Coding Standard

1. Consistency

- Maintain consistent coding style across the entire project.
- Use uniform **naming conventions**, **indentation**, and **formatting** throughout all files to enhance readability and maintainability.

---

2. Naming Conventions

2.1 Variables

- Use **snake case** for variable names.
- Variables should be descriptive and indicate their purpose.

Examples:

```
cart_items = []
sort_order = "asc"
is_sorted = True
```

2.2 Constants

- Constants must use **UPPERCASE_SNAKE_CASE** and be defined at the module level.

Examples:

```
MAX_ITEMS = 100
DEFAULT_SORT_ORDER = "newest"
```

2.3 Functions

- Use **snake_case** for all function names.
- Name functions clearly based on their behavior.

Examples:

```
def add_to_cart(book_id):
def sort_books_by_price(order):
```

2.4 Classes

- Use **CamelCase** with the first letter capitalized.
- Class names must represent what the class is doing or managing.

Examples:

```python
class CartManager:
class SortUtility:
```

2.5 Packages and Modules

- Use **short, lowercase names** for packages and modules.
- Avoid underscores in package names for OS compatibility.

Example:

```
bookstore.cart, bookstore.sorting
```

---

3. Comments and Documentation

- Use **docstrings** for all modules, functions, and classes.
- Use **inline comments** for complex logic or important notes.

Example:

```python
def calculate_total(cart_items):
    """Calculate the total price of items in the cart."""
    return sum(item.price for item in cart_items)
```

---

4. Formatting and Indentation

- Use **4 spaces** per indentation level.
- Limit lines to **79 characters** maximum for readability.

---

5. Error Handling

- Handle exceptions using **try-except blocks**.
- Catch specific exceptions before general ones.

Example:

```
try:
    process_payment()
except ValueError:
    print("Invalid value!")
except Exception as e:
    print("Unexpected error:", e)
```

---

6. Import Formatting

- Use separate lines for each import.
- Order:
    - Standard libraries
    - Third-party libraries
    - Local modules

Example:

```
import os
import sys

import requests

from bookstore.cart import add_to_cart
```

---

7. URL Formatting (if applicable for APIs or web views)

- Use **lowercase letters** and separate words using **hyphens** or **underscores**.

Example:

```
/sort-by-date
/add-to-cart
```

---

8. Template Style

- Maintain proper **HTML structure** with consistent indentation and readability.

Example:

```
<div class="book-item">
    <h2>{{ book.title }}</h2>
    <button>Add to Cart</button>
</div>
```

---

9. Code Readability

- Break large tasks into smaller, reusable functions.
- Use meaningful names and avoid abbreviations.

---

10. Code Reusability

- Reuse logic using utility functions or classes.
- Avoid duplicating logic across different modules.

---

11. Testing and Quality Assurance

- Use `unittest` or `pytest` for validating `add_to_cart()` and `sort_books()` features.

Example:

```
import unittest

class TestCart(unittest.TestCase):
    def test_add_to_cart(self):
        self.assertTrue(add_to_cart(101))
```

---

12. Security

- Sanitize and validate all user inputs.
- Prevent sorting manipulation or injection in URLs and forms.