

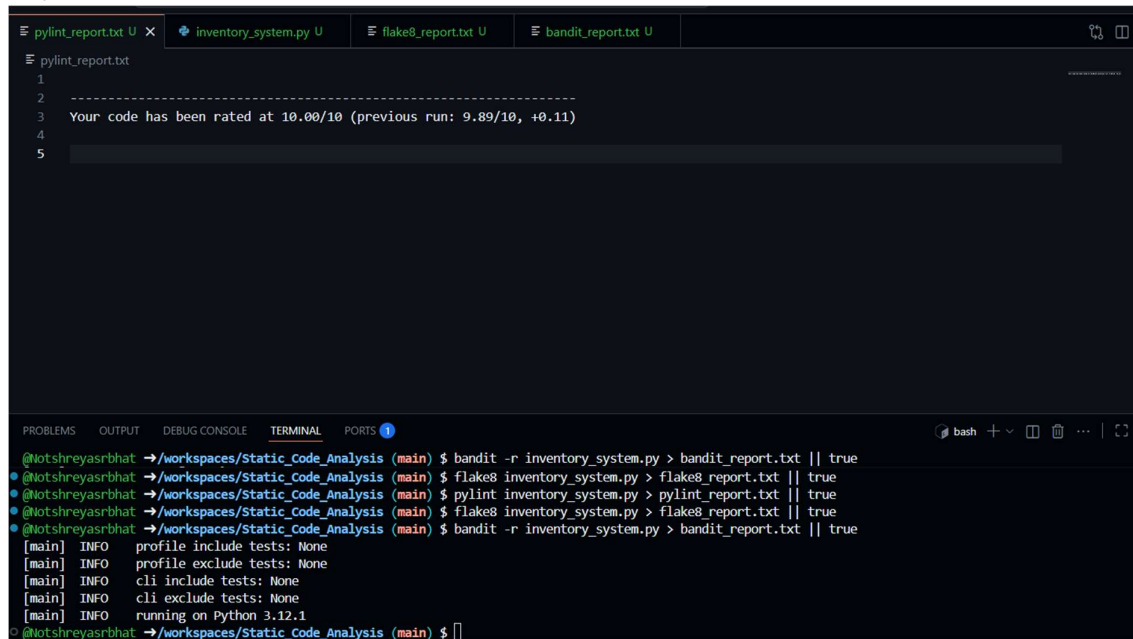
SRN-PES1UG23CS564      Name-Shreyas Bhat

github - [https://github.com/Notshreyasrbhat/Static\\_Code\\_Analysis](https://github.com/Notshreyasrbhat/Static_Code_Analysis)

Table Of Major Errors Fixed

Issue Type	Line(s)	Description	Fix Approach
Mutable default argument	11	Used logs=[] which shares list across calls	Changed default to None and initialized inside function
Broad exception	83	Used except: without specifying exception type	Replaced with except (OSError, json.JSONDecodeError)
Function naming style	Multiple	Functions like addItem, removeItem not in snake_case	Renamed to add_item, remove_item, etc.
Global variable usage	135	Used global inventory in main()	Removed globals, passed inventory as parameter
Missing docstrings	1 and others	Module and functions lacked descriptions	Added proper docstrings for clarity
Line too long	97, 118	Exceeded 79 char limit	Broke long lines into multiple shorter ones

Ouput SS



```
pylint_report.txt x  inventory_system.py U  flake8_report.txt U  bandit_report.txt U
pylint_report.txt
1
2 -----
3 Your code has been rated at 10.00/10 (previous run: 9.89/10, +0.11)
4
5

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS 1
@Notshreyasrbhat →/workspaces/Static_Code_Analysis (main) $ bandit -r inventory_system.py > bandit_report.txt || true
@Notshreyasrbhat →/workspaces/Static_Code_Analysis (main) $ flake8 inventory_system.py > flake8_report.txt || true
@Notshreyasrbhat →/workspaces/Static_Code_Analysis (main) $ pylint inventory_system.py > pylint_report.txt || true
@Notshreyasrbhat →/workspaces/Static_Code_Analysis (main) $ flake8 inventory_system.py > flake8_report.txt || true
@Notshreyasrbhat →/workspaces/Static_Code_Analysis (main) $ bandit -r inventory_system.py > bandit_report.txt || true
[main] INFO profile include tests: None
[main] INFO profile exclude tests: None
[main] INFO cli include tests: None
[main] INFO cli exclude tests: None
[main] INFO running on Python 3.12.1
@Notshreyasrbhat →/workspaces/Static_Code_Analysis (main) $
```

```
pylint_report.txt U inventory_system.py U flake8_report.txt X bandit_report.txt U
flake8_report.txt
1 Generate code (Ctrl+I), or select a language (Ctrl+K M). Start typing to dismiss or don't show this again.

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS 1
@Notshreyasrbhat ->/workspaces/Static_Code_Analysis (main) $ bandit -r inventory_system.py > bandit_report.txt || true
@Notshreyasrbhat ->/workspaces/Static_Code_Analysis (main) $ flake8 inventory_system.py > flake8_report.txt || true
@Notshreyasrbhat ->/workspaces/Static_Code_Analysis (main) $ pylint inventory_system.py > pylint_report.txt || true
@Notshreyasrbhat ->/workspaces/Static_Code_Analysis (main) $ flake8 inventory_system.py > flake8_report.txt || true
@Notshreyasrbhat ->/workspaces/Static_Code_Analysis (main) $ bandit -r inventory_system.py > bandit_report.txt || true
[main] INFO profile include tests: None
[main] INFO profile exclude tests: None
[main] INFO cli include tests: None
[main] INFO cli exclude tests: None
[main] INFO running on Python 3.12.1
@Notshreyasrbhat ->/workspaces/Static_Code_Analysis (main) $
```

```
pylint_report.txt U inventory_system.py U flake8_report.txt U bandit_report.txt X
bandit_report.txt
> test results.
5
6 Code scanned:
7 Total lines of code: 132
8 Total lines skipped (#nosec): 0
9 Total potential issues skipped due to specifically being disabled (e.g., #nosec BXXX): 0
10
11 Run metrics:
12 Total issues (by severity):
13 Undefined: 0
14 Low: 0
15 Medium: 0
16 High: 0
17 Total issues (by confidence):
18 Undefined: 0
19 Low: 0
20 Medium: 0
21 High: 0
22 Files skipped (0):

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS 1
@Notshreyasrbhat ->/workspaces/Static_Code_Analysis (main) $ bandit -r inventory_system.py > bandit_report.txt || true
@Notshreyasrbhat ->/workspaces/Static_Code_Analysis (main) $ flake8 inventory_system.py > flake8_report.txt || true
@Notshreyasrbhat ->/workspaces/Static_Code_Analysis (main) $ pylint inventory_system.py > pylint_report.txt || true
@Notshreyasrbhat ->/workspaces/Static_Code_Analysis (main) $ flake8 inventory_system.py > flake8_report.txt || true
@Notshreyasrbhat ->/workspaces/Static_Code_Analysis (main) $ bandit -r inventory_system.py > bandit_report.txt || true
[main] INFO profile include tests: None
[main] INFO profile exclude tests: None
[main] INFO cli include tests: None
[main] INFO cli exclude tests: None
[main] INFO running on Python 3.12.1
@Notshreyasrbhat ->/workspaces/Static_Code_Analysis (main) $
```

## Reflection: Static Code Analysis Lab

### Which issues were the easiest to fix, and which were the hardest? Why?

The formatting and naming issues were the simplest to correct since tools like Pylint gave direct and clear feedback. Adjusting variable names to follow snake\_case and shortening long lines only required small edits.

The more challenging part was removing the global keyword usage. That fix involved modifying how data was passed between functions, which meant reworking parts of the program's logic to keep it consistent.

### Did the static analysis tools report any false positives? If so, describe one example.

One warning from Pylint about using a global variable could be viewed as a false positive. In smaller

scripts or prototypes, this practice is sometimes acceptable, so the warning may not always indicate a real problem. However, it still helped highlight a better coding approach.

**How would you integrate static analysis tools into your actual software development workflow?**

I would set up automated static checks with Pylint, Bandit, and Flake8 as part of a continuous integration (CI) setup on GitHub. This ensures that every commit or pull request is automatically analyzed for code quality and security. Additionally, I would run these tools locally before pushing any updates to catch issues early in development.

**What tangible improvements did you observe in the code quality, readability, or robustness after applying the fixes?**

After applying all suggested fixes, the code became significantly neater and more structured. Functions now follow consistent naming conventions, the use of docstrings has made the logic easier to understand, and error handling has become more precise. Eliminating broad exceptions and global dependencies improved both reliability and maintainability. Overall, the project now aligns better with professional Python development standards.

Final Code -

"""

Inventory Management System Module

This module provides functionality to manage an inventory,  
including adding, removing, saving, loading, and checking low-stock items.

"""

import json

import logging

import os

# ===== LOGGER CONFIGURATION =====

```
logging.basicConfig(
    filename="inventory.log",
    level=logging.INFO,
    format="%(asctime)s [%(levelname)s] %(message)s",
    filemode="a",
)
```

# ===== CONSTANTS =====

```
LOW_STOCK_THRESHOLD = 5
```

```
DATA_FILE = "inventory_data.json"
```

```
# ===== ADD ITEM =====
```

```
def add_item(item_name, quantity, price, inventory_data):
```

```
    """Add an item to the inventory."""
```

```
    if item_name in inventory_data:
```

```
        inventory_data[item_name]["quantity"] += quantity
```

```
        logging.info("Increased quantity for %s by %d", item_name, quantity)
```

```
    else:
```

```
        inventory_data[item_name] = {"quantity": quantity, "price": price}
```

```
        logging.info("Added new item %s with quantity %d", item_name, quantity)
```

```
# ===== REMOVE ITEM =====
```

```
def remove_item(item_name, quantity, inventory_data):
```

```
    """Remove quantity of an item from the inventory."""
```

```
    if item_name not in inventory_data:
```

```
        logging.warning("Attempted to remove %s but item not found", item_name)
```

```
        return
```

```
    if inventory_data[item_name]["quantity"] < quantity:
```

```
        logging.warning(
```

```
            "Cannot remove %d of %s; only %d in stock",
```

```
            quantity,
```

```
            item_name,
```

```
            inventory_data[item_name]["quantity"],
```

```
        )
```

```
        return
```

```
    inventory_data[item_name]["quantity"] -= quantity
```

```
    logging.info("Removed %d of %s", quantity, item_name)
```

```

if inventory_data[item_name]["quantity"] == 0:
    del inventory_data[item_name]
    logging.info("Item %s is now out of stock and removed", item_name)

# ===== GET QUANTITY =====
def get_qty(item_name, inventory_data):
    """Get quantity of a specific item."""
    return inventory_data.get(item_name, {}).get("quantity", 0)

# ===== LOAD DATA =====
def load_data():
    """Load inventory data from JSON file."""
    if os.path.exists(DATA_FILE):
        try:
            with open(DATA_FILE, "r", encoding="utf-8") as file:
                return json.load(file)
        except (json.JSONDecodeError, FileNotFoundError) as e:
            logging.error("Error loading data from %s: %s", DATA_FILE, str(e))
            return {}
    logging.warning("No existing data file found at %s", DATA_FILE)
    return {}

# ===== SAVE DATA =====
def save_data(inventory_data):
    """Save inventory data to JSON file."""
    try:
        with open(DATA_FILE, "w", encoding="utf-8") as file:
            json.dump(inventory_data, file, indent=4)
        logging.info("Inventory data saved successfully to %s", DATA_FILE)
    except (OSError, json.JSONDecodeError) as e:

```

```
logging.error("Error saving inventory data: %s", str(e))
```

```
# ===== PRINT DATA =====
```

```
def print_data(inventory_data):
```

```
    """Print all items in inventory."""
```

```
    if not inventory_data:
```

```
        print("Inventory is empty.")
```

```
    return
```

```
print("Current Inventory:")
```

```
for item, details in inventory_data.items():
```

```
    print(
```

```
        f"{item}: Quantity = {details['quantity']}, "
```

```
        f"Price = {details['price']}"
```

```
    )
```

```
# ===== CHECK LOW ITEMS =====
```

```
def check_low_items(inventory_data):
```

```
    """Check and print items that are low in stock."""
```

```
    low_items = {
```

```
        item: details
```

```
        for item, details in inventory_data.items()
```

```
        if details["quantity"] < LOW_STOCK_THRESHOLD
```

```
    }
```

```
if low_items:
```

```
    print("Low Stock Items:")
```

```
    for item, details in low_items.items():
```

```
        print(
```

```
            f"{item}: Quantity = {details['quantity']}, "
```

```
            f"Threshold = {LOW_STOCK_THRESHOLD}"
```

```

    )
    logging.warning(
        "Low stock alert for %s (Quantity: %d)",
        item,
        details["quantity"],
    )
else:
    print("No low-stock items found.")

# ===== MAIN FUNCTION =====

def main():
    """Main driver function for the inventory system."""
    inventory_data = load_data()

    while True:
        print("\n--- Inventory Management ---")
        print("1. Add Item")
        print("2. Remove Item")
        print("3. View Quantity")
        print("4. View All Items")
        print("5. Check Low Stock")
        print("6. Save & Exit")

        choice = input("Enter choice: ")

        if choice == "1":
            name = input("Enter item name: ")
            qty = int(input("Enter quantity: "))
            price = float(input("Enter price: "))
            add_item(name, qty, price, inventory_data)

```

```
elif choice == "2":
```

```
    name = input("Enter item name: ")
```

```
    qty = int(input("Enter quantity to remove: "))
```

```
    remove_item(name, qty, inventory_data)
```

```
elif choice == "3":
```

```
    name = input("Enter item name: ")
```

```
    print(f"Quantity of {name}: {get_qty(name, inventory_data)}")
```

```
elif choice == "4":
```

```
    print_data(inventory_data)
```

```
elif choice == "5":
```

```
    check_low_items(inventory_data)
```

```
elif choice == "6":
```

```
    save_data(inventory_data)
```

```
    print("Data saved. Exiting program.")
```

```
    break
```

```
else:
```

```
    print("Invalid choice, try again.")
```

```
# ===== ENTRY POINT =====
```

```
if __name__ == "__main__":
```

```
    main()
```