[Catch a thread's exception in the caller thread in Python](https://stackoverflow.com/questions/2829329/catch-a-threads-exception-in-the-caller-thread-in-python)

There are a lot of really weirdly complicated answers to this question. Am I oversimplifying this, because this seems sufficient for most things to me.

from threading import Thread

class PropagatingThread(Thread):

def run(self):

self.exc = None

try:

if hasattr(self, '\_Thread\_\_target'):

# Thread uses name mangling prior to Python 3.

self.ret = self.\_Thread\_\_target(\*self.\_Thread\_\_args, \*\*self.\_Thread\_\_kwargs)

else:

self.ret = self.\_target(\*self.\_args, \*\*self.\_kwargs)

except BaseException as e:

self.exc = e

def join(self):

super(PropagatingThread, self).join()

if self.exc:

raise self.exc

return self.ret

If you're certain you'll only ever be running on one or the other version of Python, you could reduce the run() method down to just the mangled version (if you'll only be running on versions of Python before 3), or just the clean version (if you'll only be running on versions of Python starting with 3).

Example usage:

def f(\*args, \*\*kwargs):

print(args)

print(kwargs)

raise Exception('I suck at this')

t = PropagatingThread(target=f, args=(5,), kwargs={'hello':'world'})

t.start()

t.join()

And you'll see the exception raised on the other thread when you join.

If you are using six or on Python 3 only, you can improve the stack trace information you get when the exception is re-raised. Instead of only the stack at the point of the join, you can wrap the inner exception in a new outer exception, and get both stack traces with

six.raise\_from(RuntimeError('Exception in thread'),self.exc)

or

raise RuntimeError('Exception in thread') from self.exc