**Writing Readable Python Code: Style Guides and Best Practices**

**Why Readability Matters**

* Python is known for its **readable, human-friendly syntax**.
* Code is **read more often than it is written**—clarity matters.
* Readable code helps teams **collaborate**, **debug**, and **maintain** scripts more effectively.

**What Is a Style Guide?**

* A **style guide** provides rules and conventions for writing consistent, clean code.
* Python’s main style guide is called **PEP 8**:  
  *Python Enhancement Proposal 8*
* PEP 8 covers:
  + Formatting
  + Syntax
  + Naming conventions
  + Comments
  + Indentation
  + Line length, etc.

**Commenting in Python**

* **Comments** explain what code does and why.
* Begin with # and should be short and informative.

**Example Without a Comment:**

if failed\_attempts > 5:

print("Account locked")

**Example With a Comment:**

# Lock account if too many failed login attempts

if failed\_attempts > 5:

print("Account locked")

✅ Keep comments clear and update them when the code changes.

**Indentation and Structure**

* **Indentation** is required in Python to define code blocks.
* Commonly used with:
  + Conditionals (if)
  + Loops (for, while)
  + Functions (def)
* PEP 8 recommends using **4 spaces per indentation level**.

**Example:**

if is\_outdated:

print("Updates needed") # Correctly indented block

If indentation is missing or incorrect, Python throws an error or executes the code incorrectly.

**Real-World Importance**

* Code without consistent formatting becomes **hard to maintain**.
* In teams or long-term projects, unreadable code results in:
  + Wasted time
  + Bugs
  + Slower development
* **Adhering to style guides** ensures your code can evolve without confusion or rework.

**Summary of Key Best Practices**

| **Area** | **PEP 8 Recommendation** |
| --- | --- |
| **Comments** | Keep them relevant and updated |
| **Indentation** | Use 4 spaces per level |
| **Naming** | Use descriptive names (user\_id, not x) |
| **Line length** | Max 79 characters per line |
| **Whitespace** | Use spaces around operators (x = 5, not x=5) |

**Final Note**

Clean, well-formatted code isn't just professional—it's practical.  
Following PEP 8 makes your code **easier to understand, debug, and scale**, which is essential for both individual success and team efficiency.

Up next: we'll build more habits that make your code not just work—but **work well**.

**Ensure proper syntax and readability in Python**

Previously, you were introduced to the PEP 8 style guide and its stylistic guidelines for programmers working in Python. You also learned about how adding comments and using correct indentation makes your code more readable. Additionally, correct indentation ensures your code is executed properly. This reading explores these ideas further and also focuses on common items to check in the syntax of your code to ensure it runs.

**Comments**

A **comment** is a note programmers make about the intentions behind their code. Comments make it easier for you and other programmers to read and understand your code.

It’s important to start your code with a comment that explains what the program does. Then, throughout the code, you should add additional comments about your intentions behind specific sections.

When adding comments, you can add both single-line comments and multi-line comments.

**Single-line comments**

Single-line comments in Python begin with the (*#*) symbol. According to the PEP 8 style guide, it’s best practice to keep all lines in Python under 79 characters to maintain readability, and this includes comments.

Single-line comments are often used throughout your program to explain the intention behind specific sections of code. For example, this might be when you're explaining simpler components of your program, such as the following *for* loop:

*# Print elements of 'computer\_assets' list*

*computer\_assets = ["laptop1", "desktop20", "smartphone03"]*

*for asset in computer\_assets:*

*print(asset)*

**Note:** Comments are important when writing more complex code, like functions, or multiple loops or conditional statements. However, they're optional when writing less complex code like reassigning a variable.

**Multi-line comments**

Multi-line comments are used when you need more than 79 characters in a single comment. For example, this might occur when defining a function if the comment describes its inputs and their data types as well as its output.

There are two commonly used ways of writing multi-line comments in Python. The first is by using the hashtag (*#*) symbol over multiple lines:

*# remaining\_login\_attempts() function takes two integer parameters,*

*# the maximum login attempts allowed and the total attempts made,*

*# and it returns an integer representing remaining login attempts*

*def remaining\_login\_attempts(maximum\_attempts, total\_attempts):*

*return maximum\_attempts - total\_attempts*

Another way of writing multi-line comments is by using documentation strings and not assigning them to a variable. Documentation strings, also called docstrings, are strings that are written over multiple lines and are used to document code. To create a documentation string, use triple quotation marks (*""" """*).

You could add the comment to the function in the previous example in this way too:

*"""*

*remaining\_login\_attempts() function takes two integer parameters,*

*the maximum login attempts allowed and the total attempts made,*

*and it returns an integer representing remaining login attempts*

*"""*

**Correct indentation**

**Indentation** is space added at the beginning of a line of code. In Python, you should indent the body of conditional statements, iterative statements, and function definitions. Indentation is not only necessary for Python to interpret this syntax properly, but it can also make it easier for you and other programmers to read your code.

The PEP 8 style guide recommends that indentations should be four spaces long. For example, if you had a conditional statement inside of a *while* loop, the body of the loop would be indented four spaces and the body of the conditional would be indented four spaces beyond that. This means the conditional would be indented eight spaces in total.

*count = 0*

*login\_status = True*

*while login\_status == True:*

*print("Try again.")*

*count = count + 1*

*if count == 4:*

*login\_status = False*

**Maintaining correct syntax**

Syntax errors involve invalid usage of the Python language. They are incredibly common with Python, so focusing on correct syntax is essential in ensuring that your code runs. Awareness of common errors will help you more easily fix them.

Syntax errors often occur because of mistakes with data types or in the headers of conditional or iterative statements or of function definitions.

**Data types**

Correct syntax varies depending on data type:

* Place string data in quotation marks.
  + Example: *username = "bmoreno"*
* Do not add quotation marks around integer, float, or Boolean data types.
  + Examples: *login\_attempts = 5*, *percentage\_successful = .8*, *login\_status = True*
* Place lists in brackets and separate the elements of a list with commas.
  + Example: *username\_list = ["bmoreno", "tshah"]*

**Colons in headers**

The header of a conditional or iterative statement or of a function definition must end with a colon. For example, a colon appears at the end of the header in the following function definition:

*def remaining\_login\_attempts(maximum\_attempts, total\_attempts):*

*return maximum\_attempts - total\_attempts*

**Key takeaways**

The PEP 8 style guide provides recommendations for writing code that can be easily understood and read by other Python programmers. In order to make your intentions clear, you should incorporate comments into your code. Depending on the length of the comment, you can follow conventions for single-line or multi-line comments. It's also important to use correct indentation; this ensures your code will run as intended and also makes it easier to read. Finally, you should also be aware of common syntax issues so that you can more easily fix them.

**Resources for more information**

Learning to write readable code can be challenging, so make sure to review the PEP 8 style guide and learn about additional aspects of code readability.

* [PEP 8 - Style Guide for Python Code](https://peps.python.org/pep-0008/)

: The PEP 8 style guide contains all standards of Python code. When reading this guide, it's helpful to use the table of contents to navigate through the concepts you haven't learned yet.