traceback – Extract, format, and print exceptions and stack traces.

Purpose:	Extract, format, and print exceptions and stack traces.
Available In:	1.4 and later, with modifications over time

The traceback module works with the call stack to produce error messages. A traceback is a stack trace from the point of an exception handler down the call chain to the point where the exception was raised. You can also work with the current call stack up from the point of a call (and without the context of an error), which is useful for finding out the paths being followed into a function.

The functions in traceback fall into several common categories. There are functions for extracting raw tracebacks from the current runtime environment (either an exception handler for a traceback, or the regular stack). The extracted stack trace is a sequence of tuples containing the filename, line number, function name, and text of the source line.

Once extracted, the stack trace can be formatted using functions like `format_exception()`, `format_stack()`, etc. The format functions return a list of strings with messages formatted to be printed. There are shorthand functions for printing the formatted values, as well.

Although the functions in traceback mimic the behavior of the interactive interpreter by default, they also are useful for handling exceptions in situations where dumping the full stack trace to stderr is not desirable. For example, a web application may need to format the traceback so it looks good in HTML. An IDE may convert the elements of the stack trace into a clickable list that lets the user browse the source.

Supporting Functions

The examples below use the module traceback_example.py (provided in the source package for PyMOTW). The contents are:

```
import traceback
import sys

def produce_exception(recursion_level=2):
    sys.stdout.flush()
    if recursion_level:
        produce_exception(recursion_level-1)
    else:
        raise RuntimeError()
```

```
def call_function(f, recursion_level=2):
    if recursion_level:
        return call_function(f, recursion_level-1)
    else:
        return f()
```

Working With Exceptions

The simplest way to handle exception reporting is with `print_exc()`. It uses `sys.exc_info()` to obtain the exception information for the current thread, formats the results, and prints the text to a file handle (`sys.stderr`, by default).

```
import traceback
import sys

from traceback_example import produce_exception

print 'print_exc() with no exception:'
traceback.print_exc(file=sys.stdout)
print

try:
    produce_exception()
except Exception, err:
    print 'print_exc():'
    traceback.print_exc(file=sys.stdout)
    print
    print 'print_exc(1):'
    traceback.print_exc(limit=1, file=sys.stdout)
```

In this example, the file handle for `sys.stdout` is substituted so the informational and traceback messages are mingled correctly:

```
$ python traceback_print_exc.py

print_exc() with no exception:
None

print_exc():
Traceback (most recent call last):
  File "traceback_print_exc.py", line 20, in <module>
    produce_exception()
  File "/Users/dhellmann/Documents/PyMOTW/src/PyMOTW/traceback/traceback_example.py",
```

```
line 16, in produce_exception
    produce_exception(recursion_level-1)
  File "/Users/dhellmann/Documents/PyMOTW/src/PyMOTW/traceback/traceback_example.py",
line 16, in produce_exception
    produce_exception(recursion_level-1)
  File "/Users/dhellmann/Documents/PyMOTW/src/PyMOTW/traceback/traceback_example.py",
line 18, in produce_exception
    raise RuntimeError()
RuntimeError

print_exc(1):
Traceback (most recent call last):
  File "traceback_print_exc.py", line 20, in <module>
    produce_exception()
RuntimeError
```

`print_exc()` is just a shortcut for `print_exception()`, which requires explicit arguments:

```
import traceback
import sys

from traceback_example import produce_exception

try:
    produce_exception()
except Exception, err:
    print 'print_exception():'
    exc_type, exc_value, exc_tb = sys.exc_info()
    traceback.print_exception(exc_type, exc_value, exc_tb)
```

```
$ python traceback_print_exception.py

Traceback (most recent call last):
  File "traceback_print_exception.py", line 16, in <module>
    produce_exception()
  File "/Users/dhellmann/Documents/PyMOTW/src/PyMOTW/traceback/traceback_example.py",
line 16, in produce_exception
    produce_exception(recursion_level-1)
  File "/Users/dhellmann/Documents/PyMOTW/src/PyMOTW/traceback/traceback_example.py",
line 16, in produce_exception
    produce_exception(recursion_level-1)
  File "/Users/dhellmann/Documents/PyMOTW/src/PyMOTW/traceback/traceback_example.py",
line 18, in produce_exception
```

```
    raise RuntimeError()
RuntimeError
print_exception():
```

And `print_exception()` uses `format_exception()`:

```
import traceback
import sys
from pprint import pprint

from traceback_example import produce_exception

try:
    produce_exception()
except Exception, err:
    print 'format_exception():'
    exc_type, exc_value, exc_tb = sys.exc_info()
    pprint(traceback.format_exception(exc_type, exc_value, exc_tb))
```

```
$ python traceback_format_exception.py

format_exception():
['Traceback (most recent call last):\n',
 '  File "traceback_format_exception.py", line 17, in <module>\n
produce_exception()\n',
 '  File "/Users/dhellmann/Documents/PyMOTW/src/PyMOTW/traceback/traceback_example.py",
line 16, in produce_exception\n    produce_exception(recursion_level-1)\n',
 '  File "/Users/dhellmann/Documents/PyMOTW/src/PyMOTW/traceback/traceback_example.py",
line 16, in produce_exception\n    produce_exception(recursion_level-1)\n',
 '  File "/Users/dhellmann/Documents/PyMOTW/src/PyMOTW/traceback/traceback_example.py",
line 18, in produce_exception\n    raise RuntimeError()\n',
 'RuntimeError\n']
```

Working With the Stack

There are a similar set of functions for performing the same operations with the current call stack instead of a traceback.

print_stack()

```
import traceback
import sys
```

```
from traceback_example import call_function

def f():
    traceback.print_stack(file=sys.stdout)

print 'Calling f() directly:'
f()

print
print 'Calling f() from 3 levels deep:'
call_function(f)
```

```
$ python traceback_print_stack.py

Calling f() directly:
  File "traceback_print_stack.py", line 19, in <module>
    f()
  File "traceback_print_stack.py", line 16, in f
    traceback.print_stack(file=sys.stdout)

Calling f() from 3 levels deep:
  File "traceback_print_stack.py", line 23, in <module>
    call_function(f)
  File "/Users/dhellmann/Documents/PyMOTW/src/PyMOTW/traceback/traceback_example.py",
line 22, in call_function
    return call_function(f, recursion_level-1)
  File "/Users/dhellmann/Documents/PyMOTW/src/PyMOTW/traceback/traceback_example.py",
line 22, in call_function
    return call_function(f, recursion_level-1)
  File "/Users/dhellmann/Documents/PyMOTW/src/PyMOTW/traceback/traceback_example.py",
line 24, in call_function
    return f()
  File "traceback_print_stack.py", line 16, in f
    traceback.print_stack(file=sys.stdout)
```

format_stack()

```
import traceback
import sys
from pprint import pprint

from traceback_example import call_function
```

```
def f():
    return traceback.format_stack()

formatted_stack = call_function(f)
pprint(formatted_stack)
```

```
$ python traceback_format_stack.py

['  File "traceback_format_stack.py", line 19, in <module>\n    formatted_stack =
call_function(f)\n',
 '  File "/Users/dhellmann/Documents/PyMOTW/src/PyMOTW/traceback/traceback_example.py",
line 22, in call_function\n    return call_function(f, recursion_level-1)\n',
 '  File "/Users/dhellmann/Documents/PyMOTW/src/PyMOTW/traceback/traceback_example.py",
line 22, in call_function\n    return call_function(f, recursion_level-1)\n',
 '  File "/Users/dhellmann/Documents/PyMOTW/src/PyMOTW/traceback/traceback_example.py",
line 24, in call_function\n    return f()\n',
 '  File "traceback_format_stack.py", line 17, in f\n    return
traceback.format_stack()\n']
```

extract_stack()

```
import traceback
import sys
from pprint import pprint

from traceback_example import call_function

def f():
    return traceback.extract_stack()

stack = call_function(f)
pprint(stack)
```

```
$ python traceback_extract_stack.py

[('traceback_extract_stack.py', 19, '<module>', 'stack = call_function(f)'),
 ('/Users/dhellmann/Documents/PyMOTW/src/PyMOTW/traceback/traceback_example.py',
  22,
  'call_function',
  'return call_function(f, recursion_level-1)'),
 ('/Users/dhellmann/Documents/PyMOTW/src/PyMOTW/traceback/traceback_example.py',
  22,
  'call_function',
```

```
 'return call_function(f, recursion_level-1)'),
('/Users/dhellmann/Documents/PyMOTW/src/PyMOTW/traceback/traceback_example.py',
 24,
 'call_function',
 'return f()'),
('traceback_extract_stack.py', 17, 'f', 'return traceback.extract_stack()')]
```

See also