

🤯 Python's Sneaky Mutable Default Arguments!



The Mystery Code:

```
In [49]: # Function with a mutable default argument
def add_item(item, my_list=[]):
    my_list.append(item)
    return my_list

print(add_item(1))
print(add_item(2))
```

Output:

[1]
[1, 2]

Wait,
WHAT

🤔 What Just Happened?

- 💡 When we call `add_item(1)`, Python creates `my_list` once and stores it.
- 💡 When we call `add_item(2)`, it **doesn't** create a new list; it just modifies the same one! 🤔

🤔 Why It Happened?

- 💡 This happens because **default arguments** in Python are **evaluated only once** when the function is defined, not each time it's called.

🚀 Fix It Like a Pro!

Instead of using a mutable list as a default argument, use **None** and create a new list inside the function:

```
In [50]: def add_item(item, my_list=None):  
         if my_list is None:  
             my_list = [] # Create a new list each time  
         my_list.append(item)  
         return my_list  
  
print(add_item(1))  
print(add_item(2))
```

Output:

[1]
[2]

Finally,
fresh start



🎯 Golden Rule:

❌ Avoid mutable default arguments like lists or dictionaries. ✅ Use `` as a placeholder and initialize inside the function.

💬 Which Python quirks have surprised you the most? Drop a comment! 🔥