Clean Code

Clean Code in Python by Mariano Anaya presents a set of strategies aimed at improving code quality.

Follow the SOLID Principles

These principles guide object-oriented design:

- Single Responsibility Principle
- Open/Closed Principle
- Liskov Substitution Principle
- Interface Segregation Principle
- Dependency Inversion Principle

Write Pythonic Code

Use idiomatic Python instead of imitating patterns from other languages:

- Prefer list comprehensions and generator expressions
- Use unpacking and multiple assignment
- Use context managers

Refactor Routinely

Break down and simplify code:

- Split large functions into smaller ones
- Remove duplicate code
- Replace imperative code with declarative constructs

Use Type Hints and Static Analysis

- Add type annotations to functions and class members

Write Tests

- Use unittest, pytest, or doctest
- Cover edge cases and failure modes
- Write small, isolated tests

Design with Clean Architecture

- Decouple business logic from infrastructure
- Structure code using layers
- Follow Dependency Rule

Apply Design Patterns

Adapt patterns to Python's strengths:

- Strategy, Factory, Adapter and Command patterns
- Use first class functions and closures
- Minimise inheritance, prefer composition

Control Dependencies

- Use dependency injection instead of hard coded dependencies
- Pass interfaces or callable objects

Use Meaningful Names and Structure

- Descriptive function, variable and class names
- Group related functions into classes or modules
- Avoid deep nesting and long files

Measure and Improve

- Use code metrics: cyclomatic complexity, lines of code, coverage
- Profile performance if needed
- Continuously review and revise

My Code

This is a section of code taken from BlockWorks, a program written during Launching into Computer Science.

```
def splash_screen():
        """
        Displays the splash screen when the program starts.
        Shows a welcome message and some loading information.
        """
        clear_console()
        print("=" * 50)
        print("\033[1;37;41m              Welcome to BlockWorks               \033[0m")
        print("=" * 50)
        print("\033[1;36m  A Program for Managing Blocks and Components  \033[0m")
        print("\n\033[1;33m  Loading...\033[0m")
        print("=" * 50)
        input("Press Enter to continue to the main menu...")


def display_menu():
        """
        Displays the main menu to the user, where they can choose
        what action to perform in the program.
        """
        print("=" * 30)
        print("\033[1;37;41m    BlockWorks Main Menu      \033[0m")
        print("=" * 30)

        print("\033[36m1. View All Blocks\033[0m")
        print("\033[36m2. Search Blocks\033[0m")
        print("\033[36m3. Add Blocks\033[0m")
        print("\033[36m4. Delete Blocks\033[0m")
        print("\033[36m5. Sort Blocks\033[0m")
        print("\033[1;36m6. Exit\033[0m")

        print("=" * 30)
```

Improved Code

Separate Concerns	Use Constants	Avoid Repetition
Encapsulate ANSI Styling	Avoid Hardcoded Numbers	Make It Testable
Improve Function Statements		

```
# Constants and Helpers
LINE_SEPARATOR = "=" * 50
SHORT_SEPARATOR = "=" * 30

ANSI = {
    "cyan": "\033[36m",
    "yellow": "\033[1;33m",
    "white_red": "\033[1;37;41m",
    "cyan_bold": "\033[1;36m",
    "reset": "\033[0m"
}

MENU_OPTIONS = [
    "View All Blocks",
    "Search Blocks",
    "Add Blocks",
    "Delete Blocks",
    "Sort Blocks",
    "Exit"
]


def styled(text, colour):
    return f"{ANSI[colour]}{text}{ANSI['reset']}"


def clear_console():
    import os
    os.system('cls' if os.name == 'nt' else 'clear')


def splash_screen(wait_for_input=input):
    """
    Display welcome screen with formatted title and loading message.
    """
    clear_console()
    print(LINE_SEPARATOR)
    print(styled("              Welcome to BlockWorks               ", "white_red"))
    print(LINE_SEPARATOR)
    print(styled("  A Program for Managing Blocks and Components  ", "cyan_bold"))
    print()
    print(styled("  Loading...", "yellow"))
    print(LINE_SEPARATOR)
    wait_for_input("Press Enter to continue to the main menu...")


def display_menu():
    """
    Display the main menu and available actions.
    """
    print(SHORT_SEPARATOR)
    print(styled("    BlockWorks Main Menu      ", "white_red"))
    print(SHORT_SEPARATOR)
    for i, option in enumerate(MENU_OPTIONS, 1):
        print(styled(f"{i}. {option}", "cyan"))
    print(SHORT_SEPARATOR)
```