Python Comments

In the previous tutorial, you learned to write your first Python program. Now, let's learn about Python comments.

Important!: We are introducing comments early in this tutorial series because we will be using them to explain the code in upcoming tutorials.

Comments are hints that we add to our code to make it easier to understand. Python comments start with `#`. For example,

```
# print a number
print(25)
```

Here, `# print a number` is a comment.

Comments are completely ignored and not executed by code editors.

Important: The purpose of this tutorial is to help you understand comments, so you can ignore other concepts used in the program. We will learn about them in later tutorials.

Single-line Comment

We use the **hash (#)** symbol to write a single-line comment. For example,

```
# declare a variable
name = "John"

# print name
print(name)    # John
```

In the above example, we have used three single-line comments:

- `# declare a variable`
- `# print name`
- `# John`

A **single-line comment** starts with `#` and extends up to the end of the line. We can also use single-line comments alongside the code:

```
print(name)    # John
```

Note: Remember the keyboard shortcut to apply comments. In most text editors, it's **Ctrl** + **/** if you are on Windows & **Cmd** + **/** if you are on a Mac.

Multiline Comments

Unlike languages such as C++ and Java, Python doesn't have a dedicated method to write multi-line comments.

However, we can achieve the same effect by using the hash (`#`) symbol at the beginning of each line.

Let's look at an example.

```
# This is an example of a multiline comment
# created using multiple single-line commenced
# The code prints the text Hello World
```

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

We can also use multiline strings as comments like:

```
'''This is an example
of multiline comment'''
print("Hello, World!")
```

Output

```
Hello World
```

Note: Remember you will learn about these programming concepts in upcoming tutorials. For now. you can just focus on the usage of comments.

Prevent Executing Code Using Comments

Comments are valuable when debugging code.

If we encounter an error while running a program, instead of removing code segments, we can comment them out to prevent execution. For example,

```
number1 = 10
number2 = 15

sum = number1 + number2

print("The sum is:", sum)
print("The product is:", product)
```

Here, the code throws an error because we have not defined a *product* variable. Instead of removing the line causing an error, we can comment it.

For example,

```
number1 = 10
number2 = 15

sum = number1 + number2

print("The sum is:", sum)
# print('The product is:', product)
```

Output

```
The sum is 25
```

Here, the code runs without any errors.

We have resolved the error using a comment. Now if you need to calculate the *product* in the near future, you can uncomment it.

Note: This approach comes in handy while working with large files. Instead of deleting any line, we can use comments and identify which one is causing an error.

Why Use Comments?

We should use comments:

- For future references, as comments make our code readable.
- For debugging.
- For code collaboration, as comments help peer developers to understand each other's code.

Note: Comments are not and should not be used as a substitute to explain poorly written code. Always try to write clean, understandable code, and then use comments as an addition.

In most cases, always use comments to explain 'why' rather than 'how' and you are good to go.

Next, we will start learning fundamental concepts of Python programming.

Table of Contents