[Conditional with statement in Python](https://stackoverflow.com/questions/27803059/conditional-with-statement-in-python)

[Ask Question](https://stackoverflow.com/questions/ask)

Asked 5 years, 9 months ago

Active [11 months ago](https://stackoverflow.com/questions/27803059/conditional-with-statement-in-python?lastactivity)

Viewed 14k times

68

15

Is there a way to begin a block of code with a with statement, but conditionally?

Something like:

if needs\_with():

with get\_stuff() as gs:

# do nearly the same large block of stuff,

# involving gs or not, depending on needs\_with()

To clarify, one scenario would have a block encased in the with statement, while another possibility would be the same block, but not encased (i.e., as if it wasn't indented)

Initial experiments of course give indentation errors..

8 Answers

[Active](https://stackoverflow.com/questions/27803059/conditional-with-statement-in-python?answertab=active#tab-top)[Oldest](https://stackoverflow.com/questions/27803059/conditional-with-statement-in-python?answertab=oldest#tab-top)[Votes](https://stackoverflow.com/questions/27803059/conditional-with-statement-in-python?answertab=votes#tab-top)

55

If you want to avoid duplicating code and are using a version of Python prior to 3.7 (when contextlib.nullcontext was introduced) or even 3.3 (when contextlib.ExitStack was introduced), you could do something like:

class dummy\_context\_mgr():

def \_\_enter\_\_(self):

return None

def \_\_exit\_\_(self, exc\_type, exc\_value, traceback):

return False

or:

import contextlib

@contextlib.contextmanager

def dummy\_context\_mgr():

yield None

and then use it as:

with get\_stuff() if needs\_with() else dummy\_context\_mgr() as gs:

# do stuff involving gs or not

You alternatively could make get\_stuff() return different things based on needs\_with().

(See [Mike's answer](https://stackoverflow.com/a/34798330/179715) or [Daniel's answer](https://stackoverflow.com/a/53088625/179715) for what you can do in later versions.)

[share](https://stackoverflow.com/a/27806978)  [improve this answer](https://stackoverflow.com/posts/27806978/edit)  follow

[edited Feb 14 '19 at 16:13](https://stackoverflow.com/posts/27806978/revisions)

answered Jan 6 '15 at 20:52

[[https://www.gravatar.com/avatar/e240f1a5bb197450dfb8934dd55d9e3a?s=32&d=identicon&r=PG](https://stackoverflow.com/users/179715/jamesdlin)](https://stackoverflow.com/users/179715/jamesdlin)

[jamesdlin](https://stackoverflow.com/users/179715/jamesdlin)

**39.7k**66 gold badges8686 silver badges116116 bronze badges

* 5

This context manager should be in the standard python library, imho. Thanks for this. – [jjmontes](https://stackoverflow.com/users/401656/jjmontes" \o "14,565 reputation) [Jan 21 '16 at 18:37](https://stackoverflow.com/questions/27803059/conditional-with-statement-in-python#comment57596243_27806978)

* How about using the names *should\_...* and *dont*. Then statements like this would read with get\_stuff() if should\_get\_stuff() else dont() as gs: ? – [Riaz Rizvi](https://stackoverflow.com/users/213307/riaz-rizvi" \o "6,615 reputation) [Mar 25 '16 at 15:12](https://stackoverflow.com/questions/27803059/conditional-with-statement-in-python#comment60076469_27806978)
* @RiazRizvi I wouldn't have personally named it that way; I was using the names from the question. – [jamesdlin](https://stackoverflow.com/users/179715/jamesdlin" \o "39,664 reputation) [Mar 25 '16 at 18:01](https://stackoverflow.com/questions/27803059/conditional-with-statement-in-python#comment60081699_27806978)
* 1

@jjmontes [contextlib.ExitStack](https://docs.python.org/3/library/contextlib.html" \l "simplifying-support-for-single-optional-context-managers) (new python 3.3) can be used as a dummy context manager. – [Bryce Guinta](https://stackoverflow.com/users/487464/bryce-guinta) [May 14 '17 at 2:48](https://stackoverflow.com/questions/27803059/conditional-with-statement-in-python#comment74950659_27806978)

[add a comment](https://stackoverflow.com/questions/27803059/conditional-with-statement-in-python)

63

Python 3.3 introduced [contextlib.ExitStack](https://docs.python.org/3/library/contextlib.html" \l "contextlib.ExitStack) for just this kind of situation. It gives you a "stack", to which you add context managers as necessary. In your case, you would do this:

from contextlib import ExitStack

with ExitStack() as stack:

if needs\_with():

gs = stack.enter\_context(get\_stuff())

# do nearly the same large block of stuff,

# involving gs or not, depending on needs\_with()

Anything that is entered to stack is automatically exited at the end of the with statement as usual. (If nothing is entered, that's not a problem.) In this example, whatever is returned by get\_stuff() is exited automatically.

If you have to use an earlier version of python, you might be able to use the [contextlib2](https://contextlib2.readthedocs.org/en/latest/) module, although this is not standard. It backports this and other features to earlier versions of python. You could even do a conditional import, if you like this approach.

[share](https://stackoverflow.com/a/34798330)  [improve this answer](https://stackoverflow.com/posts/34798330/edit)  follow

[edited Jul 7 '18 at 14:49](https://stackoverflow.com/posts/34798330/revisions)

answered Jan 14 '16 at 19:45

[[https://www.gravatar.com/avatar/7143cfb952950822a883178f579113c5?s=32&d=identicon&r=PG](https://stackoverflow.com/users/1194883/mike)](https://stackoverflow.com/users/1194883/mike)

[Mike](https://stackoverflow.com/users/1194883/mike)

**14.7k**99 gold badges4646 silver badges7575 bronze badges

* 14

+1, this should be the selected answer. As pointed [here](https://docs.python.org/3/library/contextlib.html?highlight=contextmanager#simplifying-support-for-single-optional-context-managers) it is meant to deal with this kind of problem. Also, it can be used as a nifty one-liner: with get\_stuff() if needs\_with() else ExitStack() as gs. – [farsil](https://stackoverflow.com/users/930287/farsil" \o "733 reputation) [Feb 20 '17 at 14:14](https://stackoverflow.com/questions/27803059/conditional-with-statement-in-python#comment71844890_34798330)

[add a comment](https://stackoverflow.com/questions/27803059/conditional-with-statement-in-python)

30

As of Python 3.7 you can use contextlib.nullcontext:

from contextlib import nullcontext

if needs\_with():

cm = get\_stuff()

else:

cm = nullcontext()

with cm as gs:

# Do stuff

contextlib.nullcontext is pretty much just a no-op context manager. You can pass it an argument that it will yield, if you depend on something existing after the as:

>>> with nullcontext(5) as value:

... print(value)

...

5

Otherwise it'll just return None:

>>> with nullcontext() as value:

... print(value)

...

None

It's super neat, check out the docs for it here: <https://docs.python.org/3/library/contextlib.html#contextlib.nullcontext>

[share](https://stackoverflow.com/a/53088625)  [improve this answer](https://stackoverflow.com/posts/53088625/edit)  follow

[edited Oct 31 '18 at 17:14](https://stackoverflow.com/posts/53088625/revisions)

answered Oct 31 '18 at 17:02

[[https://i.stack.imgur.com/lgCDn.png?s=32&g=1](https://stackoverflow.com/users/3846032/daniel-porteous)](https://stackoverflow.com/users/3846032/daniel-porteous)

[Daniel Porteous](https://stackoverflow.com/users/3846032/daniel-porteous)

**2,794**11 gold badge1818 silver badges2727 bronze badges

* 4

This raises the question though, is it always safe to call get\_stuff() *before* entering the with statement? Is with open(file) as fh equivalent to f = open(file) followed by with f as fh? – [6005](https://stackoverflow.com/users/2038713/6005) [Jun 22 at 12:29](https://stackoverflow.com/questions/27803059/conditional-with-statement-in-python#comment110555539_53088625)

* 1

It depends on what your context manager does. Most context managers should not do stuff in their \_\_init\_\_ method and only do things on their \_\_enter\_\_ (or \_\_aenter\_\_) method, which is called when used in the with statement. So the answer unfortunately is "it depends". If you're worried about it, you could instead assign the functions to cm without calling them (with functools.partial if necessary) and then do with cm() as gs. – [Daniel Porteous](https://stackoverflow.com/users/3846032/daniel-porteous) [Oct 14 at 16:30](https://stackoverflow.com/questions/27803059/conditional-with-statement-in-python#comment113803308_53088625)

[add a comment](https://stackoverflow.com/questions/27803059/conditional-with-statement-in-python)

10

A third-party option to achieve exactly this:  
<https://pypi.python.org/pypi/conditional>

from conditional import conditional

with conditional(needs\_with(), get\_stuff()):

# do stuff

[share](https://stackoverflow.com/a/40510086)  [improve this answer](https://stackoverflow.com/posts/40510086/edit)  follow

[edited Nov 9 '16 at 15:34](https://stackoverflow.com/posts/40510086/revisions)

answered Nov 9 '16 at 15:05

[[https://www.gravatar.com/avatar/481f8ec4fe0149f2b3851eee91c23b8e?s=32&d=identicon&r=PG](https://stackoverflow.com/users/202168/anentropic)](https://stackoverflow.com/users/202168/anentropic)

[Anentropic](https://stackoverflow.com/users/202168/anentropic)

**25.7k**88 gold badges7979 silver badges121121 bronze badges

* Does it support an as ... clause at the end of the with statement? – [Craig McQueen](https://stackoverflow.com/users/60075/craig-mcqueen) [Dec 9 '16 at 0:40](https://stackoverflow.com/questions/27803059/conditional-with-statement-in-python#comment69310077_40510086)
* 1

looking at the source... yes it does. with conditional(needs\_with(), get\_stuff()) as stuff: will give you a reference to the get\_stuff() context manager (if and only if the condition is met, otherwise you get None) – [Anentropic](https://stackoverflow.com/users/202168/anentropic" \o "25,663 reputation) [Dec 9 '16 at 11:27](https://stackoverflow.com/questions/27803059/conditional-with-statement-in-python#comment69325222_40510086)

* I have found your answer is incomplete: [stackoverflow.com/questions/27803059/…](https://stackoverflow.com/questions/27803059/conditional-with-statement-in-python/58578167#58578167) – [Andry](https://stackoverflow.com/users/2672125/andry" \o "1,063 reputation) [Oct 27 '19 at 9:37](https://stackoverflow.com/questions/27803059/conditional-with-statement-in-python#comment103471416_40510086)

[add a comment](https://stackoverflow.com/questions/27803059/conditional-with-statement-in-python)

4

You can use contextlib.nested to put 0 or more context managers into a single with statement.

>>> import contextlib

>>> managers = []

>>> test\_me = True

>>> if test\_me:

... managers.append(open('x.txt','w'))

...

>>> with contextlib.nested(\*managers):

... pass

...

>>> # see if it closed

... managers[0].write('hello')

Traceback (most recent call last):

File "<stdin>", line 2, in <module>

ValueError: I/O operation on closed file

This solution has its quirks and I just noticed that as of 2.7 its been deprecated. I wrote my own context manager to handle juggling multiple context managers. Its worked for me so far, but I haven't really considered edge conditons

class ContextGroup(object):

"""A group of context managers that all exit when the group exits."""

def \_\_init\_\_(self):

"""Create a context group"""

self.\_exits = []

def add(self, ctx\_obj, name=None):

"""Open a context manager on ctx\_obj and add to this group. If

name, the context manager will be available as self.name. name

will still reference the context object after this context

closes.

"""

if name and hasattr(self, name):

raise AttributeError("ContextGroup already has context %s" % name)

self.\_exits.append(ctx\_obj.\_\_exit\_\_)

var = ctx\_obj.\_\_enter\_\_()

if name:

self.\_\_dict\_\_[name] = var

def exit\_early(self, name):

"""Call \_\_exit\_\_ on named context manager and remove from group"""

ctx\_obj = getattr(self, name)

delattr(self, name)

del self.\_exits[self.\_exits.index(ctx\_obj)]

ctx\_obj.\_\_exit\_\_(None, None, None)

def \_\_enter\_\_(self):

return self

def \_\_exit\_\_(self, \_type, value, tb):

inner\_exeptions = []

for \_exit in self.\_exits:

try:

\_exit(\_type, value, tb )

except Exception, e:

inner\_exceptions.append(e)

if inner\_exceptions:

r = RuntimeError("Errors while exiting context: %s"

% (','.join(str(e)) for e in inner\_exceptions))

def \_\_setattr\_\_(self, name, val):

if hasattr(val, '\_\_exit\_\_'):

self.add(val, name)

else:

self.\_\_dict\_\_[name] = val

[share](https://stackoverflow.com/a/27803938)  [improve this answer](https://stackoverflow.com/posts/27803938/edit)  follow

[edited Jan 6 '15 at 17:34](https://stackoverflow.com/posts/27803938/revisions)

answered Jan 6 '15 at 17:27

[[https://www.gravatar.com/avatar/0c280b71b3187d1cc113a6e302b414b9?s=32&d=identicon&r=PG](https://stackoverflow.com/users/642070/tdelaney)](https://stackoverflow.com/users/642070/tdelaney)

[tdelaney](https://stackoverflow.com/users/642070/tdelaney)

**49.5k**44 gold badges5353 silver badges7575 bronze badges

* 1

As I mention in my answer, python 3.3 has added contextlib.ExitStack, which appears to do very much what your ContextGroup does. I will say I'm a little surprised that it hasn't been backported, but if you're willing to require python >=3.3, that might be a nice robust alternative for you. – [Mike](https://stackoverflow.com/users/1194883/mike) [Jan 14 '16 at 20:02](https://stackoverflow.com/questions/27803059/conditional-with-statement-in-python#comment57340435_27803938)

* 1

contextlib2 is a pypi package which has backported ExitStack to python 2 – [Anthony Sottile](https://stackoverflow.com/users/812183/anthony-sottile) [Jul 3 '17 at 16:27](https://stackoverflow.com/questions/27803059/conditional-with-statement-in-python#comment76758428_27803938)

[add a comment](https://stackoverflow.com/questions/27803059/conditional-with-statement-in-python)

3

It was hard to find @farsil's nifty Python 3.3 one-liner, so here it is in its own answer:

with ExitStack() if not needs\_with() else get\_stuff() as gs:

# do stuff

Note that ExitStack should come first, otherwise get\_stuff() will be evaluated.