Python Programming: Handling Mutable Default Parameters

Computer Science Education

1 Problem Statement

In Python, using a mutable object (like a list or dictionary) as a default parameter can lead to unexpected behavior. This exercise demonstrates how to safely manage state across function calls without using global variables.

2 The Mutable Default Pitfall (Incorrect Approach)

The following code demonstrates a **bad practice**. In Python, default arguments are evaluated only once at the time of function definition, not at each call. This means the same list object is reused every time the function is called.

```
def add_order(order_id, orders=[]):  # BAD PRACTICE
    orders.append(order_id)
    return orders
```

Listing 1: Incorrect Usage of Mutable Default

3 The Correct Pattern (Recommended Approach)

To avoid this pitfall, we use `None` as the default value and initialize the mutable object inside the function body.

3.1 Function Implementation

```
def add_order(order_id, orders=None):
    if orders is None:
        orders = []
    orders.append(order_id)
    return orders
```

Listing 2: Safe Implementation with None

3.2 Sample Execution

The following calls demonstrate how the function correctly manages order history:

```
# First call: Starts a new history
order_history = add_order(101)
print(order_history)      # Output: [101]
```

```
# Second call: Passes the existing history to update it
order_history = add_order(102, order_history)
print(order_history)      # Output: [101, 102]
```

4 Key Features of the Solution

- **Acceptance:** Correcty accepts an `order_id` for processing.
- **Persistence:** Stores order IDs across calls by passing the list back into the function.
- **Safety:** Uses `orders=None` to avoid the shared-state bug common with `[]`.
- **Encapsulation:** No global variables are used; the state is managed through arguments and return values.

5 Conclusion

This pattern is the standard Pythonic way to handle optional mutable arguments. By using `None` and creating the list inside the function, we ensure that every "new" call starts with a fresh empty list while still allowing existing lists to be modified if provided.