

# Programación Científica y HPC

Máster Universitario en Ingeniería Matemática y Computación

---> Docente: Jesús Cigales Canga

Material Elaborado: Prf. Enrique De Miguel Ambite

## Tema 2

# ¿Cómo estudiar este tema?

1. Introducción y objetivos
2. Entorno de desarrollo
3. Elementos básicos del lenguaje
4. Manejo de archivos
5. Excepciones
6. Módulos
7. Programación orientada a objetos

## Material Complementario

[Documentación de Python - 3.9.2](#)

[Tutorial de Python](#)

[Manuales y recursos de entrenamiento para la Programación en Python](#)

[Tutorial sobre funciones de entrada y salida en Python](#)

[Tutorial sobre el manejo de archivos](#)

[Tutorial sobre los módulos de Python](#)


[Tutorial POO](#)

### Magistrales




# Modos de programación

- Interactivo
  - Ejecución en tiempo real
- Programación de script
  - Escritura del código
  - Ejecución con el interprete



Consolas →  
interactiva  
Notebooks



Spyder  
PyCharm

# Entorno de programación recomendado

Anaconda Navigator

File Help

ANACONDA.NAVIGATOR

Home

Environments

Learning

Community

Applications on

base (root)

Channels

base (root)

ProgAv\_HPC\_Tema1



CMD.exe Prompt

0.1.1

Run a cmd.exe terminal with your current environment from Navigator activated.

Launch



Datalore

Online Data Analysis Tool with smart coding assistance by JetBrains. Edit and run your Python notebooks in the cloud and share them with your team.

Launch



IBM Watson Studio Cloud

IBM Watson Studio Cloud provides you the tools to analyze and visualize data, to cleanse and shape data, to create and train machine learning models. Prepare data and build models, using open source data science tools or visual modeling.

Launch



JupyterLab

2.2.6

An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

Launch



Jupyter Notebook

6.1.4

Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

Launch



Powershell Prompt

0.0.1

Run a Powershell terminal with your current environment from Navigator activated.

Launch



Qt Console

4.7.7

PyQt GUI that supports inline figures proper multiline editing with syntax highlighting, graphical calltips, and more.

Launch



Spyder

4.1.5

Scientific Python Development Environment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features.

Launch



Glueviz

1.0.0

Multidimensional data visualization across files. Explore relationships within and among related datasets.

Install



Orange 3

3.26.0

Component based data mining Framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.

Install



PyCharm Professional

A Full-Ridged IDE by JetBrains for both Scientific and Web Python development. Supports HTML, JS, and SQL.

Install



RStudio

1.1.456

A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.

Install

# Programación interactiva (I)

```
Last login: Thu Mar 18 17:31:32 on ttys000
(base) MacBook-Pro-de-M-2:~ mluisadiez$ python
Python 3.8.5 (default, Sep  4 2020, 02:22:02)
[Clang 10.0.0 ] :: Anaconda, Inc. on darwin
Type "help", "copyright", "credits" or "license" f
>>> lista=[1,2,3]
>>>
>>> lista
[1, 2, 3]
>>> █
```

¡autocompletado!

```
Jupyter QtConsole 5.0.2
Python 3.8.3 (default, Jul  2 2020, 11:26:31)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.21.0 -- An enhanced Interactive Python. Type '?' for help.
```

```
In [1]: lista=[1,2,3]
```

```
In [2]: lista
```

```
Out[2]: [1, 2, 3]
```

```
In [3]: |
```



# Programación interactiva (II)

ANACONDA NAVIGATOR

Applications on base (root) - Channels

JupyterLab 2.2.6  
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

Jupyter Notebook 6.2.0  
Web-based, interactive computing notebook environment. Edit and run human-readable code while describing the data analysis.

Qt Console 3.0.0  
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

Spyder 4.2.1  
Scientific Python Development Environment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features.

Orange3 3.26.0  
Component-based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.

RStudio 1.1.458  
A set of integrated tools designed to help you be more productive with R, includes R essentials and notebooks.

Jupyter

- sections
- Sites
- VirtualBox VMs
- Zotero
- arbolbinario.ipynb
- Ejercicio\_4\_Algebra\_lineal\_en\_Scipy.ipynb
- Ejercicio1\_NumPy.ipynb
- Ejercicio3\_Operaciones\_con\_arrays.ipynb
- Ejercicio5\_Representacion\_sencilla\_de\_funciones.ipynb
- Ejercicio6\_Representacion\_grafica\_de\_funciones.ipynb
- as\_variables\_aleatorias.ipynb

Ejemplo 1. Arrays con NumPy

Crear un array con NumPy con valores entre 10 y 40. Mostrar el tipo de datos que almacena, su dimensión y el tamaño de las dimensiones. Dar la vuelta al array y convertir sus valores al tipo float.

Crear el array con el rango indicado

```
In [ ]: import numpy as np
```

```
In [ ]: array=np.arange(10,40)
```

```
In [ ]: print('El array es: ')
```

```
print(array)
```

Mostrar las dimensiones del array

```
In [ ]: print('El tipo de los elementos es:')
```

## Google Colab

- No requiere configuración
- Da acceso gratuito a GPUs
- Permite compartir contenido fácilmente

# Programación con script

ANACONDA NAVIGATOR

The screenshot displays the Anaconda Navigator application interface. On the left sidebar, the 'Environments' section is active. The main panel shows four application tiles: JupyterLab (2.2.6), Jupyter Notebook (6.2.0), Qt Console (5.0.2), and Spyder (4.2.3). The Spyder tile is highlighted with a blue border. Below the tiles, a terminal window is open, showing the execution of a Python script named 'ordenacionBurbuja.py'. The script's output is displayed in the terminal, showing the sorted list, execution time, and number of comparisons. A red circle highlights the 'Run' button in the terminal toolbar.

Applications on: base (root) Channels

Environments

Learning

Community

JupyterLab 2.2.6  
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.  
Launch

Jupyter Notebook 6.2.0  
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.  
Launch

Qt Console 5.0.2  
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.  
Launch

Spyder 4.2.3  
Scientific PYTHON Development Environment. Powerful PYTHON IDE with advanced editing, interactive testing, debugging and introspection features.  
Launch

Terminal Window:

```
(base) MacBook-Pro-de-M-2:~ mluisadiez$ python ordenacionBurbuja.py
Lista ordenada:
[0, 8, 17, 35, 36, 73]
Tiempo: 0.000015 segundos
Comparaciones: 15
(base) MacBook-Pro-de-M-2:~ mluisadiez$
```

# ¿Qué es Python?

- Lenguaje de alto nivel

- Multiparadigma
- Interpretado
- Tipificado dinámico

- Sintaxis

- Expresiones
- Estructuras de control  
→ if, while, for
- Definición de funciones
- Llamadas a funciones..

- Léxico

- Identificadores → distingue entre mayúsculas y minúsculas
- Palabras reservadas
- "Delimitadores de bloques" → : , "indentación"





# Tipos de datos (I)

- Numéricos → enteros, coma flotante y complejos
- Cadenas de caracteres
- Lógicos o booleanos

```
In [29]: a=3  
  
In [30]: type(a)  
Out[30]: int  
  
In [31]: b=2.3  
  
In [32]: type(b)  
Out[32]: float  
  
In [33]: c=1-3j  
  
In [34]: type(c)  
Out[34]: complex  
  
In [35]: d="casa"  
  
In [36]: type(d)  
Out[36]: str  
  
In [37]: e=True  
  
In [38]: type(e)  
Out[38]: bool
```

- No se necesita declarar las variables con un tipo.
- El tipo lo adquieren del valor que se les asigna.
- Una variable puede cambiar de tipo si se le asigna un valor de tipo distinto al que tenía.

# Tipos de datos (II)

- **Compuestos**
  - Listas
  - Tuplas
  - Conjuntos
  - Diccionarios

Lista1 = [ 1, 2, 'perro']

Lista = [1, 1, 10, 30]

```
In [41]: lista1
Out[41]: [1, 2, 'perro']
```

```
In [42]: lista[2]
Out[42]: 10
```

```
In [43]: lista1=[1,2,"perro"]
```

```
In [44]: type(lista1)
Out[44]: list
```

```
In [45]: lista1=[1,2,"perro"]
```

```
In [46]: lista1[2]
Out[46]: 'perro'
```

```
In [47]: tupla=(3,4,"gato")
```

```
In [48]: type(tupla)
Out[48]: tuple
```

```
In [49]: tupla[2]
Out[49]: 'gato'
```

```
In [50]: tupla[1]=6
```

```
TypeError                                Traceback (most recent call last)
<ipython-input-50-5ef97c3d3cc9> in <module>
----> 1 tupla[1]=6
```

```
TypeError: 'tuple' object does not support item assignment
```

```
In [51]: conjunto={1,2,5,2,6,8}
```

```
In [52]: type(conjunto)
Out[52]: set
```

```
In [53]: conjunto
Out[53]: {1, 2, 5, 6, 8}
```

```
In [54]: diccionario={'rojo':1,'azul':2,'verde':3,'negro':4}
```

```
In [55]: type(diccionario)
Out[55]: dict
```

```
In [56]: diccionario
Out[56]: {'rojo': 1, 'azul': 2, 'verde': 3, 'negro': 4}
```

# Instrucciones estilo Python

Tradicional

- **Operador ternario**

```
In [59]: if n<2:
...:     print("menor")
...: else:
...:     print("mayor o igual")
...:
mayor o igual
```

```
In [60]: print("menor") if n<2 else print("mayor o igual")
mayor o igual
```

Tradicional

```
In [61]: a=2
In [62]: b=3
In [63]: aux=a
In [64]: a=b
In [65]: b=aux
In [66]: a
Out[66]: 3
In [67]: b
Out[67]: 2
```

- **Intercambio de valores**

```
In [68]: a,b=b,a
In [69]: a
Out[69]: 2
In [70]: b
Out[70]: 3
```

# Instrucciones estilo Python. Crear listas por recorrido

A partir de tupla

```
In [47]: tupla=(3,4,"gato")  
  
In [88]: lista=[elem for elem in tupla]  
  
In [89]: lista  
Out[89]: [3, 4, 'gato']
```

A partir de  
conjunto


```
In [93]: listaConjunto=[elem for elem in conjunto]  
  
In [94]: listaConjunto  
Out[94]: [1, 2, 5, 6, 8]
```



# Instrucciones estilo Python. Crear listas por recorrido

```
In [56]: diccionario  
Out[56]: {'rojo': 1, 'azul': 2, 'verde': 3, 'negro': 4}
```

A partir de las claves o de los valores de un diccionario




```
In [96]: listaClaves=[claves for claves in diccionario.keys()]  
  
In [97]: listaClaves  
Out[97]: ['rojo', 'azul', 'verde', 'negro']  
  
In [98]: listaValores=[valores for valores in diccionario.values()]  
  
In [99]: listaValores  
Out[99]: [1, 2, 3, 4]
```

```
In [101]: listaPares=[(c,v) for c,v in diccionario.items()]  
  
In [102]: listaPares  
Out[102]: [('rojo', 1), ('azul', 2), ('verde', 3), ('negro', 4)]  
  
In [103]: listaPares[0]  
Out[103]: ('rojo', 1)  
  
In [104]: listaPares[0][0]  
Out[104]: 'rojo'
```

# Instrucciones estilo Python. Crear diccionario

```
In [102]: listaPares
Out[102]: [('rojo', 1), ('azul', 2), ('verde', 3), ('negro', 4)]
```

Crear diccionario recorriendo una lista



```
In [106]: diccionarioNuevo={elem[0]:elem[1] for elem in listaPares }
```

```
In [107]: diccionarioNuevo
Out[107]: {'rojo': 1, 'azul': 2, 'verde': 3, 'negro': 4}
```


```
In [108]: listaClaves
Out[108]: ['rojo', 'azul', 'verde', 'negro']

In [109]: listaValores
Out[109]: [1, 2, 3, 4]

In [110]: diccionarioDos={par[0]:par[1] for par in zip(listaClaves ,listaValores)}

In [111]: diccionarioDos
Out[111]: {'rojo': 1, 'azul': 2, 'verde': 3, 'negro': 4}
```

Recorrer dos  
listas a la vez



# Instrucciones estilo Python. Crear diccionario

Indexado por posiciones



```
In [150]: diccionario2={indice:valor for indice,valor in enumerate(listaValores)}
```

```
In [151]: diccionario2
```

```
Out[151]: {0: 1, 1: 2, 2: 3, 3: 4}
```

```
In [152]: listaColores={"rojo","verde","amarillo"}
```

```
In [153]: diccionario2={indice:valor for indice,valor in enumerate(listaColores)}
```

```
In [154]: diccionario2
```

```
Out[154]: {0: 'verde', 1: 'rojo', 2: 'amarillo'}
```



# Copias de objetos ¡cuidado!

```
In [120]: lista1=[2,4,6]
```

```
In [121]: lista2=lista1
```

```
In [122]: lista1  
Out[122]: [2, 4, 6]
```

```
In [123]: lista2  
Out[123]: [2, 4, 6]
```

```
In [124]: lista1[1]=20
```

```
In [125]: lista1  
Out[125]: [2, 20, 6]
```

```
In [126]: lista2  
Out[126]: [2, 20, 6]
```

```
In [146]: lista1 is lista2  
Out[146]: True
```

```
In [147]: lista1==lista2  
Out[147]: True
```

Copia referencias  
Comparten cambios

Se crea lista nueva  
Mismos valores,  
objetos distintos

```
[137]: lista1  
t[137]: [2, 30, 6]
```

```
[138]: listaCopia=lista1.copy()
```

```
In [139]: listaCopia  
Out[139]: [2, 30, 6]
```

```
In [140]: lista1 is listaCopia  
Out[140]: False
```

```
In [141]: lista1==listaCopia  
Out[141]: True
```

```
In [142]: lista1[0]=100
```

```
In [143]: lista1  
Out[143]: [100, 30, 6]
```

```
In [144]: listaCopia  
Out[144]: [2, 30, 6]
```

```
In [127]: lista1=[2,4,6]
```

```
In [128]: lista2=[elem for elem in lista1]
```

```
In [129]: lista2  
Out[129]: [2, 4, 6]
```

```
In [130]: lista1 is lista2  
Out[130]: False
```

```
In [133]: lista1==lista2  
Out[133]: True
```

```
In [134]: lista1[1]=30
```

```
In [135]: lista1  
Out[135]: [2, 30, 6]
```

Se crea lista nueva  
Mismos valores,  
objetos distintos



# Funciones lambda y filter()

```
In [160]: numeros=[i for i in range(10)]
```

```
In [161]: numeros
```

```
Out[161]: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

función lambda  
nombrada

```
In [157]: impar=lambda x:x%2!=0
```

```
In [158]: impar(5)
```


```
Out[158]: True
```

```
In [163]: impares=list(filter(impar,numeros))
```

```
In [164]: impares
```

```
Out[164]: [1, 3, 5, 7, 9]
```

Aplica función booleana a  
una lista



```
In [169]: impares=list(filter(lambda x:x%2!=0,numeros))
```

```
In [170]: impares
```

```
Out[170]: [1, 3, 5, 7, 9]
```

```
In [155]: def esImpar(x):  
...:     return x%2!=0  
...:
```

```
In [156]: esImpar(3)
```

```
Out[156]: True
```

función con nombre

# Funciones lambda y map()

función lambda  
nombrada

```
In [166]: cuadrado=lambda x:x**2
```

```
In [167]: cuadrados=list(map(cuadrado,numeros))
```

```
In [168]: cuadrados
```

```
Out[168]: [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

Aplica función a una lista

función lambda

```
In [171]: cuadrados=list(map(lambda x:x**2,numeros))
```

```
In [172]: cuadrados
```

```
Out[172]: [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

# Convenciones y Definiciones Típicas

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3 """
4 """
5 import copy
6 class Cliente:
7     numClientes=0
8
9     def __init__(self,nombre, direccion, email, clave):
10         Cliente.numClientes=Cliente.numClientes+1
11         self.id="CL"+str(Cliente.numClientes)
12         self.nombre=nombre
13         self.direccion=direccion
14         self.email=email
15         self._clave=self.encryptedClave(email,clave)
16
17     def encryptedClave(self,email,clave):
18         import hashlib
19         aux=email+clave
20         claveEncriptada=hashlib.sha3_512(aux.encode('utf-8')).hexdigest()
21         return claveEncriptada
22
23     def getClave(self):
24         return self._clave
25
26     def __eq__(self,objeto):
27         return (self.nombre==objeto.nombre and self.direccion==objeto.direccion and self.id==
28
29     def __str__(self):
30         datosCliente="El cliente es: "
31         datosCliente+="\nId de cliente: "+self.id
32         datosCliente+="\nNombre: "+self.nombre
33         datosCliente+="\nDireccion: "+self.direccion
34         datosCliente+="\nEmail: "+self.email
35         return datosCliente
36
37     def __copy__(self):
```

## Usage

Here you can get help of any object by pressing **Ctrl+I** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in **Preferences > Help**.

New to Spyder? Read our [tutorial](#)

Variable explorer Help Plots Files

Console 1/A

```
Email: maria@de.es
2891660d8d33622bfda18c2b3caddb6865decd1c0d6c16d760f3c24c09d5e6ffdb744b586582124d2ab0c68f7afb3999baffe
758e2cbfdc316121f83cbea6426
True
El cliente es:
Id de cliente: CL1
Nombre: Maria López García
Direccion: Carretas, 7. 28015 Madrid
Email: maria@de.es
Maria López García
__copy__()
El cliente es:
Id de cliente: CL3
Nombre: Maria López García
Direccion: Carretas, 7. 28015 Madrid
Email: maria@de.es
El cliente es:
Id de cliente: CL1
Nombre: Pepe
Direccion: Carretas, 7. 28015 Madrid
Email: maria@de.es
```



[www.unir.net](http://www.unir.net)