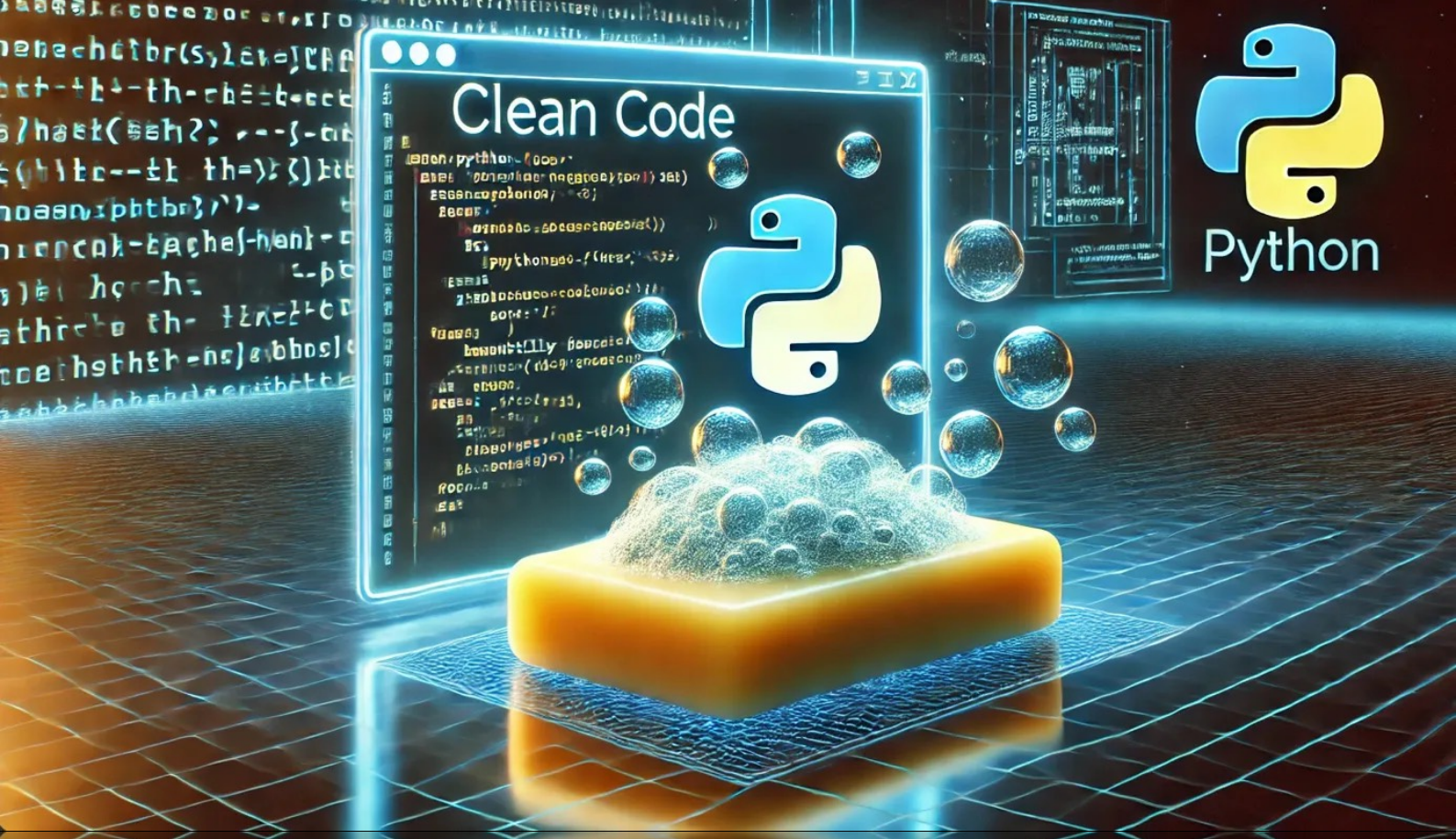
Ahmed Abdulwahid

How to Write Clean Code in Python

#DataGenius



Why Clean Code Matters? 🤔

Have you ever looked at your old code and thought, "Who wrote this mess?!" 😵‍💫 Writing clean code is not just a luxury, it's a necessity — especially in Python, where simplicity is king. 👑

Clean Code Benefits:

✅ *Easier to Read – Future-you (and teammates) will thank you!* 🙌

✅ *Easier to Debug – Bugs hide in messy code.* 🐞🚫

✅ *Easier to Maintain – Change one part without breaking everything.* 🛠️

✅ *Faster Collaboration – Work smoothly with other devs.* 🤝

🔷 1. Meaningful Variable and Function Names

❌ *Bad Example:*

```python
def calc(a, b):
    return a * b / 2
```

What is calc? What are a and b? We have no idea! 😑

✅ *Clean Version:*

```python
def calculate_triangle_area(base: float, height: float) -> float:
    return base * height / 2
```

Why? Now, the function clearly describes what it does! 🎯

🔷 2. Keep Functions Short & Focused 🧠

❌ *Bad Example:*

```
def process_data(data):
    cleaned = [d.strip().lower() for d in data]
    sorted_data = sorted(cleaned)
    filtered = [d for d in sorted_data if len(d) > 3]
    return {i: v for i, v in enumerate(filtered)}
```

This function does too much at once! 😵

✅ Clean Version:

```
def clean_data(data):
    return [d.strip().lower() for d in data]

def sort_data(data):
    return sorted(data)

def filter_short_words(data, min_length=3):
    return [d for d in data if len(d) > min_length]

def process_data(data):
    cleaned = clean_data(data)
    sorted_data = sort_data(cleaned)
    return filter_short_words(sorted_data)
```

Each function does one thing well. 🔥

🔷 3. Avoid Magic Numbers & Strings 🎩✨

❌ *Bad Example:*

```
if x > 86400:
    print("Too many seconds!")
```

Why 86400? What does it mean? 🤔

✅ *Clean Version:*

```
SECONDS_IN_A_DAY = 86400
if x > SECONDS_IN_A_DAY:
    print("Too many seconds!")
```

💡

Now it's self-explanatory!

💡

🔷 4. Use List Comprehensions Wisely 🧹

❌ *Bad Example:*

```
squared_numbers = []
for num in numbers:
    squared_numbers.append(num ** 2)
```

✅ *Clean Version:*

```
squared_numbers = [num ** 2 for num in numbers]
```

Why? More concise, Pythonic, and readable! 🐍🔥

🔷 5. Follow Python Naming Conventions 

Using proper naming conventions makes code more readable, maintainable, and professional. 🏆 Here are some standard Python naming conventions:

- **Variables & functions:** Use snake_case (e.g., user_name, calculate_total_price()).
- **Constants:** Use UPPER_CASE (e.g., MAX_USERS, SECONDS_IN_A_MINUTE).
- **Classes:** Use PascalCase (e.g., DataProcessor, UserManager).
- **Modules:** Use lowercase_with_underscores.py (e.g., data_processor.py).
- **Packages:** Use lowercase (e.g., utils, models).

More Best Practices:

✅ Avoid single-letter variables (except for loop counters like `i` and `j`).

✅ Use descriptive names, but keep them concise.

✅ Prefix private methods/variables with an underscore (`_my_private_var`).

✅ Avoid using built-in keywords for variable names (e.g., `list`, `str`).
Following these conventions ensures consistency and clarity in your codebase! 🚀

🔷 6. Write Docstrings 📜

❌ *Bad Example:*

```
def add(x, y):
    return x + y
```

✅ *Clean Version:*

```
def add(x: int, y: int) -> int:
    """Returns the sum of two integers."""
    return x + y
```

Docstrings help everyone understand your function's purpose! 📚

🔷 7. Use Type Hints for Clarity 🎯

Python allows optional type hints to make code clearer! 👀

```python
def greet(name: str) -> str:
    return f"Hello, {name}!"
```

Now we know `name` is a string, and the function returns a string. 🏆

🔷 8. Handle Errors Gracefully 🛡️

❌ *Bad Example:*

```
age = int(input("Enter your age: "))
print(f"You are {age} years old.")
```

✅ *Clean Version:*

```
def get_age():
    try:
        return int(input("Enter your age: "))
    except ValueError:
        print("Invalid input! Please enter a number.")
        return None
```

Now it's error-proof! 💪

🔷 9. Don't Repeat Yourself (DRY Principle) 🌊

❌ Bad Example:

```
def calc_circle_area(radius):
    return 3.14159 * radius * radius

def calc_sphere_volume(radius):
    return (4/3) * 3.14159 * radius ** 3
```

We're repeating 3.14159! 😡

✅ Clean Version:

```
import math

def calc_circle_area(radius):
    return math.pi * radius * radius

def calc_sphere_volume(radius):
    return (4/3) * math.pi * radius ** 3
```

Conclusion: Keep It Clean! 🧼✨

Writing clean code is an investment. 🚀

✅ *Make it readable* 🧐
✅ *Make it maintainable* 🛠️
✅ *Make it understandable* 💡

Your future self will thank you! 🙏

Repost it

Thank you