Unit 4 Practice Tasks

Shopping Inventory System

4 Separate Files • Work at Your Own Pace • Prepare for the Assessment

Complete these before the real assessment!

The 4 Practice Files

- `task1_calculate_total.py`
- `task2_create_product.py`
- `task3_filter_products.py`
- `task4_inventory_report.py`

Each file is independent — work on them in order!

🎯 What Each Task Practices

Task	File	Concepts
1	`task1_calculate_total`	List operations, slicing, basic aggregation
2	`task2_create_product`	Dictionary creation, `.get()`, safe access
3	`task3_filter_products`	List comprehensions (filter & transform)
4	`task4_inventory_report`	List of dicts, nested data, aggregation

These are the SAME concepts as the real assessment!

🏃 How to Work Through Each File

Step 1: Open the file in your code editor

Step 2: Read the docstring carefully (the big comment at the top)

Step 3: Look at the examples in the docstring

Step 4: Write your code where it says # TODO

Step 5: Run the file to test your solution

Step 6: Fix any failing tests, then move to the next file

📖 Understanding the File Structure

Each file has **3 sections**:

```
"""
SECTION 1: Instructions
- What the task is about
- What concepts you're practicing
"""

def function_name(parameters):
    """
    SECTION 2: Docstring
    - What the function should do
    - What parameters it takes
    - What it should return
    - Examples!
    """
    # TODO: Your code goes here
    pass

if __name__ == "__main__":
    # SECTION 3: Tests
    # Runs when you execute the file
```

📝 Reading the Docstring

The docstring tells you **everything** you need:

```
def create_product(name, price, category="General", in_stock=True):
    """
    Create a product dictionary.

    Args:                                ← What inputs it takes
        name: Product name (required)
        price: Product price (required)
        category: default "General"
        in_stock: default True

    Returns:                             ← What it should return
        dict with keys "name", "price", "category", "in_stock"

    Examples:                            ← What it should look like
        >>> create_product("Apple", 1.50)
        {"name": "Apple", "price": 1.50, "category": "General", "in_stock": True}
    """
```

▶️ Running the Tests

In VS Code: Click the ▶️ Run button or press `F5`

In Terminal:

```
python task1_calculate_total.py
```

What you'll see:

```
================================================
🛒 TASK 1: calculate_total
================================================
❌ Test 1 FAILED: Expected 25.97, got None
❌ Test 2 FAILED: Expected 0, got None
...
```

All ❌ means your code isn't working yet — keep trying!

✅ When You Get It Right

```
================================================
🛒 TASK 1: calculate_total
================================================
✅ Test 1 PASSED: calculate_total([...]) = 25.97
✅ Test 2 PASSED: calculate_total([]) = 0
✅ Test 3 PASSED: calculate_total([9.99]) = 9.99
✅ Test 4 PASSED: First 3 items only
================================================
```

All ✅ means you're ready for the next task!

💡 Tips for Success

Before Coding:

- Read the **entire** docstring
- Look at **ALL** examples
- Notice the **return type** (dict, list, number?)
- Check for **edge cases** (empty list, missing keys)

While Coding:

- Start simple, add complexity
- Use `print()` to debug
- Don't forget `return`!
- Check your spelling

SOS Common Mistakes to Avoid

❌ **Forgetting to handle empty lists**

```
# WRONG
def average(nums):
    return sum(nums)/len(nums)
```

```
# RIGHT
def average(nums):
    if not nums:
        return 0
    return sum(nums)/len(nums)
```

❌ **Using dict["key"] instead of dict.get("key")**

```
# CRASHES if key missing
item["discount"]
```

```
# SAFE — returns default
item.get("discount", 0)
```

🆘 More Common Mistakes

❌ Filter vs Transform confusion

```
# FILTER (if at END) — removes items
[x for x in items if x > 10]

# TRANSFORM (if-else at START) — changes all items
["big" if x > 10 else "small" for x in items]
```

❌ Modifying list while iterating

```
# WRONG
for item in items:
    if bad:
        items.remove(item)
```

```
# RIGHT
result = [item for item in items
          if not bad]
```

📊 Practice → Assessment Comparison

Practice (🛒 Shopping)	Assessment (🎮 Theme TBD)
`calculate_total`	Similar list operations
`create_product`	Similar dict creation
`filter_products`	Similar comprehensions
`inventory_report`	Similar nested data

Same structure, same concepts, different theme!

If you can do the practice, you can do the assessment!

✅ Checklist Before Assessment

Complete each practice task:

- ☐ Task 1: `calculate_total` — all 4 tests pass
- ☐ Task 2: `create_product` — all 4 tests pass
- ☐ Task 3: `filter_products` — all 4 tests pass
- ☐ Task 4: `inventory_report` — all 4 tests pass

Once all 16 tests pass, you're ready! 🎉

🚀 Get Started!

1. Open: task1_calculate_total.py
2. Read the docstring
3. Write your code
4. Run the file
5. Get all ✅, then move to Task 2!

You've got this! 💪