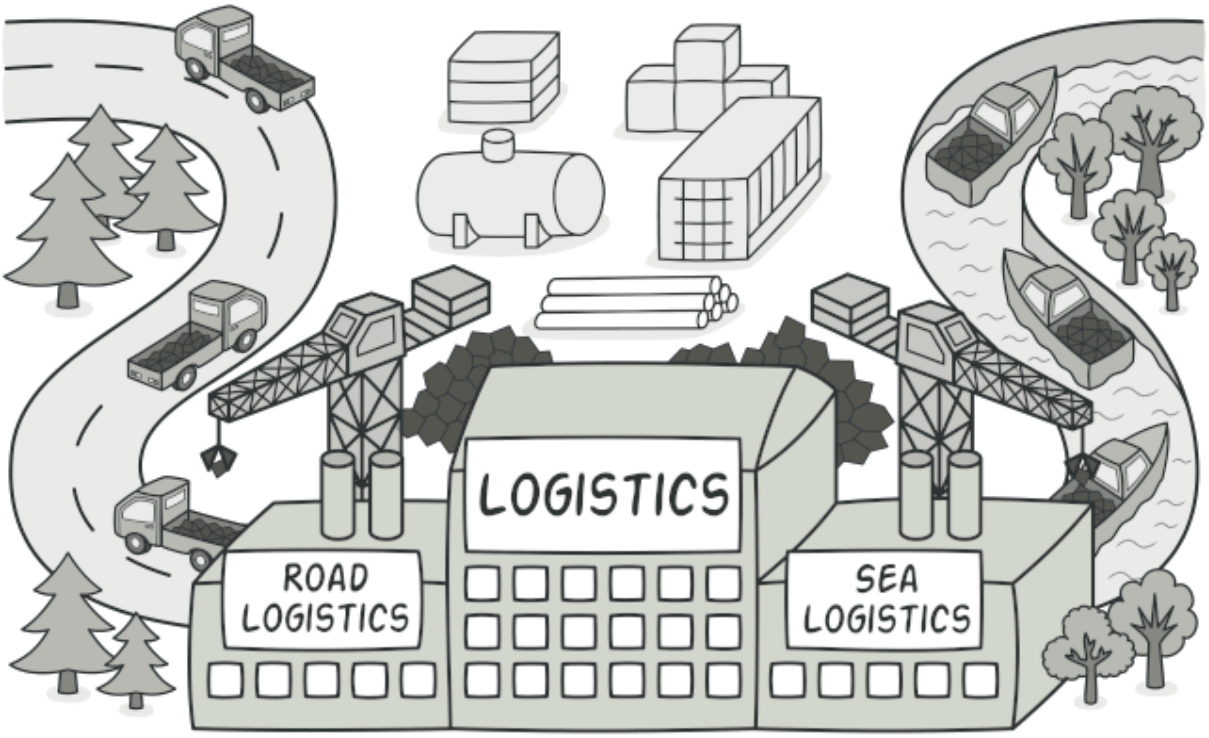
Implementación de Factory con python



Autora: Rashi Chugani Narwani

Módulos y Funciones

1. **`product.py`**:
   - **Vehicle (Clase Abstracta)**: Define `deliver` como método abstracto.
   - **Car y Truck (Clases Concretas)**: Implementan `deliver`.
2. **`creator.py`**:
   - **VehicleFactory (Clase Abstracta)**: Define el método abstracto `get_vehicle`.
3. **`concrete_factory.py`**:
   - **CarFactory y TruckFactory (Clases Concretas)**: Implementan `get_vehicle` para crear `Car` y `Truck`.
4. **`client_code.py`**:
   - **cliente_vehicle_code (Función)**: Usa una fábrica para crear y operar un vehículo.
5. **`main.py`**:
   - Punto de entrada del programa: Instancia fábricas y llama a `cliente_vehicle_code`.

Relaciones y Flujo de Trabajo

1. **Instanciación de Fábricas**: `main.py` crea instancias de `CarFactory` y `TruckFactory`.
2. **Creación de Vehículos**: `cliente_vehicle_code` llama a `get_vehicle` en la fábrica concreta.
3. **Entrega de Vehículos**: `deliver` en `Car` o `Truck` es llamado y se imprime el resultado.

Resumen del Patrón Factory

- **Desacoplamiento**: El cliente no necesita conocer las clases concretas de los vehículos.
- **Flexibilidad**: Se pueden añadir nuevos tipos de vehículos y fábricas sin modificar el código del cliente.
- **Extensibilidad**: Nuevas fábricas concretas pueden ser añadidas fácilmente.