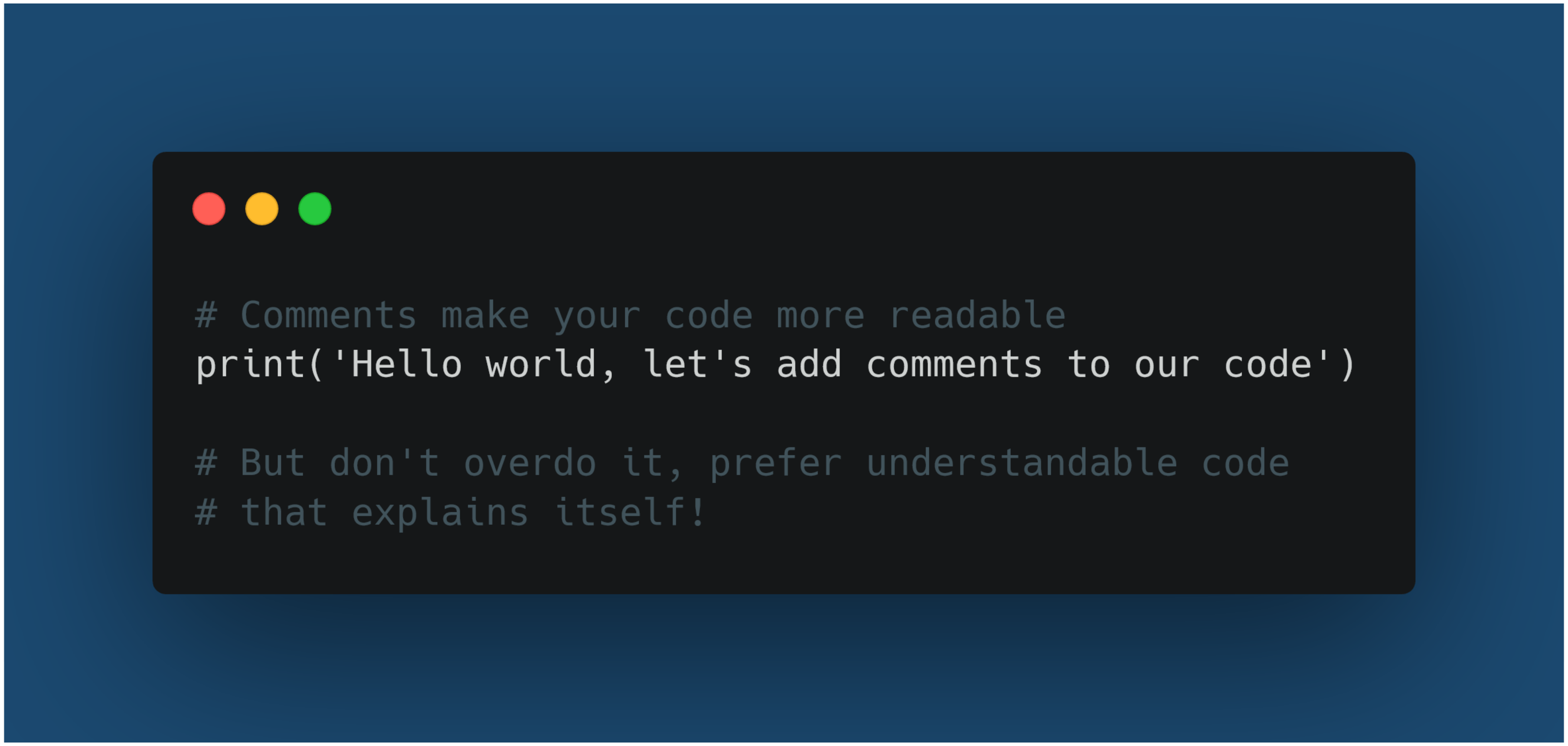
Previous: Your First Python Program

Next: Creating Python Programs ›

Home » Introduction to Python » Python Comment: What It Is And How to Create



```
# Comments make your code more readable
print('Hello world, let's add comments to our code')

# But don't overdo it, prefer understandable code
# that explains itself!
```

Python Comment: What It Is And How to Create

September 20, 2022

A Python comment is an explanation in the source code of a Python program. It doesn't do anything besides being informative and is ignored by the Python interpreter. In this article, you will learn:

- what Python comments are,
- how to add comments to your Python code,

- what the common pitfalls are when writing comments

Table of Contents [hide]

What is a Python comment?

Let's start by defining what a comment is exactly:

Comment
A comment is an explanation in the source code of a Python program. Comments are added with the purpose of making source code easier for humans to understand. They are ignored by the Python interpreter.

So, in short, a comment helps you and other readers of your code to better understand it.

A single-line Python comment

In Python, we create a comment by starting the line with the hash symbol: **#**. This symbol is known as the number sign, hash, or (in North American usage) pound sign. Whatever you call it, this is what it looks like:

```
# This is a single-line comment in Python
print('Hello')
# And here's another one!
```

Multiline Python comment

There's no special way to add multi-line comments, also called block comments. Instead, you simply use the hash symbol at the start of each line, like so:

```
# This is a multi-line comment in Python,
# there's not much difference with single-
# line comments, as you can see!
```

In Python (or any language, for that matter), it's better to write clear, understandable code. So if you're doing well, you should not need multi-line comments that often!

Comment out code

A pattern that programmers use a lot, is commenting out code temporarily. There are countless reasons why one might want to do this. Here are just a couple of examples:

- A piece of functionality is not working right now, but you plan to work on it later so you don't want to throw it away
- There's an alternative piece of code that you might want to use later on, so you leave it as a hint in a comment
- You're refactoring your code, meaning you're cleaning up the code. Often, this is a process where you create more and smaller functions, while you're not completely ready to throw away the old code.

To comment out code in Python, we again use the hash symbol:

```
# This code is disabled for now
# if password == 'welcome123':
#     allow_access()
```

Tip: In most IDEs and editors, you can comment out multi-line blocks of code by selecting all the lines and pressing **Ctrl + /**. You can reverse the operation by doing exactly the same.

Common mistakes with Python comments

There are a few mistakes that many programmers make when it comes to comments. Therefore, let's explore these, so you hopefully won't make them!

Using too many comments

You should limit the use of comments to what is absolutely necessary. Why is that?

There's a concept called self-documenting code. This means that your code is so readable and easy to understand that comments or documentation are hardly needed. If your code is readable and you don't need comments, it's more compact and easier to read. Hence, it would be best if you always strived to write self-documenting code and only resort to comments to explain the more advanced or non-obvious things.

Writing self-documenting code deserves its own article, but there are three key takeaways that already can help a lot:

1. Name your Python variables so that it's abundantly clear what they contain. E.g., don't name your list just m, or m_list, but instead, call it something like member_list.
2. Name your Python functions concisely so they clearly describe what they do. Put a verb in there if you can, like get_user, or add_item.
3. Create short functions that only do one thing. If they do more than one thing, split them into multiple functions. Short functions are also better once you start using unit tests.

Stating the obvious

Many beginning programmers tend to state the obvious. In other words, they describe in comments what they are about to do. You too might be tempted to do so, especially when a language is new to you, as a note to yourself.

Here are some examples of stating the obvious:

```python
# Ask for the user's name
name = input('Name: ')
# Call the say_hi function
say_hi(name)
# Now we divide the user input by 100
result = input / 100
```

Not maintaining your Python comments

While editing your code, it's easy to forget the comments that are there. Always check if they still apply or perhaps need a touch-up because you just changed the code.

It's a common mistake to forget comments that describe a function or a class because they are often not very close to the code you're editing. After refactoring your code, always take a look at the comments.

Commenting out code instead of removing it

As stated above, sometimes you need to comment out code. But that doesn't mean you can just leave it there. In my professional life as a programmer, I've seen (too) many commented-out sections of code. The problem with this is that nobody dares to touch it. Your colleague must have left it there for a reason, right?

No… your colleague probably just forgot about it. But can you be sure? For this reason, make a habit of cleaning up code that is commented out.

Keep learning

You may also be interested in my tutorial on Python docstrings.

Get certified with our courses

‹ Previous: Your First Python Program

Next: Creating Python Programs