Modelación Matemática: Introducción al curso

Director: Carlos Alberto Duque Daza
Expositor: Jair Hernando Tovar Tuirán

Universidad Nacional de Colombia
Departamento de Ingeniería Mecánica y Mecatrónica

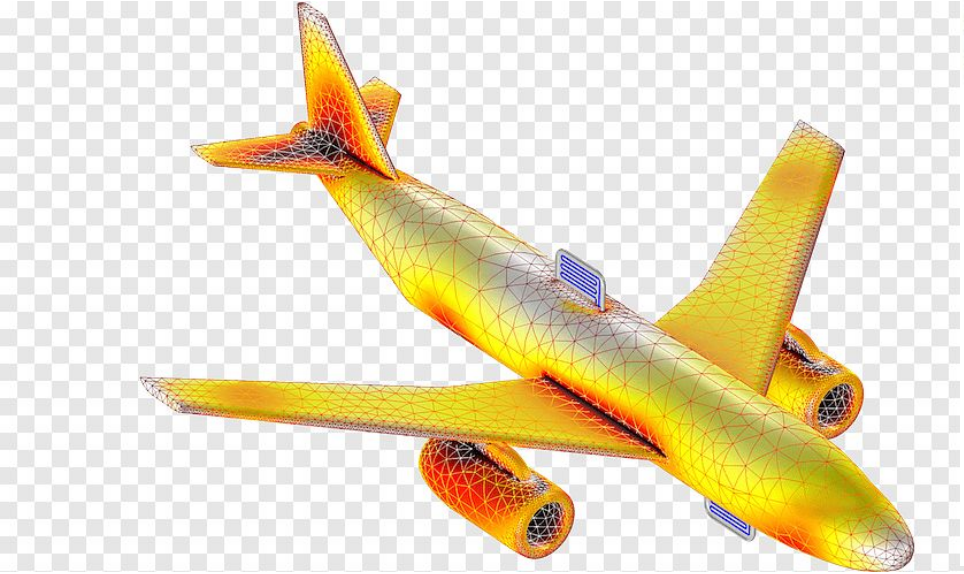
UNIVERSIDAD NACIONAL DE COLOMBIA

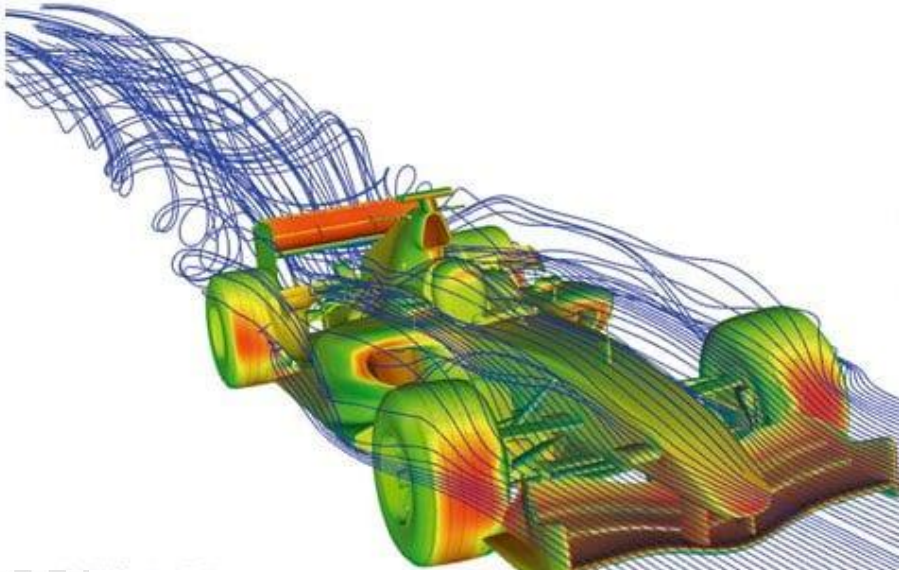
Contenido

- Motivación
- Partes del curso
  - Optimización
  - Solución de EDO
  - Solución de EDP
- Metodología

UNIVERSIDAD NACIONAL DE COLOMBIA

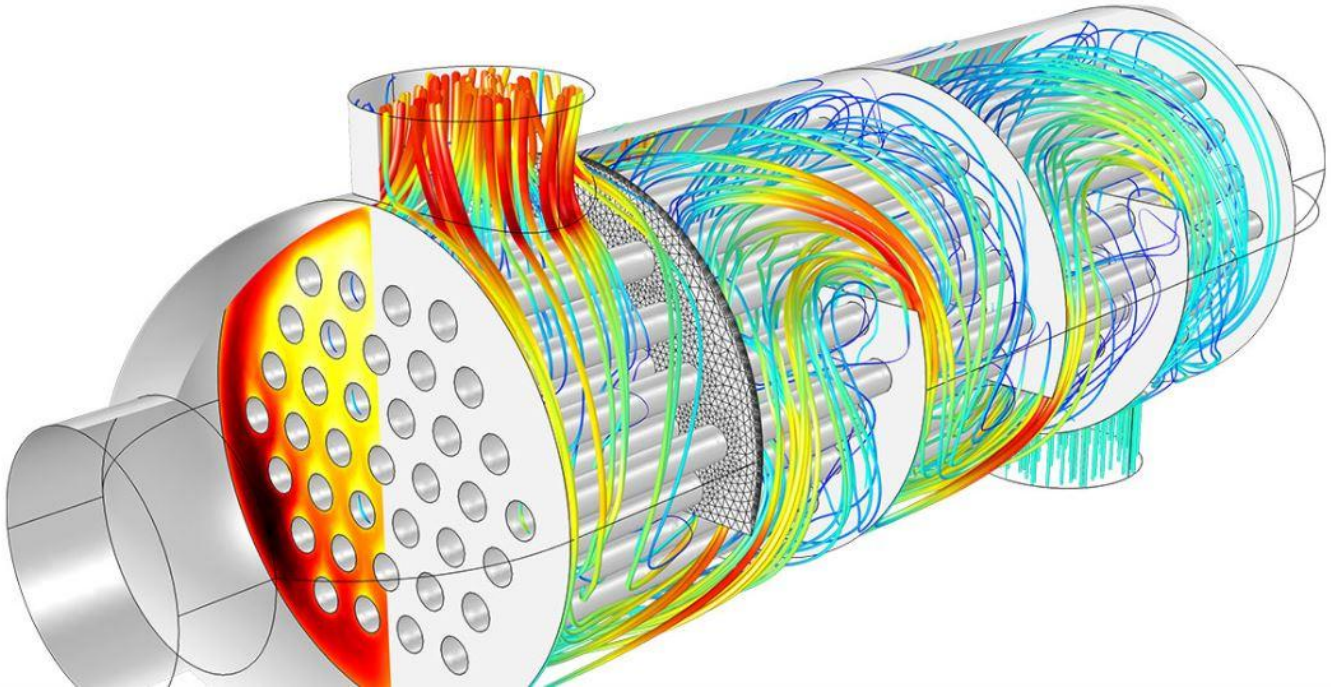
Motivación





UNIVERSIDAD NACIONAL DE COLOMBIA

Motivación

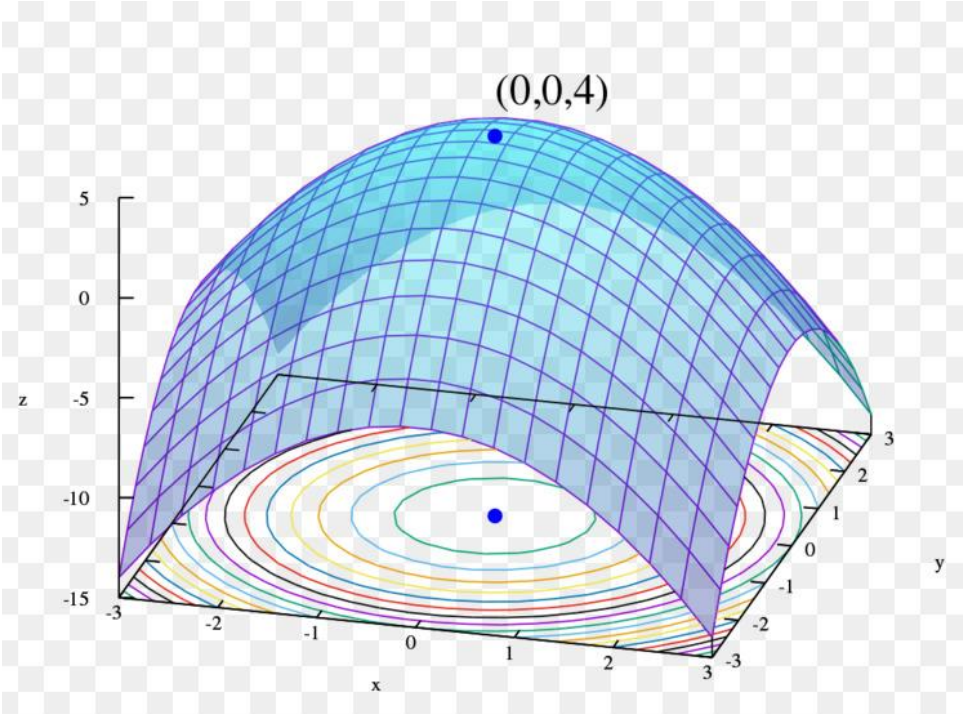


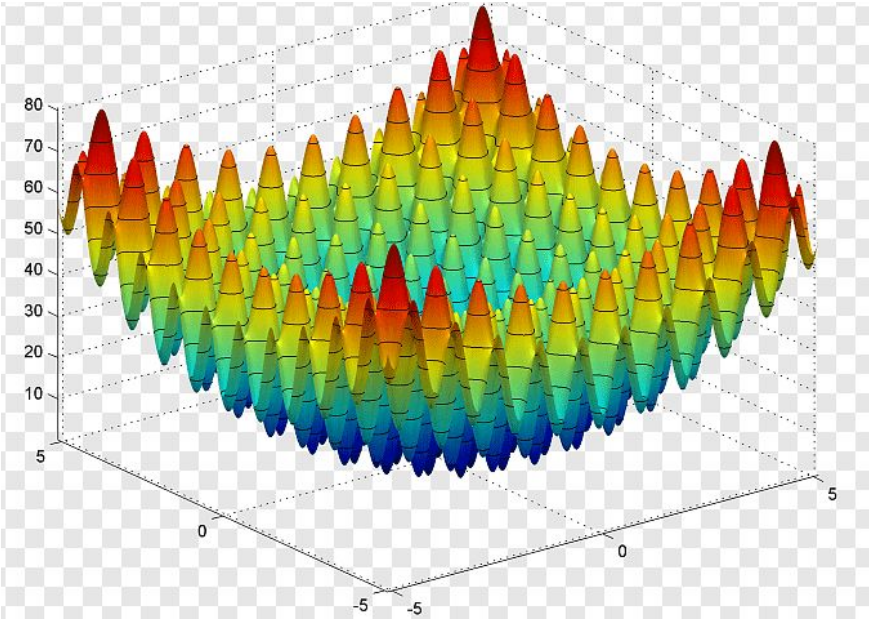
Optimización

- Es el método para encontrar la mejor solución de un problema a partir de un conjunto de soluciones factibles.
- Es un método para determinar los valores de las variables que intervienen en un proceso, para que el resultado sea el mejor posible.

UNIVERSIDAD NACIONAL DE COLOMBIA

Optimización



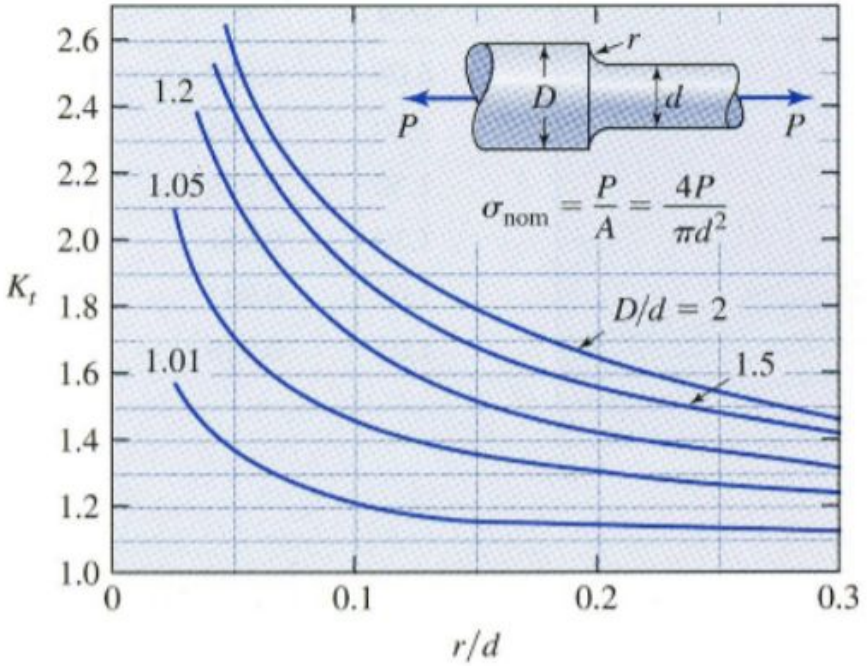


Optimización - Ejemplo

$$\sigma = \frac{P}{A} = \frac{4P}{\pi d^2}$$

Optimización - Ejemplo

$$\sigma = K_t\left(\frac{P}{A}\right) = K_t\left(\frac{4P}{\pi d^2}\right)$$

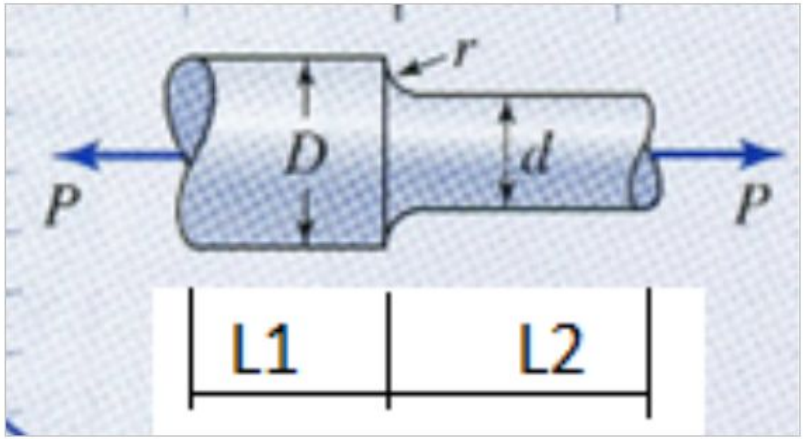


Approximate formula

$K_t \approx B\left(\frac{r}{d}\right)^a$, where:

D/d	B	a
2.00	1.015	−0.300
1.50	1.000	−0.282
1.20	0.963	−0.255
1.05	1.005	−0.171
1.01	0.984	−0.105

Optimización - Ejemplo



$$\sigma = K_t\left(\frac{P}{A}\right) = K_t\left(\frac{4P}{\pi d^2}\right)$$

Función a minimizar:

$$V = \left(\frac{\pi}{4}\right)(D^2 L_1 + d^2 L_2)$$

Sujeto a:

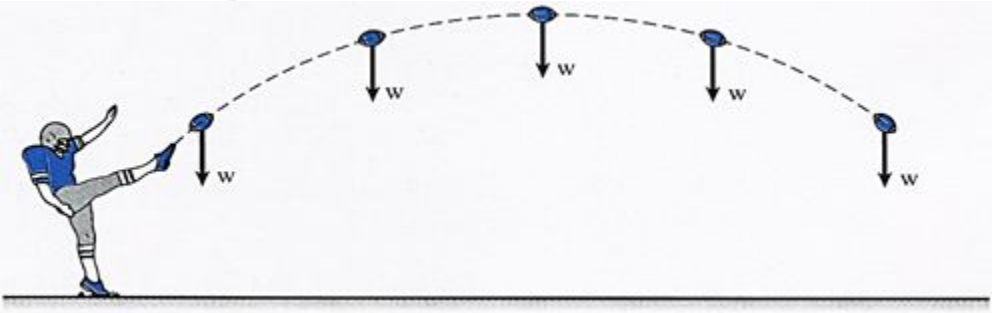
10 mm < D < 100 mm
10 mm < d < 70 mm
1 mm < r < 15 mm
r ≤ (D-d)/2
D/d ≥ 1.01
σ ≤ 90 MPa

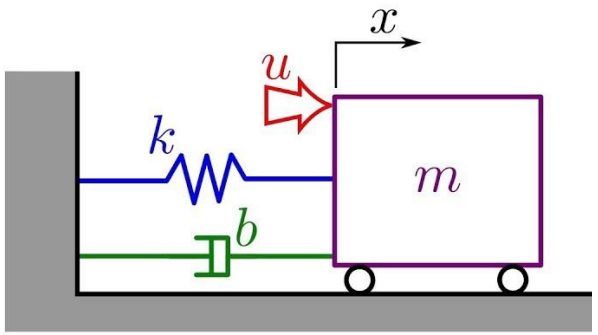
Constantes:

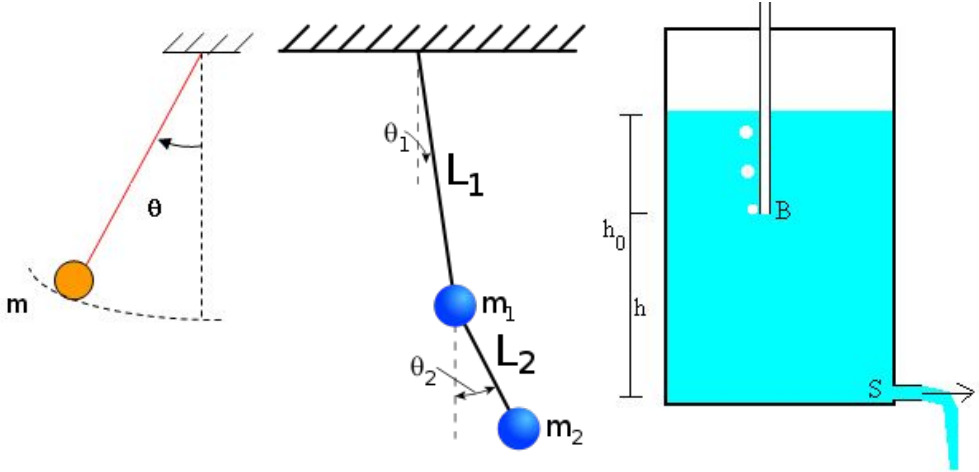
P = 130 kN
L1 = 160 mm
L2 = 220 mm

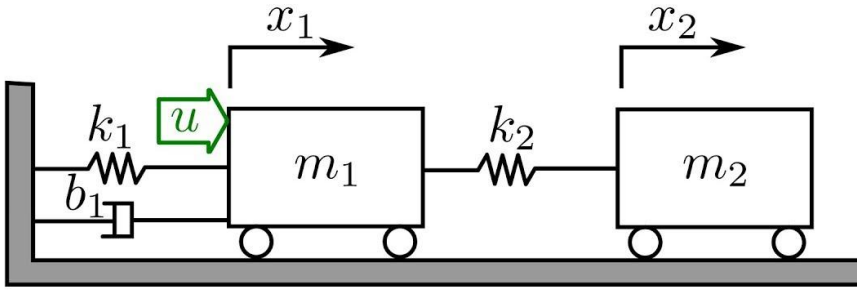
¿Cuáles son los D, d y r óptimos?

Ecuaciones diferenciales ordinarias









Ecuaciones diferenciales ordinarias

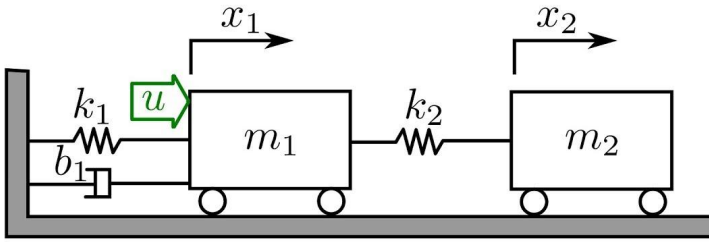
UNIVERSIDAD NACIONAL DE COLOMBIA

Una Ecuación Diferencial Ordinaria (EDO), es una ecuación que involucra una variable independiente (t), una función y(t) y una o varias derivadas de la misma.

$$\text{P.V.I} \begin{cases} \dfrac{dy}{dt} + f(t,y) = g(t) \\ \\ y(t=0) = y_0 \end{cases}$$

¿Cuál es la solución para la función y(t), formulado un P.V.I?

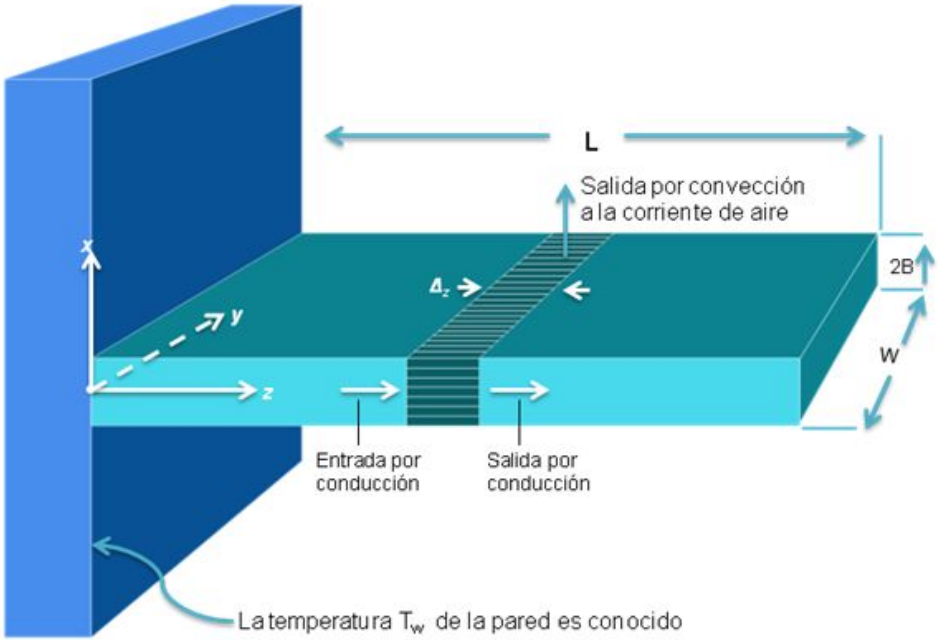
Ecuaciones diferenciales ordinarias

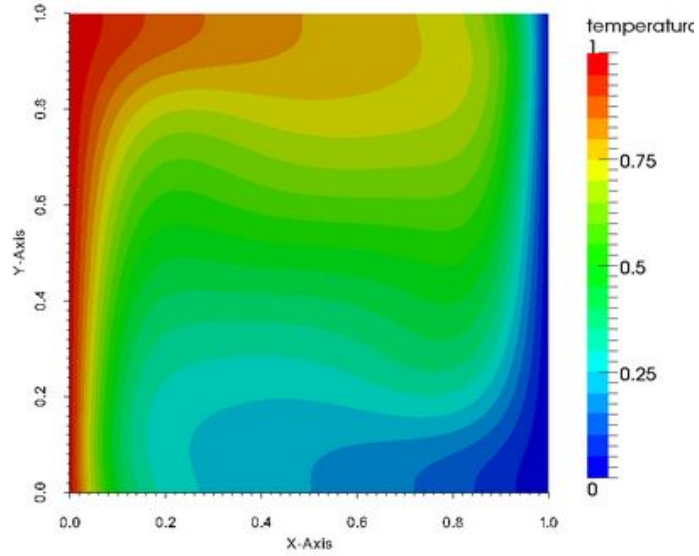


¿Cuáles son las funciones de posición de cada masa, formulado un P.V.I.?

$$\begin{cases} \dfrac{dy_1}{dt} + f_1(t, y_1, y_2) = g_1(t) \\ \dfrac{dy_2}{dt} + f_2(t, y_1, y_2) = g_2(t) \\ y_1(t=0) = y_{10} \\ y_2(t=0) = y_{20} \end{cases}$$

Ecuaciones diferenciales parciales





Ecuaciones diferenciales parciales

UNIVERSIDAD NACIONAL DE COLOMBIA

$$F(x_1, x_2, \ldots, x_n, u, \frac{\partial u}{\partial x_1}, \frac{\partial u}{\partial x_2}, \ldots, \frac{\partial u}{\partial x_n}, \frac{\partial^2 u}{\partial x_1 \partial x_1}, \frac{\partial^2 u}{\partial x_1 \partial x_2}, \cdots) = 0$$

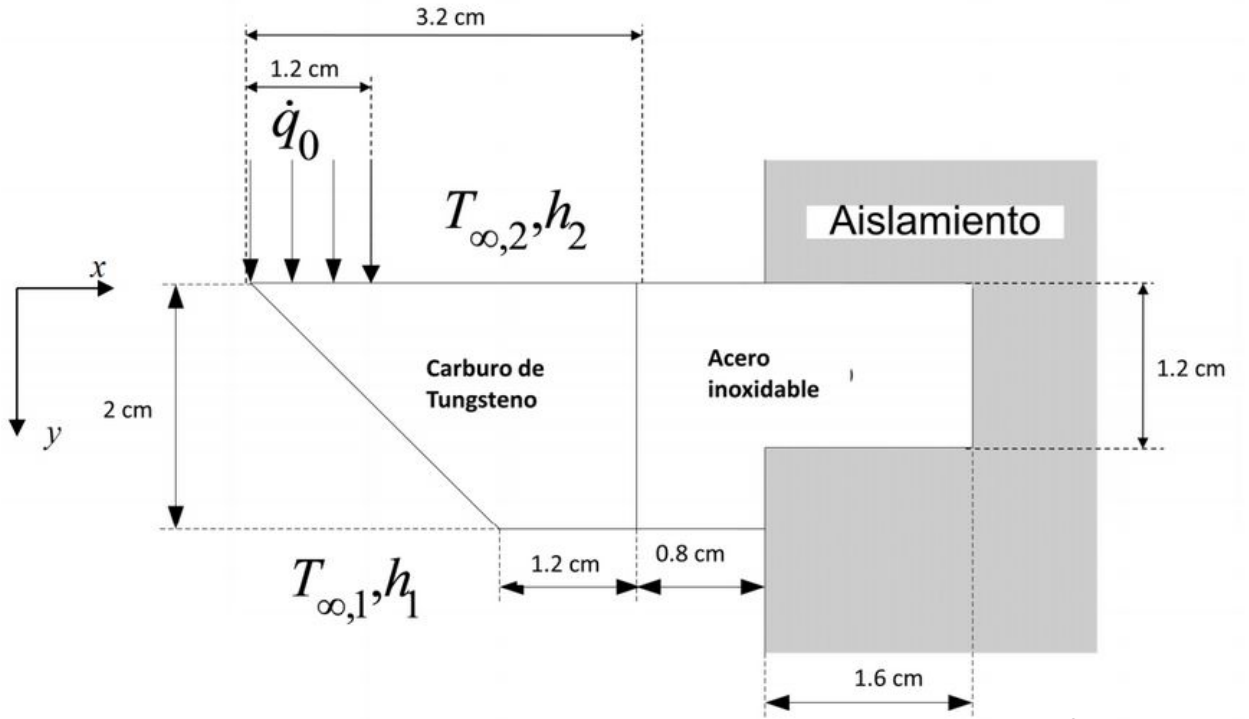
$$\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial x^2}$$

Ecuaciones diferenciales parciales

$$\frac{\partial T}{\partial t} = \alpha \left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \right)$$

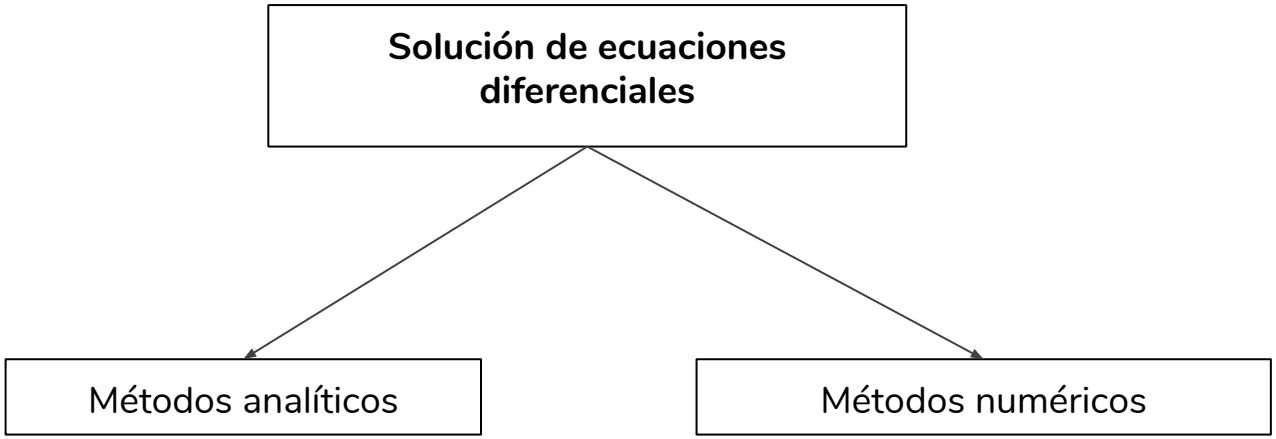
$$T \rightarrow f(t, x, y)$$

$$T(0, x, y) = T_0$$



¿Cuál será la temperatura en una coordenada (x,y) de la herramienta, para un tiempo de trabajo dado?

Metodología de solución



$$(1-t^2)\frac{dy}{dt} - ty = t(1-t^2)$$

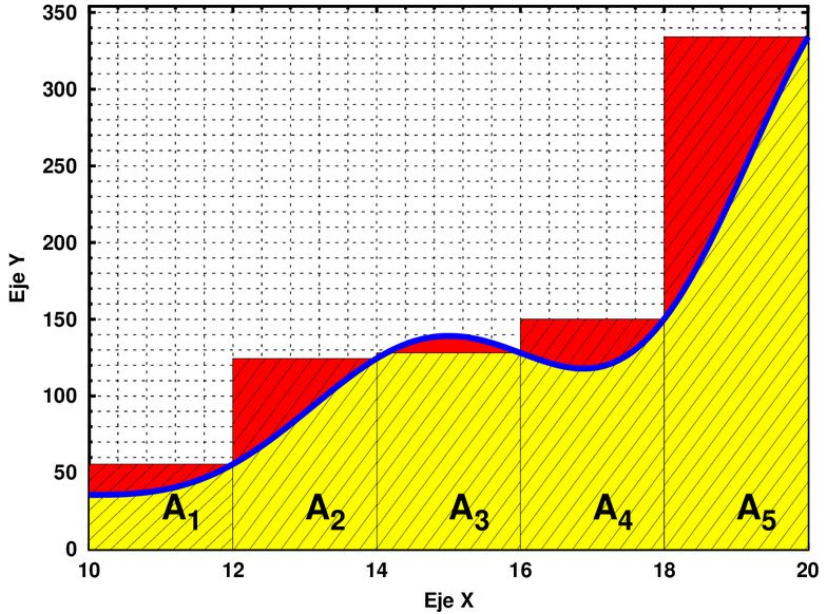
$$y(0) = 2$$

$$y(t) = \sqrt{1-t^2}(-cos(sen^{-1}(t)) + 3)$$

UNIVERSIDAD NACIONAL DE COLOMBIA

¿Qué es un método numérico?

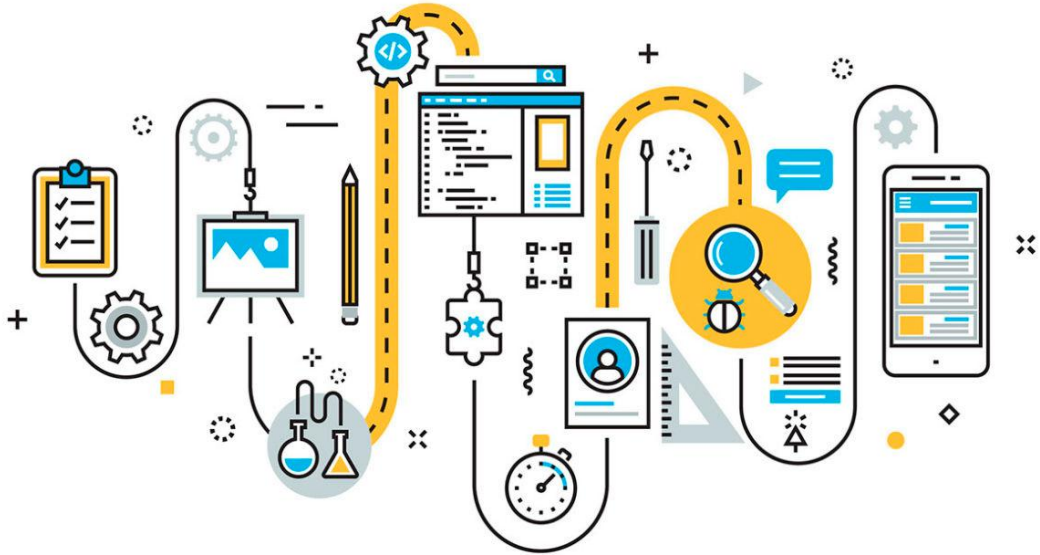
Los métodos numéricos son técnicas mediante las cuales se resuelven problemas matemáticos haciendo uso únicamente de operaciones aritméticas.



UNIVERSIDAD NACIONAL DE COLOMBIA

¿Qué es un algoritmo?

Un algoritmo es un conjunto de operaciones ordenadas y finitas que permite realizar un cálculo para la solución de un problema



UNIVERSIDAD NACIONAL DE COLOMBIA

Lenguajes de programación

¡GRACIAS POR SU ATENCIÓN!

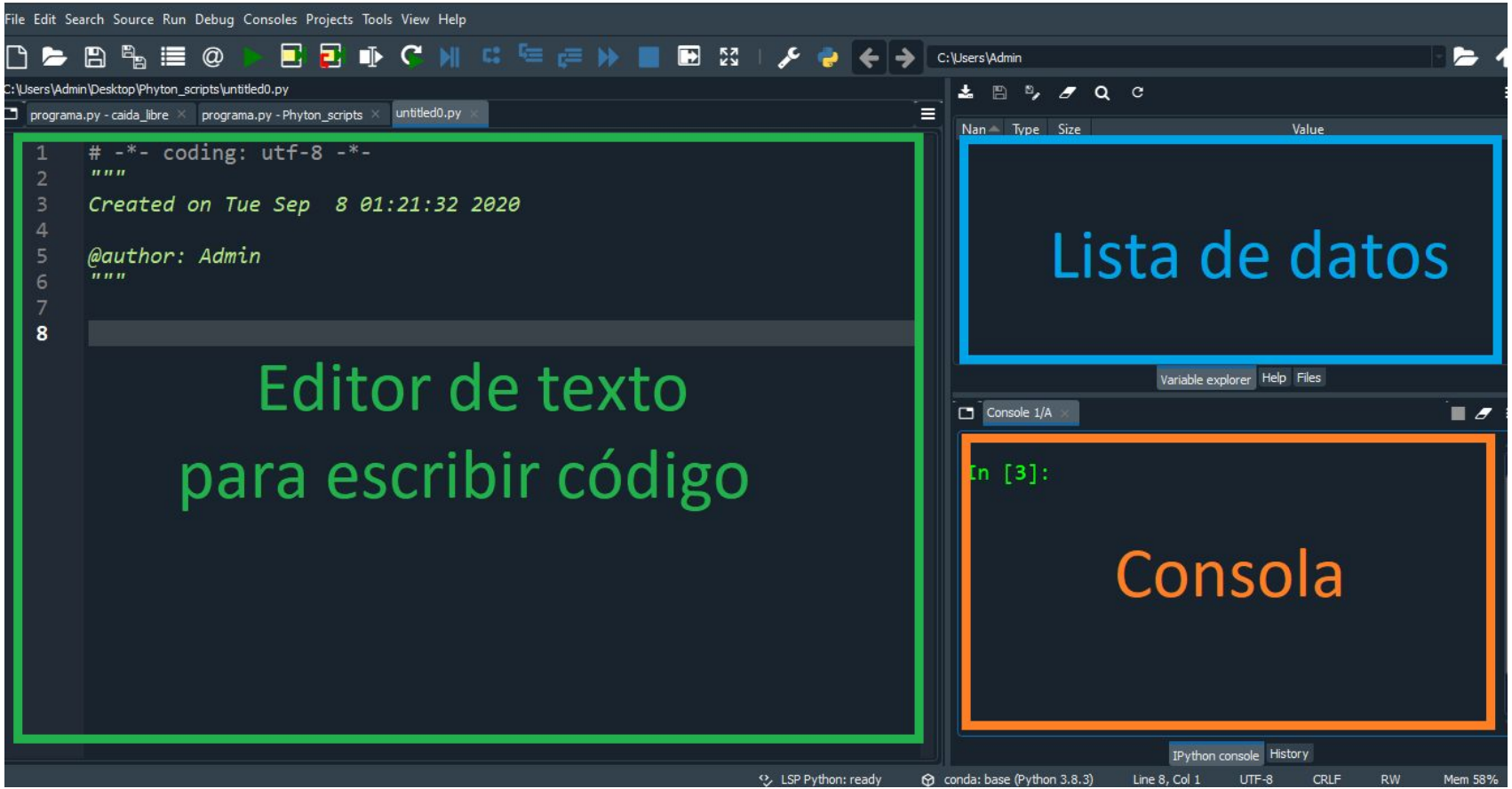
Correo: jhtovart@unal.edu.co

Modelación Matemática: Introducción a Python

Profesor: Jair Hernando Tovar Tuirán

UNIVERSIDAD NACIONAL DE COLOMBIA

Universidad Nacional de Colombia
Departamento de Ingeniería Mecánica y Mecatrónica


File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Admin
C:\Users\Admin\Desktop\Phyton_scripts\untitled0.py
programa.py - caida_libre
programa.py - Phyton_scripts
untitled0.py
1 # -*- coding: utf-8 -*-
2 """
3 Created on Tue Sep 8 01:21:32 2020
4
5 @author: Admin
6 """
7
8
Editor de texto
para escribir código
Nan Type Size Value
Lista de datos
Variable explorer Help Files
Console 1/A
In [3]:
Consola
IPython console History
LSP Python: ready
conda: base (Python 3.8.3)
Line 8, Col 1
UTF-8
CRLF
RW
Mem 58%

Sintaxis e indentación

Imprimir cadena de caracteres:

```
>>> print("Hello, World!")
Hello, World!
```

Python no utiliza comandos para declaración de variables:

```
x = 5
y = "Hello, World!"
```

```
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!")
```

Números 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

Python If... Else

```
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 while

```
while ('6' != opcion):
    opcion = input('''Por favor seleccione una operacion:
    1 Ver cuentas
    2 Crear cuenta
    3 Ver saldo
    4 Hacer ingreso
    5 Hacer retirada
    6 Exit
    ''')

    if opcion == '1':
        visualizarCuentas(clientes)
    elif opcion == '2':
        crearCuenta(clientes, numCuentas)
    elif opcion == '3':
        verSaldo(clientes)
    elif opcion == '4':
        hacerIngreso(clientes)
    elif opcion == '5':
        hacerRetirada(clientes)
    os.system("CLS")
```

Python 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.]])
```

La primera posición de un arreglo en Python es **CERO**

```
>>> from numpy import pi
>>> np.linspace( 0, 2, 9 )                 # 9 numbers from 0 to 2
array([0.  , 0.25, 0.5 , 0.75, 1.  , 1.25, 1.5 , 1.75, 2.  ])
>>> x = np.linspace( 0, 2*pi, 100 )        # useful to evaluate function a
t lots of points
>>> f = np.sin(x)
```

```
>>> 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]])
```

Python Arreglos

Inversión y transpuesta de una matriz

```
arr3 = np.array([[3.0,4.0,-1],[2.0,0.0,1.0],[1.0,3.0,-2.0]])

inv = np.linalg.inv(arr3)
tr = np.transpose(arr3)
ident = arr3@inv
```

```
In [25]: arr3
Out[25]:
array([[ 3.,  4., -1.],
       [ 2.,  0.,  1.],
       [ 1.,  3., -2.]])

In [26]: tr
Out[26]:
array([[ 3.,  2.,  1.],
       [ 4.,  0.,  3.],
       [-1.,  1., -2.]])

In [27]: inv
Out[27]:
array([[-0.6,  1. ,  0.8],
       [ 1. , -1. , -1. ],
       [ 1.2, -1. , -1.6]])
```

¿Cuál es el valor que corresponde a la posición [1,2] en la matriz arr3?

Python for

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

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

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

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

2  4.7
3  8.6
4  3.5
5  1.9
```

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 leer desde archivo de texto

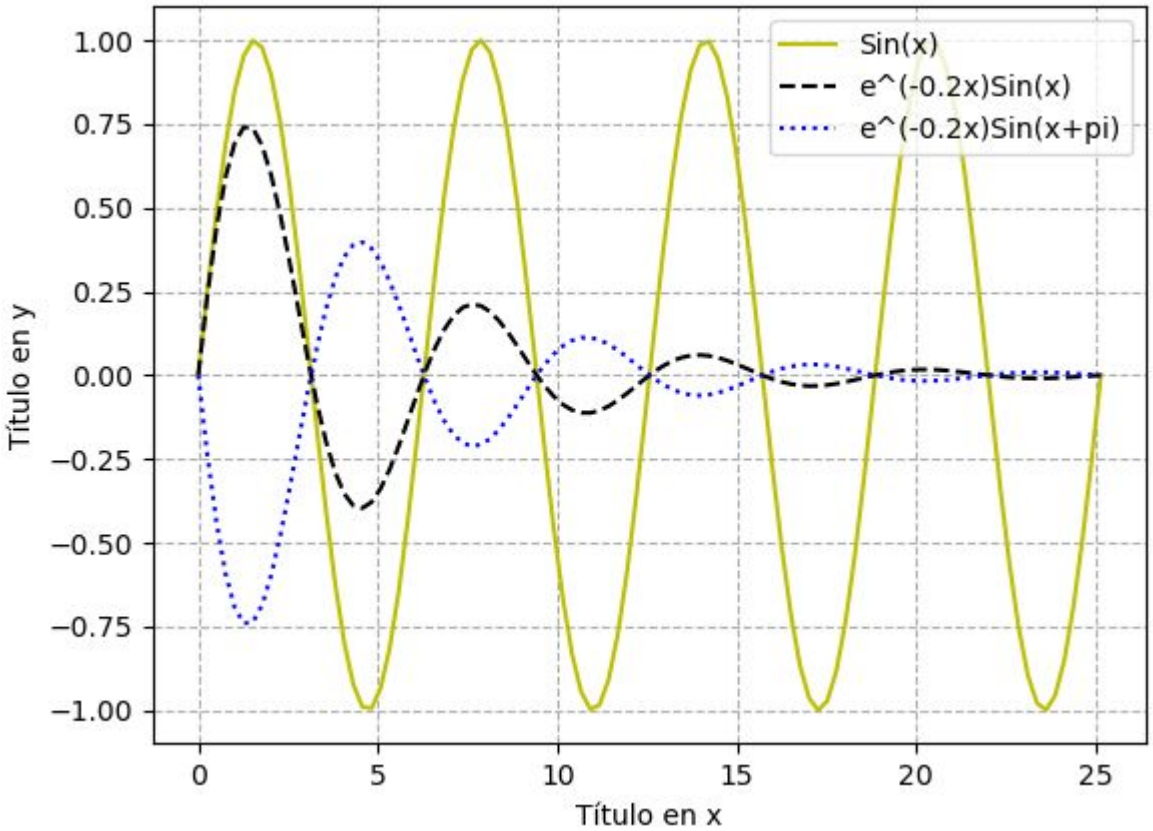
```
import numpy as np

mat = np.zeros((10,3))
j = 0
with open('datos.txt','r') as file:
    for line in range(10):
        c1,c2,c3 = [float(x) for x in next(file).split()]
        mat[line,:] = c1,c2,c3
```

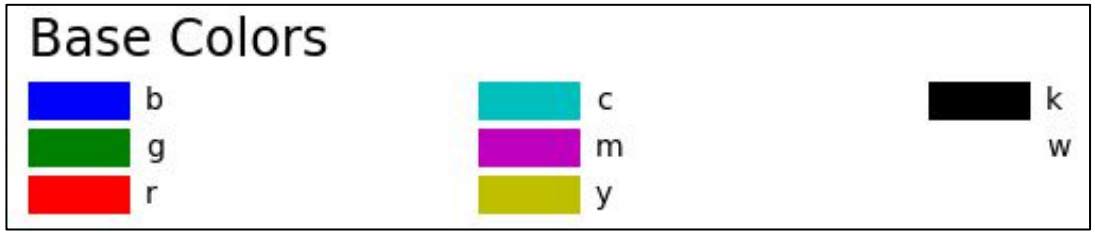
0	1.356	11.22
1	5.512	43.63
2	7.456	74.12
3	3.698	15.74
4	7.559	92.54
5	9.422	37.23
6	2.444	71.45
7	2.674	32.57
8	4.178	46.56
9	7.654	82.29

Cuando line = 5, ¿Qué valor toma la matriz en la posición mat[line,2]?

Python ploteo

```
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(linestyle='--')
```





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

Páginas web

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

Ejercicio de programación

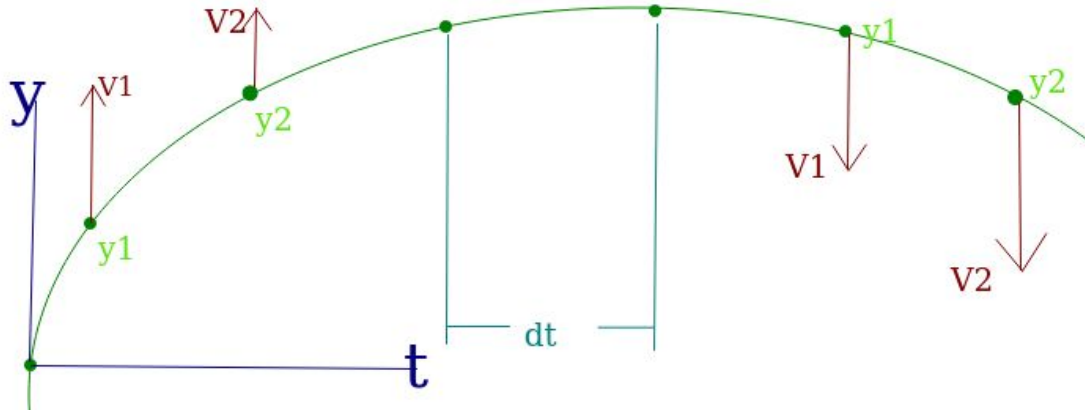
Obtener el movimiento en caída libre de una partícula cuando tiene una velocidad inicial vertical conocida (Voy) y se modifica artificialmente (numéricamente) su movimiento, de tal manera que represente el arrastre de la misma.

Condiciones: Utilizar la función de número aleatorio en Python con el siguiente comando.

x = random.random()

Donde x será una variable flotante entre cero y uno aleatoria.

Ejercicio de programación



$$y_2 = y_1 + V_1\Delta t - 0.5g\Delta t^2$$

$$y_2 = y_2(1 - 0.1x)$$

$$V_2 = \sqrt{V_1^2 - 2g(y_2 - y_1)}$$

Donde x será un número aleatorio entre 0 y 0.15

Ejercicio de programación

Se solicita imprimir en consola el tiempo, la posición y velocidad de la partícula sometida a arrastre por inducción numérica.

Imprimir en consola, para cada iteración el número de iteraciones necesarias para encontrar el x adecuado.

Plotear movimiento con y sin arrastre.

Parámetros:
Velocidad inicial de la partícula = 10 m/s
gravedad = 9.81 m/s^2
Iteraciones totales = 200
Paso de tiempo = 0.01 s

¡Gracias por su atención!

"There must be a beginning of any great matter, but the continuing unto the end until it be thoroughly finished yields the true glory."
Sir Francis Drake, 1587

Introducción a Python

Germán David Sierra Vargas
Jair Hernando Tovar Tuirán

¿Qué es Python?

- Es un lenguaje de programación de propósito general y fácil de usar.
- Admite múltiples enfoques para el diseño de software (principalmente programación estructurada y programación orientada a objetos).
- Tiene una licencia de código abierto.
- Se amplía fácilmente añadiendo nuevos módulos.
- Viene con una gran biblioteca estándar que admite muchas tareas de programación comunes.

Características de Python

- Su sintaxis facilita la lectura de los programas se que escriben.
- Administra de forma automática la memoria, por lo tanto no es necesaria liberala manualmente en el código.
- La combinación de tipos incompatibles (por ejemplo, intentar agregar una cadena y un número) provoca que se genere una excepción.

Python es un lenguaje interpretado, dinámico y multiplataforma

Python es un lenguaje interpretado

Código fuente (Python)

```
print('¡Hola mundo!')
```

Intérprete

¡Hola mundo!

Código fuente (C)

```
#include<stdio.h>
int main{}
(
        printf('¡Hola mundo!');
        return 0;
)
```

Proceso de compilado

Ejecutable (.exe)

¡Hola mundo!

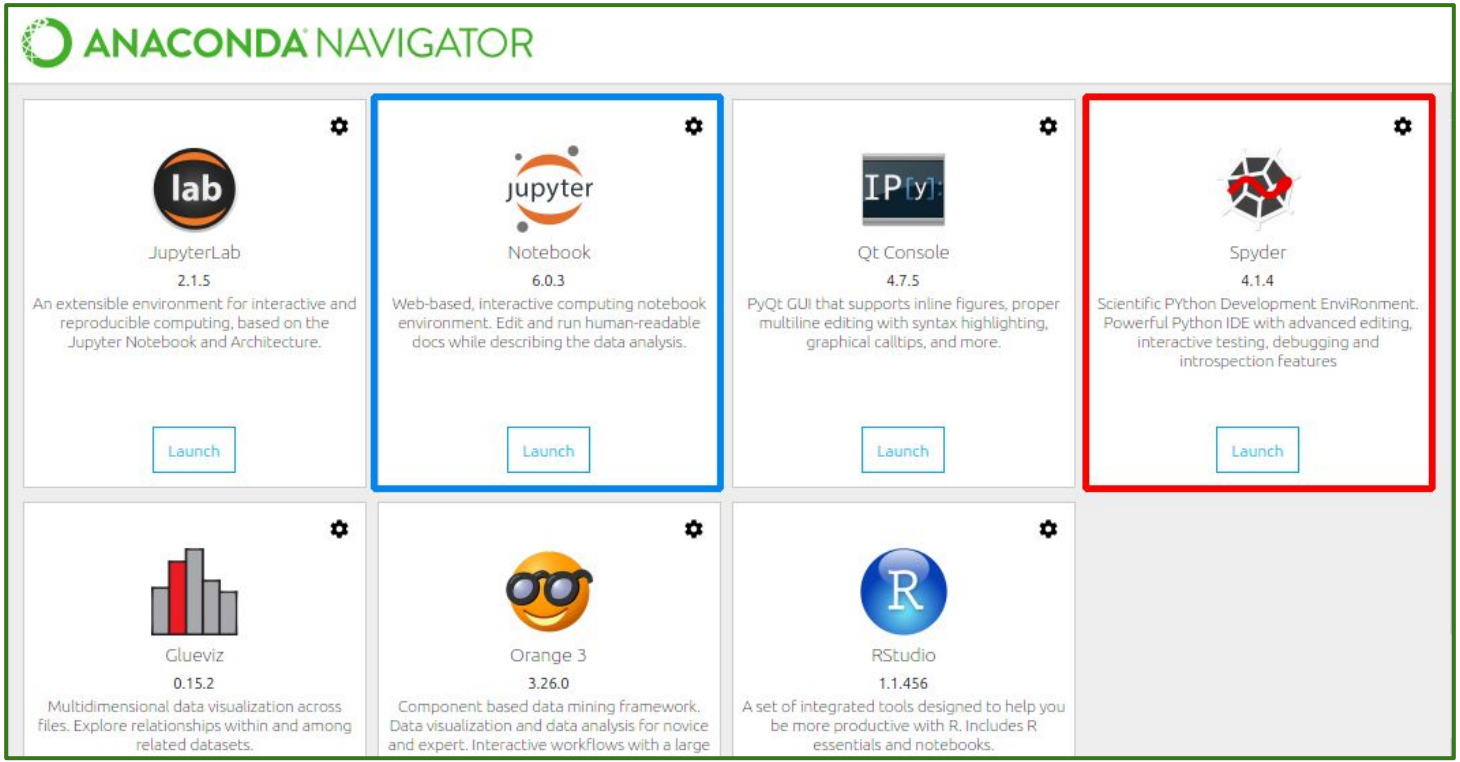
Python es un lenguaje interpretado

Lenguaje interpretado	Lenguaje compilado
● Desarrollo rápido.	● Desarrollo lento.
● Depuración más sencilla.	● Más difícil de depurar.
● Generalmente necesita un menor número de líneas de código.	● Casi siempre requiere más líneas de código.
● Programas más lentos.	● Programas más rápidos
● Menos control sobre el comportamiento del programa.	● Más control sobre el comportamiento del programa.

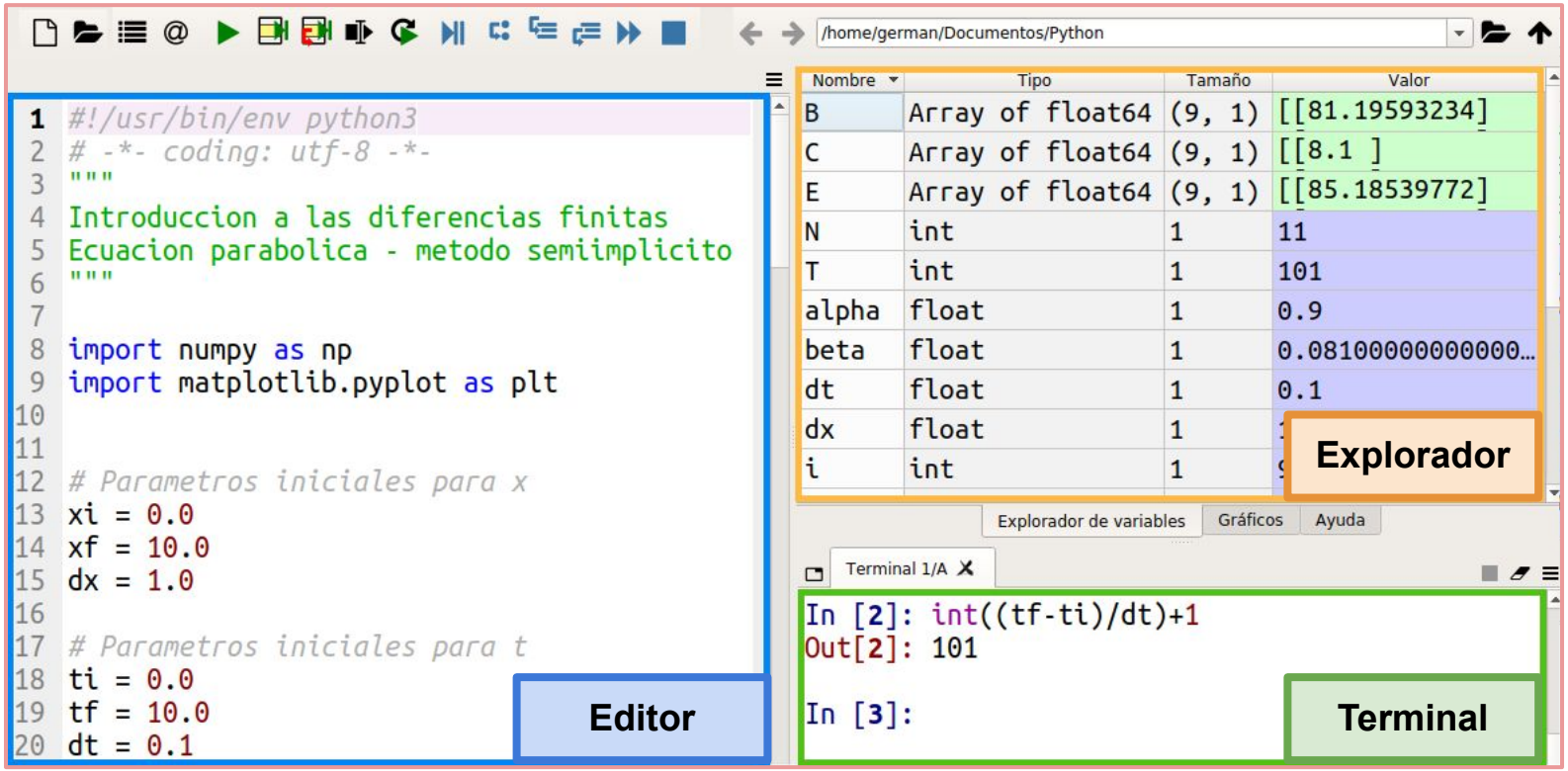
Un poco de terminología

Python	Es un lenguaje de programación interpretado y dinámico.
Anaconda	Es una distribución gratuita y de código abierto de Python que simplifica el desarrollo y la administración de paquetes.
Spyder	Es un entorno de desarrollo integrado (IDE), es decir, una aplicación informática que incluye principalmente IPython y un editor de texto integrado. Su nombre son las siglas de 'The **S**cientific **PY**thon **D**evelopment **E**nvi**R**onment'.
IPython	Es un intérprete de Python, es decir, una consola de comandos que proporciona un modo conveniente e interactivo para ejecutar programas escritos en Python.

Interfaz de Anaconda



Interfaz de Spyder



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
```

Primeros códigos

```
"""
Entrada y salida de datos
@author: Estudiante
"""

# Este programa no requiere ninguna librería

# Entrada de datos
nombre = input('Ingrese su nombre(s): ')
apellido = input('Escriba sus apellidos: ')

# Salida de datos
print('Nombre completo:',nombre,apellido)
print('Apellidos:',apellido)
```

```
"""
Asignarle un valor a una variable
@author: Estudiante
"""

# Este programa no requiere ninguna librería

# Valor de las variables ‘a’ y ‘b’
a = 135
b = 256

# Almacenamiento del resultado en ‘c’
c = a + b
print('El resultado de la suma es',c)
```

Tipos de variables

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 **X**	**Tupla** **(tuple)**	t = ('a', 45, 6.3, True) u = (t[1:3], s) t[1] = 10.34 **X**
Booleano **(bool)**	k = True l = False m = false **X**	**Diccionario** **(dict)**	v = {'name':'Sue', 'cod':2} v['status'] = True v[3] = 31.2 - 57j

Tipos de variables

```
"""
Tipos de variables
@author: Estudiante
"""

# Asignación de las variables
var1 = 'caracteres'
var2 = 18
var3 = pi                X
pi = 3.1415

# Imprimir el resultado en pantalla
print('El valor de var1 es:',var1)
print('El tipo de var1 es:'',type(var1))
```

```
# … continuación del programa

# Operaciones entre variables
var4 = var2 + var3
var5 = var1 + var2       X
var6 = var2
VAR6 = var1
var2 = True

# Imprimir el resultado en pantalla
print('El valor de var2 es:',var2)
print('El valor de var3 es:',var3)
print('El valor de var4 es:',var4)
print('El valor de var6 es:',var6)
print('El valor de VAR6 es:',VAR6)
```

Operadores

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

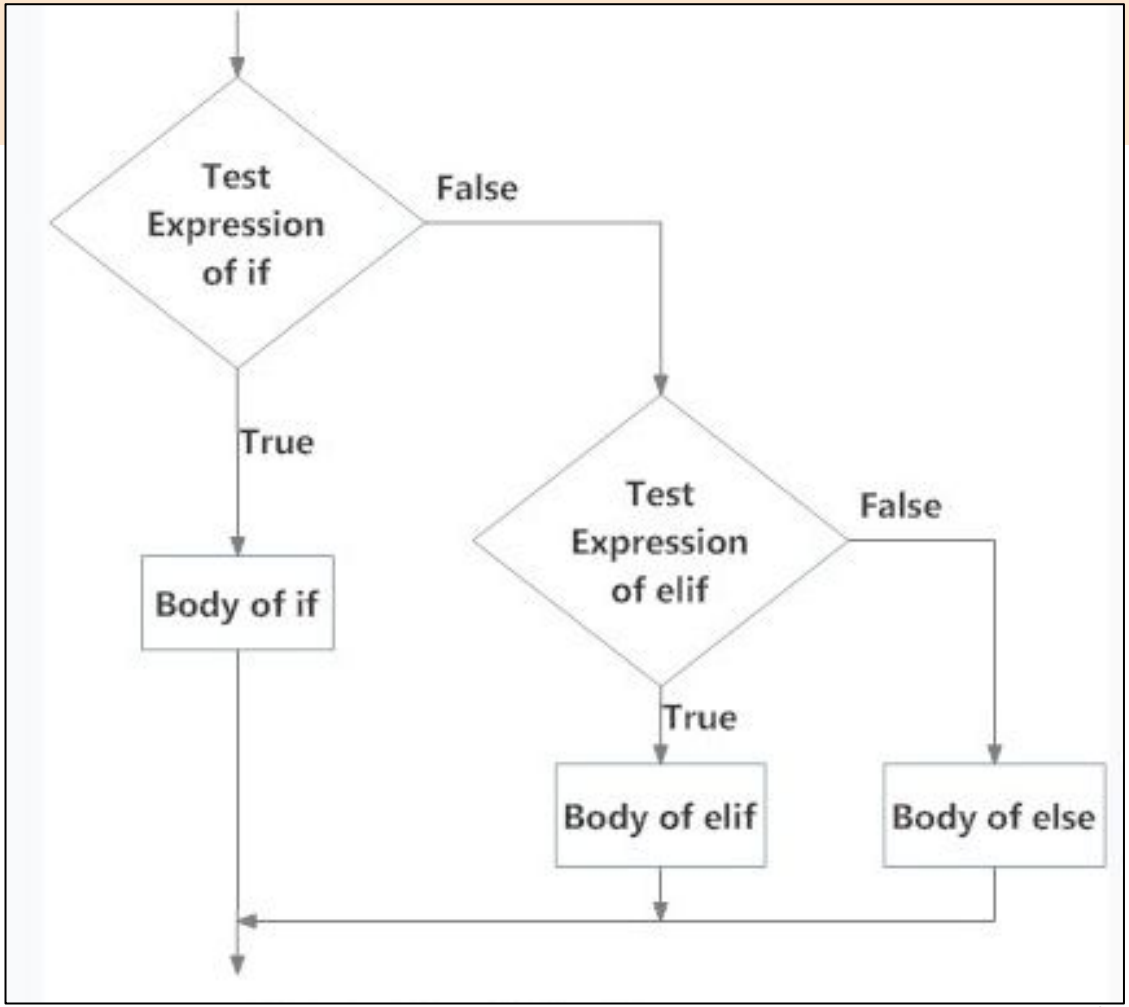
Operadores lógicos

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 (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")
```

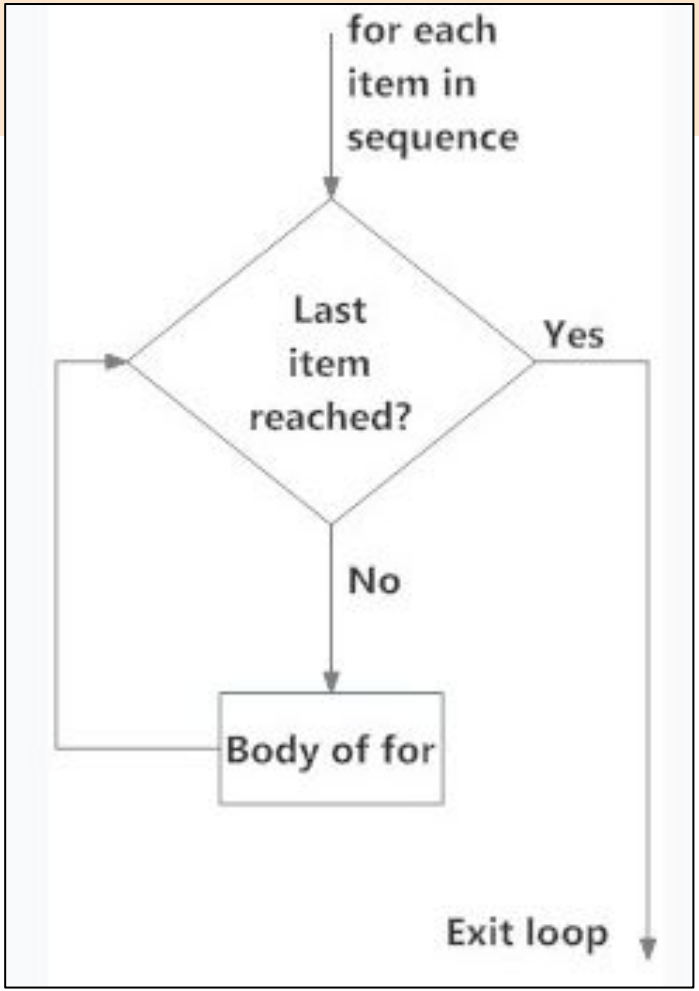


Bucle for

```
for val in sequence:
    Sentencias
```

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

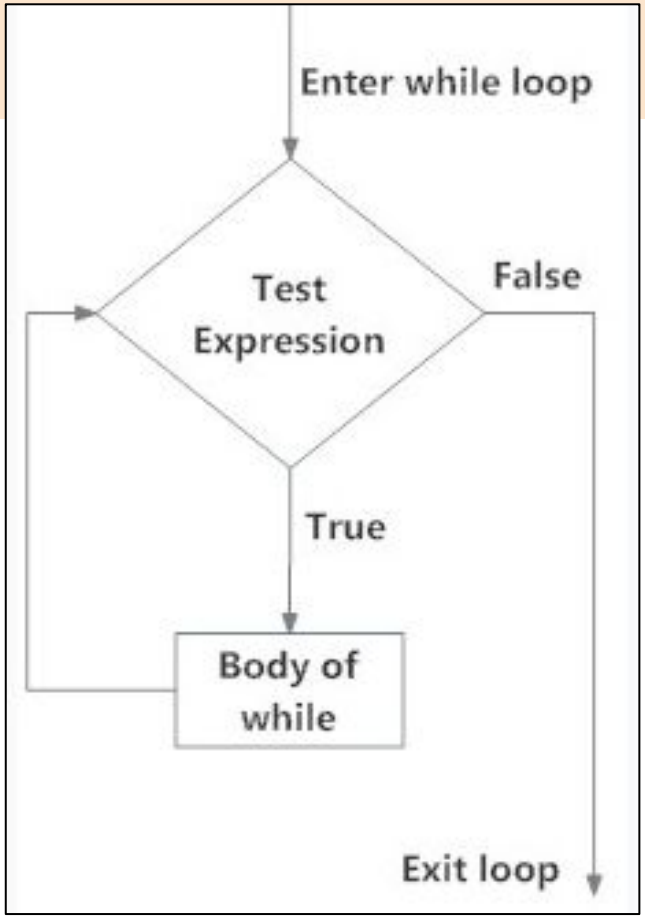
```
i: 0 n: 0
i: 1 n: 2
i: 2 n: 6
i: 3 n: 12
i: 4 n: 20
i: 5 n: 30
i: 6 n: 42
i: 7 n: 56
i: 8 n: 72
i: 9 n: 90
```



Bucle while

while (expresión de prueba):
  Sentencias

```
n = 10

# initialize sum and counter
suma = 0
i = 1

while i <= n:
    suma = suma + i
    i = i+1    # update counter

# print the sum
  print("The sum is", suma)
```



Librerías y módulos en Python

- 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	import numpy
Importar una librería con un ‘alias’	import numpy as np
Importar un módulo desde una librería	from numpy import pi

Numpy es una librería con funciones para trabajar arreglos

Tipos de arreglos

```
"""
Creación de arreglos usando listas
@author: Estudiante
"""

# Este programa no requiere ninguna librería

# Creación de las listas 'A' y 'B'
A = [0,1,2,3,4]
B = [[1,0,0],[0,1,0],[0,0,1]]

# Imprimir el resultado en pantalla
print('A =',A)
print('B =',B)
```

```
"""
Creación de arreglos usando numpy
@author: Estudiante
"""

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)
```

Operaciones básicas entre arreglos

```
"""
Operaciones básicas entre arreglos
@author: Estudiante
"""

import numpy as np

# Creación de los arreglos
A = np.array([[1,2,2],[3,4,1],[2,5,6]])
B = np.array([1,3,5])
C = 3*np.ones(5)
D = np.linspace(0,4,5)

print('Tamaño de C:',np.size(C))
```

```
# … continuación del programa

# 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)
```

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]])
```

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)
```

Funciones

```
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)
```

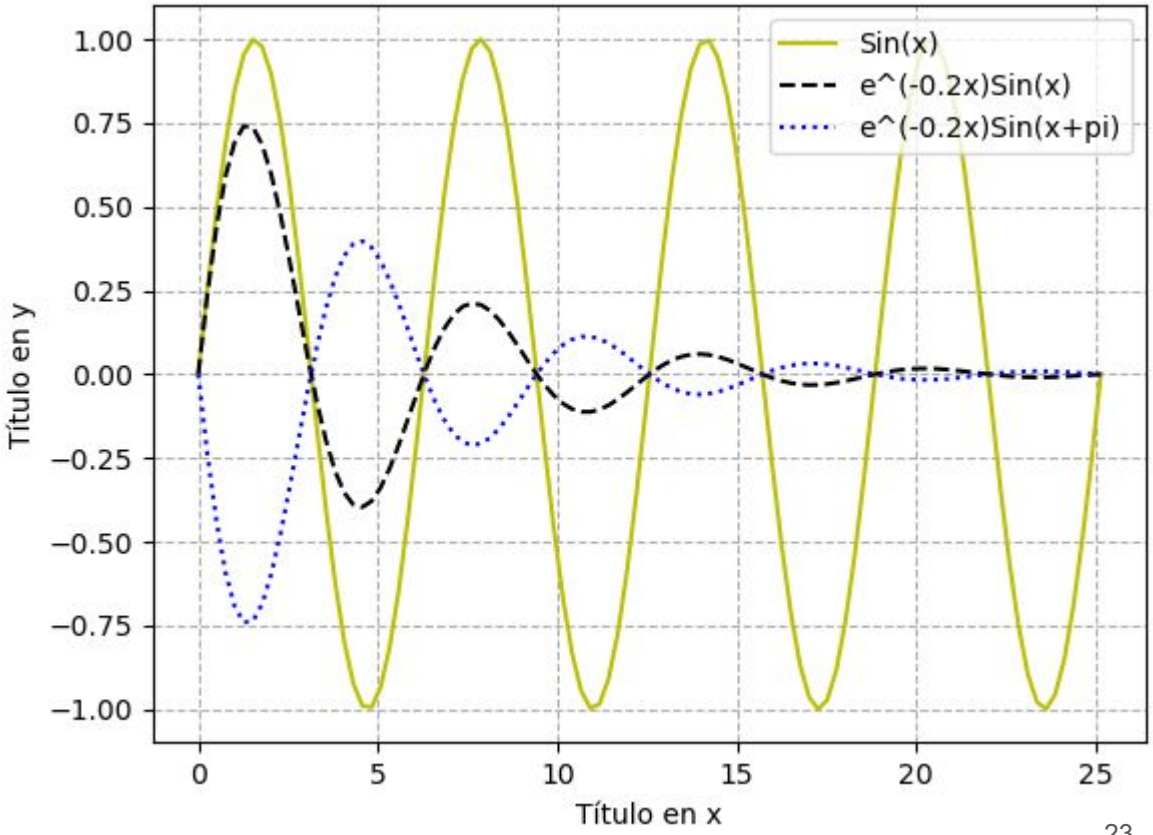
```
def NOMBRE(LISTA_DE_PARAMETROS):
    """DOCSTRING_DE_FUNCION"""
    SENTENCIAS
    RETURN [EXPRESION]
```

A continuación se detallan el significado de pseudo código fuente anterior:

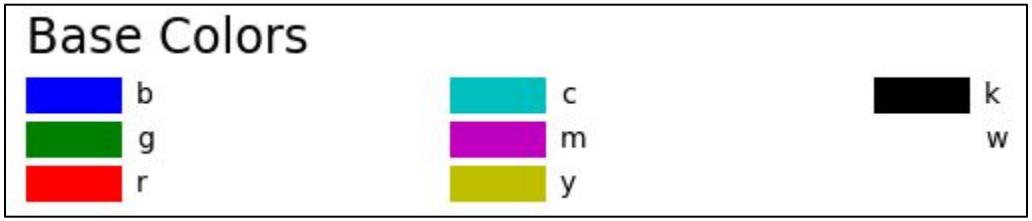
NOMBRE, es el nombre de la función.
LISTA_DE_PARAMETROS, es la lista de parámetros que puede recibir una función.
DOCSTRING_DE_FUNCION, es la cadena de caracteres usada para documentar la función.
SENTENCIAS, es el bloque de sentencias en código fuente Python que realizar cierta operación dada.
RETURN, es la sentencia return en código Python.
EXPRESION, es la expresión o variable que devuelve la sentencia return.

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(linestyle='--')
```



Graficar con matplotlib



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

Gracias por su atención