The Python Profilers

Source code: :source:`Lib/profile.py` and :source:`Lib/pstats.py`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 7); *backlink*

Unknown interpreted text role "source".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 7); *backlink*

Unknown interpreted text role "source".

Introduction to the profilers

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 16)

Unknown directive type "index".

```
.. index::
   single: deterministic profiling
   single: profiling, deterministic
```

:mod:`cProfile` and :mod:`profile` provide :dfn:`deterministic profiling` of Python programs. A :dfn:`profile` is a set of statistics that describes how often and for how long various parts of the program executed. These statistics can be formatted into reports via the :mod:`pstats` module.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 20); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 20); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 20); *backlink*

Unknown interpreted text role "dfn".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 20); *backlink*

Unknown interpreted text role "dfn".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 20); *backlink*

Unknown interpreted text role "mod".

The Python standard library provides two different implementations of the same profiling interface:

1. :mod:`cProfile` is recommended for most users; it's a C extension with reasonable overhead that makes it suitable for profiling long-running programs. Based on :mod:`lsprof`, contributed by Brett Rosen and Ted Czotter.

   System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 28); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 28); *backlink*

Unknown interpreted text role "mod".

2. :mod:`profile`, a pure Python module whose interface is imitated by :mod:`cProfile`, but which adds significant overhead to profiled programs. If you're trying to extend the profiler in some way, the task might be easier with this module. Originally designed and written by Jim Roskind.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 33); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 33); *backlink*

Unknown interpreted text role "mod".

Note

The profiler modules are designed to provide an execution profile for a given program, not for benchmarking purposes (for that, there is :mod:`timeit` for reasonably accurate results). This particularly applies to benchmarking Python code against C code: the profilers introduce overhead for Python code, but not for C-level functions, and so the C code would seem faster than any Python one.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 40); *backlink*

Unknown interpreted text role "mod".

Instant User's Manual

This section is provided for users that "don't want to read the manual." It provides a very brief overview, and allows a user to rapidly perform profiling on an existing application.

To profile a function that takes a single argument, you can do:

```
import cProfile
import re
cProfile.run('re.compile("foo|bar")')
```

(Use :mod:`profile` instead of :mod:`cProfile` if the latter is not available on your system.)

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 63); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 63); *backlink*

Unknown interpreted text role "mod".

The above action would run :func:`re.compile` and print profile results like the following:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 66); *backlink*

Unknown interpreted text role "func".

```
         214 function calls (207 primitive calls) in 0.002 seconds

   Ordered by: cumulative time

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    0.000    0.000    0.002    0.002 {built-in method builtins.exec}
        1    0.000    0.000    0.001    0.001 <string>:1(<module>)
        1    0.000    0.000    0.001    0.001 __init__.py:250(compile)
        1    0.000    0.000    0.001    0.001 __init__.py:289(_compile)
        1    0.000    0.000    0.000    0.000 _compiler.py:759(compile)
        1    0.000    0.000    0.000    0.000 _parser.py:937(parse)
        1    0.000    0.000    0.000    0.000 _compiler.py:598(_code)
        1    0.000    0.000    0.000    0.000 _parser.py:435(_parse_sub)
```

The first line indicates that 214 calls were monitored. Of those calls, 207 were :dfn:`primitive`, meaning that the call was not induced via recursion. The next line: `Ordered by: cumulative name`, indicates that the text string in the far right column was used to sort the output. The column headings include:

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 83); *backlink***
>
> Unknown interpreted text role "dfn".

ncalls
: for the number of calls.

tottime
: for the total time spent in the given function (and excluding time made in calls to sub-functions)

percall
: is the quotient of `tottime` divided by `ncalls`

cumtime
: is the cumulative time spent in this and all subfunctions (from invocation till exit). This figure is accurate *even* for recursive functions.

percall
: is the quotient of `cumtime` divided by primitive calls

filename:lineno(function)
: provides the respective data of each function

When there are two numbers in the first column (for example `3/1`), it means that the function recursed. The second value is the number of primitive calls and the former is the total number of calls. Note that when the function does not recurse, these two values are the same, and only the single figure is printed.

Instead of printing the output at the end of the profile run, you can save the results to a file by specifying a filename to the :func:`run` function:

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 114); *backlink***
>
> Unknown interpreted text role "func".

```
import cProfile
import re
cProfile.run('re.compile("foo|bar")', 'restats')
```

The :class:`pstats.Stats` class reads profile results from a file and formats them in various ways.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 121); *backlink***
>
> Unknown interpreted text role "class".

The files :mod:`cProfile` and :mod:`profile` can also be invoked as a script to profile another script. For example:

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 124); *backlink***
>
> Unknown interpreted text role "mod".

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 124); *backlink***

Unknown interpreted text role "mod".

```
python -m cProfile [-o output_file] [-s sort_order] (-m module | myscript.py)
```

`-o` writes the profile results to a file instead of to stdout

`-s` specifies one of the :func:`~pstats.Stats.sort_stats` sort values to sort the output by. This only applies when `-o` is not supplied.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 131); *backlink*

Unknown interpreted text role "func".

`-m` specifies that a module is being profiled instead of a script.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 136)

Unknown directive type "versionadded".

```
.. versionadded:: 3.7
   Added the ``-m`` option to :mod:`cProfile`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 139)

Unknown directive type "versionadded".

```
.. versionadded:: 3.8
   Added the ``-m`` option to :mod:`profile`.
```

The :mod:`pstats` module's :class:`~pstats.Stats` class has a variety of methods for manipulating and printing the data saved into a profile results file:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 142); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 142); *backlink*

Unknown interpreted text role "class".

```
import pstats
from pstats import SortKey
p = pstats.Stats('restats')
p.strip_dirs().sort_stats(-1).print_stats()
```

The :meth:`~pstats.Stats.strip_dirs` method removed the extraneous path from all the module names. The :meth:`~pstats.Stats.sort_stats` method sorted all the entries according to the standard module/line/name string that is printed. The :meth:`~pstats.Stats.print_stats` method printed out all the statistics. You might try the following sort calls:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 150); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 150); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 150); *backlink*

Unknown interpreted text role "meth".

```
p.sort_stats(SortKey.NAME)
p.print_stats()
```

The first call will actually sort the list by function name, and the second call will print out the statistics. The following are some interesting calls to experiment with:

```
p.sort_stats(SortKey.CUMULATIVE).print_stats(10)
```

This sorts the profile by cumulative time in a function, and then only prints the ten most significant lines. If you want to understand what algorithms are taking time, the above line is what you would use.

If you were looking to see what functions were looping a lot, and taking a lot of time, you would do:

```
p.sort_stats(SortKey.TIME).print_stats(10)
```

to sort according to time spent within each function, and then print the statistics for the top ten functions.

You might also try:

```
p.sort_stats(SortKey.FILENAME).print_stats('__init__')
```

This will sort all the statistics by file name, and then print out statistics for only the class init methods (since they are spelled with `__init__` in them). As one final example, you could try:

```
p.sort_stats(SortKey.TIME, SortKey.CUMULATIVE).print_stats(.5, 'init')
```

This line sorts statistics with a primary key of time, and a secondary key of cumulative time, and then prints out some of the statistics. To be specific, the list is first culled down to 50% (re: `.5`) of its original size, then only lines containing `init` are maintained, and that sub-sub-list is printed.

If you wondered what functions called the above functions, you could now (`p` is still sorted according to the last criteria) do:

```
p.print_callers(.5, 'init')
```

and you would get a list of callers for each of the listed functions.

If you want more functionality, you're going to have to read the manual, or guess what the following functions do:

```
p.print_callees()
p.add('restats')
```

Invoked as a script, the :mod:`pstats` module is a statistics browser for reading and examining profile dumps. It has a simple line-oriented interface (implemented using :mod:`cmd`) and interactive help.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 205); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 205); *backlink*

Unknown interpreted text role "mod".

:mod:`profile` and :mod:`cProfile` Module Reference

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 209); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 209); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 212)

Unknown directive type "module".

```
.. module:: cProfile
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 213)

Unknown directive type "module".

```
.. module:: profile
   :synopsis: Python source profiler.
```

Both the :mod:`profile` and :mod:`cProfile` modules provide the following functions:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 216); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 216); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 219)

Unknown directive type "function".

```
.. function:: run(command, filename=None, sort=-1)

   This function takes a single argument that can be passed to the :func:`exec`
   function, and an optional file name.  In all cases this routine executes::

      exec(command, __main__.__dict__, __main__.__dict__)

   and gathers profiling statistics from the execution. If no file name is
   present, then this function automatically creates a :class:`~pstats.Stats`
   instance and prints a simple profiling report. If the sort value is specified,
   it is passed to this :class:`~pstats.Stats` instance to control how the
   results are sorted.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 232)

Unknown directive type "function".

```
.. function:: runctx(command, globals, locals, filename=None, sort=-1)

   This function is similar to :func:`run`, with added arguments to supply the
   globals and locals dictionaries for the *command* string. This routine
   executes::

      exec(command, globals, locals)

   and gathers profiling statistics as in the :func:`run` function above.
```

This class is normally only used if more precise control over profiling is needed than what the :func:`cProfile.run` function provides.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 244); *backlink*

Unknown interpreted text role "func".

A custom timer can be supplied for measuring how long code takes to run via the *timer* argument. This must be a function that returns a single number representing the current time. If the number is an integer, the *timeunit* specifies a multiplier that specifies the duration of each unit of time. For example, if the timer returns times measured in thousands of seconds, the time unit would be `.001`.

Directly using the :class:`Profile` class allows formatting profile results without writing the profile data to a file:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 254); *backlink*

Unknown interpreted text role "class".

```
import cProfile, pstats, io
from pstats import SortKey
pr = cProfile.Profile()
pr.enable()
# ... do something ...
pr.disable()
s = io.StringIO()
sortby = SortKey.CUMULATIVE
ps = pstats.Stats(pr, stream=s).sort_stats(sortby)
ps.print_stats()
print(s.getvalue())
```

The :class:`Profile` class can also be used as a context manager (supported only in :mod:`cProfile` module. see :ref:`typecontextmanager`):

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 269); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 269); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 269); *backlink*

Unknown interpreted text role "ref".

```
import cProfile

with cProfile.Profile() as pr:
    # ... do something ...

pr.print_stats()
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 279)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.8
   Added context manager support.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 282)

Unknown directive type "method".

```
.. method:: enable()

   Start collecting profiling data. Only in :mod:`cProfile`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 286)

Unknown directive type "method".

```
.. method:: disable()

   Stop collecting profiling data. Only in :mod:`cProfile`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 290)

Unknown directive type "method".

```
.. method:: create_stats()

   Stop collecting profiling data and record the results internally
   as the current profile.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 295)

Unknown directive type "method".

```
.. method:: print_stats(sort=-1)

   Create a :class:`~pstats.Stats` object based on the current
   profile and print the results to stdout.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 300)

Unknown directive type "method".

```
.. method:: dump_stats(filename)

   Write the results of the current profile to *filename*.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 304)

Unknown directive type "method".

```
.. method:: run(cmd)

   Profile the cmd via :func:`exec`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 308)

Unknown directive type "method".

```
.. method:: runctx(cmd, globals, locals)

   Profile the cmd via :func:`exec` with the specified global and
   local environment.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 313)

Unknown directive type "method".

```
.. method:: runcall(func, /, *args, **kwargs)

   Profile ``func(*args, **kwargs)``
```

Note that profiling will only work if the called command/function actually returns. If the interpreter is terminated (e.g. via a :func:`sys.exit` call during the called command/function execution) no profiling results will be printed.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 317); *backlink*

Unknown interpreted text role "func".

The :class:`Stats` Class

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 324); *backlink*

Unknown interpreted text role "class".

Analysis of the profiler data is done using the :class:`~pstats.Stats` class.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 327); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 329)

Unknown directive type "module".

```
.. module:: pstats
   :synopsis: Statistics object for use with the profiler.
```

This class constructor creates an instance of a "statistics object" from a *filename* (or list of filenames) or from a :class:`Profile` instance. Output will be printed to the stream specified by *stream*.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 334); *backlink*

Unknown interpreted text role "class".

The file selected by the above constructor must have been created by the corresponding version of :mod:`profile` or :mod:`cProfile`. To be specific, there is *no* file compatibility guaranteed with future versions of this profiler, and there is no compatibility with files produced by other profilers, or the same profiler run on a different operating system. If several files are provided, all the statistics for identical functions will be coalesced, so that an overall view of several processes can be considered in a single report. If additional files need to be combined with data in an existing :class:`~pstats.Stats` object, the :meth:`~pstats.Stats.add` method can be used.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 338); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 338); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 338); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 338); *backlink*

Unknown interpreted text role "meth".

Instead of reading the profile data from a file, a :class:`cProfile.Profile` or :class:`profile.Profile` object can be used as the profile data source.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 349); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 349); *backlink*

Unknown interpreted text role "class".

:class:`Stats` objects have the following methods:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 352); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 354)

Unknown directive type "method".

```
.. method:: strip_dirs()

   This method for the :class:`Stats` class removes all leading path
   information from file names.  It is very useful in reducing the size of
   the printout to fit within (close to) 80 columns.  This method modifies
   the object, and the stripped information is lost.  After performing a
   strip operation, the object is considered to have its entries in a
   "random" order, as it was just after object initialization and loading.
   If :meth:`~pstats.Stats.strip_dirs` causes two function names to be
   indistinguishable (they are on the same line of the same filename, and
   have the same function name), then the statistics for these two entries
   are accumulated into a single entry.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 368)

Unknown directive type "method".

```
.. method:: add(*filenames)

   This method of the :class:`Stats` class accumulates additional profiling
   information into the current profiling object.  Its arguments should refer
   to filenames created by the corresponding version of :func:`profile.run`
   or :func:`cProfile.run`. Statistics for identically named (re: file, line,
   name) functions are automatically accumulated into single function
   statistics.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 378)

Unknown directive type "method".

```
.. method:: dump_stats(filename)

   Save the data loaded into the :class:`Stats` object to a file named
   *filename*.  The file is created if it does not exist, and is overwritten
   if it already exists.  This is equivalent to the method of the same name
   on the :class:`profile.Profile` and :class:`cProfile.Profile` classes.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 386)

Unknown directive type "method".

```
.. method:: sort_stats(*keys)

   This method modifies the :class:`Stats` object by sorting it according to
   the supplied criteria.  The argument can be either a string or a SortKey
   enum identifying the basis of a sort (example: ``'time'``, ``'name'``,
   ``SortKey.TIME`` or ``SortKey.NAME``). The SortKey enums argument have
   advantage over the string argument in that it is more robust and less
   error prone.

   When more than one key is provided, then additional keys are used as
   secondary criteria when there is equality in all keys selected before
```

```
them.  For example, ``sort_stats(SortKey.NAME, SortKey.FILE)`` will sort
all the entries according to their function name, and resolve all ties
(identical function names) by sorting by file name.

For the string argument, abbreviations can be used for any key names, as
long as the abbreviation is unambiguous.

The following are the valid string and SortKey:

+------------------+---------------------+----------------------+
| Valid String Arg | Valid enum Arg      | Meaning              |
+==================+=====================+======================+
| ``'calls'``      | SortKey.CALLS       | call count           |
+------------------+---------------------+----------------------+
| ``'cumulative'`` | SortKey.CUMULATIVE  | cumulative time      |
+------------------+---------------------+----------------------+
| ``'cumtime'``    | N/A                 | cumulative time      |
+------------------+---------------------+----------------------+
| ``'file'``       | N/A                 | file name            |
+------------------+---------------------+----------------------+
| ``'filename'``   | SortKey.FILENAME    | file name            |
+------------------+---------------------+----------------------+
| ``'module'``     | N/A                 | file name            |
+------------------+---------------------+----------------------+
| ``'ncalls'``     | N/A                 | call count           |
+------------------+---------------------+----------------------+
| ``'pcalls'``     | SortKey.PCALLS      | primitive call count |
+------------------+---------------------+----------------------+
| ``'line'``       | SortKey.LINE        | line number          |
+------------------+---------------------+----------------------+
| ``'name'``       | SortKey.NAME        | function name        |
+------------------+---------------------+----------------------+
| ``'nfl'``        | SortKey.NFL         | name/file/line       |
+------------------+---------------------+----------------------+
| ``'stdname'``    | SortKey.STDNAME     | standard name        |
+------------------+---------------------+----------------------+
| ``'time'``       | SortKey.TIME        | internal time        |
+------------------+---------------------+----------------------+
| ``'tottime'``    | N/A                 | internal time        |
+------------------+---------------------+----------------------+

Note that all sorts on statistics are in descending order (placing most
time consuming items first), where as name, file, and line number searches
are in ascending order (alphabetical). The subtle distinction between
``SortKey.NFL`` and ``SortKey.STDNAME`` is that the standard name is a
sort of the name as printed, which means that the embedded line numbers
get compared in an odd way.  For example, lines 3, 20, and 40 would (if
the file names were the same) appear in the string order 20, 3 and 40.
In contrast, ``SortKey.NFL`` does a numeric compare of the line numbers.
In fact, ``sort_stats(SortKey.NFL)`` is the same as
``sort_stats(SortKey.NAME, SortKey.FILENAME, SortKey.LINE)``.

For backward-compatibility reasons, the numeric arguments ``-1``, ``0``,
``1``, and ``2`` are permitted.  They are interpreted as ``'stdname'``,
``'calls'``, ``'time'``, and ``'cumulative'`` respectively.  If this old
style format (numeric) is used, only one sort key (the numeric key) will
be used, and additional arguments will be silently ignored.

.. For compatibility with the old profiler.

.. versionadded:: 3.7
   Added the SortKey enum.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 460)

Unknown directive type "method".

```
.. method:: reverse_order()

   This method for the :class:`Stats` class reverses the ordering of the
   basic list within the object.  Note that by default ascending vs
   descending order is properly selected based on the sort key of choice.

   .. This method is provided primarily for compatibility with the old
      profiler.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 470)

Unknown directive type "method".

```
.. method:: print_stats(*restrictions)

   This method for the :class:`Stats` class prints out a report as described
   in the :func:`profile.run` definition.

   The order of the printing is based on the last
   :meth:`~pstats.Stats.sort_stats` operation done on the object (subject to
   caveats in :meth:`~pstats.Stats.add` and
   :meth:`~pstats.Stats.strip_dirs`).

   The arguments provided (if any) can be used to limit the list down to the
   significant entries.  Initially, the list is taken to be the complete set
   of profiled functions.  Each restriction is either an integer (to select a
   count of lines), or a decimal fraction between 0.0 and 1.0 inclusive (to
   select a percentage of lines), or a string that will interpreted as a
   regular expression (to pattern match the standard name that is printed).
   If several restrictions are provided, then they are applied sequentially.
   For example::

      print_stats(.1, 'foo:')

   would first limit the printing to first 10% of list, and then only print
   functions that were part of filename :file:`.\*foo:`.  In contrast, the
   command::

      print_stats('foo:', .1)

   would limit the list to all functions having file names :file:`.\*foo:`,
   and then proceed to only print the first 10% of them.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 501)

Unknown directive type "method".

```
.. method:: print_callers(*restrictions)

   This method for the :class:`Stats` class prints a list of all functions
   that called each function in the profiled database.  The ordering is
   identical to that provided by :meth:`~pstats.Stats.print_stats`, and the
   definition of the restricting argument is also identical.  Each caller is
   reported on its own line.  The format differs slightly depending on the
   profiler that produced the stats:

   * With :mod:`profile`, a number is shown in parentheses after each caller
     to show how many times this specific call was made.  For convenience, a
     second non-parenthesized number repeats the cumulative time spent in the
     function at the right.

   * With :mod:`cProfile`, each caller is preceded by three numbers: the
     number of times this specific call was made, and the total and
     cumulative times spent in the current function while it was invoked by
     this specific caller.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 521)

Unknown directive type "method".

```
.. method:: print_callees(*restrictions)

   This method for the :class:`Stats` class prints a list of all function
   that were called by the indicated function.  Aside from this reversal of
   direction of calls (re: called vs was called by), the arguments and
   ordering are identical to the :meth:`~pstats.Stats.print_callers` method.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 529)

Unknown directive type "method".

```
.. method:: get_stats_profile()

   This method returns an instance of StatsProfile, which contains a mapping
   of function names to instances of FunctionProfile. Each FunctionProfile
   instance holds information related to the function's profile such as how
   long the function took to run, how many times it was called, etc...

   .. versionadded:: 3.9
      Added the following dataclasses: StatsProfile, FunctionProfile.
      Added the following function: get_stats_profile.
```

What Is Deterministic Profiling?

:dfn:`Deterministic profiling` is meant to reflect the fact that all *function call*, *function return*, and *exception* events are monitored, and precise timings are made for the intervals between these events (during which time the user's code is executing). In contrast, :dfn:`statistical profiling` (which is not done by this module) randomly samples the effective instruction pointer, and deduces where time is being spent. The latter technique traditionally involves less overhead (as the code does not need to be instrumented), but provides only relative indications of where time is being spent.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 545); *backlink*

Unknown interpreted text role "dfn".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 545); *backlink*

Unknown interpreted text role "dfn".

In Python, since there is an interpreter active during execution, the presence of instrumented code is not required in order to do deterministic profiling. Python automatically provides a :dfn:`hook` (optional callback) for each event. In addition, the interpreted nature of Python tends to add so much overhead to execution, that deterministic profiling tends to only add small processing overhead in typical applications. The result is that deterministic profiling is not that expensive, yet provides extensive run time statistics about the execution of a Python program.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 554); *backlink*

Unknown interpreted text role "dfn".

Call count statistics can be used to identify bugs in code (surprising counts), and to identify possible inline-expansion points (high call counts). Internal time statistics can be used to identify "hot loops" that should be carefully optimized. Cumulative time statistics should be used to identify high level errors in the selection of algorithms. Note that the unusual handling of cumulative times in this profiler allows statistics for recursive implementations of algorithms to be directly compared to iterative implementations.

Limitations

One limitation has to do with accuracy of timing information. There is a fundamental problem with deterministic profilers involving accuracy. The most obvious restriction is that the underlying "clock" is only ticking at a rate (typically) of about .001 seconds. Hence no measurements will be more accurate than the underlying clock. If enough measurements are taken, then the "error" will tend to average out. Unfortunately, removing this first error induces a second source of error.

The second problem is that it "takes a while" from when an event is dispatched until the profiler's call to get the time actually *gets* the state of the clock. Similarly, there is a certain lag when exiting the profiler event handler from the time that the clock's value was obtained (and then squirreled away), until the user's code is once again executing. As a result, functions that are called many times, or call many functions, will typically accumulate this error. The error that accumulates in this fashion is typically less than the accuracy of the clock (less than one clock tick), but it *can* accumulate and become very significant.

The problem is more important with :mod:`profile` than with the lower-overhead :mod:`cProfile`. For this reason, :mod:`profile` provides a means of calibrating itself for a given platform so that this error can be probabilistically (on the average) removed. After the profiler is calibrated, it will be more accurate (in a least square sense), but it will sometimes produce negative numbers (when call counts are exceptionally low, and the gods of probability work against you :-).) Do *not* be alarmed by negative numbers in the profile. They should *only* appear if you have calibrated your profiler, and the results are actually better than without calibration.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 596); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 596); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 596); *backlink*

Unknown interpreted text role "mod".

Calibration

The profiler of the :mod:`profile` module subtracts a constant from each event handling time to compensate for the overhead of calling the time function, and socking away the results. By default, the constant is 0. The following procedure can be used to obtain a better constant for a given platform (see :ref:`profile-limitations`).

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 612); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 612); *backlink*

Unknown interpreted text role "ref".

```
import profile
pr = profile.Profile()
for i in range(5):
    print(pr.calibrate(10000))
```

The method executes the number of Python calls given by the argument, directly and again under the profiler, measuring the time for both. It then computes the hidden overhead per profiler event, and returns that as a float. For example, on a 1.8Ghz Intel Core i5 running macOS, and using Python's time.process_time() as the timer, the magical number is about 4.04e-6.

The object of this exercise is to get a fairly consistent result. If your computer is *very* fast, or your timer function has poor resolution, you might have to pass 100000, or even 1000000, to get consistent results.

When you have a consistent answer, there are three ways you can use it:

```
import profile

# 1. Apply computed bias to all Profile instances created hereafter.
profile.Profile.bias = your_computed_bias

# 2. Apply computed bias to a specific Profile instance.
pr = profile.Profile()
pr.bias = your_computed_bias

# 3. Specify computed bias in instance constructor.
pr = profile.Profile(bias=your_computed_bias)
```

If you have a choice, you are better off choosing a smaller constant, and then your results will "less often" show up as negative in profile statistics.

Using a custom timer

If you want to change how current time is determined (for example, to force use of wall-clock time or elapsed process time), pass the timing function you want to the :class:`Profile` class constructor:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 655); *backlink*

Unknown interpreted text role "class".

```
pr = profile.Profile(your_time_func)
```

The resulting profiler will then call `your_time_func`. Depending on whether you are using :class:`profile.Profile` or :class:`cProfile.Profile`, `your_time_func`'s return value will be interpreted differently:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 661); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 661); *backlink*

Unknown interpreted text role "class".

:class:`profile.Profile`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 679); *backlink*

Unknown interpreted text role "class".

`your_time_func` should return a single number, or a list of numbers whose sum is the current time (like what :func:`os.times` returns). If the function returns a single time number, or the list of returned numbers has length 2, then you will get an especially fast version of the dispatch routine.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 666); *backlink*

Unknown interpreted text role "func".

Be warned that you should calibrate the profiler class for the timer function that you choose (see :ref:`profile-calibration`). For most machines, a timer that returns a lone integer value will provide the best results in terms of low overhead during profiling. (:func:`os.times` is *pretty* bad, as it returns a tuple of floating point values). If you want to substitute a better timer in the cleanest fashion, derive a class and hardwire a replacement dispatch method that best handles your timer call, along with the appropriate calibration constant.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 672); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 672); *backlink*

Unknown interpreted text role "func".

:class:`cProfile.Profile`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 693); *backlink*

Unknown interpreted text role "class".

`your_time_func` should return a single number. If it returns integers, you can also invoke the class constructor with a second argument specifying the real duration of one unit of time. For example, if `your_integer_time_func` returns times measured in thousands of seconds, you would construct the :class:`Profile` instance as follows:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 682); *backlink*

Unknown interpreted text role "class".

```
pr = cProfile.Profile(your_integer_time_func, 0.001)
```

As the :class:`cProfile.Profile` class cannot be calibrated, custom timer functions should be used with care and should be as fast as possible. For the best results with a custom timer, it might be necessary to hard-code it in the C source of the internal :mod:`_lsprof` module.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 690); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 690); *backlink*

Unknown interpreted text role "mod".

Python 3.3 adds several new functions in :mod:`time` that can be used to make precise measurements of process or wall-clock time. For example, see :func:`time.perf_counter`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 695); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)profile.rst, line 695); *backlink*

Unknown interpreted text role "func".