El Problema:

Un equipo de ingeniería está diseñando una pieza mecánica. Los objetivos son:

1. Maximizar la resistencia a la fatiga (f1): Queremos una pieza muy duradera.
2. Minimizar el costo de producción (f2): Queremos una pieza barata de fabricar.

Variables de Decisión (x):

Para simplificar, definiremos dos variables de decisión que influencian ambos objetivos:

- x1: Índice de Material (ej. contenido de aleación, calidad del material). Un valor más alto significa un material más resistente pero también más caro.
    - Rango: [0.5, 2.0]
- x2: Espesor del Componente (ej. en cm). Un mayor espesor generalmente aumenta la resistencia pero también el costo debido a más material y/o tiempo de procesamiento.
    - Rango: [1.0, 5.0]

Funciones Objetivo (Simplificadas):

Dado que pymoo por defecto *minimiza* todos los objetivos, para "maximizar la resistencia a la fatiga", minimizaremos su negativo.

- f1_obj = - (100 * x1 * x2):
    - La resistencia a la fatiga aumenta linealmente con el índice de material (x1) y el espesor (x2). Un coeficiente de 100 lo hace significativo. Al negarlo, para pymoo es una minimización.
- f2_obj = 50 * x1**2 + 5 * x2:
    - El costo de producción aumenta más rápidamente con la calidad del material (x1^2) y linealmente con el espesor (x2).

Enfoque de la Solución con pymoo:

1. Definir el Problema (Problem Class): Implementaremos la clase MechanicalDesignProblem que extiende pymoo.core.problem.Problem. Aquí se definen el número de variables, objetivos, restricciones y las funciones objetivo.

2. Seleccionar el Algoritmo: Usaremos NSGA-II (pymoo.algorithms.moo.nsga2.NSGA2), uno de los algoritmos de optimización multiobjetivo más populares y efectivos.
3. Definir la Terminación: Estableceremos el número de generaciones para que el algoritmo busque soluciones.
4. Ejecutar la Optimización: Usaremos pymoo.optimize.minimize.
5. Visualizar el Frente de Pareto: Usaremos pymoo.visualization.scatter para graficar los resultados en el espacio de objetivos y mostrar el frente de Pareto. También graficaremos en el espacio de decisión para ver las combinaciones de x1 y x2 que llevan a estos puntos.
6. Interpretar los Resultados: Explicaremos qué representa el frente de Pareto y cómo se pueden tomar decisiones a partir de él.

Explicación del Código y Gráficas:

1. Clase MechanicalDesignProblem:
    - n_var=2: Tenemos dos variables de decisión (x1, x2).
    - n_obj=2: Tenemos dos objetivos (Resistencia a la Fatiga y Costo).
    - xl y xu: Definen los límites inferior y superior para x1 y x2, respectivamente.
    - _evaluate(self, X, out, ...): Este es el corazón del problema. Recibe una matriz X donde cada fila es un conjunto de variables de decisión (un diseño de pieza). Para cada fila, calcula la Resistencia a la Fatiga y el Costo de Producción.
        - Fíjate que f1 es fatigue_strength *negativo*. Esto es porque pymoo está diseñado para *minimizar* objetivos. Para "maximizar" algo, se minimiza su negativo.
        - Los resultados se guardan en out["F"] como una matriz de NumPy donde cada fila son los valores de los objetivos para un diseño específico.
2. Algoritmo (NSGA2):
    - NSGA2 es un algoritmo genético multiobjetivo muy robusto.
    - pop_size=100: Cada "generación" del algoritmo trabajará con una población de 100 diseños de piezas.

3. Terminación:
    - get_termination("n_gen", 200): El algoritmo se ejecutará durante 200 generaciones. Con cada generación, la población evoluciona para acercarse más al frente de Pareto.
4. minimize:
    - Orquesta todo el proceso: toma el problema, el algoritmo y el criterio de terminación, y ejecuta la simulación de optimización.
    - seed=1: Esto es importante para la reproducibilidad. Si ejecutas el código con la misma semilla, obtendrás los mismos resultados exactos.
5. Resultados y Visualización (Scatter):
    - res.X: Contiene las combinaciones *óptimas* de x1 (Índice de Material) y x2 (Espesor del Componente) que se encontraron.
    - res.F: Contiene los valores de Resistencia a la Fatiga (negativos) y Costo correspondientes a los res.X.
    - ¡Punto Clave para la Gráfica!: Antes de graficar, convertimos F[:, 0] de -(Resistencia a la Fatiga) de nuevo a Resistencia a la Fatiga positiva. Esto hace que la gráfica sea intuitiva: valores más grandes a la derecha significan mayor resistencia.
    - pymoo.visualization.scatter es una herramienta conveniente para plotear los resultados. Se utiliza junto con matplotlib.pyplot para añadir etiquetas, cuadrículas y títulos personalizados, haciendo el gráfico más informativo.
    - Primera Gráfica (Frente de Pareto):
        - Muestra el "Frente de Pareto" en el espacio de objetivos. El eje X es la Resistencia a la Fatiga y el eje Y es el Costo de Producción.
        - Puedes ver una curva de puntos. Cada punto en esta curva es un diseño "óptimo" donde no puedes mejorar la resistencia sin aumentar el costo, ni puedes reducir el costo sin disminuir la resistencia. Es la frontera de soluciones no dominadas.
    - Segunda Gráfica (Espacio de Decisión):

        - Muestra las combinaciones de x1 (Índice de Material) y x2 (Espesor) que dan como resultado los puntos en el Frente de Pareto. Esto te dice qué parámetros de diseño específicos debes usar para lograr ciertos niveles de rendimiento y costo.

6. Interpretación:
    - Se muestra una tabla con ejemplos de diseños en el frente de Pareto.
    - Extremo izquierdo-inferior de la curva de Pareto: Representa diseños con baja Resistencia a la Fatiga pero también bajo Costo de Producción (ej. usar un material menos denso y un espesor pequeño).
    - Extremo derecho-superior de la curva de Pareto: Representa diseños con alta Resistencia a la Fatiga pero alto Costo de Producción (ej. usar un material de alta calidad y un espesor mayor).
    - Puntos intermedios: Ofrecen un equilibrio. Un ingeniero puede elegir un punto en esta curva basándose en las necesidades y restricciones específicas del proyecto. Si la durabilidad es más importante que el costo, se moverá a la derecha. Si el costo es lo principal, se moverá a la izquierda.

Este ejemplo proporciona una base para entender cómo funciona la optimización multiobjetivo con pymoo y cómo interpretar sus resultados para la toma de decisiones de diseño.