Coding Standards For Python

---

1. Naming Conventions

1.1 Variable Names

- All variable names should be in lowercase..For multiple words, use underscores in between the words **(`lower_case_with_underscores`)**.
- Variable names should be meaningful – so anyone reading your code can understand what it does easily..
- For Boolean variables the variable names should start with **`is`**, **`has`**, or **`can`** to indicate status.

Examples:

```
product_price = 1500
user_email = "customer@123.com"
is_order_confirmed = True
has_discount = False
```

1.2 Constants

- Constants should be written in **all capital letters** with underscores between words
- .Use them for values that remain the constant throughout the project.
- Constants should usually be defined at the top of the module.

Examples:

```
MAX_CART_ITEMS = 20
FREE_SHIPPING_THRESHOLD = 2000
DEFAULT_CURRENCY = "BDT"
```

1.3 Function and Method Names

- Function and method names should be lowercase, for more than one word use underscores in between .
- Function names should be **descriptive** and usually start with a **verb**.

Examples:

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

1.4 Package and Module Names

- Packages and modules should have **short and simple names**.
- Use underscores only when necessary for readability.

Examples:

```
orders
payments
user_profiles
product_catalog
shopping_cart
```

1.5 Class Names

- Classes should be named using **PascalCase** convention (each word starts with a capital letter).
- Class names should represent entities or objects in the system.

Examples:

```
class Product:
    def __init__(self, name, price, stock):
        self.name = name
        self.price = price
        self.stock = stock


class ShoppingCart:
    def __init__(self, user):
        self.user = user
        self.items = []
```

1.6 Exception Names

- Exception classes are also a type of class and should follow **PascalCase** and end with "Error."
- Use meaningful names that describe the type of error.

Examples:

```
class PaymentError(Exception):
    """Raised when a payment fails."""
    pass
```

```
class OutOfStockError(Exception):
    """Raised when a product is not available."""
    pass
```

---

2. Indentation & Code Structure

- Always use **4 spaces** for indentation
- Lines should be under **79 characters** so the code stays easy to read.
- Use blank lines between sections or functions to make the code look clean.

Example:

```
if is_order_confirmed and has_payment_completed:
    process_shipment(order_id)
else:
    notify_admin("Order processing issue detected.")
```

---

3. General Practices

- Avoid writing the same code— use functions or modules so you can reuse them.
- Keep each function short and focused on doing just one thing.
- Use try and except to handle errors safely.
- Make sure everyone follows the same naming style and code structure to keep things consistent.
- Write **unit tests** for all major features like login, checkout, and order processing.

Example:

```
try:
    number = int(input("Enter a number: "))
    print("You entered:", number)
except ValueError:
    print("That's not a valid number!")
```
