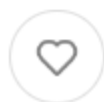


# 7 Uses of Underscore in Python

Must-know for Python programmers.

AVI CHAWLA

MAR 2



READ IN APP ↗

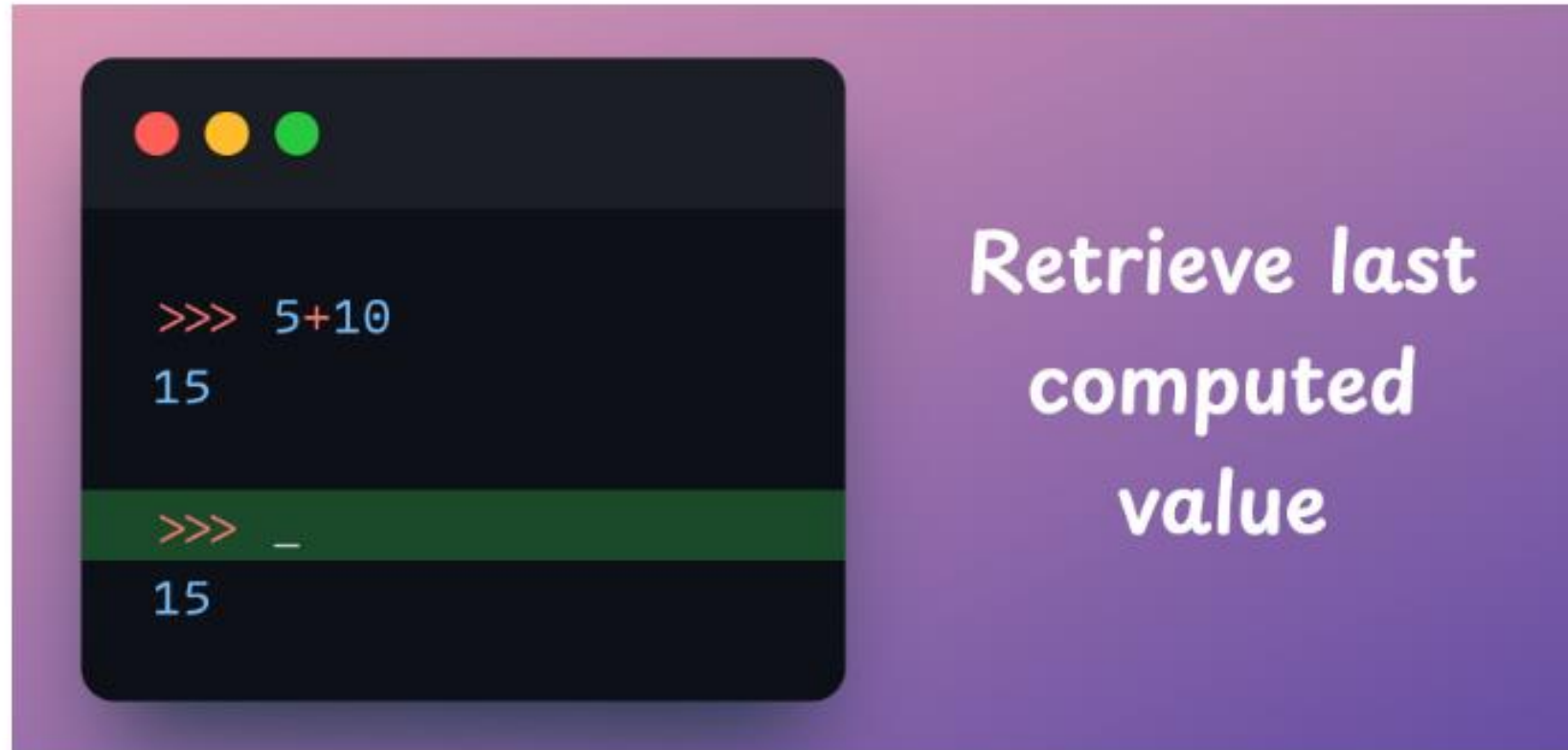
Underscore (`_`) has so many usages in Python.

Today, I want to walk you through 7 of them, which I find pretty useful from time to time.

Let's begin!

## #1) Retrieve the last computed value


One of the most elegant utilities of underscore is to retrieve the last computed value, as demonstrated below:



This works both in a script (.py) and an interactive environment like Jupyter Notebook.

## #2) Placeholder for loop variable

Instead of explicitly declaring a loop variable, one can also run loops as follows:




```
for _ in range(100):  
    ...
```

**No loop  
variable**

### #3) Digit separator

When declaring large numbers, it can be difficult to interpret them.

Underscore simplifies this:



```
>>> big_num = 10000000000000000
```

Difficult to  
interpret



```
>>> big_num = 100_000_000_000_000
```



Easy to  
read and  
interpret

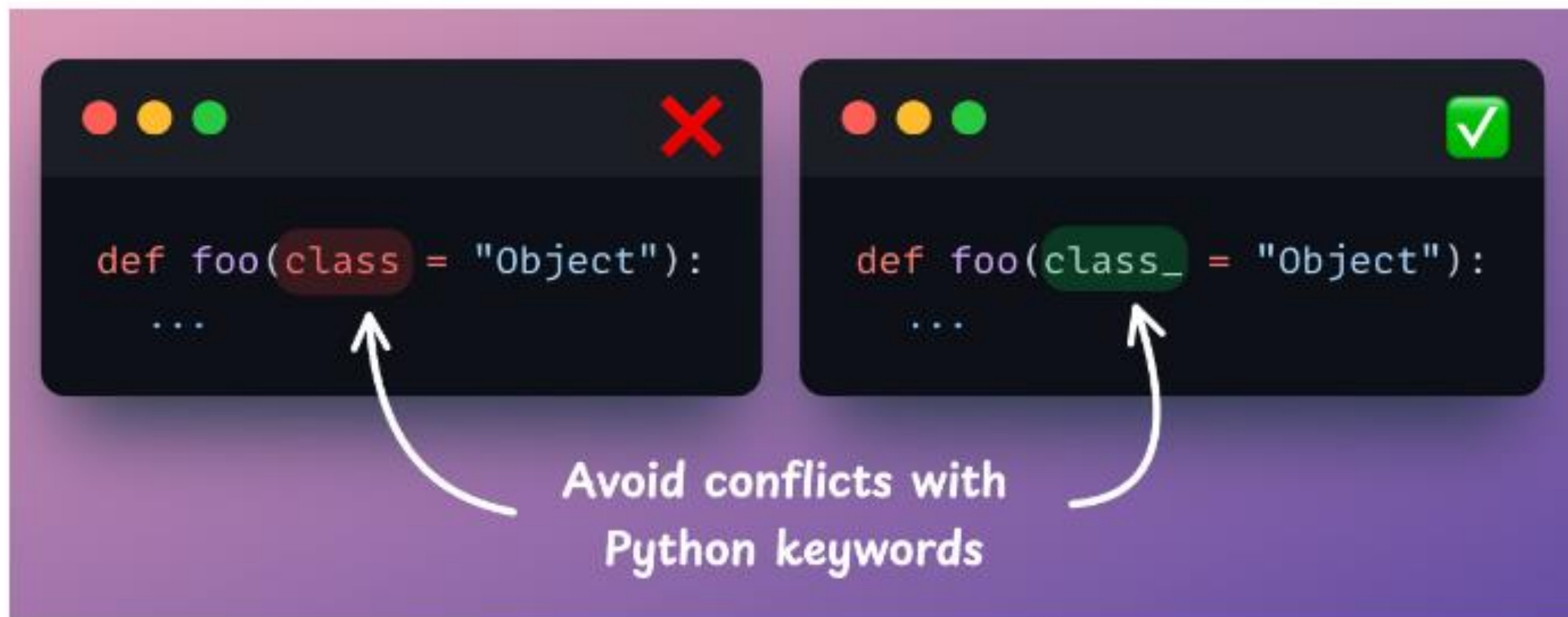
## #4-7) Declaring names

We can also use underscore while naming objects.

- A single leading underscore is used to declare variables for internal use. Thus, they cannot be imported during wild imports (`from file import *`)  
\*)



- A single trailing underscore is used to avoid conflict with reserved keywords, as depicted below:



- Double leading underscores are used to invoke name mangling. This way, one can prevent direct access to private variables outside a class:

```
class MyClass:
    def __init__(self):
        self.__variable = 10
```

**Name  
mangling  
for private  
attributes**

```
obj = MyClass()
```

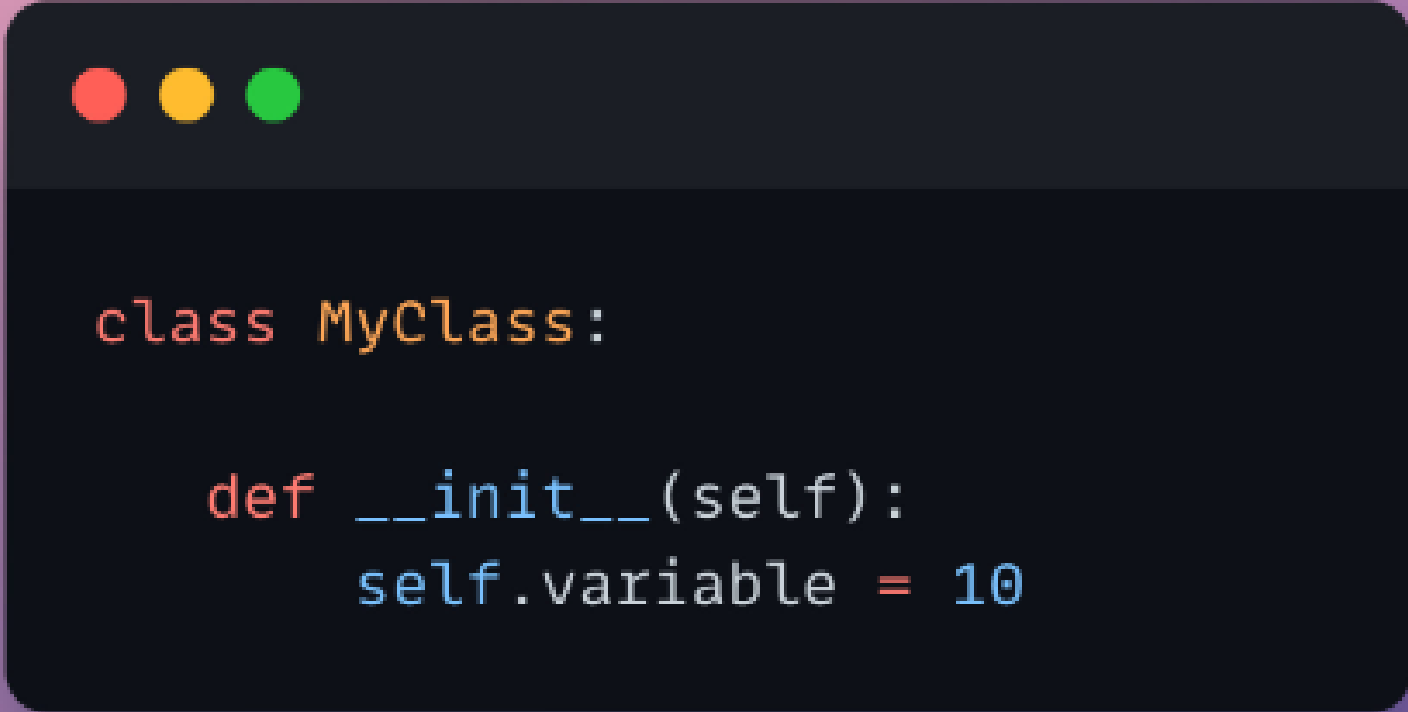
```
>>> obj.__variable
AttributeError: 'MyClass' object
has no attribute '__variable'
```

```
>>> obj._MyClass__variable
10
```

We covered this in detail in this issue: [Python Does Not Fully Deliver OOP Encapsulation Functionalities](#).



- Finally, double leading and trailing underscores, as you may already know, are used to define magic methods:



```
class MyClass:  
  
    def __init__(self):  
        self.variable = 10
```

**Magic  
Method**

