razortrace Documentation

Release 0.1

Kevin (eales)

Feb 28, 2022

CONTENTS:

CHAPTER

ONE

LINKS

https://github.com/manbehindthemadness/razortrace

https://pypi.org/project/razortrace

CHAPTER

TWO

DESCRIPTION

Razortrace is a memory diagnostic tool based on the `tracemalloc` library. It's aim is to provide rapid identification of memory leaks and produce straightforward, human-readable reports.

CHAPTER

THREE

REQUIREMENTS

The main library requires no additional packages; however, pytest and PIL are needed to run tests.

CHAPTER

FOUR

USAGE

Razortrace can be used as a decorator *(recommended)* or alternatively as an imported class *(useful for more specific scenarios)*. Leak detection is achieved by starting tracemalloc and capturing a memory snapshot. Once arbitrary code has completed execution a second snapshot is taken and compared against the first. Results are filtered based on two sets of criteria:

- Execution has been increasing in memory usage throughout the sampling process.
- Execution has not reclaimed any memory throughout the sampling process *(configurable)*.

When used as a decorator, each probe is activates by a trigger in the form of an environment variable. This allows cherry-picking of many selectively placed tests throughout a project with minimal alteration of the business logic.

NOTES:

- Only detections originating from within the working directory are returned, if a dependency or extraneous file needs to be inspected, tracemalloc will likely be required: https://docs.python.org/3/library/tracemalloc.html

Parameters

- **`trigger` An environment variable that enables the trace based on its truth-state**
  - default - `str()`
- **`traceback` Specifies the inclusion of tracebacks in the final report**
  - default - `False`
- **`clear` Specifies the memory trace will be cleared after each execution**
  - default - `False`
- **`strict` Shows only executions that have not reclaimed memory during the sampling process**
  - default - `True`
- **`debug` Specifies the final report will include the recorded memory samples in addition to allowing trace items from within**
  - default - `False`
- **`here` The current working directory *(only required when initializing the probe as a class)***
  - default - root installation directory

CHAPTER

FIVE

EXAMPLES

NOTES:

- These examples can be found in the tests folder
- Examples will report improperly when executed from a virtual python console.
- If two traces are active simultaneously it is likely they will capture each other and provide undesired results.

```
import os
from pathlib import Path
from razortrace import probe


HOLD = list()


@probe(trigger='TRACE_TEXT', traceback=True, clear=True, debug=True)
def text():
    """
    Creates a memory leak from a text file.
    """
    global HOLD
    with open(Path('tests/text.txt'), "r") as txt:  # <--- Leak
        for line in txt.readlines():  # <--- Leak
            HOLD.append(line)
    for cycle in range(0, 1000):
        HOLD.append([cycle, HOLD])
    return


def test_text():
    """
    Fires off the above logic.
    """
    os.environ["TRACE_TEXT"] = "1"  # Enable trace.
    txt = text()
    txt.trace.reset()  # Clear memory tracer and samples.
```

```
>>> \razortrace\tests\test_mem.py line: 76 command: for line in txt.readlines():  # <---␣
↪Leak  average: 12.007161458333334 total kb 35.904296875

>>>                                                -----trace-----
```

(continues on next page)

(continued from previous page)

```
>>> \my-project-dir\lib\site-packages\pluggy\_callers.py:39
>>> \my-project-dir\lib\site-packages\_pytest\runner.py:168
>>> \my-project-dir\lib\site-packages\_pytest\python.py:1718
>>> \my-project-dir\lib\site-packages\pluggy\_hooks.py:265
>>> \my-project-dir\lib\site-packages\pluggy\_manager.py:80
>>> \my-project-dir\lib\site-packages\pluggy\_callers.py:39
>>> \my-project-dir\lib\site-packages\_pytest\python.py:192
>>> \razortrace\tests\test_mem.py:88
>>> \razortrace\razortrace\main.py:273
>>> \razortrace\tests\test_mem.py:76


>>>                                                  -----sizes-----

>>> 0.0546875 0.0625 35.904296875


>>> ------------------------------------------------------------------------------

>>> \razortrace\tests\test_mem.py line: 75 command: with open(Path(HERE + '/text.txt'),
↪"r") as txt:  # <--- Leak average: 0.12027994791666667 total kb 0.2763671875

>>>                                                  -----trace-----

>>> \my-project-dir\lib\site-packages\pluggy\_callers.py:39
>>> \my-project-dir\lib\site-packages\_pytest\runner.py:168
>>> \my-project-dir\lib\site-packages\_pytest\python.py:1718
>>> \my-project-dir\lib\site-packages\pluggy\_hooks.py:265
>>> \my-project-dir\lib\site-packages\pluggy\_manager.py:80
>>> \my-project-dir\lib\site-packages\pluggy\_callers.py:39
>>> \my-project-dir\lib\site-packages\_pytest\python.py:192
>>> \razortrace\tests\test_mem.py:88
>>> \razortrace\razortrace\main.py:273
>>> \razortrace\tests\test_mem.py:75


>>>                                                  -----sizes-----

>>> 0.046875 0.046875 0.0625 0.0625 0.2265625 0.2763671875


>>> ------------------------------------------------------------------------------


>>> ==============================================================================
```

CHAPTER

SIX

INSTALLATION

razortrace can be installed using pip:

```
pip install razortrace
```

or alternatively:

```
git clone https://github.com/manbehindthemadness/razortrace.git
cd razortrace
python setup.py install
```

CHAPTER

SEVEN

DISCLAIMER

This library is still in development, please use at your own risk and test sufficiently before using it in a production environment.

CHAPTER

EIGHT

INDICES AND TABLES

- genindex
- modindex
- search