
****DESCARGAMOS LA BBIBLOTECA Y EL SOLVER****

```
!pip install gurobipy
from gurobipy import Model, GRB
import pandas as pd
```

****Ejercicios****

**1. Logística de Distribución en ElectroNova S.A.**

ElectroNova S.A. es una empresa colombiana que fabrica equipos electrónicos en sus dos plantas industriales: una ubicada en Medellín y otra en Bucaramanga. Estos productos deben distribuirse diariamente a tres centros logísticos ubicados en Bogotá, Cali y Barranquilla, desde donde serán entregados a clientes finales.

Para reducir costos, la dirección logística de ElectroNova desea diseñar un plan de transporte que minimice los gastos totales de envío, garantizando que cada centro reciba exactamente la cantidad requerida y que no se exceda la capacidad diaria de envío de cada planta.

| Plantas | Ciudad | Capacidad diaria |

|---|---|---|

| Planta 1 | Medellín | 300 und |

| Planta 2 | Bucaramanga | 500 und |

* Demanda en centros de distribución:

| Centro de Distribución # | Ciudad | Demanda diaria |

|---|---|---|

| CD1 | Bogotá | 250 und |

| CD2 | Cali | 300 und |

| CD3 | Barranquilla | 250 und |

* Costos de transporte (en miles de pesos COP por unidad):

| Desde / Hacia | CD1 (Bogotá) | CD2 (Cali) | CD3 (Barranquilla) |

|---|---|---|---|

| Planta 1 | 18 | 22 | 30 |

| Planta 2 | 25 | 20 | 16 |

OBJETIVO

ElectroNova desea minimizar el costo total de transporte, asegurando que:

* Cada centro de distribución reciba su demanda completa.

* Las plantas no envíen más unidades de las que pueden producir por día.

Crear el modelo

```
model_electro = Model("Ejercicio_Logistica_ElectroNova")
```

Conjuntos

```
plantas = ['Planta1', 'Planta2']
```

```
centros = ['CD1', 'CD2', 'CD3']
```

Parametros

```
Oferta = [300, 500]
```

```
Demanda = [250, 300, 250]
```

```
Costo = [[18, 22, 30], [25, 20, 16]]
```

Variables

```
x = model_electro.addVars(plantas, centros, vtype=GRB.CONTINUOUS, name="x", lb=0)
```

Función objetivo

```
model_electro.setObjective(sum(Costo[i][j] * x[plantas[i], centros[j]] for i in range(2) for j  
in range(3)), GRB.MINIMIZE)
```

Restricciones de demanda

for j in centros:

```
    model_electro.addConstr(sum(x[i,j] for i in plantas) == Demanda[centros.index(j)],  
f"Demanda_{j}")
```

```
# Restricciones de oferta
```

```
for i in plantas:
```

```
    model_electro.addConstr(sum(x[i,j] for j in centros) <= Oferta[plantas.index(i)],  
f"Oferta_{i}")
```

```
# Resolver
```

```
model_electro.optimize()
```

```
# Resultados
```

```
if model_electro.status == GRB.OPTIMAL:
```

```
    print("--- Plan de transporte óptimo ---")
```

```
    for i in plantas:
```

```
        for j in centros:
```

```
            if x[i,j].X > 0:
```

```
                print(f"Enviar {x[i,j].X} unidades de {i} a {j}")
```

```
print(f"\nCosto total de transporte: ${model_electro.ObjVal:,.1f}")
```

```
### **2. Producción óptima de jugos naturales**
```

Una empresa produce 3 tipos de jugos naturales: naranja (i=1), mango (i=2) y fresa (i=3). Cada jugo requiere una cantidad específica de fruta natural, agua purificada y botellas, todos recursos limitados. Cada tipo de jugo aporta una utilidad por litro vendida.

| Jugo | Utilidad | Fruta | Agua | Botellas |

|---|---|---|---|

| Naranja | 1.20 | 0.5 | 0.4 | 1 |

| Mango | 1.50 | 0.6 | 0.3 | 1 |

| Fresa | 1.80 | 0.8 | 0.5 | 1 |

Recursos disponibles:

* Fruta natural: 500 kg

* Agua purificada: 300 litros

* Botellas disponibles: 800 unidades

* Cada tipo de jugo debe producirse en al menos 100 litros y como máximo 300 litros.

OBJETIVO

Determinar cuántos litros de cada jugo producir para maximizar la utilidad total, respetando los recursos y los límites de producción.

Crear el modelo

model_jugo = Model("Ejercicio_Produccion_Jugo")

```
# Parametros
```

```
jugos = ['Naranja', 'Mango', 'Fresa']
```

```
utilidad = [1.20, 1.50, 1.80]
```

```
fruta = [0.5, 0.6, 0.8]
```

```
agua = [0.4, 0.3, 0.5]
```

```
botellas = [1, 1, 1]
```

```
min_prod = [100, 100, 100]
```

```
max_prod = [300, 300, 300]
```

```
# Limites
```

```
lim_fruta = 500
```

```
lim_agua = 300
```

```
lim_botellas = 800
```

```
# Variables de decisión
```

```
x = model_jugo.addVars(3, vtype=GRB.CONTINUOUS, name="x", lb=min_prod,  
ub=max_prod)
```

```
# Función objetivo
```

```
model_jugo.setObjective(sum(utilidad[i] * x[i] for i in range(3)), GRB.MAXIMIZE)
```

```
# Restricciones
```

```
model_jugo.addConstr(sum(fruta[i] * x[i] for i in range(3)) <= lim_fruta, "Fruta")
```

```
model_jugo.addConstr(sum(agua[i] * x[i] for i in range(3)) <= lim_agua, "Agua")
```

```
model_jugo.addConstr(sum(botellas[i] * x[i] for i in range(3)) <= lim_botellas, "Botellas")
```

```
# Resolver
```

```
model_jugo.optimize()
```

```
# Resultados
```

```
if model_jugo.status == GRB.OPTIMAL:
```

```
    print("--- Solución óptima ---")
```

```
    for i in range(3):
```

```
        print(f"Litros de jugo {jugos[i]}: {x[i].X:.2f}")
```

```
    print(f"Utilidad total: ${model_jugo.ObjVal:.2f}")
```

```
else:
```

```
    print("No se encontró solución óptima.")
```

```
# Parametros
```

```
I = [1, 2, 3]
```

```
utilidad = {1: 1.20, 2: 1.50, 3: 1.80}
```

```
fruta = {1: 0.5, 2: 0.6, 3: 0.8}
```

```
agua = {1: 0.4, 2: 0.3, 3: 0.5}
```

```
botellas = {1: 1, 2: 1, 3: 1}
```

```
min_prod = {i: 100 for i in I}
```

```
max_prod = {i: 300 for i in I}
```

```
# Limite
```

```
lim_fruta = 500
```

```
lim_agua = 300
```

```
lim_botellas = 800
```

```
# Crear modelo
```

```
model = Model("Produccion_Jugos")
```

```
# Variables de decisión
```

```
x = model.addVars(I, vtype=GRB.CONTINUOUS, name="x", lb=min_prod, ub=max_prod)
```

```
# Función objetivo
```

```
model.setObjective(sum(utilidad[i] * x[i] for i in I), GRB.MAXIMIZE)
```

```
# Restricciones de recursos
```

```
model.addConstr(sum(fruta[i] * x[i] for i in I) <= lim_fruta, name="Fruta")
```

```
model.addConstr(sum(agua[i] * x[i] for i in I) <= lim_agua, name="Agua")
```

```
model.addConstr(sum(botellas[i] * x[i] for i in I) <= lim_botellas, name="Botellas")
```

```
# Resolver
```

```
model.optimize()
```

```
# Mostrar resultados
```

```
if model.status == GRB.OPTIMAL:
```

```
    print("\n--- Solución óptima ---")
```

```
    for i in I:
```

```
        print(f"Litros de jugo {i}: {x[i].X:.2f}")
```

```
    print(f"Utilidad total: ${model.ObjVal:.2f}")
```

*** **3. Mezcla de suplementos alimenticios**

Una empresa elabora 4 tipos de suplementos en polvo que contienen diferentes proporciones de tres ingredientes esenciales: proteína vegetal, fibra y antioxidantes. Cada suplemento se vende en bolsas de 1 kg y tiene una utilidad por unidad. Todos los productos se fabrican en la misma planta y comparten recursos.

Ingredientes disponibles:

* Proteína vegetal: 600 kg

* Fibra: 400 kg

* Antioxidantes: 300 kg

* Presupuesto máximo: \$1200

Información de productos:

Productos	Utilidad	Proteína	Fibra	Antioxidantes	Costo por unidad	Producción mínima	producción máxima
-----------	----------	----------	-------	---------------	------------------	-------------------	-------------------

---	---	---	---	---	---	---	---
-----	-----	-----	-----	-----	-----	-----	-----

Energy Fit	4.5	0.6	0.2	0.1	2.5	50	200
------------	-----	-----	-----	-----	-----	----	-----

Fiber Plus	3.2	0.2	0.5	0.1	2.0	40	150
------------	-----	-----	-----	-----	-----	----	-----

Antiox Defense	5.0	0.3	0.1	0.4	3.0	30	100
----------------	-----	-----	-----	-----	-----	----	-----

| Protein Boost | 4.0 | 0.7 | 0.1 | 0.1 | 2.8 | 60 | 180 |

OBJETIVO:

Determinar cuántas unidades producir de cada suplemento para maximizar la utilidad total, cumpliendo con las restricciones de disponibilidad de ingredientes y presupuesto.

Crear modelo

model_suplementos = Model("Ejercicio_Mezcla_Suplementos")

Parametros

Productos = ['Energy Fit', 'Fiber Plus', 'Antiox Defense', 'Protein Boost']

utilidad = [4.5, 3.2, 5.0, 4.0]

proteina = [0.6, 0.2, 0.3, 0.7]

fibra = [0.2, 0.5, 0.1, 0.1]

antiox = [0.1, 0.1, 0.4, 0.1]

costo = [2.5, 2.0, 3.0, 2.8]

min_prod = [50, 40, 30, 60]

max_prod = [200, 150, 100, 180]

Limite

max_proteina = 600

max_fibra = 400

max_antiox = 300

max_presupuesto = 1200

```
# Variables de decisión
```

```
x = model_suplementos.addVars(Productos, vtype=GRB.CONTINUOUS, name="x",  
lb=min_prod, ub=max_prod)
```

```
# Función objetivo
```

```
model_suplementos.setObjective(sum(utilidad[i] * x[Productos[i]] for i in range(4)),  
GRB.MAXIMIZE)
```

```
# Restricciones
```

```
model_suplementos.addConstr(sum(proteina[i] * x[Productos[i]] for i in range(4)) <=  
max_proteina, "Proteina")
```

```
model_suplementos.addConstr(sum(fibra[i] * x[Productos[i]] for i in range(4)) <=  
max_fibra, "Fibra")
```

```
model_suplementos.addConstr(sum(antiox[i] * x[Productos[i]] for i in range(4)) <=  
max_antiox, "Antiox")
```

```
model_suplementos.addConstr(sum(costo[i] * x[Productos[i]] for i in range(4)) <=  
max_presupuesto, "Presupuesto")
```

```
# Resolver
```

```
model_suplementos.optimize()
```

```
# Resultados
```

```
if model_suplementos.status == GRB.OPTIMAL:
```

```
    print("--- Solución óptima ---")
```

```
    for i in range(4):
```

```
        print(f"Unidades de {Productos[i]}: {x[Productos[i]].X:.2f}")
```

```
    print(f"Utilidad total: ${model_suplementos.ObjVal:.2f}")
```

else:

print("No se encontró solución óptima.")

4. Problema de Producción de Mezcla de Jugos Naturales HIT

La empresa productora de jugos naturales HIT ubicada en Bucaramanga desea planificar su producción diaria para maximizar sus ingresos. La empresa produce 10 tipos de jugos, identificados como

i

\in

{

1

,

2

,

.

.

.

,

10

}

. Cada jugo requiere dos tipos de insumos, fruta y agua purificada, cuyas disponibilidades diarias máximas son 500 litros para la fruta y 800 litros para el agua purificada. La cantidad de insumos necesarios para producir una unidad de cada jugo se presenta en la siguiente tabla.

Productos	Agua Purificada (L)	Concentrado de Fruta (L)	Precio de Venta (\$)
Naranja	5	6	4000
Mango	4	7	4500
Fresa	6	5	4200
Mora	3	8	4300
Piña	4	6	4100
Guanábana	5	5	4400
Maracuyá	6	4	4600
Limón	4	7	4700
Manzana	5	6	4800
Durazno	3	8	4900

La empresa ha establecido que la cantidad máxima de unidades de cualquier jugo que se puede producir por día es 50 unidades. ¿Cuántas unidades de cada jugo debe producir la empresa para maximizar el ingreso total?

```
# Crear el modelo
```

```
model_hit = Model("Ejercicio_Mezcla_Productos_Hit")
```

```
# Parámetros
```

```
jugos = ['Naranja', 'Mango', 'Fresa', 'Mora', 'Piña', 'Guanábana', 'Maracuyá', 'Limón', 'Manzana', 'Durazno']
```

```
agua = [5, 4, 6, 3, 4, 5, 6, 4, 5, 3]
```

```
fruta = [6, 7, 5, 8, 6, 5, 4, 7, 6, 8]
```

```
precio = [4000, 4500, 4200, 4300, 4100, 4400, 4600, 4700, 4800, 4900]
```

```
# Limites
```

```
max_fruta = 500
```

```
max_agua = 800
```

```
max_unidades = 50
```

```
# Variables de decisión
```

```
x = model_hit.addVars(jugos, vtype=GRB.INTEGER, name="x", lb=0, ub=max_unidades)
```

```
# Función objetivo
```

```
model_hit.setObjective(sum(precio[i] * x[jugos[i]] for i in range(10)), GRB.MAXIMIZE)
```

```
# Restricciones
```

```
model_hit.addConstr(sum(agua[i] * x[jugos[i]] for i in range(10)) <= max_agua, "Agua")
```

```
model_hit.addConstr(sum(fruta[i] * x[jugos[i]] for i in range(10)) <= max_fruta, "Fruta")
```

```
# Resolver
```

```
model_hit.optimize()
```

```
# Resultados
```

```
if model_hit.status == GRB.OPTIMAL:
```

```
    print("--- Producción óptima de jugos ---")
```

```
    for j in jugos:
```

```
if x[j].X> 0:

    print(f"jugo{j}: {x[j].X} unidades")

print(f"\nIngreso total maximo: ${model_hit.ObjVal:,.1f}")

else:

    print("No se encontró solución óptima.")
```

****OTRA FORMA DE REALIZAR****

```
# Crear el modelo
```

```
model_hit = Model("Ejercicio_Mezcla_Productos_Hit")
```

```
# Parámetros
```

```
l = list(range(1, 10))
```

```
jugos = ['Naranja', 'Mango', 'Fresa', 'Mora', 'Piña', 'Guanábana', 'Maracuyá', 'Limón',  
'Manzana', 'Durazno']
```

```
agua = [5, 4, 6, 3, 4, 5, 6, 4, 5, 3]
```

```
fruta = [6, 7, 5, 8, 6, 5, 4, 7, 6, 8]
```

```
precio = [4000, 4500, 4200, 4300, 4100, 4400, 4600, 4700, 4800, 4900]
```

```
# Limites
```

```
max_fruta = 500
```

```
max_agua = 800
```

```
max_unidades = 50
```

```
# Variables de decisión
```

```
x = model_hit.addVars(jugos, vtype=GRB.INTEGER, name="x", lb=0, ub=max_unidades)
```

```
# Función objetivo
```

```
model_hit.setObjective(sum(precio[i] * x[jugos[i]] for i in I), GRB.MAXIMIZE)
```

```
# Restricciones
```

```
model_hit.addConstr(sum(agua[i] * x[jugos[i]] for i in I) <= max_agua, "Agua")
```

```
model_hit.addConstr(sum(fruta[i] * x[jugos[i]] for i in I) <= max_fruta, "Fruta")
```

```
# Resolver
```

```
model_hit.optimize()
```

```
# Resultados
```

```
if model_hit.status == GRB.OPTIMAL:
```

```
    print("--- Producción óptima de jugos ---")
```

```
    for j in jugos:
```

```
        if x[j].X > 0:
```

```
            print(f"jugo{j}: {x[j].X} unidades")
```

```
    print(f"\nIngreso total maximo: ${model_hit.ObjVal:,.1f}")
```

```
else:
```

```
    print("No se encontró solución óptima.")
```

```
### **5. Producción de jugos Valle**
```

La empresa productora de jugos naturales Valle elabora diferentes tipos de jugos utilizando tres ingredientes principales: fruta, agua y edulcorante. La empresa tiene tres plantas de producción ubicadas en diferentes regiones de Colombia y desea optimizar la cantidad de cada jugo producido en cada planta para maximizar sus ingresos.

Cada planta (

p

) puede producir diferentes tipos de jugos (

j

), pero tiene limitaciones en la disponibilidad de ingredientes y en la capacidad de producción. La cantidad de cada ingrediente requerida por litro de jugo varía según el tipo de jugo. La disponibilidad de ingredientes en cada planta también está limitada por día.

La tabla siguiente muestra la cantidad de ingredientes necesarios por litro de cada jugo y la disponibilidad máxima en cada planta:

Ingrediente	Jugo 1		Jugo 2		Jugo 3		Disponibilidad Planta 1	
Disponibilidad Planta 2		Disponibilidad Planta 3						
--- --- --- --- --- --- ---								
Fruta (kg)	2	3	1	100	150	120		
Agua (L)	1	2	2	80	100	90		
Edulcorante (g)	50	30	40	10,000	12,000	9,000		

Cada planta tiene una capacidad máxima de producción diaria en litros:

- * Planta 1: 50 litros
- * Planta 2: 70 litros
- * Planta 3: 60 litros

El precio de venta por litro de jugo es:

- * Jugo 1: 5000

- * Jugo 2: 6000

- * Jugo 3: 5500

¿Cuántos litros de cada jugo debe producir cada planta para maximizar los ingresos, respetando las restricciones de disponibilidad de ingredientes y capacidad de producción?

Crear el modelo

```
model_valle = Model("Ejercicio_Mezcla_Productos_Valle")
```

Índices

```
plantas = [1, 2, 3]
```

```
jugos = [1, 2, 3]
```

Datos

```
precio = {1: 5000, 2: 6000, 3: 5500}
```

```
fruta = {1: 2, 2: 3, 3: 1}
```

```
agua = {1: 1, 2: 2, 3: 2}
```

```
edulcorante = {1: 50, 2: 30, 3: 40}
```

Disponibilidades por planta

```
disp_fruta = {1: 100, 2: 150, 3: 120}
```

```
disp_agua = {1: 80, 2: 100, 3: 90}
```

```
disp_edulcorante = {1: 10000, 2: 12000, 3: 9000}
```

```
capacidad = {1: 50, 2: 70, 3: 60}
```

```
# Variables de decisión
```

```
x = model_valle.addVars(plantas, jugos, vtype=GRB.CONTINUOUS, name="x", lb=0)
```

```
# Función objetivo
```

```
model_valle.setObjective(sum(precio[j] * x[i,j] for i in plantas for j in jugos),  
GRB.MAXIMIZE)
```

```
# # Restricciones por planta
```

```
for i in plantas:
```

```
    model_valle.addConstr(sum(fruta[j] * x[i,j] for j in jugos) <= disp_fruta[i], f"Fruta_{i}")
```

```
    model_valle.addConstr(sum(agua[j] * x[i,j] for j in jugos) <= disp_agua[i], f"Agua_{i}")
```

```
    model_valle.addConstr(sum(edulcorante[j] * x[i,j] for j in jugos) <= disp_edulcorante[i],  
f"Edulcorante_{i}")
```

```
    model_valle.addConstr(sum(x[i,j] for j in jugos) <= capacidad[i], f"Capacidad_{i}")
```

```
# Resolver
```

```
model_valle.optimize()
```

```
# Resultados
```

```
if model_valle.status == GRB.OPTIMAL:
```

```
    print("\n--- Producción óptima por planta y jugo ---")
```

```
    for i in plantas:
```

```
        for j in jugos:
```

```
            litros = x[i,j].X
```

```
if litros > 0:

    print(f"Planta {i}, Jugo {j}: {litros:.2f} litros")

    print(f"\nIngresos máximos: ${model_valle.ObjVal:,.2f}")

else:

    print("No se encontró solución óptima.")
```

6. Producción bebidas saludables FrutiMix

La empresa FrutiMix S.A.S., dedicada a la producción de bebidas saludables, desea optimizar su plan de producción semanal de 8 tipos de jugos funcionales. Cada jugo requiere una combinación de tres recursos: agua purificada, concentrado de fruta, y un nuevo insumo llamado extracto funcional (como cúrcuma, jengibre o moringa).

Cada jugo tiene una demanda máxima semanal de 70 unidades y una ganancia unitaria determinada. Además, algunos jugos deben cumplir con restricciones adicionales (por ejemplo, no producir más de 50 unidades de jugos energéticos: Limón, Jengibre y Moringa combinados).

Las disponibilidades semanales son:

* Agua purificada: 1200 litros

* Concentrado de fruta: 900 litros

* Extracto funcional: 300 litros

La empresa desea saber cuántas unidades de cada jugo debe producir para maximizar la ganancia, respetando las restricciones operativas.

Producto	Agua (L)	Fruta (L)	Extracto (L)	Precio de Venta	Costo
Total por unidad	Ganancia (\$)				

--- --- --- --- --- --- ---

Naranja	4	5	0		5000	2200	2800	
---------	---	---	---	--	------	------	------	--

Mango	3		6		0		5200	2400	2800	
-------	---	--	---	--	---	--	------	------	------	--

Limón	4	4	2		5300	2700	2600	
-------	---	---	---	--	------	------	------	--

Guayaba	5		5		1		5400	2500	2900	
---------	---	--	---	--	---	--	------	------	------	--

Fresa	3		7		1		5600	2600	3000	
-------	---	--	---	--	---	--	------	------	------	--

Zanahoria	2		6		1		5500	2550	2950	
-----------	---	--	---	--	---	--	------	------	------	--

Jengibre	4		3		2		5800	2900	2900	
----------	---	--	---	--	---	--	------	------	------	--

Moringa	5		4		3		6000	3000	3000	
---------	---	--	---	--	---	--	------	------	------	--

Crear el modelo

```
model_frutimix = Model("Ejercicio_Mezcla_Productos_FrutiMix")
```

Parametros

```
jugos = ['Naranja', 'Mango', 'Limón', 'Guayaba', 'Fresa', 'Zanahoria', 'Jengibre', 'Moringa']
```

```
agua = [4, 3, 4, 5, 3, 2, 4, 5]
```

```
fruta = [5, 6, 4, 5, 7, 6, 3, 4]
```

```
extracto = [0, 0, 2, 1, 1, 1, 2, 3]
```

```
ganancia = [2800, 2800, 2600, 2900, 3000, 2950, 2900, 3000]
```

```
costo = [2200, 2400, 2700, 2500, 2600, 2550, 2900, 3000]
```

```
precio = [5000, 5200, 5300, 5400, 5600, 5500, 5800, 6000]
```

```
max_agua = 1200
```

```
max_fruta = 900
```

```
max_extracto = 300
```

```
max_prod = 70
```

```
# Variables de decisión
```

```
x = model_frutimix.addVars(jugos, vtype=GRB.CONTINUOUS, name="x", lb=0,  
ub=max_prod)
```

```
# Función objetivo
```

```
model_frutimix.setObjective(sum(ganancia[i] * x[jugos[i]] for i in range(8)),  
GRB.MAXIMIZE)
```

```
# model.setObjective(sum(ganancia[i] * x[jugos[i]] for i in range(len(jugos))),  
GRB.MAXIMIZE)
```

```
# Restricciones
```

```
model_frutimix.addConstr(sum(agua[i] * x[jugos[i]] for i in range(8)) <= max_agua, "Agua")
```

```
model_frutimix.addConstr(sum(fruta[i] * x[jugos[i]] for i in range(8)) <= max_fruta, name =  
"Fruta")
```

```
model_frutimix.addConstr(sum(extracto[i] * x[jugos[i]] for i in range(8)) <= max_extracto,  
name = "Extracto")
```

```
# Restricción adicional
```

```
model_frutimix.addConstr(x['Limón'] + x['Jengibre'] + x['Moringa'] <= 50, name =  
"Energeticos")
```

```

# Resolver

model_frutimix.optimize()

# Resultados

if model_frutimix.status == GRB.OPTIMAL:

    print("--- Producción óptima de jugos FrutiMix ---")

    for j in jugos:

        if x[j].X > 0:

            print(f"Jugo {j}: {x[j].X:.2f} unidades")

    print(f"\nGanancia total: ${model_frutimix.ObjVal:,.2f}")

else:

    print("No se encontró solución óptima.")

```

*** **7. Producción jugos naturales FrutiLots S.A.S**

La empresa FrutiLots S.A.S produce jugos naturales en dos plantas de fabricación, ubicadas en Cali y Medellín. Existen tres tipos de jugo: Naranja, Mango y Fresa, cada uno con requerimientos específicos de materia prima y una demanda máxima en el mercado. Cada planta tiene una capacidad máxima de producción y un costo de operación por tonelada procesada. El objetivo es maximizar el ingreso total de la empresa, cumpliendo con las restricciones de disponibilidad de materia prima y capacidad de cada planta.

Jugo	Ingrediente A (kg/ton)		Ingrediente B (kg/ton)		Demanda Máxima (ton)
	Precio de Venta (\$/ton)				
---	---	---	---	---	---
Naranja	2	3	50	5000	

Mango	4	2	40	6000	
Fresa	3	5	30	7000	

Se debe tener en cuenta la información de las plantas:

Planta	Capacidad Máxima (ton)	Costo Operación (\$/ton)
--- --- ---		
Cali	60 1000	1 100 150 120
Medellín	50 1200 2 80	100 90

Ingrediente	Disponibilidad Máxima (kg)
--- ---	
A	400
B	500

Crear el modelo

```
model_frutilot = Model("Ejercicio_Mezcla_Productos_FrutiLots")
```

Índices

```
plantas = ['Cali', 'Medellin']
```

```
jugos = ['Naranja', 'Mango', 'Fresa']
```

Parámetros

```
precio = {'Naranja': 5000, 'Mango': 6000, 'Fresa': 7000}
```

```
costo_operacion = {'Cali': 1000, 'Medellin': 1200}
```

```
utilidad = {(p, j): precio[j] - costo_operacion[p] for p in plantas for j in jugos}
```

```
ingrediente_A = {'Naranja': 2, 'Mango': 4, 'Fresa': 3}
ingrediente_B = {'Naranja': 3, 'Mango': 2, 'Fresa': 5}
demanda_max = {'Naranja': 50, 'Mango': 40, 'Fresa': 30}
capacidad = {'Cali': 60, "Medellin": 50}
```

```
disp_A = 400
```

```
disp_B = 500
```

```
# Variables
```

```
x = model_frutilot.addVars(plantas, jugos, vtype=GRB.CONTINUOUS, name="x", lb=0)
```

```
# Objetivo
```

```
model_frutilot.setObjective(sum(utilidad[p, j] * x[p, j] for p in plantas for j in jugos), GRB.MAXIMIZE)
```

```
# Restricciones
```

```
for j in jugos:
```

```
    model_frutilot.addConstr(sum(x[p, j] for p in plantas) <= demanda_max[j],
    name=f"Demanda_{j}")
```

```
for p in plantas:
```

```
    model_frutilot.addConstr(sum(x[p, j] for j in jugos) <= capacidad[p],
    name=f"Capacidad_{p}")
```

```
model_frutilot.addConstr(sum(ingrediente_A[j] * x[p, j] for p in plantas for j in jugos) <=
disp_A, name="IngredienteA")
```

```
model_frutilot.addConstr(sum(ingrediente_B[j] * x[p, j] for p in plantas for j in jugos) <=
disp_B, name="IngredienteB")
```

```

# Resolver modelo

model_frutilotos.optimize()

# Resultados

if model_frutilotos.status == GRB.OPTIMAL:

    print("--- Producción óptima con nombres explícitos ---")

    for p in plantas:

        for j in jugos:

            val = x[p, j].X

            if val > 0:

                print(f"Planta {p} debe producir {val:.2f} toneladas de Jugo {j}")

    print(f"\nUtilidad total máxima: ${model_frutilotos.ObjVal:,.2f}")

else:

    print("No se encontró solución óptima.")

```

****OTRA FORMA DE HACERLO****

```

# Parámetros

plantas = ['Cali', 'Medellin']

jugos = ['Naranja', 'Mango', 'Fresa']

precio = [5000, 6000, 7000]          # $/ton
costo = [1000, 1200]                # $/ton por planta
utilidad = [[precio[j] - costo[p] for j in range(3)] for p in range(2)]

ingrediente_A = [2, 4, 3]           # kg/ton

```

ingrediente_B = [3, 2, 5] # kg/ton

demanda_max = [50, 40, 30] # toneladas

capacidad = [60, 50] # toneladas por planta

disp_A = 400 # kg totales

disp_B = 500 # kg totales

Crear modelo

model = Model("FrutiLots")

Variables x[p][j]: toneladas del jugo j en planta p

x = {}

for p in range(2):

 for j in range(3):

 x[p, j] = model.addVar(vtype=GRB.CONTINUOUS, name=f"x_{{plantas[p]}}_{{jugos[j]}}",
lb=0)

Objetivo: maximizar utilidad total

model.setObjective(

 sum(utilidad[p][j] * x[p, j] for p in range(2) for j in range(3)),

 GRB.MAXIMIZE

)

Restricciones de demanda

for j in range(3):

 model.addConstr(sum(x[p, j] for p in range(2)) <= demanda_max[j],
name=f"Demanda_{{jugos[j]}}")

```
# Restricciones de capacidad por planta
```

```
for p in range(2):
```

```
    model.addConstr(sum(x[p, j] for j in range(3)) <= capacidad[p],  
name=f"Capacidad_{plantas[p]}")
```

```
# Restricciones de ingredientes
```

```
model.addConstr(sum(ingrediente_A[j] * x[p, j] for p in range(2) for j in range(3)) <=  
disp_A, name="IngredienteA")
```

```
model.addConstr(sum(ingrediente_B[j] * x[p, j] for p in range(2) for j in range(3)) <=  
disp_B, name="IngredienteB")
```

```
# Resolver modelo
```

```
model.optimize()
```

```
# Resultados
```

```
if model.status == GRB.OPTIMAL:
```

```
    print("--- Plan óptimo de producción ---")
```

```
    for p in range(2):
```

```
        for j in range(3):
```

```
            valor = x[p, j].X
```

```
            if valor > 0:
```

```
                print(f"Planta {plantas[p]}, Jugo {jugos[j]}: {valor:.2f} toneladas")
```

```
    print(f"\nUtilidad total máxima: ${model.ObjVal:,.2f}")
```

```
else:
```

```
    print("No se encontró solución óptima.")
```

8. Optimización de Producción y Transporte de Baterías – ElectroPower S.A.S.

La empresa ElectroPower S.A.S. fabrica 3 tipos de baterías recargables** en **3 centros de producción: Bogotá, Medellín y Barranquilla.

Estas baterías se distribuyen a **4 centros de distribución regionales**: Norte, Centro, Occidente y Sur.

Cada planta tiene una **capacidad máxima de producción**, disponibilidad limitada de **litio** y **cobre**, y cada batería tiene diferentes requerimientos.

El objetivo es **maximizar la utilidad total** teniendo en cuenta:

- Costos de producción
- Costos de transporte
- Precio de venta
- Restricciones de capacidad y materiales

****Datos de plantas****

Planta	Capacidad (unid)	Litio (kg)	Cobre (kg)	
-----	-----	-----	-----	
Bogotá	400	600	800	

Medellín	300	500	600	
Barranquilla	350	550	700	

****Demanda mínima por centro****

Centro	Demanda (unid)	
-----	-----	
Norte	250	
Centro	300	
Occidente	200	
Sur	250	

****Características de baterías****

Tipo	Litio (kg)	Cobre (kg)	Precio (\$)	Costo Prod.	
-----	-----	-----	-----	-----	
Hogar (B1)	1	2	80	40	
Indus (B2)	2	2.5	120	60	
EV (B3)	2.5	3	150	75	

****Costos de transporte planta-centro (por unidad)****

Planta / Centro	Norte	Centro	Occidente	Sur	
-----	-----	-----	-----	-----	

Bogotá	10	8	12	15	
Medellín	9	7	14	20	
Barranquilla	12	10	8	25	

Índices

plantas = ['Bogota', 'Medellin', 'Barranquilla']

centros = ['Norte', 'Centro', 'Occidente', 'Sur']

baterias = ['B1', 'B2', 'B3']

Parámetros

capacidad = {'Bogota': 400, 'Medellin': 300, 'Barranquilla': 350}

litio_disp = {'Bogota': 600, 'Medellin': 500, 'Barranquilla': 550}

cobre_disp = {'Bogota': 800, 'Medellin': 600, 'Barranquilla': 700}

demanda = {'Norte': 250, 'Centro': 300, 'Occidente': 200, 'Sur': 250}

precio = {'B1': 80, 'B2': 120, 'B3': 150}

costo_prod = {'B1': 40, 'B2': 60, 'B3': 75}

litio = {'B1': 1, 'B2': 2, 'B3': 2.5}

cobre = {'B1': 2, 'B2': 2.5, 'B3': 3}

costo_trans = {

 ('Bogota', 'Norte'): 10, ('Bogota', 'Centro'): 8, ('Bogota', 'Occidente'): 12, ('Bogota', 'Sur'): 15,

 ('Medellin', 'Norte'): 9, ('Medellin', 'Centro'): 7, ('Medellin', 'Occidente'): 14, ('Medellin', 'Sur'): 20,

```
    ('Barranquilla', 'Norte'): 12, ('Barranquilla', 'Centro'): 10, ('Barranquilla', 'Occidente'): 8,  
    ('Barranquilla', 'Sur'): 25,  
}
```

```
# Crear modelo
```

```
model = Model("ElectroPower")
```

```
# Variables de decisión:  $x[p,b,c]$  = unidades de batería b fabricadas en planta p y enviadas  
al centro c
```

```
x = model.addVars(plantas, baterias, centros, vtype=GRB.CONTINUOUS, name="x", lb=0)
```

```
# Función objetivo: maximizar utilidad
```

```
model.setObjective(
```

```
    sum((precio[b] - costo_prod[b] - costo_trans[p,c]) * x[p,b,c] for p in plantas for b in  
    baterias for c in centros),
```

```
    GRB.MAXIMIZE
```

```
)
```

```
# Restricciones por planta: capacidad
```

```
for p in plantas:
```

```
    model.addConstr(sum(x[p,b,c] for b in baterias for c in centros) <= capacidad[p],  
    name=f"Capacidad_{p}")
```

```
    model.addConstr(sum(litio[b] * x[p,b,c] for b in baterias for c in centros) <= litio_disp[p],  
    name=f"Litio_{p}")
```

```
    model.addConstr(sum(cobre[b] * x[p,b,c] for b in baterias for c in centros) <=  
    cobre_disp[p], name=f"Cobre_{p}")
```

```
# Restricciones de demanda mínima por centro
```

```

for c in centros:

    model.addConstr(sum(x[p,b,c] for p in plantas for b in baterias) >= demanda[c],
name=f"Demanda_{c}")

# Resolver modelo

model.optimize()

# Resultados

if model.status == GRB.OPTIMAL:

    print("\nPlan de producción y distribución óptimo:")

    for p in plantas:

        for b in baterias:

            for c in centros:

                val = x[p,b,c].X

                if val > 0:

                    print(f"{val:.2f} unidades de batería {b} desde {p} a {c}")

    print(f"\nUtilidad máxima total: ${model.ObjVal:,.2f}")

else:

    print("No se encontró solución óptima.")

```

9. Problema de Producción de Jugos Naturales (Análisis de sencibilidad)

Una empresa productora de jugos naturales embotellados cuenta con tres fábricas (A, B y C) encargadas de producir cinco tipos de jugos (Naranja, Mango, Piña, Fresa y Guayaba). Cada fábrica tiene una capacidad máxima de producción medida en litros diarios. Para la producción de los jugos, se requiere una combinación específica de ingredientes naturales, cuya disponibilidad es limitada. Además, cada tipo de jugo tiene una demanda mínima diaria en el mercado y un precio de venta establecido por litro.

La tabla siguiente muestra la cantidad de materia prima requerida por litro de jugo en cada fábrica, la disponibilidad de materia prima y la capacidad máxima de producción por fábrica:

Ingrediente	Naranja	Mango	Piña	Fresa	Guayaba	Disponibilidad Máx.
(litros)						
--- --- --- --- --- ---						
Agua	2	1.5 2	1.2 1.8	5000		
Pulpa	1 2	1.5 2.5	1.2 4000			
Azúcar	0.5 0.7	30 0.9 0.6	3000			

Cada fábrica tiene una capacidad máxima de producción diaria, siendo de 2000 litros para la Fábrica A, 3000 litros para la Fábrica B y 2500 litros para la Fábrica C. Los precios de venta por litro de jugo en el mercado varían según el sabor, con la Naranja a 1500, Mango a 1800, Piña a 1600, Fresa a 2000 y Guayaba a 1700.

El objetivo es determinar cuántos litros de cada tipo de jugo debe producir cada fábrica para maximizar los ingresos de la empresa, cumpliendo con las restricciones de capacidad y disponibilidad de insumos. Se debe realizar un análisis de sensibilidad, obteniendo los precios sombra de las restricciones y los intervalos de sensibilidad para la capacidad de producción de cada fábrica y la disponibilidad de materia prima.

Conjuntos

Productos = ['Naranja', 'Mango', 'Pina', 'Fresa', 'Guayaba']

```
Fabricas = ['A', 'B', 'C']
```

```
Ingredientes = ['Agua', 'Pulpa', 'Azúcar']
```

```
# Parámetros
```

```
capacidad_fabrica = {'A': 2000, 'B': 2500, 'C': 1500}
```

```
disp_ingr = {'Agua': 5000, 'Pulpa': 4000, 'Azúcar': 3000}
```

```
precio_venta = {'Naranja': 1500, 'Mango': 1800, 'Pina': 1300, 'Fresa': 2000, 'Guayaba':  
1700}
```

```
demanda_min = {'Naranja': 500, 'Mango': 400, 'Pina': 300, 'Fresa': 600, 'Guayaba': 450}
```

```
uso_ingr = {
```

```
    'Agua': {'Naranja': 2, 'Mango': 1.5, 'Pina': 2, 'Fresa': 1.2, 'Guayaba': 1.8},
```

```
    'Pulpa': {'Naranja': 1, 'Mango': 2, 'Pina': 1.5, 'Fresa': 2.5, 'Guayaba': 1.2},
```

```
    'Azúcar': {'Naranja': 0.5, 'Mango': 0.7, 'Pina': 3, 'Fresa': 0.9, 'Guayaba': 0.6}
```

```
}
```

```
# Crear modelo
```

```
model = Model("Sensibilidad_Jugos")
```

```
# Variables de decisión
```

```
X = model.addVars(Fabricas, Productos, name="X", lb=0)
```

```
# Función objetivo
```

```
model.setObjective(sum(precio_venta[p] * X[f, p] for f in Fabricas for p in Productos),  
GRB.MAXIMIZE)
```

```
# Restricciones de capacidad por fábrica
```

```
for f in Fabricas:
```

```
    model.addConstr(sum(X[f, p] for p in Productos) <= capacidad_fabrica[f],  
name=f"Capacidad_{f}")
```

```
# Restricciones de disponibilidad de ingredientes
```

```
for i in Ingredientes:
```

```
    model.addConstr(sum(uso_ingr[i][p] * X[f, p] for f in Fabricas for p in Productos) <=  
disp_ingr[i], name=f"Ingr_{i}")
```

```
# Restricciones de demanda mínima
```

```
for p in Productos:
```

```
    model.addConstr(sum(X[f, p] for f in Fabricas) >= demanda_min[p],  
name=f"Demanda_{p}")
```

```
# Resolver modelo
```

```
model.optimize()
```

```
# Imprimir resultados
```

```
if model.status == GRB.OPTIMAL:
```

```
    print("\n--- Producción óptima ---")
```

```
    for f in Fabricas:
```

```
        for p in Productos:
```

```
            cantidad = X[f, p].x
```

```
            if cantidad > 0:
```

```
                print(f"Fábrica {f} produce {cantidad:.2f} litros de jugo de {p}")
```

```
print(f"\nIngreso total: ${model.objVal:,.2f}")
```

```
# Precios sombra
```

```

print("\n--- Precios Sombra ---")

for i in Ingredientes:

    restr = model.getConstrByName(f"Ingr_{i}")

    print(f"{restr.constrName}: {restr.pi:.2f}")

for f in Fabricas:

    restr = model.getConstrByName(f"Capacidad_{f}")

    print(f"{restr.constrName}: {restr.pi:.2f}")


# Intervalos de sensibilidad

print("\n--- Intervalos de Sensibilidad (rango de precios aceptables) ---")

for f in Fabricas:

    for p in Productos:

        var = model.getVarByName(f"X[{f},{p}]")

        if var.x > 0:

            print(f"{var.VarName}: [{var.SAObjLow:.2f}, {var.SAObjUp:.2f}]")

else:

    print("No se encontró una solución óptima.")


# Conjuntos

Productos = ['Naranja', 'Mango', 'Pina', 'Fresa', 'Guayaba']

Fabricas = ['A', 'B', 'C']

Ingredientes = ['Agua', 'Pulpa', 'Azúcar']


# Parámetros

capacidad_fabrica = {'A': 2000, 'B': 2500, 'C': 1500}

disp_ingr = {'Agua': 5000, 'Pulpa': 4000, 'Azúcar': 3000}

```

```

precio_venta = {'Naranja': 1500, 'Mango': 1800, 'Pina': 1300, 'Fresa': 2000, 'Guayaba': 1700}

demanda_min = {'Naranja': 500, 'Mango': 400, 'Pina': 300, 'Fresa': 600, 'Guayaba': 450}

uso_ingr = {
    'Agua': {'Naranja': 2, 'Mango': 1.5, 'Pina': 2, 'Fresa': 1.2, 'Guayaba': 1.8},
    'Pulpa': {'Naranja': 1, 'Mango': 2, 'Pina': 1.5, 'Fresa': 2.5, 'Guayaba': 1.2},
    'Azúcar': {'Naranja': 0.5, 'Mango': 0.7, 'Pina': 3, 'Fresa': 0.9, 'Guayaba': 0.6}
}

```

Crear modelo

```
model = Model("Sensibilidad_Jugos")
```

Variables de decisión

```
X = model.addVars(Fabricas, Productos, name="X", lb=0)
```

Función objetivo

```
model.setObjective(sum(precio_venta[p] * X[f, p] for f in Fabricas for p in Productos),
GRB.MAXIMIZE)
```

Restricciones de capacidad por fábrica

for f in Fabricas:

```
    model.addConstr(sum(X[f, p] for p in Productos) <= capacidad_fabrica[f],
name=f"Capacidad_{f}")
```

Restricciones de disponibilidad de ingredientes

for i in Ingredientes:

```
model.addConstr(sum(uso_ingr[i][p] * X[f, p] for f in Fabricas for p in Productos) <=
disp_ingr[i], name=f"Ingr_{i}")
```

```
# Restricciones de demanda mínima
```

```
for p in Productos:
```

```
    model.addConstr(sum(X[f, p] for f in Fabricas) >= demanda_min[p],
name=f"Demanda_{p}")
```

```
# Resolver modelo
```

```
model.optimize()
```

```
# Imprimir resultados
```

```
if model.status == GRB.OPTIMAL:
```

```
    print("\n--- Producción óptima ---")
```

```
    for f in Fabricas:
```

```
        for p in Productos:
```

```
            cantidad = X[f, p].x
```

```
            if cantidad > 0:
```

```
                print(f"Fábrica {f} produce {cantidad:.2f} litros de jugo de {p}")
```

```
    print(f"\nIngreso total: ${model.objVal:.2f}")
```

```
# Precios sombra
```

```
print("\n--- Precios Sombra ---")
```

```
for i in Ingredientes:
```

```
    restr = model.getConstrByName(f"Ingr_{i}")
```

```
    print(f"{restr.constrName}: {restr.pi:.2f}")
```

```
for f in Fabricas:
```

```

restr = model.getConstrByName(f"Capacidad_{f}")

print(f"{restr.constrName}: {restr.pi:.2f}")

# Intervalos de sensibilidad

print("\n--- Intervalos de Sensibilidad (rango de precios aceptables) ---")

for f in Fabricas:

    for p in Productos:

        var = model.getVarByName(f"X[{f},{p}]")

        if var.x > 0:

            print(f"{var.VarName}: [{var.SAObjLow:.2f}, {var.SAObjUp:.2f}]")

else:

    print("No se encontró una solución óptima.")

```

****10. Optimización de Jugos VitalJugos desde Excel****

La empresa VitalJugos S.A.S. produce cinco jugos: Manzana, Uva, Mora, Papaya, Kiwi en tres fábricas: Centro, Norte, Sur.

Tienes un archivo Excel llamado ****datos_jugos.xlsx**** que contiene:

1. Leer los datos del archivo Excel

```
archivo = 'datos_jugos.xlsx'
```

```
productos_df = pd.read_excel(archivo, sheet_name='Productos')
```

```
fabricas_df = pd.read_excel(archivo, sheet_name='Fabricas')
ingredientes_df = pd.read_excel(archivo, sheet_name='Ingredientes')
consumo_df = pd.read_excel(archivo, sheet_name='Consumo')
```

2. Crear listas

```
Productos = list(productos_df['Producto'])
Fabricas = list(fabricas_df['Fabrica'])
Ingredientes = list(ingredientes_df['Ingrediente'])
```

3. Crear parámetros

```
precio_venta = dict(zip(productos_df['Producto'], productos_df['PrecioVenta']))
demanda_min = dict(zip(productos_df['Producto'], productos_df['DemandaMinima']))
capacidad_fabrica = dict(zip(fabricas_df['Fabrica'], fabricas_df['Capacidad']))
disp_ingr = dict(zip(ingredientes_df['Ingrediente'], ingredientes_df['Disponibilidad']))
```

```
consumo = {
    i: dict(zip(Productos, consumo_df.loc[consumo_df['Ingrediente'] == i,
Productos].values.flatten()))
    for i in Ingredientes
}
```

4. Crear el modelo

```
model = Model("Optimización_Jugos_Desde_Excel")
```

5. Variables de decisión

```
X = model.addVars(Fabricas, Productos, name="Producción", lb=0)
```

6. Función objetivo: maximizar ingresos

```
model.setObjective(sum(precio_venta[p] * X[f, p] for f in Fabricas for p in Productos),  
GRB.MAXIMIZE)
```

7. Restricciones

Capacidad de fábricas

for f in Fabricas:

```
    model.addConstr(sum(X[f, p] for p in Productos) <= capacidad_fabrica[f],  
name=f"Capacidad_{f}")
```

Disponibilidad de ingredientes

for i in Ingredientes:

```
    model.addConstr(sum(consumo[i][p] * X[f, p] for f in Fabricas for p in Productos) <=  
disp_ingr[i], name=f"Ingr_{i}")
```

Demanda mínima

for p in Productos:

```
    model.addConstr(sum(X[f, p] for f in Fabricas) >= demanda_min[p],  
name=f"Demanda_{p}")
```

8. Resolver el modelo

```
model.optimize()
```

9. Mostrar resultados

if model.status == GRB.OPTIMAL:

```
    print("\n--- Producción óptima ---")
```

```
for f in Fabricas:
    for p in Productos:
        cantidad = X[f, p].x
        if cantidad > 0:
            print(f"Fábrica {f} produce {cantidad:.2f} litros de {p}")
print(f"\nIngreso total: ${model.objVal:,.2f}")
```

```
else:
    print("No se encontró solución óptima.")
```

11. Producción Óptima de Computadoras

La empresa **TecnoFab S.A.S.** fabrica tres productos electrónicos: **Portátiles Básicos**, **Portátiles Gaming**, y **PCs de Escritorio** en tres plantas: **Planta Norte**, **Planta Sur** y **Planta Centro**.

Cada tipo de producto requiere ciertas cantidades de **Microprocesadores**, **Memoria RAM** y **Unidades SSD**, y cada planta tiene una **capacidad máxima diaria de ensamblaje** (en unidades).

La empresa desea **maximizar los ingresos totales por ventas**, asegurando que se cumplan las siguientes condiciones:

- No se puede exceder la **capacidad de ensamblaje** de cada planta.
- La disponibilidad total de **componentes** es limitada.

- Debe cumplirse una ****demanda mínima diaria**** para cada tipo de producto.

A continuación se presentan las tablas con la información correspondiente.

Requerimientos de componentes por producto (por unidad)

Ingrediente	Portátil Básico	Portátil Gaming	PC Escritorio	
-----	-----	-----	-----	
Microprocesador	1	1	1	
RAM	2	4	3	
SSD	1	2	2	

Capacidad de las plantas

Fábrica	Capacidad (unidades/día)	
-----	-----	
Planta Norte	300	
Planta Sur	250	
Planta Centro	350	

Disponibilidad diaria de componentes

Componente	Disponibilidad
Microprocesador	1000
RAM	1500
SSD	900

Precio de venta y demanda mínima diaria

Producto	Precio de Venta	Demanda Mínima
Portátil Básico	3,200,000	100
Portátil Gaming	6,500,000	80
PC Escritorio	4,500,000	120

```
archivo = 'datos_computadoras.xlsx'
```

```
productos_df = pd.read_excel(archivo, sheet_name='Productos')
```

```
fabricas_df = pd.read_excel(archivo, sheet_name='Fabricas')
```

```
ingredientes_df = pd.read_excel(archivo, sheet_name='Ingredientes')
```

```
consumo_df = pd.read_excel(archivo, sheet_name='Consumo')
```

```
# Listas
```

```
Productos = list(productos_df['Producto'])
```

```
Fabricas = list(fabricas_df['Fabrica'])
```

```
Ingredientes = list(ingredientes_df['Ingrediente'])
```

```
# Parámetros
```

```
precio_venta = dict(zip(productos_df['Producto'], productos_df['PrecioVenta']))
```

```
demanda_min = dict(zip(productos_df['Producto'], productos_df['DemandaMinima']))
```

```
capacidad_fabrica = dict(zip(fabricas_df['Fabrica'], fabricas_df['Capacidad']))
```

```
disp_ingr = dict(zip(ingredientes_df['Ingrediente'], ingredientes_df['Disponibilidad']))
```

```
# Consumo por ingrediente-producto
```

```
consumo = {
```

```
    i: dict(zip(Productos, consumo_df.loc[consumo_df['Ingrediente'] == i,  
Productos].values.flatten()))
```

```
    for i in Ingredientes
```

```
}
```

```
# Crear modelo
```

```
model = Model("Computadoras_Producción")
```

```
X = model.addVars(Fabricas, Productos, name="Producción", lb=0)
```

```
model.setObjective(sum(precio_venta[p] * X[f, p] for f in Fabricas for p in Productos),  
GRB.MAXIMIZE)
```

```
# Restricciones
```

```
for f in Fabricas:
```

```
    model.addConstr(sum(X[f, p] for p in Productos) <= capacidad_fabrica[f],  
name=f"Capacidad_{f}")
```

```
for i in Ingredientes:
```

```
    model.addConstr(sum(consumo[i][p] * X[f, p] for f in Fabricas for p in Productos) <=  
disp_ingr[i], name=f"Ingr_{i}")
```

```
for p in Productos:
```

```
    model.addConstr(sum(X[f, p] for f in Fabricas) >= demanda_min[p],  
name=f"Demanda_{p}")
```

```
model.optimize()
```

```
# Resultados
```

```
if model.status == GRB.OPTIMAL:
```

```
    print("\n--- Producción óptima ---")
```

```
    for f in Fabricas:
```

```
        for p in Productos:
```

```
            cantidad = X[f, p].x
```

```
            if cantidad > 0:
```

```
                print(f"{f} produce {cantidad:.2f} unidades de {p}")
```

```
    print(f"\nIngreso total: ${model.ObjVal:,.2f}")
```

```
else:
```

```
    print("No se encontró solución óptima.")
```