Tuple Operations Explained

✅ 1. **Concatenating Tuples**

python Copy Edit

```python
more_tea = ("Herbal", "Earl Grey") all_tea = more_tea + tea_types
```

You're combining two tuples. This produces:

python Copy Edit

```python
('Herbal', 'Earl Grey', 'Black', 'Green', 'Oolong')
```

✅ 2. **Membership Testing with** `in`

python Copy Edit

```python
if "Green" in all_tea: print("I have green tea")
```

This checks if `"Green"` is in the tuple. Correctly prints:

css Copy Edit

```css
I have green tea
```

⚠️ The earlier check:

python Copy Edit

```python
if "Gree" in all_tea:
```

fails because `"Gree"` is not an exact match.

✅ 3. **Counting Items with** `.count()`

```python
more_tea = ("Herbal", "Earl grey", "Herbal") more_tea.count("Herbal") # 2
more_tea.count("Herb") # 0
```

`.count()` checks **exact matches only**. It doesn't match substrings.

✅ 4. Tuple Unpacking

```python
tea_types = ('Black', 'Green', 'Oolong') (black, green, Oolong) = tea_types
```

This is **unpacking**. Each value from the tuple is assigned to a variable:

```python
black -> 'Black' green -> 'Green' Oolong -> 'Oolong'
```

You can now use them individually.

🧠 Bonus Tip: Case Sensitivity

Be careful:

```python
"Earl Grey" ≠ "Earl grey"
```

Python is **case-sensitive**, so `"Earl Grey"` and `"Earl grey"` are different strings.

✅ Summary

Operation	Example	Result
Concatenation	`t1 + t2`	Combines tuples

Operation	Example	Result
Membership test	`"Green" in all_tea`	`True` if exists
Count element	`t.count("Herbal")`	Counts exact matches
Tuple unpacking	`(a, b, c) = tuple`	Assigns values to variables
Case-sensitive check	`"Herb" != "Herbal"`	Must match exactly