Software Refactoring Overview

The term **refactoring** refers to a process of simplifying an application's source code with the goal of removing duplication. During the refactoring process, developers strive to adhere to the "DRY" principle, which stands for "Don't Repeat Yourself".

Benefits of Refactoring

Business benefits of refactoring are mainly in terms of cost and time savings, and quality assurance/control. Main benefits include:

- Improved Maintainability
- Improved Readability
- Improved Quality
- Decreased Complexity

Maintainability

Refactoring can reduce the need to maintain the same or similar code in multiple different places. This makes the software easier and more efficient to maintain.

Quality

Refactoring decreases the probability of future errors relating to a developer forgetting to update one of the duplicative code locations.

Extension

Refactored code is more efficient to extend in the sense that a developer need only make a change in one place instead of multiple similar or duplicative places.

When to Refactor

Even though refactoring is beneficial, sometimes refactoring too early in the development process is not. Your professor recommends you focus on writing code to achieve desired functionality before turning your focus to refactoring the code you have written. Maybe make comments for yourself during the development process such as `TODO: refactor`. However, in general, if you find yourself writing the same or similar code two or three times during the development process, consider refactoring at that time. Or if you need to make an update to your code and find the need to make the same or similar update in multiple places, consider refactoring at that time.

Examples

Each of these examples of refactoring allow a program to use consistent print formatting without duplication, and allow a developer to change that formatting in a single place and have it apply to all of its uses.

Removing Duplication

Before refactoring:

```
print("--------------------")
print("MY MESSAGE")
print("--------------------")

print("--------------------")
print("MY MESSAGE")
print("--------------------")

print("--------------------")
print("MY MESSAGE")
print("--------------------")
```

After refactoring:

```
def print_message():
    print("--------------------")
    print("MY MESSAGE")
    print("--------------------")

print_message()

print_message()

print_message()
```

Simplifying Terms

Before refactoring:

```
print("--------------------")
print("FIRST HEADING")
print("--------------------")

print("--------------------")
print("SECOND HEADING")
print("--------------------")

print("--------------------")
print("THIRD HEADING")
print("--------------------")
```

After refactoring:

```
def print_message(message):
    print("--------------------")
    print(message)
    print("--------------------")

print_message("FIRST HEADING")

print_message("SECOND HEADING")

print_message("THIRD HEADING")
```