How to write commands, indent, and comments in Python

What are comments in Python? This article will tell you everything about how to write comments in Python.

Comments are very important in Python programming. This article will **guide you in detail on how to write comments in Python** .

Python is one of the most popular programming languages ??today. You can easily find it in many of the most popular programming products currently available. Using Python isn't difficult; basically, you just need to understand the most common concepts and features.

When writing code in Python, it's important to ensure your code is easily understandable to others. Using clear variable names, defining functions clearly, and organizing your code are excellent ways to achieve this.

Another great and easy way to improve your code readability is to use comments!

In this tutorial, you'll learn some basics about writing comments in Python. You'll learn how to write clear and concise comments, when you might not need to write any comments at all, how to indent, and much more.

Write code in Python

Python syntax can be implemented by writing directly into the Command Line:

```
>>> print("Hello, World!") Hello, World!
```

In this context, `print`print` is a Python function used to print a string to the screen. The content inside the double quotes ("") is what you want to display on the screen.

Alternatively, you can create a Python file on the server, use the .py file extension, and run it from the Command Line:

```
C:UsersYour Name>python myfile.py
```

To run a Python program, you can use an online Python compiler or a Python executable installed on your computer.

Write multi-line Python statements

In Python, a statement is typically terminated by a line break. However, sometimes you need to write a longer statement in Python and want a line break for easier readability.

To write a multi-line statement in Python, you can use parentheses `()`, square brackets `[]`, or curly braces `{}`. The parentheses here imply that the line of code continues until a closing bracket is placed.

For example, to write a statement `print`across multiple lines, you can use wrapping characters like this:

```
mau_sac = ['vàng', 'xanh', 'cam']
```

Note that when writing a statement across multiple lines, you need to ensure that subsequent lines are indented to let Python know that they are part of the preceding statement.

```
b = (1 + 3 + 5 + 7 + 9 + 11 + 13 + 15 + 17)
```

However, we don't always need parentheses to create line breaks for statements. For flexibility, you can also extend a statement across multiple lines using the line continuation character ( ). The example above can be rewritten as follows:

```
b = 1 + 3 + 5 +  7 + 9 + 11 +  13 + 15 + 17
```

Alternatively, you can also place multiple commands on the same line, separated by semicolons, `;`like this:

```
a = 1; b = 2; c = 3
```

Indent in Python

Python indentation is the whitespace (space or tab) before a block of code. It's fundamental to any Python syntax and isn't just a convention, it's a requirement. In other words, indentation indicates that all statements with the same whitespace to the right are part of the same block of code.

Indentation is crucial in Python because it ensures code readability. Unlike C, C++, and other languages ??where curly braces denote a block of code, Python uses indentation (the number of leading spaces) to render the same group of statements in black.

In Python, indentation serves not only to define code flow but also the following purposes:

Python is designed to increase code readability, a key feature of Python programming. That's where indentation plays a crucial role. Consistent indentation helps keep code organized, clean, and easy to understand. Developers can adhere to best programming practices and ensure seamless team collaboration.

Some programming languages ??like C, C++, and Java use curly braces {} to define a block of code. However, Python uses indentation, which helps to clarify its code structure. Therefore, managing indentation in Python is essential as it can affect logic and code flow.

Python uses indentation to indicate a block of code. For example:

```
if 5 > 2: print("Five is greater than two!")
```

Python will throw an error if you omit indentation:

```
File "", line 2 print("Five is greater than two!") ^ IndentationError: expected a
```

The number of indentation spaces depends on your web or app programming habits. The most common is 4, but the minimum should be 1 space.

```
if 5 > 2: print("Five is greater than two!") if 5 > 2: print("Five is greater tha
```

Typically, you'll use the spacebar four times to indent. The tab key, however, is preferred by programmers, as in the example below:

```
for i in range(1,11): print(i) if i == 5: break
```

The code block `for`consists of `print(i)`and `if`, which are indented equally. `break`is the command within `if`, so it's indented even further. You don't need to understand exactly what the code above says yet; just remember how Python indents work. We'll learn the details later, one by one.

If the above command were written as:

```
for i in range(1,11): print(i) if i == 5: break
```

You will receive an error message immediately, and the error will appear before the command `print(i)`.

Indentation makes Python code look neater and clearer. You can skip indentation for multi-line statements, but experienced programmers recommend always using indentation; it makes the code easier to read.

For example:

```
if True: print('Xin chào!') q = 10
```

And:

```
if True: print('Xin chào!'); q = 10
```

Both ways of writing are correct and perform the same task, but do you find the first way clearer?

Comments and annotations in Python

Comments are a way for code writers to communicate with code readers; arguably, they are an indispensable component of programming code. They help describe what is happening in the program so that code readers don't waste too much time trying to understand or guess.

More practically, you might be writing multiple programs simultaneously, overlapping them, and it's normal to mix up or forget important details of code you wrote months ago. That's where comments become a lifesaver. Start making it a habit to write comments now.

In Python, we use a character **#**to start a comment. Comments begin after the character **#**until the start of a new line. When interpreted, Python ignores these comments.

```
#?ây là chú thích #In dòng ch? TipsMake.com.com print('TipsMake.com')
```

You can write comments on the same line as the command or expression without any problem, like this:

```
print('TipsMake.com') #?ây là chú thích in dòng ch? TipsMake.com.com
```

If you want to write comments on multiple lines, it's very simple; just add a # before each line, like this:

```
#?ây là chú thích #trên nhi?u dòng #In dòng ch?
 TipsMake.com.com #trong Python print('TipsMake.com')
```

Another way to write comments is to use three single quotes `' ' '` or double quotes `" " "`. These quotes are commonly used for multi-line strings. But they can also be used to write multi-line comments. As long as it's not a docstring, it won't generate any additional code.

```
"""?ây là chú thích trên nhi?u dòng In dòng ch?
 TipsMake.com.com trong Python"""
```

How to use comments in Python

Make the code easier to understand.

Writing comments in your code makes it easier to reference them in the future. Additionally, your code is easier for other programmers to understand.

Use comments to debug.

If you encounter an error while running the program, you can leave a comment on the line of code causing the error instead of removing it. For example:

```
print('Python') # print('Error Line ) print('Django')
```

Here, **print('Error Line')** was causing an error, so it was changed to a comment. Now the program runs normally without errors.

Here's how comments can become a valuable debugging tool.

Note: Always use comments to explain why certain data in the code was changed, rather than explaining how something was done. Comments should not be used to explain code quality.

Is writing comments in Python easy? This article will guide you in detail on how to write indentation and comments in Python.

How to properly add comments in Python

Comments or notes are an important part of any program. Therefore, you need to learn how to write effective notes. Here are some characteristics that identify well-written comments and notes:

1. Make sure they are accurate.
2. Don't write generic comments; write clearly and specifically (a=10 #attach to 10 to avoid writing generic comments).

3. Write comments that describe the overall task of a function or method, not its specific details.
4. Good comments and notes are always clear and concise.
5. Don't write unnecessary notes.

Docstring in Python

Docstring stands for *Documentation string* , and it appears as the first statement in a module, function, or method definition. You will need to write what the function or class will do within the docstring.

Three double quotation marks are used to write a docstring, as in the example below:

```
def double(num): """Ham nhan doi gia tri""" return 2*num
```

The docstring will appear as a `__doc__`function property. To view the function's docstring, you can use the command `print()`below:

```
def double(num): """Ham nhan doi gia tri""" return 2*num print(double.__doc__)
```

Result:

```
Ham nhan doi gia tri
```

Wait for user action.

The following command will display a prompt saying " *Press Enter to exit!* " and wait for user action.

```
input("nnNh?n phím Enter ?? thoát!")
```

Here, " `nn`" is used to create two new lines before displaying the actual line. Once the user presses Enter, the program will terminate. This is a good trick to keep the program window displayed until the user finishes interacting with the application.

Reviewing lesson knowledge

In short, comments in Python are lines of code that the interpreter ignores during program execution.

1. Comments help improve the readability of code.
2. Comments can be used to identify the function or structure of the codebase.
3. Comments can help understand unusual or tricky situations that the code handles to prevent accidental deletion or modification.
4. Comments can be used to prevent the execution of any specific part of your code when making changes or performing tests.

Some best practices for writing comments in Python.

Here are some tips you can follow to add effective comments in Python.

1. The annotations should be concise and accurate.
2. Only use comments when necessary; don't clutter your source code with comments.
3. Avoid writing generic or basic annotations.
4. Please write the annotations in an easy-to-understand manner.

Above are the things you need to know about writing code, comments, and indentation in Python. By mastering these rules and good commenting techniques in Python, programmers can easily read and understand code. It's just one of many fundamental concepts in Python that you must learn to master this programming language.