# Python para I.O.

Investigación Operativa

# Motivación

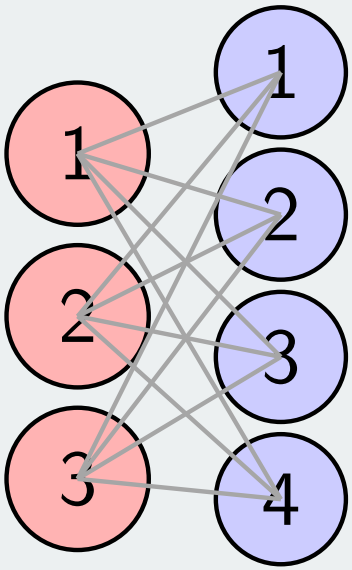
Python es un lenguaje de programación bastante versátil que nos va a permitir resolver problemas de optimización. Los pasos típicos van a ser:

1. Identificar el problema
2. Crear el modelo matemático
3. Implementar y resolver usando Python

En Investigación Operativa usamos Python porque los problemas con los que nos vamos a encontrar no son resolubles a mano.

# Proceso

**Problema** $\rightarrow$ **Modelado** $\rightarrow$ **Optimización**



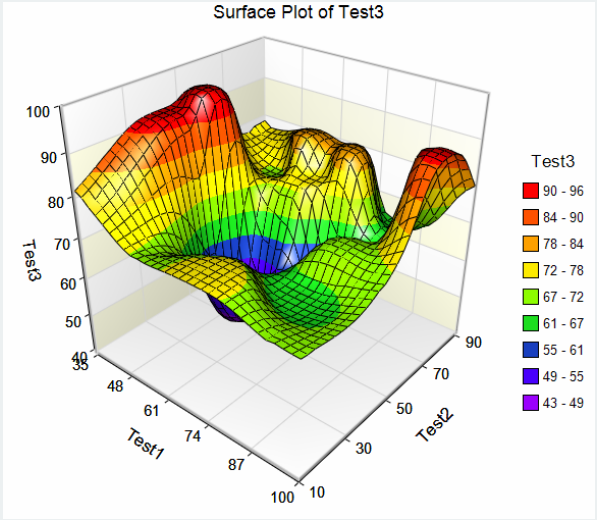

$$Z = 3000x_1 + 5000x_2$$

$$2x_2 \leq 12$$

$$3x_1 + 2x_2 \leq 18$$

$$x_1 \geq 0$$

$$x_2 \geq 0$$



# Variables en Python

```
1 texto = "Hello World"
2 numero = 5
3 numero_con_coma = 1.3
4 mi_lista = [1,2,3,4]
5 mi_tupla = (1,2,4)
```

# Operaciones

En Python podemos usar las operaciones matemáticas comunes escritas de la siguiente manera:

- Suma: +
- Resta:
- Multiplicación: *
- División: /
- División entera: //
- Potencia: **
- Resto: %

# Print

Si queremos imprimir una variable o el resultado de una operación usamos print:

```
1 a = 10
2 b = 20
3 print("El resultado es", a+b)
4 # Output:
5 # El resultado es 30
```

# Listas

Dada una lista a = [1,2,3] podemos realizar varias cosas:

- Los elementos de a se pueden obtener haciendo a[i] donde i es la posición del elemento.
- ¡Ojo! En python el primer elemento es 0, no 1
- Podemos cambiar el elemento i de la lista escribiendo a[i] = 4
- Podemos agregar un elemento al final haciendo a.append(4)

(!) ¿Es lo mismo una lista y una tupla? ¿En qué difieren?

# Booleanos y Operadores Lógicos

Los valores booleanos son True y False, y cuando comparamos dos variables, se nos devuelve el valor booleano correspondiente a esa relación.

- **and**: Retorna True si AMBAS condiciones son True
- **or**: Retorna True si AL MENOS UNA condición es True
- **not**: Invierte el valor booleano

Ejemplo:

```
1 a = True
2 b = False
3 print(a and b)  # False
4 print(a or b)   # True
5 print(not a)    # False
```

# Operadores de Comparación

Operadores lógicos:

- $>$ Mayor que
- $<$ Menor que
- $==$ Igual (son dos porque si fuera uno sería asignación!)
- $\geq$ Mayor o igual
- $\leq$ Menor o igual
- $!=$ Distinto

# If/else

La sintaxis de if/else es la siguiente:

```
if boolean:
    Statement1
else:
    Statement2

# Ejemplo:
if a == b:
    print('Son iguales')
else:
    print('Son distintos')
```

También es posible anidarlos o poner muchos condicionales con elif.

# For loop

La sintaxis del for loop que típicamente usaremos es la siguiente:

```
for i in range(N):
    routine

# Ejemplo:
for i in range(4):
    print(i)

# Output:
# 0
# 1
# 2
# 3
```

La función range(N) genera una lista con una secuencia de números, hasta N-1.

Sintaxis: range(comienzo, final, paso)

# While

La sintaxis de while es la siguiente:

```
while boolean:
    statement

# Ejemplo:
i = 0
while i < 4:
    print(i)
    i = i + 1
# Output:
# 0
# 1
# 2
# 3
```

¡Cuidado con los loops infinitos!

# Librerías

Vamos a usar en principio 4 librerías distintas:

- **Numpy**: Para fácil manipulación de matrices y vectores
- **Matplotlib**: Nos va a permitir graficar funciones
- **PICOS**: Contiene paquetes de optimización que nos serán muy útiles
- **SciPy**: Para optimización no lineal

Este será nuestro estándar:

```
1 import numpy as np
2 import scipy as scp
3 import picos
4 import matplotlib.pyplot as plt
```

# Numpy

Numpy nos permite trabajar con vectores y matrices fácilmente. El elemento básico que uno usa en Numpy son los 'numpy arrays'

Para definir un Numpy array debemos escribirlo de la siguiente manera:

```
1 array = np.array([1, 2, 3, 4])
```

Podemos acceder a los lugares del array de la misma forma que con las listas. La diferencia es que si en este caso tenemos una matriz y queremos ver el lugar i,j de la matriz podemos escribir array[i,j] en vez de array[i][j]

Además, en este caso, multiplicar un np.array por un escalar, equivale a multiplicar cada lugar del array por el escalar.

# Ejercicios Prácticos

## Ejercicio 1: Listas y Promedios

1. Generar una lista L con todos los números pares hasta el 20
2. Solo guardar los múltiplos de 4
3. Calcular la media: $\langle L \rangle = \frac{\sum L_i}{N}$

## Ejercicio 2: Función Múltiplos

1. Definir la función múltiplos que acepte como input el número hasta donde quiero obtener los números pares y el número del que quiero que sean múltiplos
2. Graficar los puntos usando matplotlib

# A programar se aprende programando

# Terminamos

**¿Dudas?**

**¿Consultas?**