

Is there a way around coding in Python without the tab, indent & whitespace criteria?

Asked 16 years, 3 months ago Modified 11 months ago

Viewed 21k times



12



I want to start using Python for small projects but the fact that a misplaced tab or indent can throw a compile error is really getting on my nerves. Is there some type of setting to turn this off?

I'm currently using NotePad++. Is there maybe an IDE that would take care of the tabs and indenting?

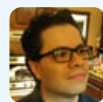
python

Share

Improve this question

Follow

asked Sep 15, 2008 at 13:55



Donny V.

23.5k ● 13 ● 68 ● 81

No. How would Python parse your script if there was a way to disable this? – [innaM](#) Sep 15, 2008 at 13:56

- 1 I use Notepad++ too (2013). Added a shortcut for menu / View / Show Symbol / Show whitespace and tab. In menu / Settings / Preferences / Tab Settings you can set whether to use tabs or (custom amount of, default 4)

spaces. (I [assigned a shortcut](#) for changing that setting too but it's quite hacky and buggy.) If I receive a file with random indentation, it can uniformize them by pressing *Ctrl+A Tab *Shift+Tab*. It works more consistently than Eclipse for me.
– [n611x007](#) Nov 28, 2013 at 17:24

Use bython it is literally python without the dumb whitespaces. – [Vladimir_314159](#) Jun 19, 2018 at 16:14

if you are coming from C, then maybe Cling suits you better.
– [Foad S. Farimani](#) Oct 4, 2018 at 22:59

I think in Notepad++ only, saving the file or setting the file as .py would do this for you. – [Amit Amola](#) Jan 6, 2019 at 17:24

23 Answers

Sorted by:

Highest score (default)



The answer is no.

51

At least, not until something like the following is implemented:



```
from __future__ import braces
```



Share Improve this answer

Follow

edited Oct 15, 2021 at 18:53



[Cody Gray](#) ♦

244k ● 52 ● 501 ● 581

answered Sep 16, 2008 at 1:55



[Chris Calloway](#)

1,478 ● 1 ● 10 ● 12

10 A cruel, cruel joke. I saw that and almost tried it. Indentation and dynamic typing is going to kill this language. I load up

someone's code in an IDE (which the apologists all say I should be using as it solves all of python's problems) and everything is underlined with squiggles because the author didn't follow any conventions. It's like Word underlining your whole document for spelling and grammar. Ten years from now there will be braces in python 4. Right now we are being forced to use Guido's pet code format. I feel like I'm writing VBA. – [Sean Anderson](#) Sep 20, 2018 at 22:39

That is the solution, if it existed. Can we make it exist? Using a change in indenting as a delimiter is cruel and unusual syntax. I have recently fallen into this trap again. I had a line that accidentally went up a for loop level and changed the code without giving an error. If we replaced ':' with '{' and <change of indentation> with '}' then everything would be fine. – [Richard Kirk](#) Aug 23, 2021 at 11:27

Implemented it. pypi.org/project/pyend – [Mircode](#) Feb 26, 2023 at 13:21



No. Indentation-as-grammar is an integral part of the Python language, for better and worse.

38



Share Improve this answer

answered Sep 15, 2008 at 13:56

Follow



[Asaf Bartov](#)

2,790 ● 3 ● 20 ● 18



3 Yup, for the better. It makes it harder to write hard to read code. – [Echo says Reinstate Monica](#) Sep 19, 2008 at 14:58

The question asked about available IDEs &c., and there are certainly such; thus, this is a bad answer. – [Charles Duffy](#) Sep 22, 2008 at 23:30

3 @Xiong Chiamiov It isn't necessarily that the OP suck at dealing with the indention him/herself. It's perferctly possible for people to work on the same python file, and one uses this indentation, and the other uses a different one. Most IDEs won't render whitespaces by default, and it *will* drive you nuts if you're getting a whitespace error. The lack of tokens to indicate the scope makes it impossible to get an auto-indent that would at least *show* what the problem is. – [Calyth](#) Oct 8, 2009 at 16:44

2 Pythons' indentation rules are meant to father everybody into following some lame rules.If indentation rules are so popular,make them optional. – [You Know Nothing Jon Snow](#) Jul 23, 2013 at 7:05

2 for worse. It becomes impossible to copy paste some snippets in an interactive python shell to test them quickly. Also moving around some blocks of code into or outside loops makes it a pain – [Alex F](#) Dec 18, 2017 at 13:06



Emacs! Seriously, its use of "tab is a *command*, not a *character*", is absolutely perfect for python development.

11



Share Improve this answer

answered Sep 15, 2008 at 23:56

Follow



user10470





7

All of the whitespace issues I had when I was starting Python were the result mixing tabs and spaces. Once I configured everything to just use one or the other, I stopped having problems.



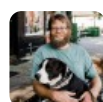
In my case I configured UltraEdit & vim to use spaces in place of tabs.



[Share](#) [Improve this answer](#)

[Follow](#)

answered Sep 15, 2008 at 13:59



[Mark Biek](#)

151k ● 54 ● 158 ● 201



5

It's possible to write a pre-processor which takes randomly-indented code with pseudo-python keywords like "endif" and "endwhile" and properly indents things. I had to do this when using python as an "ASP-like" language, because the whole notion of "indentation" gets a bit fuzzy in such an environment.



Of course, even with such a thing you really ought to indent sanely, at which point the conveter becomes superfluous.

[Share](#) [Improve this answer](#)

[Follow](#)

answered Sep 15, 2008 at 14:06



[Captain Obvious](#)



I find it hard to understand when people flag this as a problem with Python. I took to it immediately and actually

5

find it's one of my favourite 'features' of the language :)



In other languages I have two jobs: 1. Fix the braces so the computer can parse my code 2. Fix the indentation so I can parse my code.



So in Python I have half as much to worry about ;-)

(nb the only time I ever have problem with indendation is when Python code is in a blog and a forum that messes with the white-space but this is happening less and less as the apps get smarter)

Share Improve this answer

answered Sep 15, 2008 at 15:15

Follow



[Andy Baker](#)

51 ● 1

You find it hard to understand that people see using indentation to delimit blocks is a problem? Really? – [Ben](#) Jul 27, 2011 at 9:34

After you fix the braces, you can use a text editor to automatically fix indentation. – [ThePiercingPrince](#) Jul 15, 2013 at 4:58



4



I'm currently using NotePad++. Is there maybe an IDE that would take care of the tabs and indenting?

I liked [pydev](#) extensions of eclipse for that.





Share Improve this answer

Follow

answered Sep 15, 2008 at 14:22



Raz

1,932 ● 2 ● 18 ● 23



3

I do not believe so, as Python is a whitespace-delimited language. Perhaps a text editor or IDE with auto-indentation would be of help. What are you currently using?



Share Improve this answer

Follow

answered Sep 15, 2008 at 13:56



Thomas Owens

116k ● 99 ● 317 ● 436



3

No, there isn't. Indentation is syntax for Python. You can:

1. Use tabnanny.py to check your code
2. Use a syntax-aware editor that highlights such mistakes (vi does that, emacs I bet it does, and then, most IDEs do too)
3. (far-fetched) write a preprocessor of your own to convert braces (or whatever block delimiters you love) into indentation



Share Improve this answer

Follow

answered Sep 15, 2008 at 13:59



tzot

95.8k ● 30 ● 149 ● 208



3



You should disable tab characters in your editor when you're working with Python (always, actually, IMHO, but especially when you're working with Python). Look for an option like "Use spaces for tabs": any decent editor should have one.



Share Improve this answer

answered Sep 15, 2008 at 14:00



Follow



[Chris Conway](#)

56k ● 43 ● 131 ● 155

2 Yes, that is true. But Python has one interpretation of tabs and your text editor may have another. For this reason, the Python language manual recommends not mixing tabs and spaces in indentation. IMHO, the easiest way to guarantee this is to always use spaces. – [Chris Conway](#) Sep 1, 2009 at 21:10



2



Not really. There are a few ways to modify whitespace rules for a given line of code, but you will still need indent levels to determine scope.



You can terminate statements with `;` and then begin a new statement on the same line. (Which people often do when [golfing](#).)



If you want to break up a single line into multiple lines you can finish a line with the `\` character which means the current line effectively continues from the first non-whitespace character of the next line. This visually *appears* violate the usual whitespace rules but is legal.

My advice: don't use tabs if you are having tab/space confusion. Use spaces, and choose either 2 or 3 spaces as your indent level.

A good editor will make it so you don't have to worry about this. (python-mode for [emacs](#), for example, you can just use the tab key and it will keep you honest).

Share Improve this answer

edited Sep 15, 2008 at 14:10

Follow

answered Sep 15, 2008 at 14:00



Justin Standard

21.5k ● 22 ● 82 ● 90



Tabs and spaces confusion can be fixed by setting your editor to use spaces instead of tabs.

2



To make whitespace completely intuitive, you can use a stronger code editor or an IDE (though you don't need a full-blown IDE if all you need is proper automatic code indenting).



A list of editors can be found in the Python wiki, though that one is a bit too exhausting: -

<http://wiki.python.org/moin/PythonEditors>

There's already a question in here which tries to slim that down a bit:

- <https://stackoverflow.com/questions/60784/poll-which-python-ideeditor-is-the-best>

Maybe you should add a more specific question on that:
"Which Python editor or IDE do you prefer on Windows -
and why?"

Share Improve this answer

edited May 23, 2017 at 12:17

Follow



Community Bot

1 • 1

answered Sep 15, 2008 at 14:28



Arne Babenhauserheide

2,503 • 29 • 23



1



I agree with justin and others -- pick a good editor and use spaces rather than tabs for indentation and the whitespace thing becomes a non-issue. I only recently started using Python, and while I thought the whitespace issue would be a real annoyance it turns out to not be the case. For the record I'm using emacs though I'm sure there are other editors out there that do an equally fine job.

If you're really dead-set against it, you can always pass your scripts through a pre-processor but that's a bad idea on many levels. If you're going to learn a language, embrace the features of that language rather than try to work around them. Otherwise, what's the point of learning a new language?

Share Improve this answer

answered Sep 15, 2008 at 14:09

Follow



Bryan Oakley

385k ● 53 ● 569 ● 727



I was a bit reluctant to learn Python because of tabbing. However, I almost didn't notice it when I used Vim.

1



Share Improve this answer

answered Sep 15, 2008 at 14:34

Follow



metadave

111 ● 3



If you don't want to use an IDE/text editor with automatic indenting, you can use the pindent.py script that comes in the Tools\Scripts directory. It's a preprocessor that can convert code like:

1



```
def foobar(a, b):  
    if a == b:  
        a = a+1  
    elif a < b:  
        b = b-1  
    if b > a: a = a-1  
    end if  
    else:  
        print 'oops!'  
    end if  
end def foobar
```

into:

```
def foobar(a, b):  
    if a == b:  
        a = a+1  
    elif a < b:  
        b = b-1  
        if b > a: a = a-1  
        # end if  
    else:  
        print 'oops!'  
    # end if  
# end def foobar
```

Which is valid python.

Share Improve this answer

answered Sep 15, 2008 at 23:53

Follow



Ryan

15.3k ● 7 ● 50 ● 50

But why would want to write code like in the first example?
It's harder to read. – [Echo says Reinstate Monica](#) Sep 19, 2008 at 15:00

You would only input it the first way. Perhaps you have physical limitations, such as being visually impaired and using a screen reader, or are copy/pasting code from something that eats tabs. – [Ryan](#) Sep 22, 2008 at 3:32

How does it know that the second "if" goes inside the "elif", or that the final "else" doesn't? – [detly](#) Apr 22, 2010 at 23:32

The same way that other compilers match up parenthesis. It keeps it on a stack :) – [Ryan](#) Apr 23, 2010 at 2:44

Ah, I did not notice the `end if` lines! Makes more sense.
– [detly](#) Apr 23, 2010 at 11:39



Getting your indentation to work correctly is going to be important in any language you use.

1



Even though it won't affect the execution of the program in most other languages, incorrect indentation can be very confusing for anyone trying to read your program, so you need to invest the time in figuring out how to configure your editor to align things correctly.



Python is pretty liberal in how it lets you indent. You can pick between tabs and spaces (but you really should use spaces) and can pick how many spaces. The only thing it requires is that you are consistent which ultimately is important no matter what language you use.

Share Improve this answer

edited Sep 19, 2008 at 14:26

Follow

answered Sep 15, 2008 at 14:16



indentation

9,985 ● 6 ● 23 ● 14



Nope, there's no way around it, and it's by design:

1



```
>>> from __future__ import braces
      File "<stdin>", line 1
SyntaxError: not a chance
```



Most Python programmers simply don't use tabs, but use spaces to indent instead, that way there's no editor-to-



editor inconsistency.

Share Improve this answer

answered Sep 22, 2008 at 23:27

Follow



Dan Lenski

79.6k ● 13 ● 84 ● 127



1



I'm surprised no one has mentioned IDLE as a good default python editor. Nice syntax colors, handles indents, has intellisense, easy to adjust fonts, and it comes with the default download of python. Heck, I write mostly IronPython, but it's so nice & easy to edit in IDLE and run ipy from a command prompt.



Oh, and what is the big deal about whitespace? Most easy to read C or C# is well indented, too, python just enforces a really simple formatting rule.

Share Improve this answer

answered Oct 8, 2009 at 16:35

Follow



epic_fil

669 ● 7 ● 15



0



Many Python IDEs and generally-capable text/source editors can handle the whitespace for you.



However, it is best to just "let go" and enjoy the whitespace rules of Python. With some practice, they won't get into your way at all, and you will find they have many merits, the most important of which are:

1. Because of the forced whitespace, Python code is simpler to understand. You will find that as you read code written by others, it is easier to grok than code in, say, Perl or PHP.
2. Whitespace saves you quite a few keystrokes of control characters like { and }, which litter code written in C-like languages. Less {s and }s means, among other things, less RSI and wrist pain. This is not a matter to take lightly.

Share Improve this answer

answered Sep 15, 2008 at 15:57

Follow



[Eli Bendersky](#)

273k ● 91 ● 362 ● 422



In Python, indentation is a semantic element as well as providing visual grouping for readability.

0



Both space and tab can indicate indentation. This is unfortunate, because:



- The interpretation(s) of a tab varies among editors and IDEs and is often configurable (and often configured).
- OTOH, some editors are not configurable but apply their own rules for indentation.
- Different sequences of spaces and tabs may be visually indistinguishable.
- Cut and pastes can alter whitespace.

So, unless you know that a given piece of code will only be modified by yourself with a single tool and an unvarying config, you must avoid tabs for indentation (configure your IDE) and make sure that you are warned if they are introduced (search for tabs in leading whitespace).

And you can still expect to be bitten now and then, as long as arbitrary semantics are applied to control characters.

Share Improve this answer

answered Sep 15, 2008 at 17:25

Follow



Elmer



0

Check the options of your editor or find an editor/IDE that allows you to convert TABs to spaces. I usually set the options of my editor to substitute the TAB character with 4 spaces, and I never run into any problems.



Share Improve this answer

answered Sep 16, 2008 at 3:03

Follow



Curro

852 ● 2 ● 10 ● 14



0

Yes, there is a way. I hate these "no way" answers, there is no way until you discover one.



And in that case, whatever it is worth, there is one.



I read once about a guy who designed a way to code so that a simple script could re-indent the code properly. I didn't managed to find any links today, though, but I swear I read it.

The main tricks are to *always* use `return` at the end of a function, *always* use `pass` at the end of an `if` or at the end of a class definition, and always use `continue` at the end of a `while`. Of course, any other *no-effect* instruction would fit the purpose.

Then, a simple `awk` script can take your code and detect the end of block by reading `pass/continue/return` instructions, and the start of code with `if/def/while/...` instructions.

Of course, because you'll develop your indenting script, you'll see that you don't have to use `continue` after a `return` inside the `if`, because the `return` will trigger the indent-back mechanism. The same applies for other situations. Just get use to it.

If you are diligent, you'll be able to cut/paste and add/remove `if` and correct the indentations automagically. And incidentally, pasting code from the web will require you to understand a bit of it so that you can adapt it to that "non-classical" setting.

Share Improve this answer


answered Oct 4, 2009 at 0:20

Follow



Gzorg

857 ● 4 ● 10 ● 25

-
- 1 The "whitespace thing" in Python exists specifically to **avoid** requiring any specific closing mark or word at the end of `if / while / def` blocks. **How would you distinguish the end of an `if` block within a `class` block, from the end of the `class` block itself?** There is only one standard no-op statement, `pass`, which is valid in both contexts.
- [Dan Lenski](#) Jun 19, 2014 at 17:26 
-



0



Yes, I agree with the workout suggested by Ryan. I also face the problem of misindentation of python code especially when I was coping some loops or ifs between different codes. For the multi-level for loops, a mis-indentation will cost a lot of time to fix. So I just try to manually add commented endif or endfor to clarify the loop structure. eg:

```
for i in range(0, N):  
    x = x + i  
    for j in range(0, M):  
        if x < 0:  
            y = -x*x  
        else:  
            y = x*x  
        #endif  
    #endfor  
#endfor
```

Of course this will cause some extra coding. But when the indentation is misplaced, it will be easy to get the correct loop structure with the end notice.

I personally hope that future python version could accept the end experssion and also made it compatable with old version python without end.

Share Improve this answer

answered Dec 29, 2023 at 10:57

Follow



DocNan

11



Highly active question. Earn 10 reputation (not counting the [association bonus](#)) in order to answer this question. The reputation requirement helps protect this question from spam and non-answer activity.