**Key Modifications and Improvements**

Based on the initial observations, it seems the major change is refactoring the code into a class-based structure. Here are the steps to provide a detailed comparison:

1. **Encapsulation**:
   * **Original**: Functions and the task list are defined in the global scope.
   * **Updated**: A TodoList class is defined, encapsulating the task list and its related methods.
2. **Code Organization**:
   * **Original**: Code is organized in a procedural manner.
   * **Updated**: Code is organized using Object-Oriented Programming (OOP) principles.
3. **Reusability and Maintainability**:
   * **Original**: Global variables and functions may lead to namespace pollution and less modular code.
   * **Updated**: The class structure promotes reusability and better maintainability by keeping the task list and its operations self-contained.
4. **Efficiency**:
   * **Original**: While procedural code can be efficient, it may become cumbersome as the application grows.
   * **Updated**: The OOP approach can improve efficiency in managing larger codebases, as classes and methods are easier to manage, extend, and debug.

**Detailed Comparison**

Let's compare the complete contents of both files to highlight the specific changes.

**todo\_list.py**

python

Copy code

# Define an empty list to store tasks

tasks = []

# Function to display the to-do list

def display\_tasks():

if not tasks:

print("Your to-do list is empty.")

else:

print("To-Do List:")

for i, task in enumerate(tasks, start=1):

status = "Done" if task["completed"] else "Not Done"

print(f"{i}. {task['task']} ({status})")

# Function to add a task to the to-do list

def add\_task(task\_name):

task = {"task": task\_name, "completed": False}

tasks.append(task)

print(f"Task '{task\_name}' added to your to-do list.")

# Function to mark a task as completed

def mark\_completed(task\_number):

if 1 <= task\_number <= len(tasks):

tasks[task\_number - 1]["completed"] = True

print(f"Task {task\_number} marked as completed.")

else:

print("Invalid task number. Please enter a valid task number.")

# Function to remove a task from the to-do list

def remove\_task(task\_number):

if 1 <= task\_number <= len(tasks):

removed\_task = tasks.pop(task\_number - 1)

print(f"Task '{removed\_task['task']}' removed from your to-do list.")

else:

print("Invalid task number. Please enter a valid task number.")

**Todo\_listUpdated.py**

python

Copy code

class TodoList:

def \_\_init\_\_(self):

self.tasks = []

def display\_tasks(self):

if not self.tasks:

print("Your to-do list is empty.")

else:

print("To-Do List:")

for i, task in enumerate(self.tasks, start=1):

status = "Done" if task["completed"] else "Not Done"

print(f"{i}. {task['task']} ({status})")

def add\_task(self, task\_name):

task = {"task": task\_name, "completed": False}

self.tasks.append(task)

print(f"Task '{task\_name}' added to your to-do list.")

def mark\_completed(self, task\_number):

if 1 <= task\_number <= len(self.tasks):

self.tasks[task\_number - 1]["completed"] = True

print(f"Task {task\_number} marked as completed.")

else:

print("Invalid task number. Please enter a valid task number.")

def remove\_task(self, task\_number):

if 1 <= task\_number <= len(self.tasks):

removed\_task = self.tasks.pop(task\_number - 1)

print(f"Task '{removed\_task['task']}' removed from your to-do list.")

else:

print("Invalid task number. Please enter a valid task number.")

**Summary of Improvements**

* **Encapsulation**: All task-related data and operations are encapsulated within the TodoList class.
* **Namespace Management**: Reduces the risk of global namespace pollution.
* **Modularity**: Improves code modularity, making it easier to manage and extend.
* **Reusability**: The class can be instantiated multiple times, allowing for multiple independent to-do lists.

**Conclusion**

The updated Todo\_listUpdated.py is more efficient and maintainable due to its object-oriented structure. This approach is particularly beneficial for larger projects and for scenarios where multiple instances of the to-do list may be required.