Python para IO

Investigación Operativa, Universidad de San Andrés

Si encuentran algún error en el documento o hay alguna duda, mandenmé un mail a rodriguezf@udesa.edu.ar y lo revisamos.

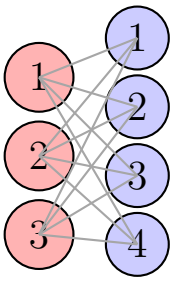
1. Motivación

Python es un lenguaje de programación versátil que nos va a permitir resolver problemas de optimización. El proceso típico va a ser:

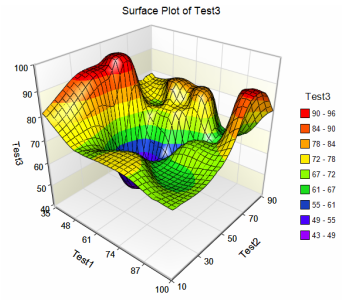
1. Identificar el problema
2. Crear el modelo matemático
3. Implementar y resolver usando Python

En Investigación Operativa usamos Python porque los problemas con los que nos vamos a encontrar no son resolubles a mano. Necesitamos herramientas que nos permitan resolverlos de manera eficiente, y Python es una de las herramientas más fáciles de usar para resolver problemas de optimización.

Problema → **Modelado** → **Optimización**



$$
\begin{aligned}
Z &= 3000x_1 + 5000x_2 \\
2x_2 &\leq 12 \\
3x_1 + 2x_2 &\leq 18 \\
x_1 &\geq 0 \\
x_2 &\geq 0
\end{aligned}
$$



2. Conceptos Básicos de Python

2.1. Variables y Tipos de Datos

En Python podemos asignar un objeto a una variable usando el signo =.

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

2.2. Operaciones Matemáticas

Las operaciones matemáticas básicas en Python son:

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

2.3. Print

Para mostrar valores o resultados usamos la función print:

```python
a = 10
b = 20
print("El resultado es", a+b)
# Output:
# El resultado es 30
```

2.4. Estructuras de Control

2.4.1. Condicionales (if/else)

```python
if a == b:
    print("Son iguales")
else:
    print("Son distintos")
# Output:
# Son distintos
```

2.4.2. Bucles

El bucle for es muy útil para iterar sobre secuencias:

```
for i in range(4):
    print(i)
# Output:
# 0
# 1
# 2
# 3
```

El bucle while se ejecuta mientras una condición sea verdadera:

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

3. Librerías Principales

Para nuestro trabajo en IO, usaremos principalmente:

- **NumPy**: Para manipulación de matrices y vectores
- **Matplotlib**: Para visualización de datos
- **PICOS**: Para problemas de optimización
- **SciPy**: Para optimización no lineal

El estándar de importación que usaremos es:

```
import numpy as np
import scipy as scp
import picos
import matplotlib.pyplot as plt
```

4. NumPy

NumPy es fundamental para trabajar con arrays y matrices:

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

# Acceso a elementos
print(array[0])  # Primer elemento
# Output:
# 1

# Operaciones matriciales
matriz = np.array([[1, 2], [3, 4]])
print(matriz[0, 1])  # Elemento en fila 0, columna 1
# Output:
# 2
```

5. Ejercicios Prácticos

5.1. Ejercicio 1: Listas y Promedios

Vamos a resolver los siguientes puntos:

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

```
L = []
for i in range(20):
    if i % 2 == 0:
        L.append(i)

L_filtrado = []
for i in L:
    if i % 4 == 0:
        L_filtrado.append(i)

media = np.mean(L_filtrado)
print(media)
# Output:
# 6.0
```

5.2. 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

```
def multiplos(n, m):
    L = []
    for i in range(n):
        if i % m == 0:
            L.append(i)
    return L

plt.plot(multiplos(100, 4))
plt.show()
```

A programar se aprende programando.