Do you have a hard time keeping to 80 columns with Python? [closed]

Asked 15 years, 10 months ago Modified 9 years, 6 months ago

Viewed 8k times

16

Closed. This question is opinion-based. It is not currently accepting answers.

Want to improve this question? Update the question so it can be answered with facts and citations by editing this post.

Closed 9 years ago.

Improve this question

I find myself breaking strings constantly just to get them on the next line. And of course when I go to change those strings (think logging messages), I have to reformat the breaks to keep them within the 80 columns.

How do most people deal with this?

python

asked Feb 24, 2009 at 18:49

Share

Improve this question

Follow

ApplePieIsGood
3,731 ● 6 ● 37 ● 51

I'm not sure if you're implying this, but Python doesn't force you to keep line lengths within 80 columns. I find shorter lines easier to read, but you're free to make them as long as you wish. – Cerin Apr 30, 2010 at 16:35

13 Answers

Sorted by: Highest score (default)

27

I recommend trying to stay true to 80-column, but not at *any cost*. Sometimes, like for logging messages, it just makes more sense to keep 'em long than breaking up. But for most cases, like complex conditions or list comprehensions, breaking up is a good idea because it will help you divide the complex logic to more understandable parts. It's easier to understand:

```
print sum(n for n in xrange(1000000)
             if palindromic(n, bits_of_n) and palindro
```

Than:

```
print sum(n for n in xrange(1000000) if palindromic(n,
palindromic(n, digits))
```

It may look the same to you if you've just written it, but after a few days those long lines become hard to understand.

Finally, while PEP 8 dictates the column restriction, it also says:

> A style guide is about consistency. Consistency with this style guide is important. Consistency within a project is more important. Consistency within one module or function is most important.
>
> But most importantly: know when to be inconsistent -- sometimes the style guide just doesn't apply. When in doubt, use your best judgment. Look at other examples and decide what looks best. And don't hesitate to ask!

Share Improve this answer Follow

edited Nov 24, 2013 at 18:27

answered Feb 24, 2009 at 19:01

Eli Bendersky
273k ● 91 ● 362 ● 422

Another example is long if conditions. I think its easier to read a 100 character if condition than to break it across two/three lines using (). – Richard Levasseur Feb 25, 2009 at 2:20

12

"A foolish consistency is the hobgoblin of little minds, adored by little statesmen and philosophers and divines."

The important part is "foolish".

The 80-column limit, like other parts of PEP 8 is a pretty strong suggestion. But, there is a limit, beyond which it could be seen as foolish consistency.

I have the indentation guides and edge line turned on in Komodo. That way, I know when I've run over. The questions are "why?" and "is it worth fixing it?"

Here are our common situations.

logging messages. We try to make these easy to wrap. They look like this

```
logger.info( "unique thing %s %s %s",
   arg1, arg2, arg3 )
```

Django filter expressions. These can run on, but that's a good thing. We often knit several filters together in a row. But it doesn't have to be one line of code, multiple lines can make it more clear what's going on.

This is an example of functional-style programming, where a long expression is sensible. We avoid it, however.

Unit Test Expected Result Strings. These happen because we cut and paste to create the unit test code and don't spend a lot of time refactoring it. When it bugs us we pull the strings out into separate string variables and clean the `self.assertXXX()` lines up.

We generally don't have long lines of code because we don't use lambdas. We don't strive for fluent class design. We don't pass lots and lots of arguments (except in a few cases).

We rarely have a lot of functional-style long-winded expressions. When we do, we're not embarrassed to break them up and leave an intermediate result lying around. If we were functional purists, we might have gas with intermediate result variables, but we're not purists.

Share Improve this answer Follow

edited Apr 30, 2010 at 16:32
Nick Presta
28.6k ● 6 ● 59 ● 76

answered Feb 24, 2009 at 19:09
S.Lott
391k ● 82 ● 517 ● 788

"Next Stop Wonderland", one of my favorite movies, used that quote from Emerson as a running bit throughout. Terrific stuff. – duffymo Feb 24, 2009 at 19:12

9

It doesn't matter what year is it or what output devices you use (to some extent). Your code should be readable if possible by *humans*. It is hard for humans to read long lines.

It depends on the line's content how long it should be. If It is a log message then its length matters less. If it is a complex code then its big length won't be helping to comprehend it.

Share Improve this answer Follow

answered Feb 24, 2009 at 20:00

jfs
413k ● 205 ● 1k ● 1.7k

8

Temporary variables. They solve almost every problem I have with long lines. Very occasionally, I'll need to use some extra parens (like in a longer if-statement). I won't make any arguments for or against 80 character limitations since that seems irrelevant.

Specifically, for a log message; instead of:

```
self._log.info('Insert long message here')
```

Use:

```
msg = 'Insert long message here'
self._log.info(msg)
```

The cool thing about this is that it's going to be two lines no matter what, but by using good variable names, you also make it self-documenting. E.g., instead of:

```
obj.some_long_method_name(subtotal * (1 + tax_rate))
```

Use:

```
grand_total = subtotal * (1 + tax_rate)
obj.some_long_method_name(grand_total)
```

Most every long line I've seen is trying to do more than one thing and it's trivial to pull one of those things out into a temp variable. The primary exception is very long strings, but there's usually something you can do there too, since strings in code are often structured. Here's an example:

```
br = mechanize.Browser()
ua = '; '.join(('Mozilla/5.0 (Macintosh', 'U', 'Intel
                'en-US', 'rv:1.9.0.6) Gecko/2009011912
br.addheaders = [('User-agent', ua)]
```

Share Improve this answer

Follow

answered Feb 24, 2009 at 23:42

Ryan Bright
3,555 ● 23 ● 21

This doesn't solve *every* problem certainly, but I have to agree with the premise that it often gives the win-win of shorter lines and clearer code. +1. – Beska Feb 27, 2009 at 21:06

6

This is a good rule to keep to a large part of the time, but don't pull your hair out over it. The most important thing is that stylistically your code looks readable and clean, and keeping your lines to reasonable length is part of that.

Sometimes it's nicer to let things run on for more than 80 columns, but most of the time I can write my code such that it's short and concise and fits in 80 or less. As some responders point out the limit of 80 is rather dated, but it's not bad to have such a limit and many people have terminals

Here are some of the things that I keep in mind when trying to restrict the length of my lines:

- is this code that I expect other people to use? If so, what's the standard that those other people and use for this type of code?
- do those people have laptops, use giant fonts, or have other reasons for their screen real estate being limited?
- does the code look better to me split up into multiple lines, or as one long line?

This is a stylistic question, but style is really important because you want people to read and understand your code.

Share Improve this answer answered Feb 24, 2009 at 19:12

Follow

James Thompson
48k ● 18 ● 68 ● 84

4

I would suggest being willing to go beyond 80 columns. 80 columns is a holdover from when it was a hard limit based on various output devices.

Now, I wouldn't go hog wild...set a reasonable limit, but an arbitary limit of 80 columns seems a bit overzealous.

EDIT: Other answers are also clarifing this: it matters what you're breaking. Strings can more often be "special cases" where you may want to bend the rules a bit, for the sake of clarity. If your *code*, on the other hand, is getting long, that's a good time to look at where it is logical to break it up.

Share Improve this answer Follow

edited Feb 24, 2009 at 19:04

answered Feb 24, 2009 at 18:52

Beska
12.7k ● 14 ● 79 ● 113

Python style guides do not like your answer. . .python.org/dev/peps/pep-0008 – Baltimark Feb 24, 2009 at 18:54

2 It's not just a holdover. Long lines are difficult to read even if you resize your window to be wide enough to display them. That's why newspapers and magazines lay out text in columns. I wish programmers would try to stick to 80 columns

or perhaps 100 in verbose languages (Java, C#). – Tmdean Feb 24, 2009 at 18:55

@Tmdean: I would agree that the lines shouldn't get overly long...but his point is that he is breaking strings in places that seem awkward...ie it's getting *less* readable by the breaks he's inserting. A bit of flexibility may well go a long way. – Beska Feb 24, 2009 at 18:59

2 Using any hard limit is stupid. If adding breaks makes it more readable, add them. Otherwise, go past 80. – Matthew Olenik Feb 24, 2009 at 19:04

1 I agree with Matt. Part of the Zen of Python is to keep things clear, and I don't think you'll find many hardcore Pythonistas who would be willing to let much get in the way of clarity, within reason. – Wayne Koorts Feb 24, 2009 at 20:19

4

80 character limits? What year is it?

Make your code readable. If a long line is readable, it's fine. If it's hard to read, split it.

For example, I tend to make long lines when there is a method call with lots of arguments, and the arguments are the normal arguments you'd expect. So, let's say I'm passing 10 variables around to a bunch of methods. If every method takes a transaction id, an order id, a user id, a credit card number, etc, and these are stored in appropriately named variables, then it's ok for the method call to appear on one line with all the variables one after another, because there are no surprises.

If, however, you are dealing with multiple transactions in one method, you need to ensure that the next programmer can see that THIS time you're using transId1, and THAT time transId2. In that case make sure it's clear. (Note: sometimes using long lines HELPS that too).

Just because a "style guide" says you should do something doesn't mean you have to do it. Some style guides are just plain wrong.

Share Improve this answer Follow

answered Feb 24, 2009 at 19:38

Mr. Shiny and New 安宇
13.9k ● 6 ● 46 ● 65

I'd like to vote this up for the good points made, but simply can't because of the last sentence: an unnecessary, irrelevant stab at Strunk & White. :) (Partially outdated, deserving criticism? Sure. Plain wrong? No.) – Jonik Feb 24, 2009 at 20:22

@Jonik: Thanks for the Strunk and White reference. Swiping at Strunk and White's too easy. Most of their advice is really sound, even if White didn't choose to follow it strictly. The blog is helpful because it's so heavily one-sided. – S.Lott Feb 24, 2009 at 20:32

Maybe "Plain Wrong" is too strong but people brandish S&W as if it's the Gospel of writing and never wrong. Heck, even SO has the S&W badge :(– Mr. Shiny and New 安宇 Feb 24, 2009 at 21:51

4

The 80 column count is one of the few places I disagree with the Python style guide. I'd recommend you take a look at the audience for your code. If everyone you're working with uses a modern IDE on a monitor with a reasonable resolution, it's probably not worth your time. My monitor is cheap and has a relatively weak resolution, but I can still fit 140 columns plus scroll bars, line markers, break markers, and a separate tree-view frame on the left.

However, you will probably end up following some kind of limit, even if it's not a fixed number. With the exception of messages and logging, long lines are hard to read. Lines that are broken up are also harder to read. Judge each situation on its own, and do what you think will make life easiest for the person coming after you.

Share Improve this answer Follow

answered Feb 24, 2009 at 21:37

DNS
38.2k ● 20 ● 96 ● 133

Since no reason was given for voting this down and it's obviously a reasonable answer I'm voting it back up for balance. – Wayne Koorts Feb 25, 2009 at 17:16

2 Try working on low quality code written by someone else with the line lengths all up around 120 chars and you might reconsider. It just gets too dense to swallow without lots of tiresome chewing. – John Mee Aug 12, 2011 at 7:05

1 Comments and string literals are harder to read at 79 characters than 100-120. If you're seeing actual code lines at 120 or more on a regular basis, then you probably have a

more fundamental problem, like excessive nesting or over-complicated conditions, that isn't going to be solved by artificially imposing a length limit. – DNS Aug 12, 2011 at 17:13

3

Strings are special because they tend to be long, so break them when you need and don't worry about it.

When your actual *code* starts bumping the 80 column mark it's a sign that you might want to break up deeply nested code into smaller logical chunks.

Share Improve this answer Follow

answered Feb 24, 2009 at 18:56

dwc
24.9k ● 7 ● 45 ● 55

3

I deal with it by not worrying about the length of my lines. I know that some of the lines I write are longer than 80 characters but most of them aren't.

I know that my position is not considered "pythonic" by many and I understand their points. Part of being an engineer is knowing the trade-offs for each decision and then making the decision that you think is the best.

Share Improve this answer Follow

answered Feb 24, 2009 at 19:03

David Locke
18.1k ● 9 ● 34 ● 53

3

Sticking to 80 columns is important not only for readability, but because many of us like to have narrow terminal windows so that, at the same time as we are coding, we can also see things like module documentation loaded in our web browser and an error message sitting in an xterm. Giving your whole screen to your IDE is a rather primitive, if not monotonous, way to use screen space.

Generally, if a line stretches to more than 80 columns it means that something is going wrong anyway: either you are trying to do too much on one line, or have allowed a section of your code to become too deeply indented. I rarely find myself hitting the right edge of the screen unless I am also failing to refactor what should be separate functions; name temporary results; and do other things like that will make testing and debugging much easier in the end. Read Linus's Kernel Coding Style guide for good points on this topic, albeit from a C perspective:

http://www.kernel.org/doc/Documentation/CodingStyle

And always remember that long strings can either be broken into smaller pieces:

```
print ("When Python reads in source code"
       " with string constants written"
       " directly adjacent to one another"
       " without any operators between"
       " them, it considers them one"
       " single string constant.")
```

Or, if they are really long, they're generally best defined as a constant then used in your code under that abbreviated name:

```
STRING_MESSAGE = (
        "When Python reads in source code"
        " with string constants written directly adjac
        " another without any operators between them,
        " them one single string constant.")

...
print STRING_MESSAGE
...
```

Share Improve this answer

Follow

edited Sep 13, 2012 at 5:48

bignose
32.2k ● 15 ● 79 ● 116

answered Apr 30, 2010 at 16:25

Brandon Rhodes
88.9k ● 16 ● 109 ● 149

1

Pick a style you like, apply a layer of common sense, and use it consistently.

PEP 8 is a style guide for libraries included as part of the Python standard library. It was never intended to be picked up as the style rules for all Python code. That said, there's no reason people shouldn't use it, but it's definitely not a set of hard rules. Like any style, there is no single correct way and the most important thing is consistency.

Share Improve this answer

edited Dec 23, 2013 at 20:31

Follow

answered Feb 24, 2009 at 20:21

Wayne Koorts
11.1k ● 13 ● 48 ● 72

I'm not the original downvoter. But your answer has no content. Sure pick a style, common sense it and then use consistently. It can't be relevant because it's not quite there. – muhuk Feb 26, 2009 at 16:47

I disagree. He's obviously sticking to 80 columns because of PEP-8 but the point I'm making is that these are just style *guidelines*, not *rules*. – Wayne Koorts Feb 26, 2009 at 17:16

@Wayne: I'm also not a downvoter on this answer. But if your comment was part of your answer it'd be clearer. – Mr. Shiny and New 安宇 Feb 27, 2009 at 13:55

i agree with @Wayne Koorts, PEP8 is coding convention for more readability not a rule to impose to make your application works. – James Sapam Dec 23, 2013 at 5:46

0

I do run into code that spills past 79 columns on every now and then. I've either broken them up with '\' (although recently, I've read about using parenthesis instead as a preferred alternative, so I'll give that a shot), or just let it be if it's no more than 15 or so past. And this coming from someone who indents only 2, not 4 spaces (I know, shame on me :\ )! It isn't entirely science. It's also part style, and sometimes, keeping things on one line is just easier to manage or read. Other times, excessive side-to-side scrolling can be worse.

Much of the time has to do with longer variable names. For variables beyond temp values and iterators, I don't want to reduce them to 1 to 5 letters. These that are 7 to 15 characters long actually do provide context as to their uses and what classes they refer to.

When I need to print stuff out where parts of the output are dynamic, I'll replace those portions with function calls that cuts down on the conditional statements and sheer content that would've been in that body of code.

Share Improve this answer Follow

edited Jun 17, 2015 at 17:45

answered Jun 17, 2015 at 17:27

ackmondual

385 ● 3 ● 12