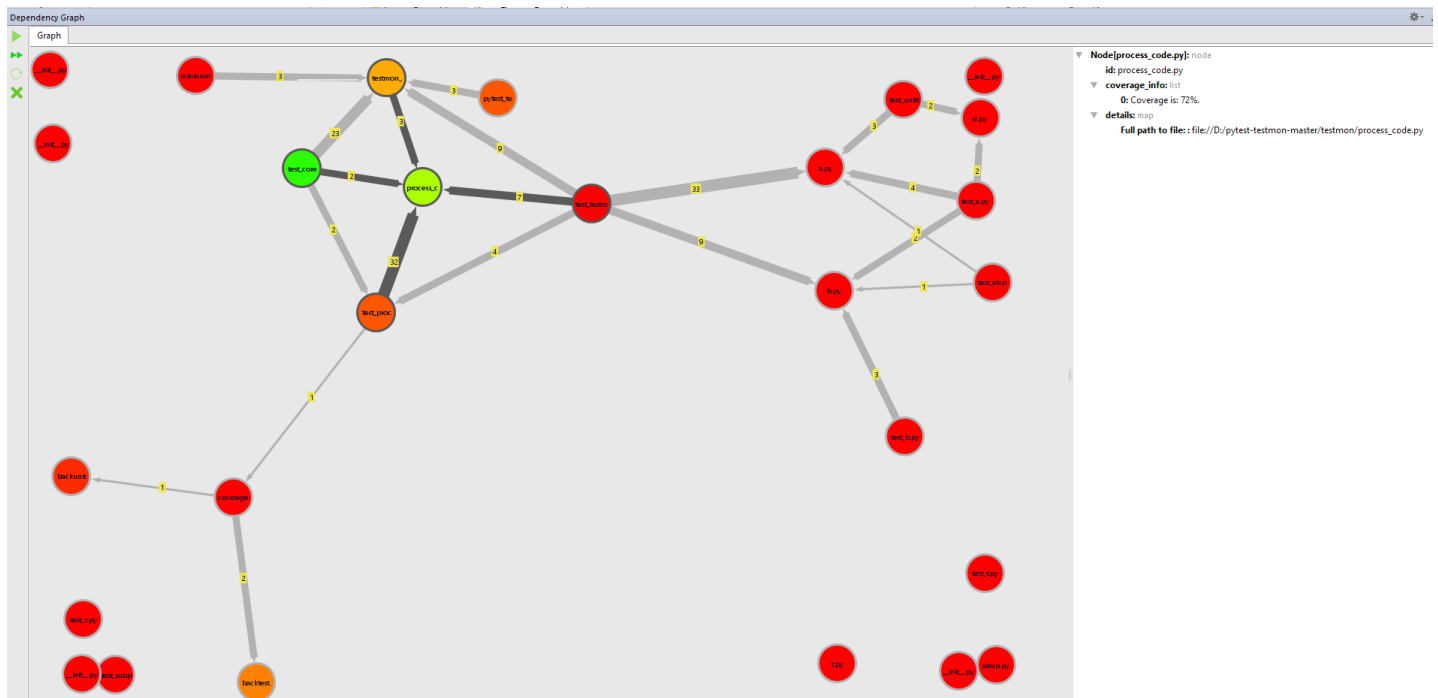
## Creating a dependency graph from project

In the view->Tool Windows the option of Dependency Graph is now available. This graph visualizes a dependency graph, where each node is a .py file in the project.


Each of these nodes is coloured differently using the coverage of the node (0%-red to 100%-green). The relations between the nodes represent the times a function/object from the imported node is used using static code analysis.


Clicking on a relation or node will show more info about them.


Example done using the [Testmon](#) project with one test run with coverage:




The layout used is the [Fruchterman-reingold](#) layout for dynamically changing graphs.


**Visualize the entire dependency graph**


**Visualize only covered nodes**


**Reset focus**

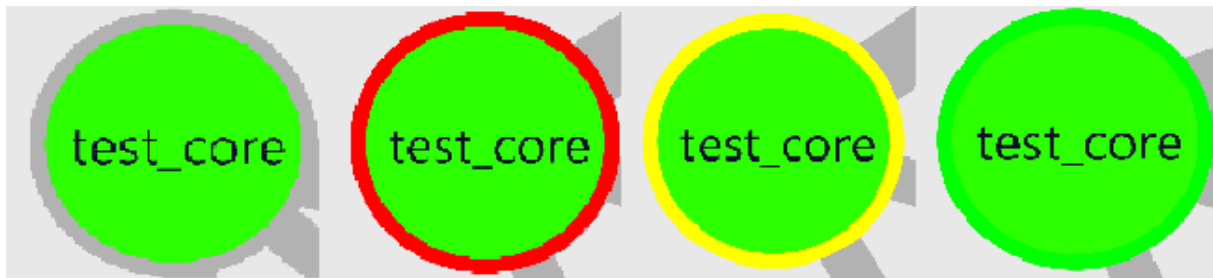

**Clean canvas**

The visualization is automatically triggered by each test run (after test suite is finished, the visualization is done new) if the window was opened before (does not apply if the window wasn't opened in the current session).

The graph keeps also a history of test runs. The first time a history is kept, the nodes have **dark grey** edges.

Now, if a test in a certain node fails from the last test run, the edge becomes **red**. If a test succeeds and it was failing before, the edge of the node, where the test belongs, is **green**.

If both happen at the same time at a certain node, the node will have **yellow** edges.



How the nodes look like in different stages.

## Notes

- For best results use the graph with the highlighter whilst running tests, so both features are used
- This plugin is a work for my bachelor thesis Advanced testing during software development
- Please report any bugs, feature proposals and all the general stuff here at: [jan.cegin@gmail.com](mailto:jan.cegin@gmail.com)