



UNIVERSIDAD
NACIONAL
DE COLOMBIA

PROYECTO CULTURAL, CIENTÍFICO Y COLECTIVO DE NACIÓN

Introducción a Python - Tutorial

Profesor: Carlos A. Duque Daza
Asistente docente: Sebastián Gómez P.

Facultad de Ingeniería – Departamento de Ingeniería Mecánica y
Mecatrónica – Sede Bogotá

Universidad Nacional de Colombia

PROYECTO CULTURAL, CIENTÍFICO Y COLECTIVO DE NACIÓN

¿Qué es Python?

Lenguaje de propósito general y fácil de usar

Enfoque en diseño de software: programación estructurada y POO.

Licencia de código abierto

Biblioteca estándar, amplio uso y difusión

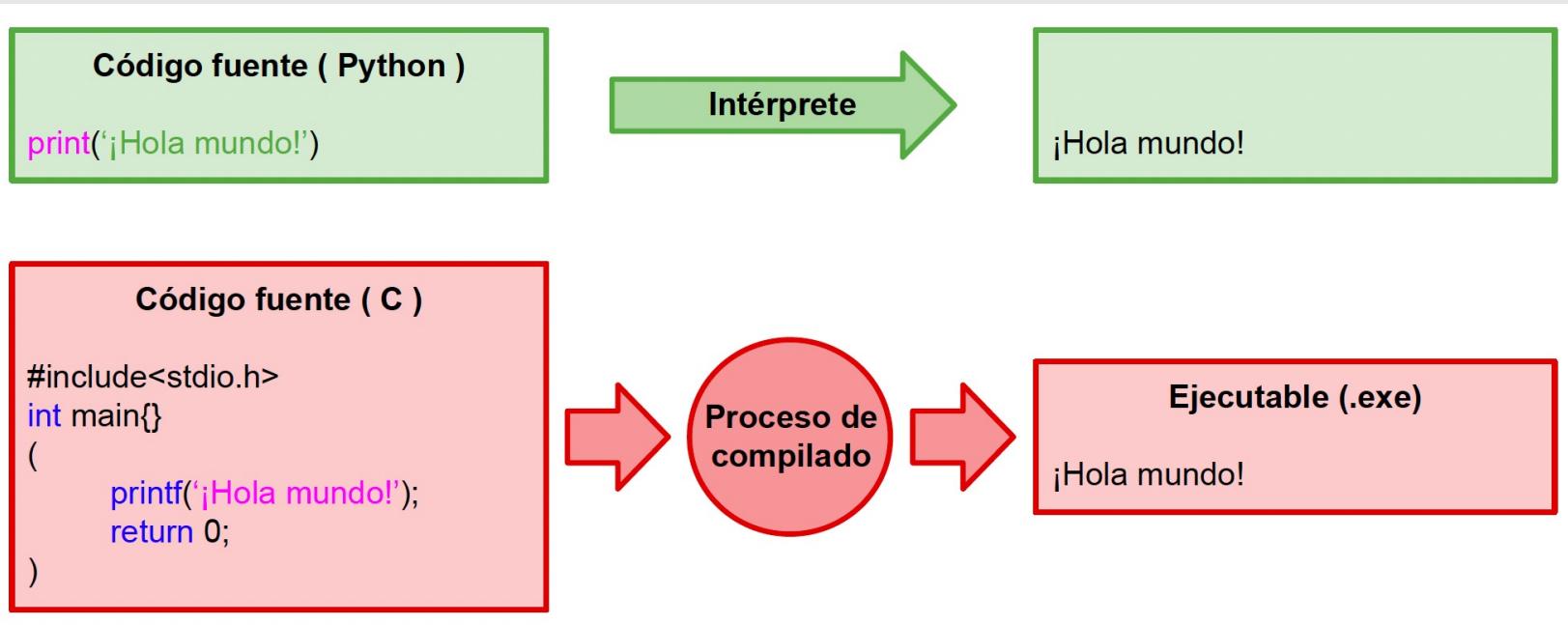
Lenguaje interpretado, dinámico y multiplataforma

Python: ¿Qué es lenguaje interpretado?

Lenguaje interpretado	Lenguaje compilado
<ul style="list-style-type: none">• Desarrollo rápido.	<ul style="list-style-type: none">• Desarrollo lento.
<ul style="list-style-type: none">• Depuración más sencilla.	<ul style="list-style-type: none">• Más difícil de depurar.
<ul style="list-style-type: none">• Generalmente necesita un menor número de líneas de código.	<ul style="list-style-type: none">• Casi siempre requiere más líneas de código.
<ul style="list-style-type: none">• Programas más lentos.	<ul style="list-style-type: none">• Programas más rápidos
<ul style="list-style-type: none">• Menos control sobre el comportamiento del programa.	<ul style="list-style-type: none">• Más control sobre el comportamiento del programa.

Fuente: Jair Tovar y Germán Sierra

Python: ¿Qué es lenguaje interpretado?



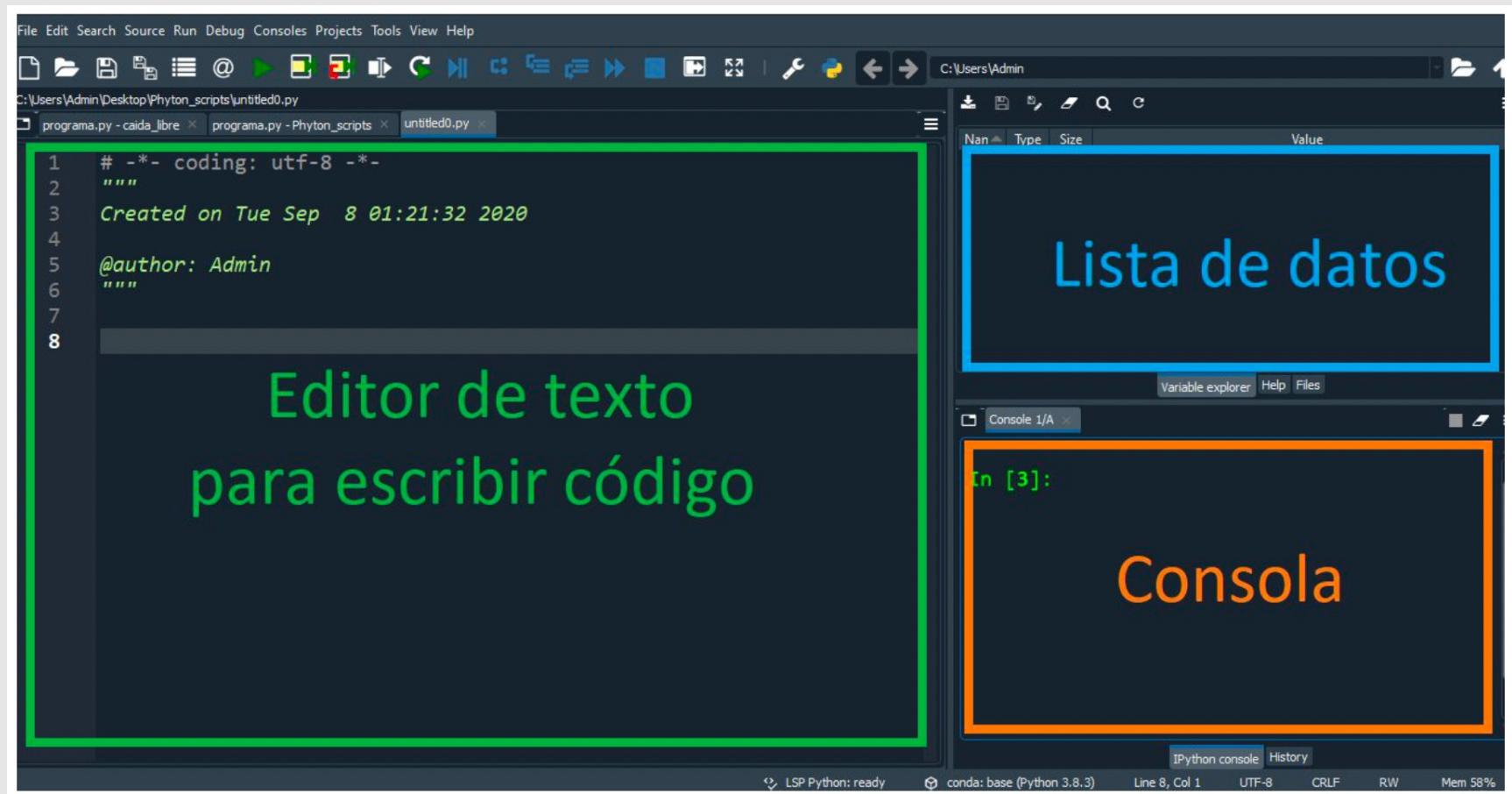
Fuente: Jair Tovar y Germán Sierra

Interfaz de Anaconda

The screenshot shows the Anaconda Navigator interface with a grid of tools:

- JupyterLab** 2.1.5: An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture. [Launch](#)
- jupyter Notebook** 6.0.3: Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis. [Launch](#)
- Qt Console** 4.7.5: PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more. [Launch](#)
- Spyder** 4.1.4: Scientific Python Development Environment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features. [Launch](#) (This tool is highlighted with a red border)
- Glueviz** 0.15.2: Multidimensional data visualization across files. Explore relationships within and among related datasets.
- Orange 3** 3.26.0: Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large
- RStudio** 1.1.456: A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.

Interfaz de Spyder



Python

ESTRUCTURA BÁSICA DE UN PROGRAMA

.....

Estructura básica de un programa
@author: Estudiante

.....

Este es un comentario

{ Listado de librerías y módulos }

{ Función 1 }

Instrucciones de la f1
Retorno de valores de f1

{ Función n }

Instrucciones de la fn
Retorno de valores de fn

... continuación del programa

{ Inicialización de variables }

{ Condicional if }

Instrucciones del condicional

{ ... else }

Instrucciones del condicional

{ Bucle for / while }

Instrucciones del for / while

{ Generación de gráficas }

Fin del programa

Fuente: Jair Tovar y Germán Sierra

Tipos de variables en Python

Entero (int)	a = 2 b = -256 c = 0	Cadena (str)	n = 'True' o = n[0:2] p = "He's a 9 years old"
Real (float)	d = 13. e = -102.56 f = 0.0	Lista (list)	q = ['a', 45, 6.3, True] r = q[1:2] s = list(range(10))
Complejo (complex)	g = 2 + 5j h = 7.4 - 18j j = -23 + 10.5i	Tupla (tuple)	t = ('a', 45, 6.3, True) u = (t[1:3], s) t[1] = 10.34
Booleano (bool)	k = True l = False m = false	Diccionario (dict)	v = {'name':'Sue', 'cod':2} v['status'] = True v[3] = 31.2 - 57j

Fuente: Jair Tovar y Germán Sierra

Tipos de variables en Python

NUMEROS EN PYTHON

- int
- float
- complex

```
x = 1      # int
y = 2.8    # float
z = 1j     # complex
```

```
#convert from int to float:
a = float(x)

#convert from float to int:
b = int(y)

#convert from int to complex:
c = complex(x)
```

Operadores en Python

OPERADORES ARITMÉTICOS

Operator	Name
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus
**	Exponentiation
//	Floor division

OPERADORES DE COMPARACIÓN

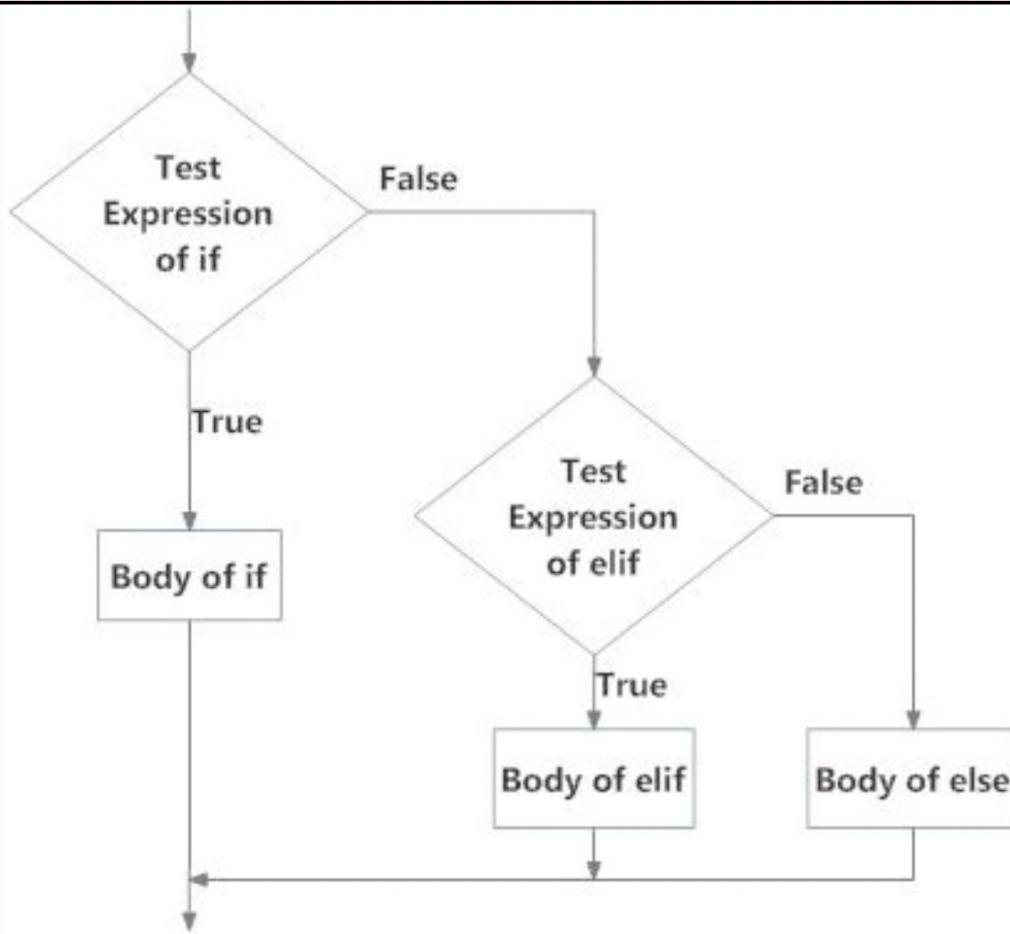
==	Equal
!=	Not equal
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to

OPERADORES LÓGICOS

and	Returns True if both statements are true
or	Returns True if one of the statements is true
not	Reverse the result, returns False if the result is true

Condicionales

IF - ELSE



if (expresión de prueba):
Sentencia if
elif (expresión de prueba):
Sentencia elif
else:
Sentencia else

```
num = 3.4

# Try these two variations as well:
# num = 0
# num = -4.5

if num > 0:
    print("Positive number")
elif num == 0:
    print("Zero")
else:
    print("Negative number")
```

Sintaxis e indentación

```
if 5 > 2:  
    print("Five is greater than two!")
```

```
if 5 > 2:  
    print("Five is greater than two!")
```

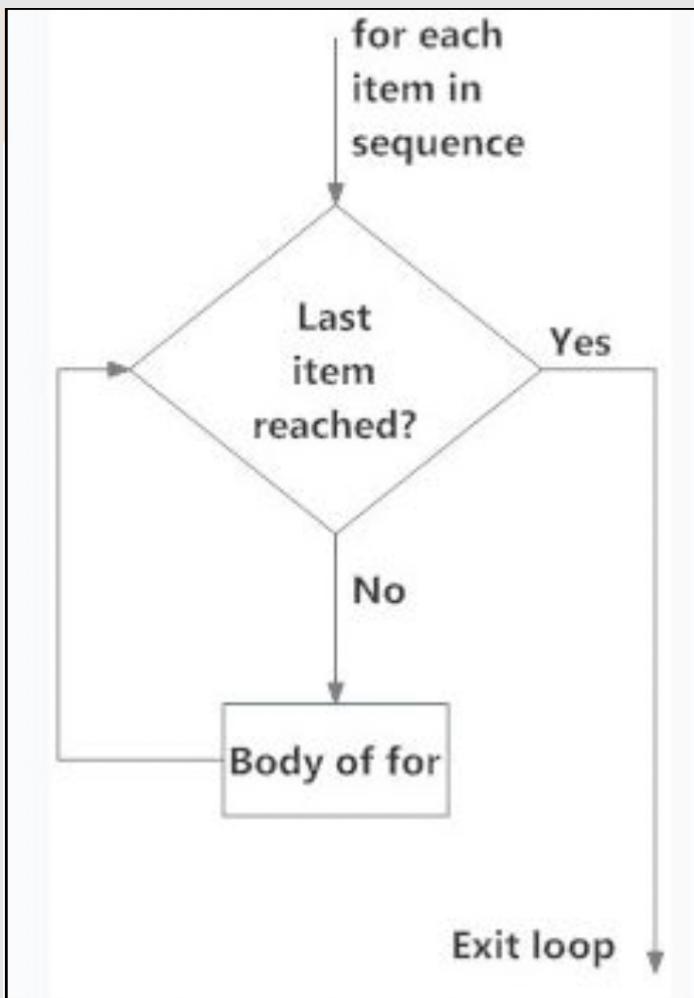
```
if 5 > 2:  
    print("Five is greater than two!")  
    print("Five is greater than two!")
```

```
In [4]: if b > a:  
....:     print("b is greater than a")  
....:     elif a == b:  
....:         print("a and b are equal")  
....: else:  
....:     print("a is greater than b")  
File "<ipython-input-4-f229db10c5f8>", line 3  
    elif a == b:  
        ^  
  
SyntaxError: invalid syntax
```

```
a=200  
b=33|  
if b > a:  
    print("b is greater than a")  
elif a == b:  
    print("a and b are equal")  
else:  
    print("a is greater than b")
```

Python

BUCLE FOR



for val in sequence:
Sentencias

```
x = np.array([2.3,3.5,4.7,8.6,3.5,1.9])
```

```
for i in range(6):  
    print(i,x[i])
```

0	2.3
1	3.5
2	4.7
3	8.6
4	3.5
5	1.9

```
for i in range(2, 6):  
    print(i,x[i])
```

2	4.7
3	8.6
4	3.5
5	1.9

Python

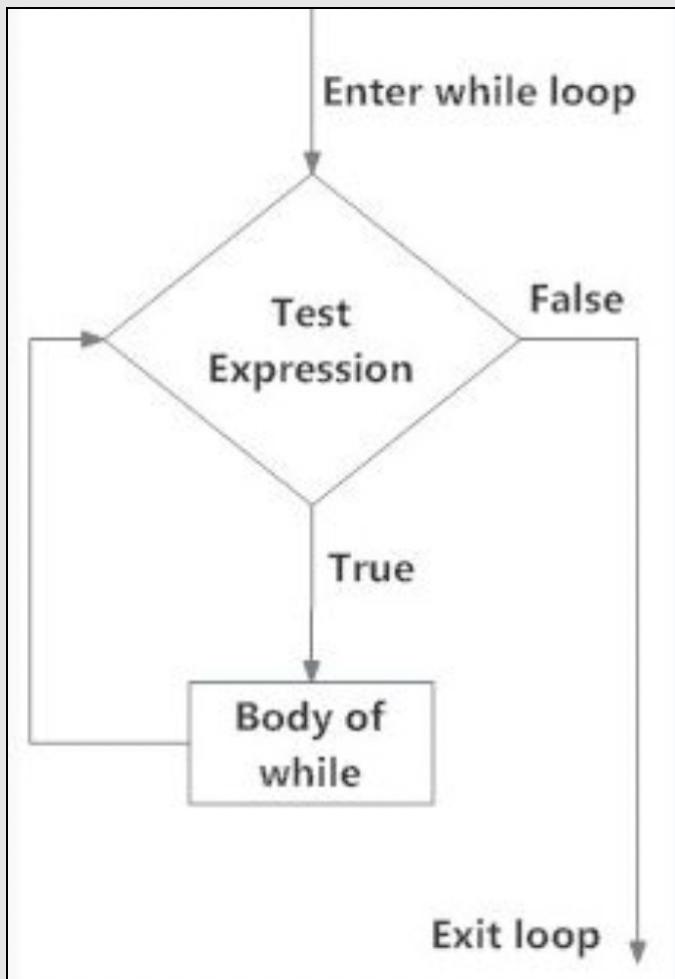
BUCLE FOR

```
n = 0
for i in range(10):
    n = n + 2*i
    print("i:",i,"n:",n)
```

```
i: 0 n: 0
i: 1 n: 2
i: 2 n: 6
i: 3 n: 12
i: 4 n: 20
i: 5 n: 30
i: 6 n: 42
i: 7 n: 56
i: 8 n: 72
i: 9 n: 90
```

Python

BUCLE WHILE



while (expresión de prueba):
Sentencias

```
n=10
suma=0
i=1
while i<=n:
    suma=suma+i
    i= i + 1
    print("i:",i,"n:",n,"the sum is",suma)
```

```
i: 2 n: 10 the sum is 1
i: 3 n: 10 the sum is 3
i: 4 n: 10 the sum is 6
i: 5 n: 10 the sum is 10
i: 6 n: 10 the sum is 15
i: 7 n: 10 the sum is 21
i: 8 n: 10 the sum is 28
i: 9 n: 10 the sum is 36
i: 10 n: 10 the sum is 45
i: 11 n: 10 the sum is 55
```

Python

LIBRERÍAS Y MÓDULOS

Un módulo es un fichero (.py) que contiene funciones o porciones de código que puede llamarse en el programa principal. Una librería, en cambio, es un conjunto de módulos con una finalidad específica.

Importar un módulo o librería	<code>import numpy</code>
Importar una librería con un 'alias'	<code>import numpy as np</code>
Importar un módulo desde una librería	<code>from numpy import pi</code>

Numpy es una librería con funciones para trabajar arreglos

Python

TIPOS DE ARREGLOS

```
>>> import numpy as np  
>>> a = np.array([2,3,4])  
>>> a  
array([2, 3, 4])  
>>> a.dtype  
dtype('int64')  
>>> b = np.array([1.2, 3.5, 5.1])  
>>> b.dtype  
dtype('float64')
```

```
>>> b = np.array([(1.5,2,3), (4,5,6)])  
>>> b  
array([[1.5, 2. , 3. ],  
       [4. , 5. , 6. ]])
```

```
>>> np.zeros((3, 4))  
array([[0., 0., 0., 0.],  
      [0., 0., 0., 0.],  
      [0., 0., 0., 0.]])
```

```
import numpy as np  
# Creación de los arreglos 'C' y 'D'  
C = np.array([0,1,2,3,4],dtype=np.float64)  
D = np.eye(3)  
# Imprimir el resultado en pantalla  
print("C =",C)  
print("D =",D)
```

```
C = [0. 1. 2. 3. 4.]  
D = [[1. 0. 0.]  
     [0. 1. 0.]  
     [0. 0. 1.]]
```

Python

OPERACIONES CON ARREGLOS

```
>>> A = np.array( [[1,1],  
...                 [0,1]] )  
>>> B = np.array( [[2,0],  
...                 [3,4]] )  
>>> A * B                      # elementwise product  
array([[2, 0],  
       [0, 4]])  
>>> A @ B                      # matrix product  
array([[5, 4],  
       [3, 4]])
```

```
# Operaciones básicas  
E = C - 2.5*D + 5  
  
# Producto punto entre vectores  
F = np.dot(B,B)  
  
# Multiplicación entre matrices  
G = A*B  
H = np.matmul(A,B)  
I = np.dot(A,B)  
  
print('Resultado de A*B:',G)  
print('Resultado de AB:',H)
```

Python

FUNCIONES DE NUMPY PARA OPERACIONES MATRICIALES

```
# Declaración matriz M de 3x3 (ejemplo)
M = np.array([[3,4,-1],[2,0,1],[1,3,-2]])

# Calcula la inversa de la matriz M
M_inv = np.linalg.inv(M)

# Calcula la transpuesta de la matriz M
M_tr = np.transpose(M)

# Calcula el determinante de la matriz M
M_det = np.linalg.det(M)

# Calcula los valores y vectores propios de la matriz M
# eigen_val es un vector que almacena los valores propios
# eigen_vec es una matriz que almacena por columnas...
# ...los vectores propios de cada valor propio

eigen_val,eigen_vec = np.linalg.eig(M)
```

```
In [40]: M
Out[40]:
array([[ 3,   4,  -1],
       [ 2,   0,   1],
       [ 1,   3,  -2]])

In [41]: M_inv
Out[41]:
array([[-0.6,   1. ,   0.8],
       [ 1. ,  -1. ,  -1. ],
       [ 1.2,  -1. ,  -1.6]])

In [42]: M_tr
Out[42]:
array([[ 3,   2,   1],
       [ 4,   0,   3],
       [-1,   1,  -2]])
```

```
In [43]: M_det
Out[43]: 5.000000000000001

In [44]: eigen_val
Out[44]: array([ 4.66203005, -0.32100779, -3.34102227])

In [45]: eigen_vec
Out[45]:
array([[-0.84364622, -0.44317766,  0.42573961],
       [-0.43068599,  0.54574116, -0.4837342 ],
       [-0.32057859,  0.71116816,  0.76468752]])
```

Python

SOLUCIÓN DE SISTEMAS DE ECUACIONES LINEALES

.....
Solución de sistemas AX=B
@author: Estudiante
.....

```
import numpy as np

# 8x + 3y - 2z = 9
# -4x + 7y + 5z = 15
# 3x + 4y - 12z = 35

# Creación de los arreglos
A = np.array([[8,3,-2],[-4,7,5],[3,4,-12]])
B = np.array([[9],[15],[35]])
```

...continuación del programa

Método 1:
invA = np.linalg.inv(A)
X1 = invA.dot(B)

Método 2:
X2 = np.linalg.solve(A,B)

Método 3:
Métodos iterativos como el algoritmo de Thomas (TDMA), Gauss-Seidel, Jacobi...

```
print('La solución del sistema es:',X1)
print('La solución del sistema es:',X2)
```

```
La solución del sistema es: [[-0.58226371]
 [ 3.22870478]
 [-1.98599767]]
La solución del sistema es: [[-0.58226371]
 [ 3.22870478]
 [-1.98599767]]
```

Python

FUNCIONES

```
def NOMBRE(LISTA_DE_PARAMETROS):
    """DOCSTRING_DE_FUNCION"""
    SENTENCIAS
    RETURN [EXPRESION]
```

NOMBRE, es el nombre de la función.

LISTA_DE_PARAMETROS, es la lista de parámetros que puede recibir una función.

DOCSTRING_DE_FUNCION, es la cadena de caracteres usada para documentar la función.

SENTENCIAS, es el bloque de sentencias en código fuente Python que realizar cierta operación dada.

RETURN, es la sentencia return en código Python.

EXPRESION, es la expresión o variable que devuelve la sentencia return.

```
import numpy as np

def volumen_cilindro(d,h):
    r = 0.5*d
    vol = (np.pi*r**2)*h
    return vol

diametro = 2.0
altura = 5.0

volumen = volumen_cilindro(diametro,altura)
print("El volumen del cilindro es:",volumen)
```

Python

FUNCIones

```
def f1(a,b,c):  
    y = a+b+c  
    return y  
  
n1 = 1.5  
n2 = 3.9  
n3 = 4.0  
x = f1(n1,n2,n3)
```

```
def f2(a,b,c,y):  
    y = y*(a+b+c)  
    return y
```

```
n1 = 1.5  
n2 = 3.9  
n3 = 4.0  
x = 5.0  
x = f2(n1,n2,n3,x)
```

```
def f3(a,b,c,y1,y2):  
    y1 = y1*(a+b+c)  
    y2 = y1*a  
    return y1,y2
```

```
n1 = 1.5  
n2 = 3.9  
n3 = 4.0  
x1 = 5.0  
x2 = 6.0  
[x1,x2] = f3(n1,n2,n3,x1,x2)
```

Python

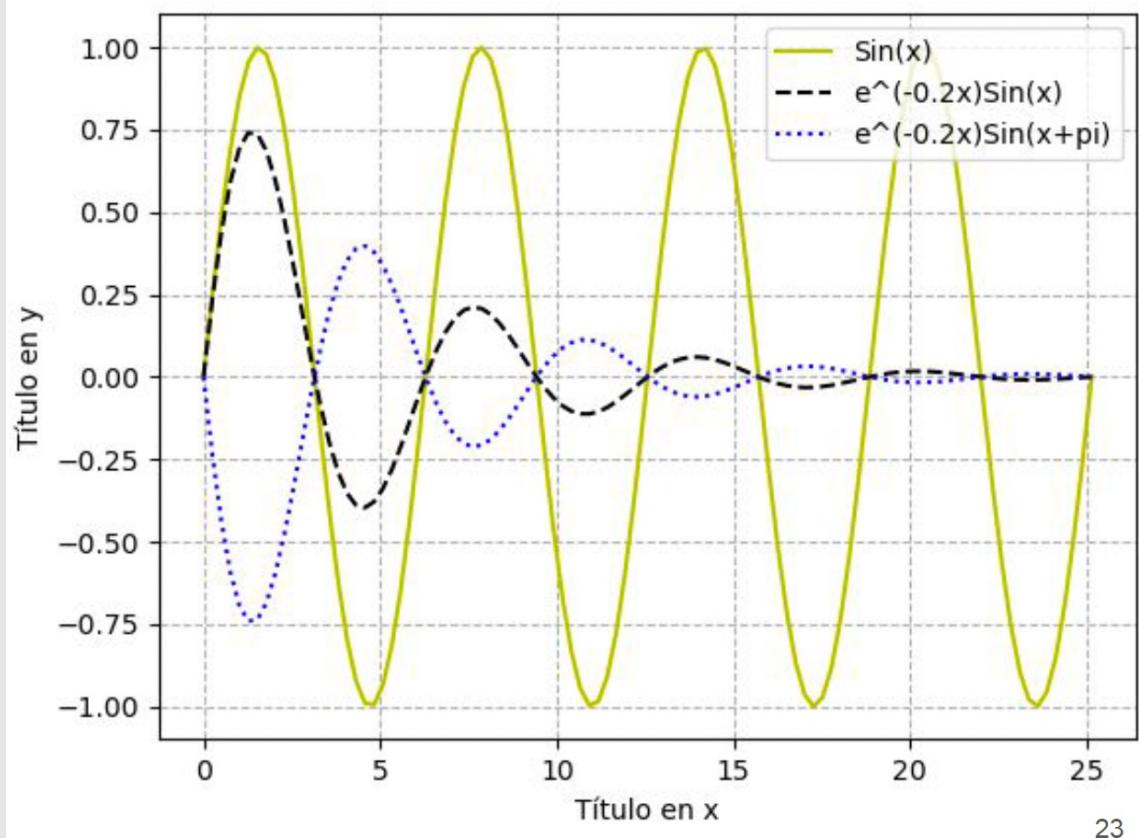
GRAFICAR CON MATPLOTLIB

```
import numpy as np
import matplotlib.pyplot as plt

pi = np.pi

x = np.linspace(0,8*pi,100)

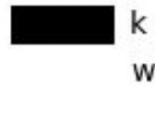
plt.plot(x,np.sin(x),"-y",label="Sin(x)")
plt.plot(x,np.exp(-0.2*x)*np.sin(x),"--k",
         label="e^(-0.2x)Sin(x)")
plt.plot(x,np.exp(-0.2*x)*np.sin(x+pi),
         ":b",label="e^(-0.2x)Sin(x+pi)")
plt.legend(loc="upper right")
plt.xlabel('Título en x')
plt.ylabel('Título en y')
plt.grid(style='--')
```



Python

GRAFICAR CON MATPLOTLIB

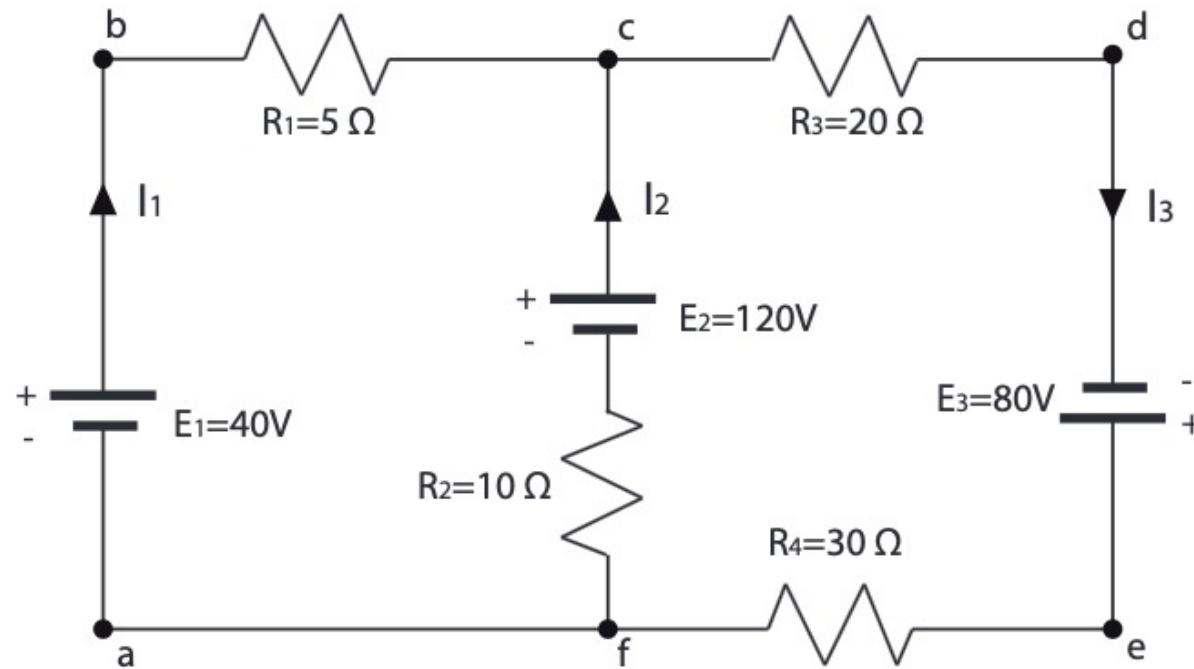
Base Colors



Location String	Location Code
'best'	0
'upper right'	1
'upper left'	2
'lower left'	3
'lower right'	4
'right'	5
'center left'	6
'center right'	7
'lower center'	8
'upper center'	9
'center'	10

character	description
'-'	solid line style
'--'	dashed line style
'-. '	dash-dot line style
':'	dotted line style
'.'	point marker
', '	pixel marker
'o'	circle marker
'v'	triangle_down marker
'^'	triangle_up marker
'<'	triangle_left marker
'>'	triangle_right marker
'1'	tri_down marker
'2'	tri_up marker
'3'	tri_left marker
'4'	tri_right marker
's'	square marker
'p'	pentagon marker
'*'	star marker
'h'	hexagon1 marker
'H'	hexagon2 marker
'+'	plus marker
'x'	x marker
'D'	diamond marker
'd'	thin_diamond marker
' '	vline marker
'_'	hline marker

Ley de Kirchhoff Determine los valores de las corrientes I_1 , I_2 e I_3 , de la siguiente figura:



Python

EJERCICIOS

Resolviendo obtendriamos el siguiente sistema de ecuaciones
(Verifique):

$$I_1 + I_2 - I_3 = 0$$

$$I_1 - 2I_2 = -16$$

$$I_2 + 5I_3 = 20$$

Estas ecuaciones conducen al sistema lineal:

$$\begin{bmatrix} 1 & 1 & -1 \\ 1 & -2 & 0 \\ 0 & 1 & 5 \end{bmatrix} \begin{bmatrix} I_1 \\ I_2 \\ I_3 \end{bmatrix} = \begin{bmatrix} 0 \\ -16 \\ 20 \end{bmatrix}$$

Al resolver para I_1 , I_2 e I_3 obtenemos:

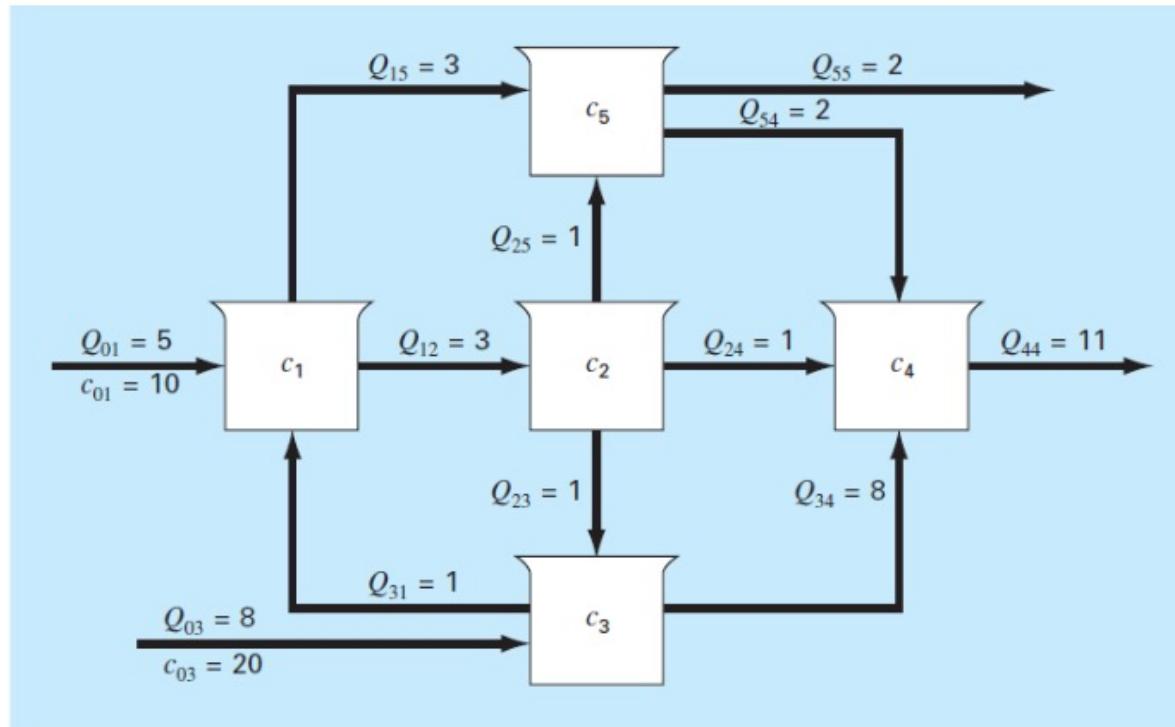
$$I_1 = -3,5A \quad I_2 = 6,25A \quad I_3 = 2,75A$$



Python

EJERCICIOS

Ejercicio 3 Determine la concentración en cada uno de los reactores mostrados en la figura. Considere el sistema en estado estable



Python

EJERCICIOS

$$-6 c_1 + c_3 = -50$$

$$3 c_1 - 3 c_2 = 0$$

$$c_2 - 9 c_3 = -160$$

$$c_2 + 8 c_3 + 2 c_5 - 11 c_4 = 0$$

$$3 c_1 + c_2 - 4 c_5 = 0$$

Python

BIBLIOGRAFÍA ADICIONAL

En las siguientes páginas web podrán complementar más herramientas que proporciona este lenguaje, en dado caso que la información mostrada en estas diapositivas no les sea suficiente:

Python Tutorial

https://www.w3schools.com/python/python_arrays.asp

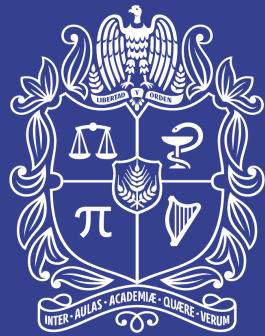
Numpy Tutorial

<https://numpy.org/doc/stable/user/quickstart.html#array-creation>

Gracias

Universidad Nacional de Colombia

PROYECTO CULTURAL, CIENTÍFICO Y COLECTIVO DE NACIÓN



UNIVERSIDAD
NACIONAL
DE COLOMBIA

PROYECTO CULTURAL, CIENTÍFICO Y COLECTIVO DE NACIÓN

Aproximaciones y errores de solución

Profesor: Carlos A. Duque Daza

Asistente docente: Sebastián Gómez P.

Facultad de Ingeniería – Departamento de Ingeniería Mecánica y
Mecatrónica – Sede Bogotá

Universidad Nacional de Colombia

PROYECTO CULTURAL, CIENTÍFICO Y COLECTIVO DE NACIÓN

Aproximación

Empleo de un valor cercano al verdadero,
¿puede ser este usado como cierto?

Aproximación

Empleo de un valor cercano al verdadero,
¿puede ser este usado como cierto?

¿En Bogotá, cuánto tiempo se gasta en
desplazamiento en transporte público?



Aproximación

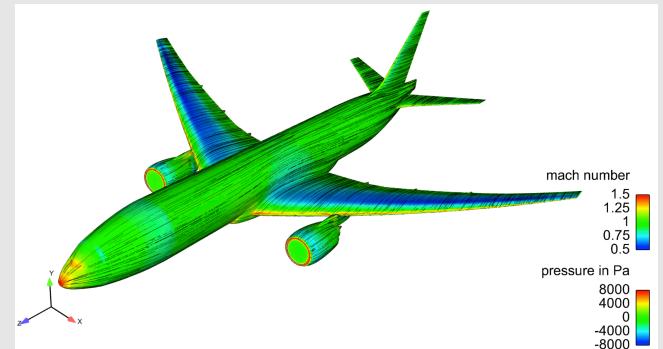
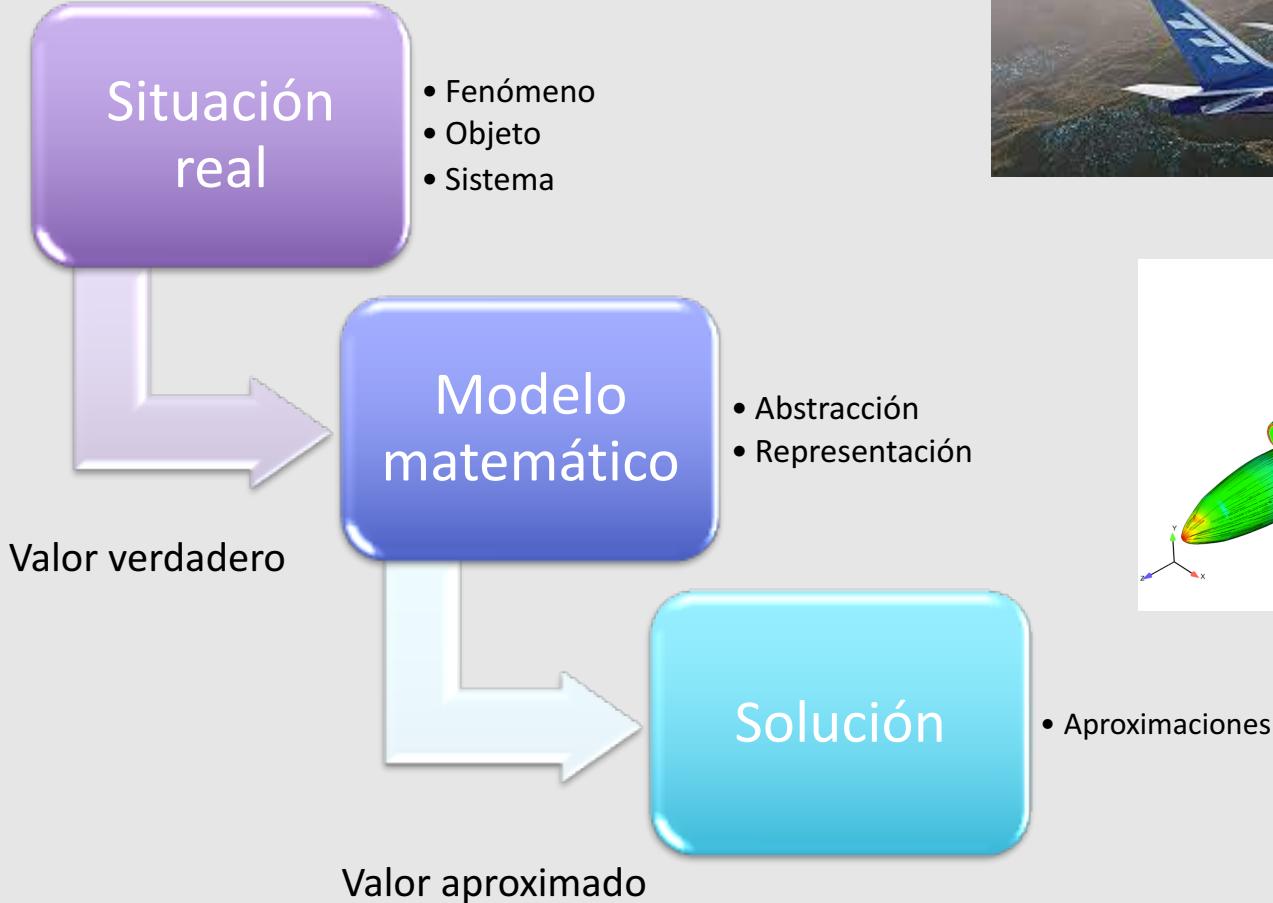
Empleo de un valor cercano al verdadero,
¿puede ser este usado como cierto?

¿En Bogotá, cuánto tiempo se gasta en
desplazamiento en transporte público?

La media de tiempo que las personas en
Bogotá emplean desplazándose en transporte
público, en un día entre semana es **64 min.**

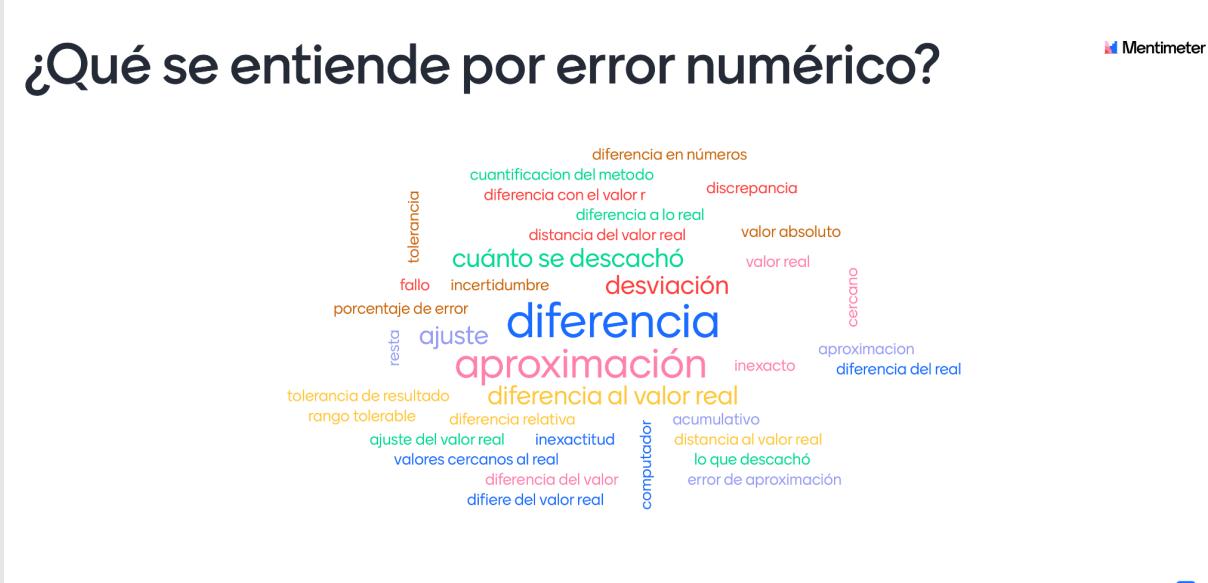


Aproximación



Definición del error

¿QUÉ SE ENTIENDE POR ERROR NUMÉRICO?



- Diferencia entre valor verdadero y valor aproximado.
- De acuerdo a Chapra&Canale (2006), "Los errores numéricos surgen del uso de aproximaciones para representar operaciones y cantidades matemáticas exactas".

Definición del error

ERROR VERDADERO

- Diferencia entre valor verdadero y valor aproximado.

$$\text{Error} = \text{Valor verdadero} - \text{Valor aproximado}$$

- Desventaja del error verdadero



No toma en consideración el orden de magnitud del valor que se desea estimar.

Definición del error

ERROR RELATIVO VERDADERO

- Normalizando el error verdadero:

$$\text{Error relativo verdadero} = \frac{\text{Error verdadero}}{\text{Valor verdadero}}$$

- Desventaja 2: sólo se puede calcular cuando existe información de la solución verdadera o solución analítica: estimación del error.

Definición del error

ERROR APROXIMADO

$$Error_a = \frac{Error\逼近ado}{Valor\逼近ado}$$

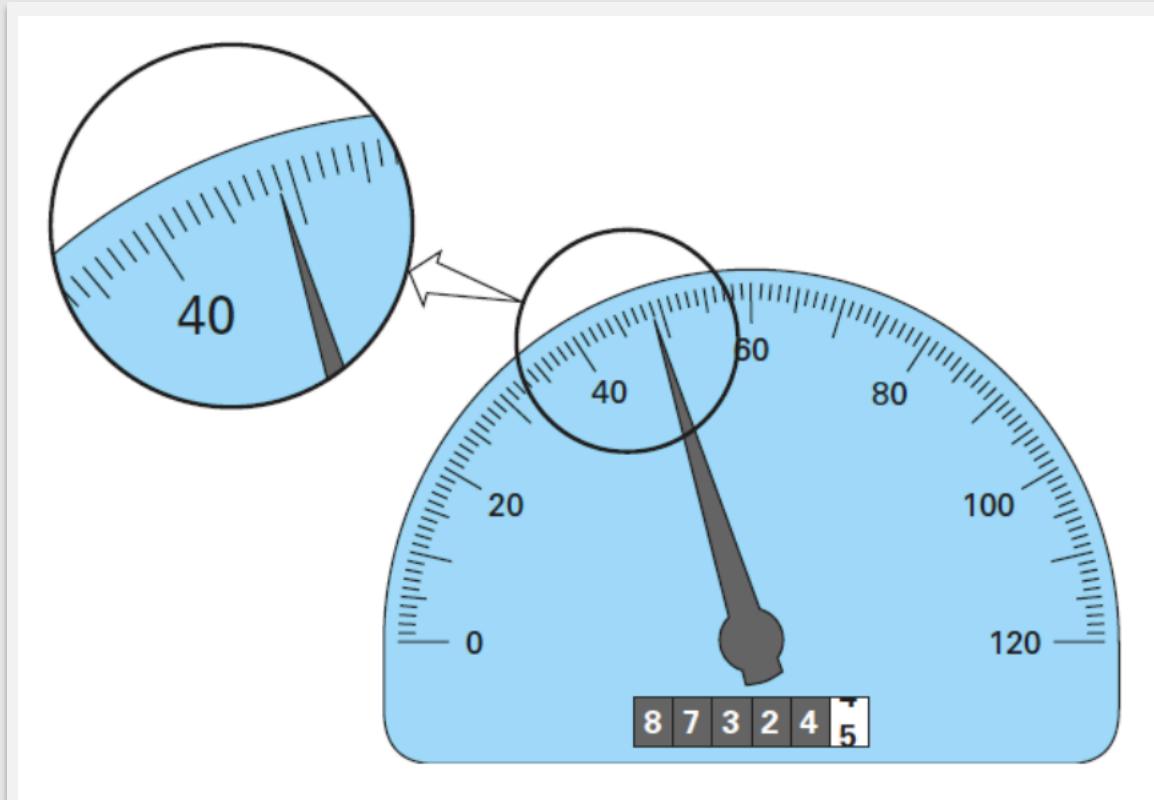
$$Error_a = \frac{AproxActual - AproxAnterior}{AproxActual}$$

Uso de esquemas iterativos donde se obtiene una aproximación actual sobre la base de una aproximación anterior.

Errores de solución

CIFRAS SIGNIFICATIVAS

Las cifras significativas, corresponden a la confianza con la que se puede leer un número. Ejemplo:



49 km/h

87324,5 km

Velocímetro y Odómetro (Chapra&Canale,2006)

Errores de solución

CIFRAS SIGNIFICATIVAS

Según Chapra&Canale, (2006): “aquellas que pueden utilizarse en forma confiable. Se trata del número de dígitos que se ofrecen con certeza, más uno estimado.”

Nota: Por convención, al dígito estimado se le da el valor de la mitad de la menor escala del instrumento de medición.

Tipos de errores

ERRORES DE REDONDEO

Surgen de una representación aproximada de un número exacto. Los computadores usan un determinado número de cifras significativas al momento de realizar un cálculo.

$$\pi, e, \sqrt{7}$$

A la discrepancia de cifras significativas respecto al valor exacto se le denomina **error de redondeo**.

Nota: La cantidad de cifras significativas que maneja un ordenador está directamente relacionada con la forma en la que se guardan los números en la memoria (cantidad de bits)

Tipos de errores

FORMAS DE REDUCIR ERRORES DE REDONDEO

1. Hacer uso de precisión extendida.
2. Evitar sumar números muy grandes con números muy pequeños.
3. Evitar realizar operaciones aritméticas con valores o datos que se encuentren muy dispersos. Ejemplo: Una sumatoria donde los términos de suma sean mucho mayores que la sumatoria total.

Tipos de errores

ERRORES POR EQUIVOCACIÓN

Atribuibles a fallas humanas o negligencia en varios momentos:

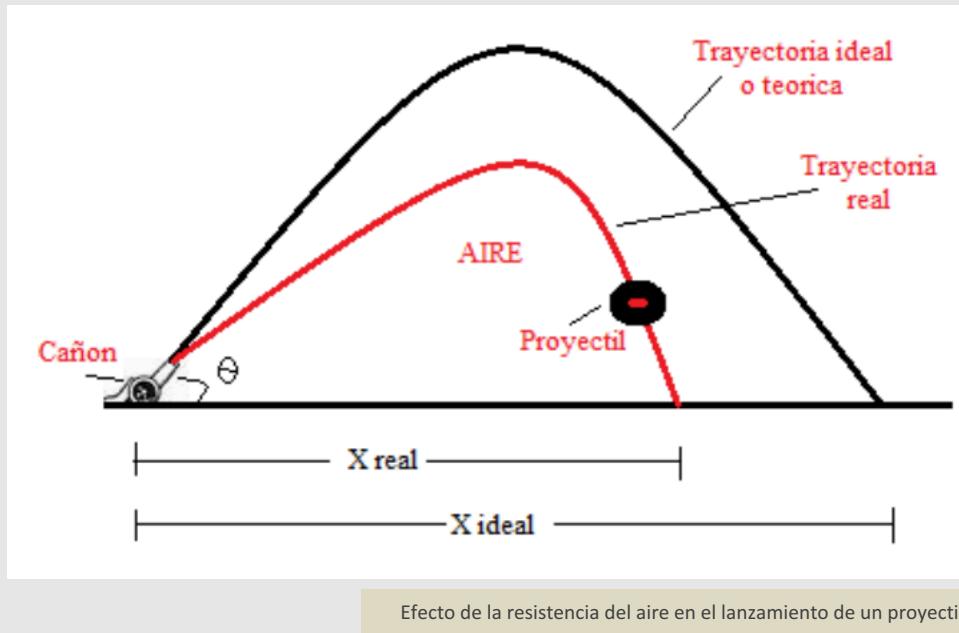
- Desarrollo del modelo matemático.
- Aplicación del método de solución.
- Tratamiento y análisis de los resultados.
- Otros inherentes a fallas humanas.



Tipos de errores

ERRORES DE FORMULACIÓN

Se atribuyen a el uso de modelos matemáticos que no describen y/o representan correctamente algún fenómeno de interés.



Ejemplo: Al lanzar un proyectil, suponer que la resistencia del aire es despreciable.

Tipos de errores

ERRORES DE TRUNCAMIENTO

“Resultan al usar una aproximación en lugar de un procedimiento matemático exacto”, Chapra&Canale,(2006).

SERIES DE TAYLOR

$$f(x) \approx f(x_0) + f'(x_0) \cdot (x - x_0) + \frac{f''(x_0)}{2!} \cdot (x - x_0)^2 + \dots + \frac{f^{(n)}(x_0)}{n!} \cdot (x - x_0)^n$$

$$f(x) \approx \sum_{n=0}^{\infty} \frac{f^{(n)}(x_0)}{n!} \cdot (x - x_0)^n$$

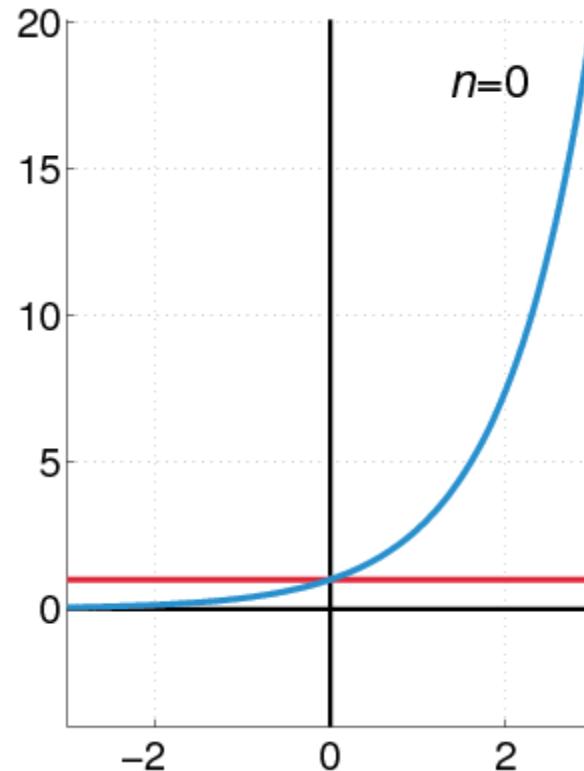
Aproximar una función usando una serie de polinomios.

Mientras más términos, más exacto el resultado.

Tipos de errores

ERRORES DE TRUNCAMIENTO – SERIE DE TAYLOR

- Número infinito de términos: imposible de evaluar computacionalmente
- En la práctica se debe decidir cuántos términos se usan.
- La eliminación de términos se conoce como truncamiento



Representación numérica en computador

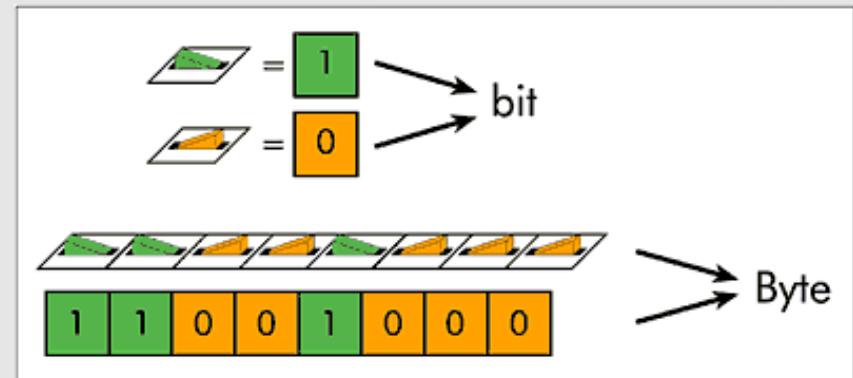
DEFINICIONES

BIT: “Binary Digit” : Hace referencia a un dígito binario. Es la unidad de almacenamiento (0 ó 1)

BYTE: Conjunto de 8 bits. Normalmente un número en Python / Matlab / Scilab ocupa 8 bytes, lo que corresponde a 64 bits.

PALABRA: Conjunto de bits que representan un número en **base 2**. Según el estándar IEEE 754 se puede almacenar un número usando:

- **32 bits:** precisión simple.
- **64 bits:** precisión doble



Representación numérica en computador

SISTEMA POSICIONAL

Dígitos decimales: 0, 1, 2, 3, ..., 9. α

$$a \cdot 10^{n-1} \quad \leftarrow \cdot \rightarrow \quad a \cdot 10^{-n}$$

472.83
321

En general, la representación decimal

$$\begin{array}{l} s=0 \\ s=1 \end{array}$$

$$(-1)^s (a_n a_{n-1} \dots a_1 a_0, a_{-1} a_{-2})$$

Corresponde al número

Representación numérica en computador

BASES NUMÉRICAS

Representación de un número en base 10:

$$0.\underline{1}0\underline{1}_{10} = \underline{1} \cdot \underline{10}^{-1} + 0 \cdot \underline{10}^{-2} + 1 \cdot \underline{10}^{-3} = \frac{1}{10} + \frac{1}{1000} = 0.101$$

Representación de un número en base 2:

$$0.101_2 = 1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3} = 0.625_{10}$$

$$\begin{aligned} 11011.01_2 &= 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2} \\ &\stackrel{s}{=} 27.25_{10} \end{aligned}$$

¿Cómo es en general?

Representación numérica en computador

BASES NUMÉRICAS

En general,

$$x = (a_n a_{n-1} \dots a_1 a'_0 \cdot a_{-1} a_{-2} \dots a_{-m})_2 = \\ a_n \cdot 2^n + a_{n-1} \cdot 2^{n-1} + \dots + a_1 \cdot 2^1 + a_0 \cdot 2^0 + a_{-1} \cdot 2^{-1} + a_{-2} \cdot 2^{-2} + \dots + a_{-m} \cdot 2^{-m}$$

Representación en base 2 de un número entero

$$x = (a_n a_{n-1} \dots a_1 a_0)_2 = a_n \cdot 2^n + a_{n-1} \cdot 2^{n-1} + \dots + a_1 \cdot 2^1 + a_0 \cdot 2^0 \\ - \quad - \quad - \\ 2 \left(\underbrace{a_n \cdot 2^{n-1} + a_{n-1} \cdot 2^{n-2} + \dots + a_1}_x \right) + a_0 \\ r_0$$
$$x = x_1 + r_0$$

Representación numérica en computador

BASES NUMÉRICAS

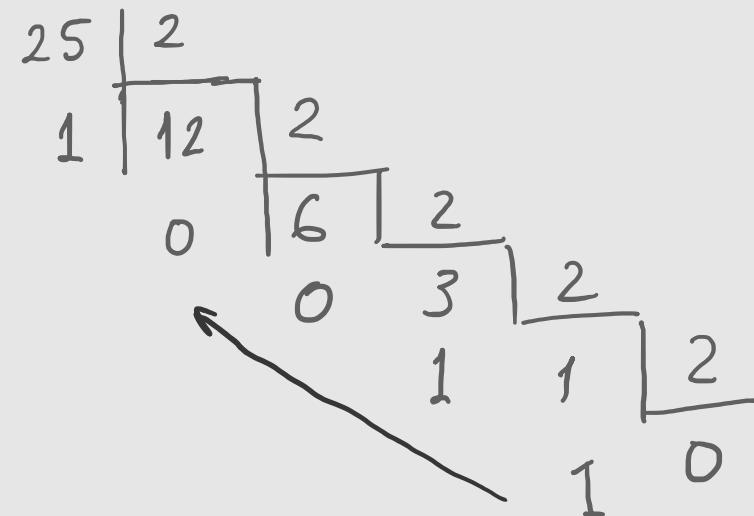
Para hallar el siguiente dígito, se aplica el mismo procedimiento:

$$x_1 = \underbrace{a_n \cdot 2^{n-1} + a_{n-1} \cdot 2^{n-2} + \cdots + a_1}_{x_2} = 2 \cdot x_2 + r_1$$

Ejemplo: Representación en base 2

$$x = 25$$

$$25_{10} = (11001)_2$$



Representación numérica en computador

BASES NUMÉRICAS

Representación en base 2 de la parte fraccionaria . →

$$x = (. a_{-1} a_{-2} \dots a_{-m})_2 = a_{-1} \cdot 2^{-1} + a_{-2} \cdot 2^{-2} + \dots + a_{-m} \cdot 2^{-m}$$

$$x = \frac{1}{2} (a_{-1} + a_{-2} \cdot 2^{-1} + \dots + a_{-m} \cdot 2^{-m+1})$$

$$2x = a_{-1} + a_{-2} \cdot 2^{-1} + \dots + a_{-m} \cdot 2^{-m+1}$$

$$x = 0.8125$$

$$2x = 1.\underline{625}$$

$$a_{-1} = 1$$

$$x_1 = 0.625$$

$$2x_1 = 1.\underline{25}$$

$$a_{-2} = 1$$

$$x_2 = 0.25$$

$$2x_2 = 0.50$$

$$a_{-3} = 0$$

$$x_3 = 0.50$$

$$2x_3 = 1.\underline{00}$$

$$a_{-4} = 1$$

$$(0.8125)_{10} \rightarrow (0.1101)_2$$

Representación numérica en computador

REPRESENTACIÓN BASE BETA

En principio, cualquier número natural ≥ 2 puede ser usado como base.

Con una base fija, todo número real admite una representación posicional en base beta. Normalización:

$$x \in \mathbb{R}$$

$$x = \sigma \underbrace{(d_0 . d_1 d_2 \dots d_t)}_{\text{Mantisa}} \beta^{(e)_\beta} \quad \begin{array}{l} \sigma: \text{Signo } x \\ (e)_\beta: \text{exponente } \beta \end{array}$$

$$d_0 \quad 1 \quad \beta^{-1}$$

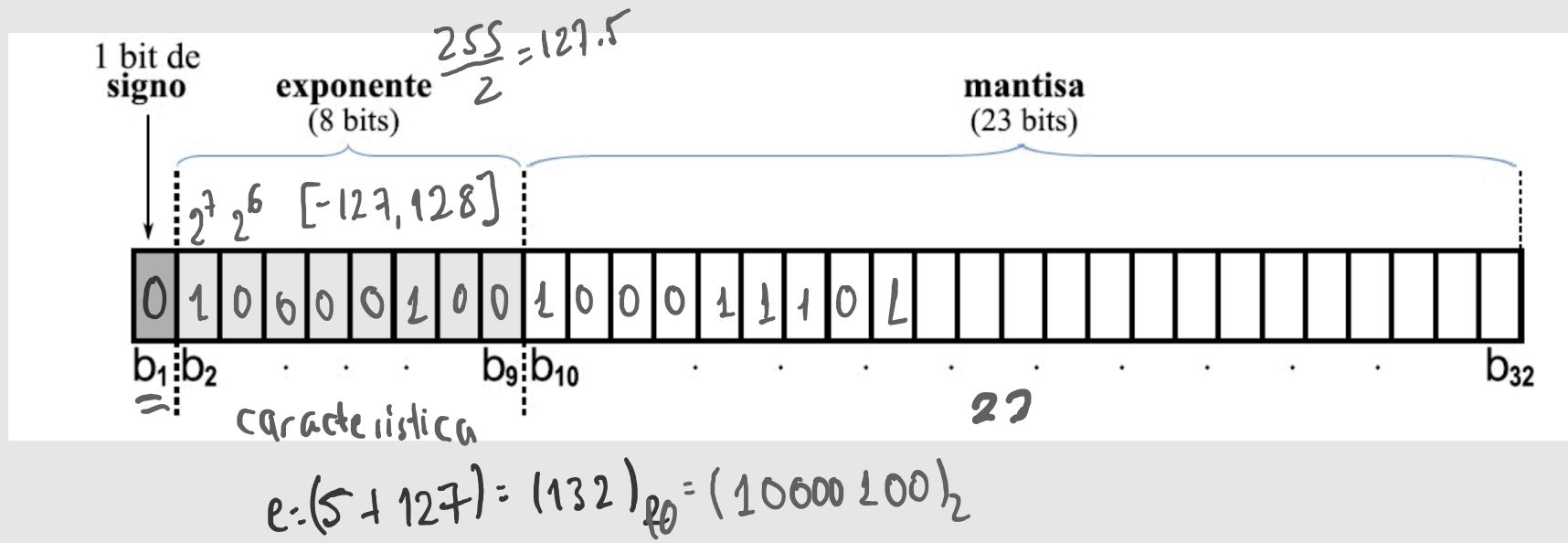
$$x = (49,8125)_{10}$$

$$x = (110001.1101)_2 = (\underline{1.100011101})_2 \cdot \underline{\underline{2^{(101)_2}}}$$

Representación numérica en computador

REPRESENTACIÓN PUNTO FLOTANTE

Estandar IEEE 754 para la representación en punto flotante en **precisión simple**, supone el uso de una palabra de 32 bits



Representación numérica en computador

REPRESENTACIÓN PUNTO FLOTANTE

Retomando el ejemplo,

$$x = \underbrace{(1.100011101)}_{2} \cdot 2^{(101)_2}$$

mantisa

Representación numérica en computador

ERROR – EPSILON DE LA MÁQUINA

Se define el épsilon de la máquina como la distancia entre 1 y el siguiente número representable mayor que 1.

De acuerdo al estándar IEEE 754, el épsilon de la máquina en precisión simple es $2^{-23} \approx 1.192092895507813e-007$, mientras que en precisión doble es $2^{-52} \approx 2.220446049250313e-016$

En general, si en la representación de punto flotante se usan n bits para almacenar la mantisa, entonces el épsilon de la máquina es 2^{-n} .

ERROR DE REDONDEO

Acotado por el épsilon de la máquina

Trabajo individual

ERRORES Y REPRESENTACIÓN NUMÉRICA

¿Cuál es el valor más grande y más pequeño que se pueden representar en precisión simple?

¿Cuál es el valor más grande y más pequeño que se pueden representar en precisión doble?

Quiz sobre errores y representación numérica. Fecha por definir

Gracias

Universidad Nacional de Colombia

PROYECTO CULTURAL, CIENTÍFICO Y COLECTIVO DE NACIÓN