Week 2: **Coding Standards, SOLID Principles, and Code Smells**

***Session 1***: Theory

1. **Coding Standards and Conventions**

- Coding standards improve code quality, readability, and maintainability.

- Best practices for Python, JavaScript, and MySQL:

**1.1 Python (Django) Coding Standards**

- Follow PEP 8 for python styling:

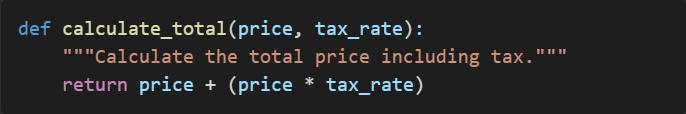
PEP 8 specifically provides guidelines and best practices for writing clean, readable, and consistent Python code. It covers various aspects of coding style, including indentation, naming conventions, line length, and more.

- Use snake\_case for variables and functions.

- Keep indentation at 4 spaces.

- Use docstrings for documentation. (swagger)

Example:



**1.2 JavaScript (Node.js, React) Coding Standards**

- Use camelCase for variables and functions.

- Follow ESLint and Prettier for formatting.

- Prefer const and let over var.

A black background with blue text

Description automatically generated

**1.3 MySQL Best Practices**

- Use snake\_case for table and column names.

- Define primary and foreign keys properly.

- Optimize queries using indexes.

- Example:

A black screen with white text

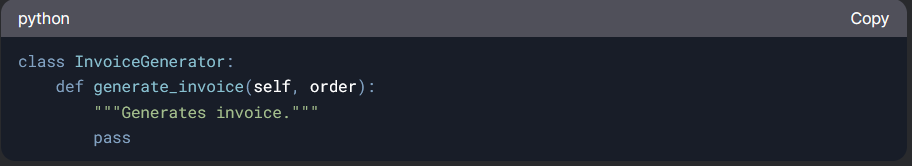
Description automatically generated

2. **Introduction to SOLID Principles, DRY, and KISS**

***2.1 SOLID Principles***

- *Single Responsibility Principle (SRP): A class should have only one reason to change.*

Example:



- *Open/Closed Principle (OCP): Software entities should be open for extension but closed for modification.*

Example:

A screen shot of a computer code

Description automatically generated

- *Liskov Substitution Principle (LSP): Derived classes should be substitutable for their base classes.*

Example:

A black background with green dots

Description automatically generated with medium confidence

- *Interface Segregation Principle (ISP): Clients should not depend on interfaces they do not use.*

Example:



- *Dependency Inversion Principle (DIP): High-level modules should not depend on low-level modules; both should depend on abstractions.*

Example:

A screenshot of a computer screen

Description automatically generated

***2.2 DRY (Don’t Repeat Yourself)***

- Avoid duplication by reusing functions and classes.

- Example:

A computer screen shot of text

Description automatically generated

***2.3 KISS (Keep It Simple, Stupid)***

- Write simple and understandable code.

- Example:

A black screen with white text

Description automatically generated

**3. Code Smells: Identifying and Addressing Them**

***3.1 What Are Code Smells?***

- Indicators of poor design or implementation that make code harder to maintain.

***3.2 Common Code Smells and Fixes***

- ***Long Functions***: Break into smaller, reusable functions.

- ***Duplicate Code***: Use functions or classes to reuse code.

- ***God Object***: Split responsibilities into smaller classes.

- ***Hardcoded Values***: Use constants or configuration files.

***3.3 Example Fix for a Long Function (Python)***

- Before:



- After:

A screen shot of a computer code

Description automatically generated

**Conclusion**

- Coding standards improve readability and maintainability.

- SOLID principles ensure flexible and scalable software design.

- DRY and KISS help avoid complexity and redundancy.

- Code smells should be actively identified and fixed to maintain high-quality code.