Introducción a Python

Laboratorio 3, DF, FCEyN, UBA
1C 2022 - Matías Zanini

Lo que vamos a aprender hoy:

- Qué es Python
- Instalación: Anaconda, Spyder, Librerías
- Interfaz de Spyder
- Conceptos y comandos básicos
- Definición de variables y funciones
- Librerías importantes

¿Qué es Python?

Es un lenguaje de programación **interpretado**.

Pros:

- Fácil de leer
- Dinámico
- Multiplataforma
- **Todo el mundo lo usa**

Contras:

- Más lento que los compilados
- Muy “permisivo”

Se ejecuta línea por línea

```
7
8   ready = True
9
10  if ready:
11
12      print('Hola Mundo!')
```

Instalación y Preparación

Link: https://www.anaconda.com/products/individual

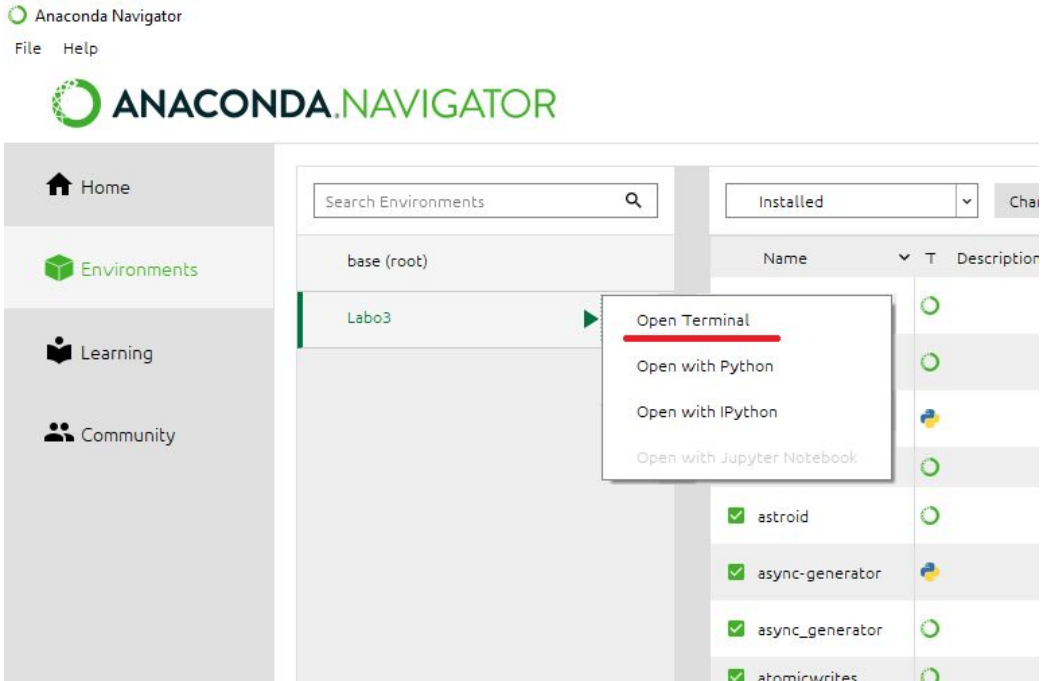
Se encarga de todo. Instala Python, Spyder, librerías y provee una interfaz gráfica para ordenarlo.

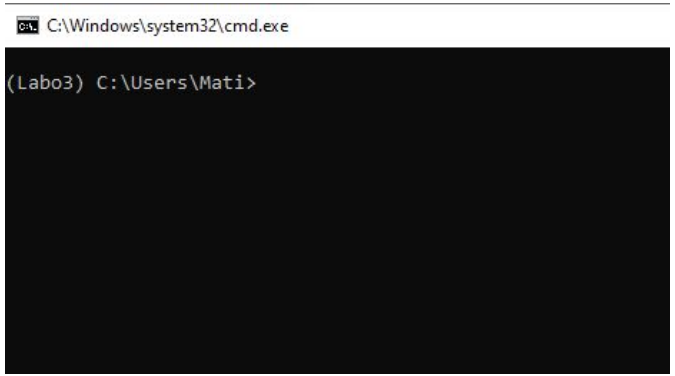
Necesitamos un intérprete:

El intérprete nos permite escribir el código y ejecutarlo

Usando Anaconda: Entornos y Consola

Creamos un nuevo *environment* (Python 3.8) → Lo seleccionamos y abrimos una terminal o **consola**.





Comandos que usaremos:

- conda install → instala paquetes
- conda update → los actualiza
- spyder → ejecuta spyder
- pip install (**no recomendado**)

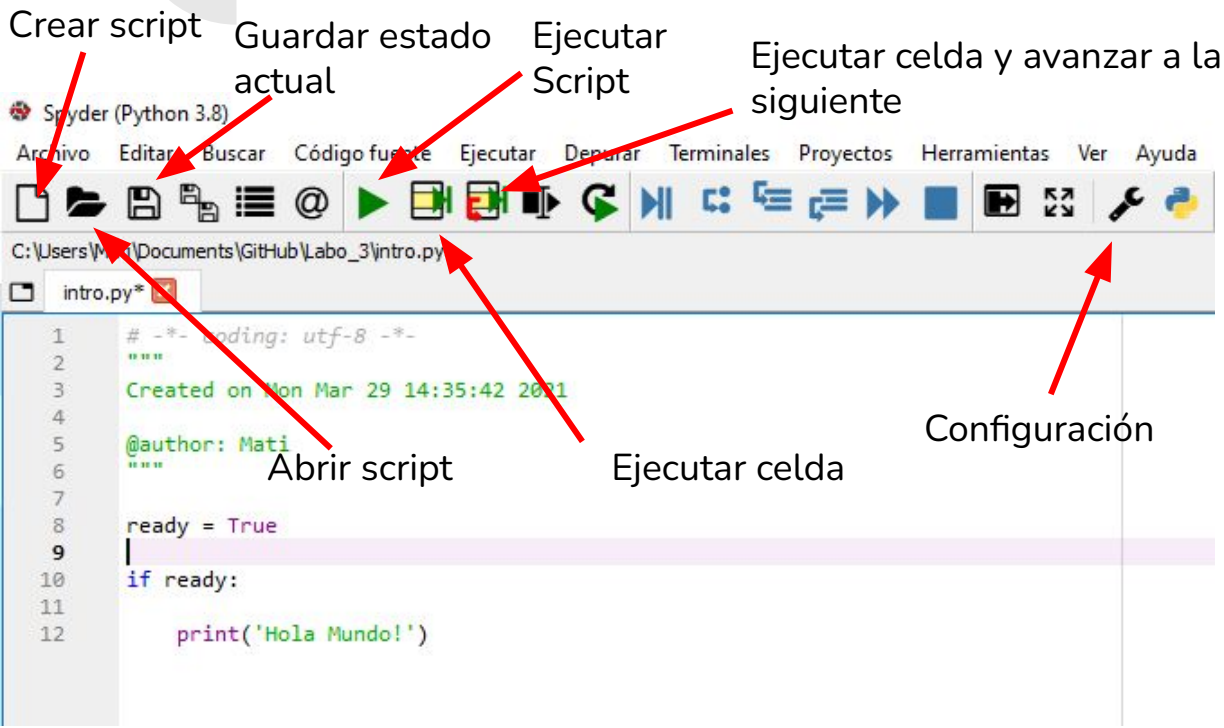
Usando Anaconda: Librerías

Las librerías son paquetes descargables e instalables que contienen funciones útiles que vamos a necesitar.

Las instalamos con: **conda install “nombre de la librería”**

- **Numpy:** permite crear objetos que funcionan como vectores y matrices.
- **Matplotlib:** permite crear gráficos a partir de los datos.
- **Scipy:** funciones matemáticas, constantes, ajuste de funciones.
- **Pandas:** permite organizar datos en forma de tablas.

Interfaz de Spyder

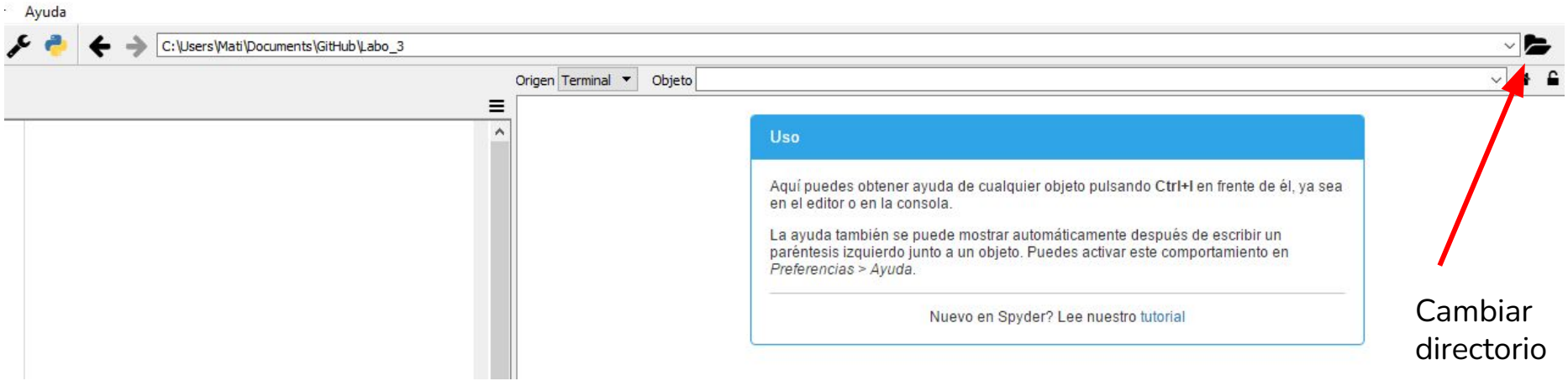


Podemos crear **scripts** nuevos, guardarlos, ejecutarlos en la consola, configurar spyder, etc.

Un **script** es un archivo donde se escribe código para luego ser ejecutado. Suele tener la extensión “.py”.

(Veámoslo en vivo)

Interfaz de Spyder: Directorio de Trabajo

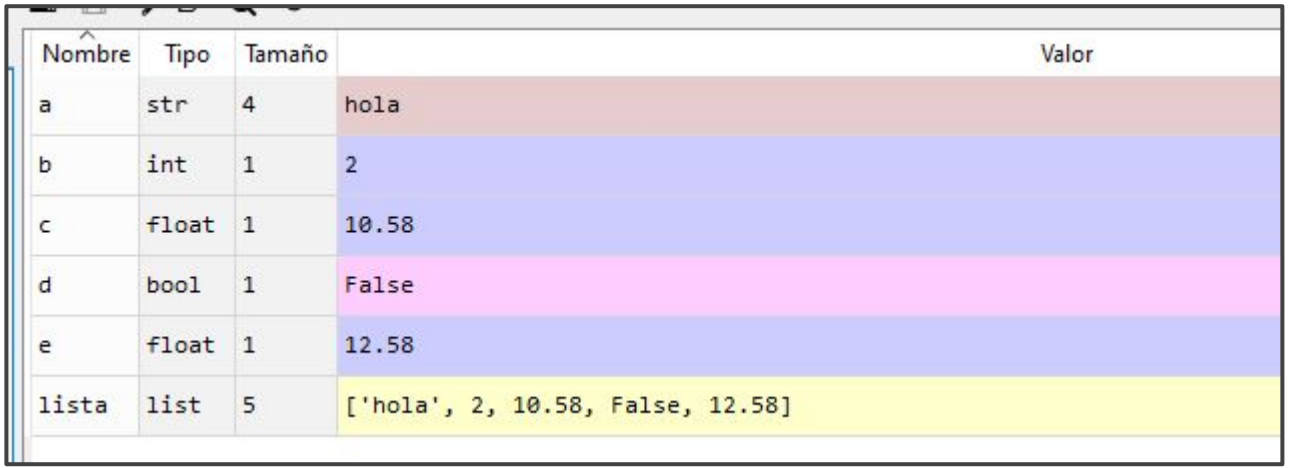


Es la carpeta donde Python, por defecto, buscará scripts para abrir o importar, guardará archivos, etc.

Comandos Básicos y Tipos de Variables

Vamos a aprender cómo definir variables y a hacer operaciones básicas con ellas.

Abrir el script “**intro.py**”.



Nombre	Tipo	Tamaño	Valor
a	str	4	hola
b	int	1	2
c	float	1	10.58
d	bool	1	False
e	float	1	12.58
lista	list	5	['hola', 2, 10.58, False, 12.58]

Comandos If, For, While

Python permite realizar rutinas de manera sencilla.

If: Permite decidir si un bloque de código se ejecuta o no, bajo una condición. Puede ir junto con **else** (sino) o **elif** (de lo contrario, si...).

For: Repite un bloque de código la cantidad de veces deseada.

While: Repite un bloque de código siempre, hasta que una dada condición se deje de cumplir.

Para definir los bloques de código hay que tener en cuenta la **identación**.

Identación

La identación es la forma que tiene Python de definir bloques de código con jerarquía.

Es como una “mamuschka”:

```
69
70  if 2+2==4:
71
72      print('Este es el primer bloque de identación. Lo armamos presionando "Tab" ')
73
74      for elemento in ['Agua','Aire','Fuego','Tierra']:
75
76          # Este es el segundo bloque de identación.
77
78          print(elemento) # Se mostrará en la consola cada elemento de la lista en cada iteración.
79
80
81
82
83
84
85
```

Bloque 1

Bloque 2

Creando Funciones

```
def nombre_funcion(x,y):

    z = x+y

    return z
```

declaramos el nombre y las variables.

bloque de código. Puede ser cualquier cosa.

declaramos lo que la función debe devolver como resultado.

¡No olvidar la identación!

Creando Funciones: Funciones Anónimas

Una forma rápida de definir una función sin necesidad de ponerle nombre.

```
a = lambda x: x+2

>> a(2)

>> 4
```

No las vamos a usar. Su aplicación es más avanzada y excede al objetivo del curso.

Funciones Internas Útiles

len(): devuelve el tamaño del arreglo.

type(): devuelve qué tipo de variable es el objeto entre paréntesis (float, string, list, etc.).

range(): objeto que itera en un dado rango, con pasos de 1 en 1. Por defecto comienza en 0.

list(): convierte los objetos en listas (siempre que sean iterables).

int(): convierte los objetos en números enteros (si es posible).

float(): convierte los objetos en números de punto flotante (si es posible).

sum(): devuelve la suma de todos los elementos de un iterable (si está hecho de números).

Librerías: Cómo Importarlas

este comando define la abreviatura que usaremos para llamar a la librería

```
233  import numpy as np
234
235  import scipy as sp
236
237  import pandas as pd
238
239  from matplotlib import pyplot as plt
240
```

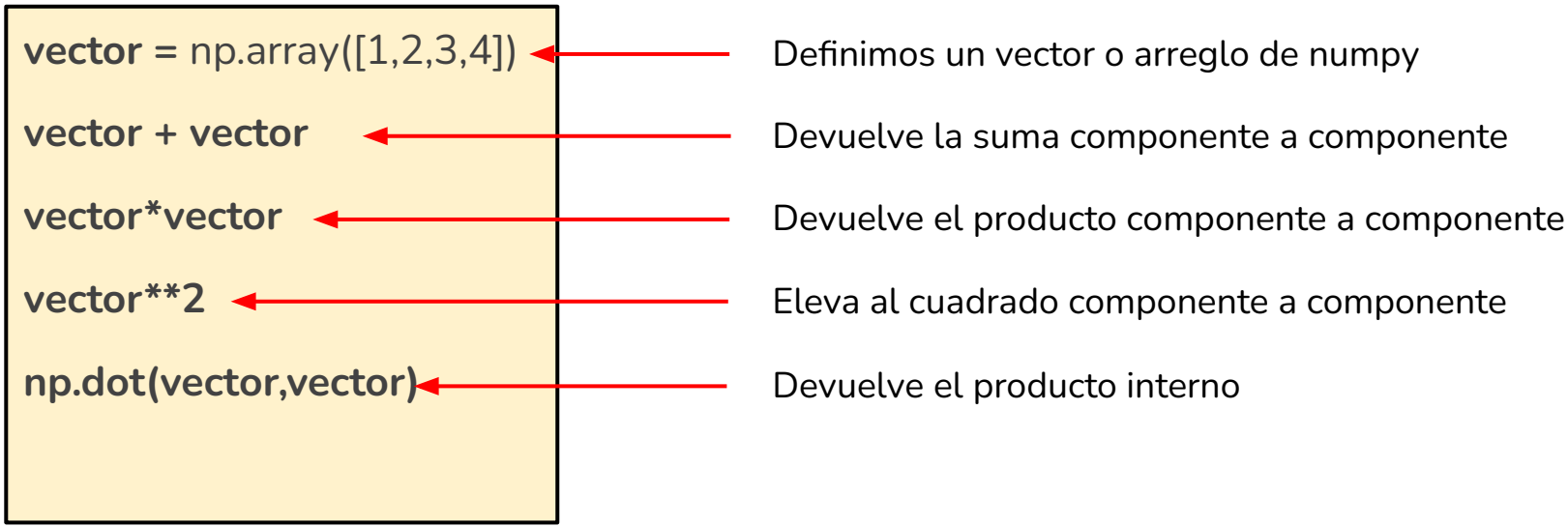
si solo nos interesa una subclase de la librería podemos llamarla así

Para utilizar los objetos dentro de las librerías, deberemos anteponer la abreviatura, seguido de un punto “.”

Ejemplos:

np.array()
np.dot()
plt.plot()

Librerías Importantes: Numpy



Librerías Importantes: Numpy

Para definir matrices, hacemos una “lista de listas”. Cada lista es una fila.

matriz = np.array(**[[**1,2**], [**2,1**]]**) → $\begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix}$

Las funciones anteriores también funcionan.

np.dot(matriz, matriz) devuelve el producto matricial (otra matriz).

Guardar/Importar Archivos con Datos

Utilizaremos los siguientes comandos

Array o matriz de numpy

Para guardar: **np.savetxt**(nombre_archivo, vector)

Para cargar: vector = **np.loadtxt**(nombre_archivo)

Si no especifican lo contrario, el archivo se guardará en su **directorio de trabajo.**
Recordar agregar la extensión ".txt" al final del string con el nombre.

Otra Librería Útil: Pandas

Permite generar tablas con los datos que recopilemos.

tabla = pd.DataFrame(index = **iterable** , columns = **lista con nombres de columnas**)

Ejemplo:

index = range(8)

columns = [‘Tiempo’, ‘Voltaje’]

	Tiempo	Voltaje
0	NaN	NaN
1	NaN	NaN
2	NaN	NaN
3	NaN	NaN
4	NaN	NaN
5	NaN	NaN
6	NaN	NaN
7	NaN	NaN

Tabla vacía

NaN = Not a Number

Otra Librería Útil: Pandas

Si tenemos los siguientes datos:

```
tiempo = [1,2,3,4,5,6,7,8]

señal = [4,5.4,5.3, 3, 4.3, 2.2, 2.4, 8]
```

Los agregamos a la tabla como:

tabla[‘Tiempo’] = tiempo

tabla[‘Voltaje’] = señal

	Tiempo	Voltaje
0	1	4.0
1	2	5.4
2	3	5.3
3	4	3.0
4	5	4.3
5	6	2.2
6	7	2.4
7	8	8.0

Guardar/Importar Archivos con Datos

Utilizaremos los siguientes comandos

Evita que se guarden los números de los índices

Para guardar: **tabla.to_csv**(nombre_archivo, index=False)

Para cargar: tabla_cargada = **pd.read_csv**(nombre_archivo)

Si no especifican lo contrario, el archivo se guardará en su **directorio de trabajo.**
Recordar agregar la extensión ".csv" al final del string con el nombre.

En futuras clases veremos...

- Manejar mejor el guardado/cargado de archivos.
- Gráficos y Análisis
- Ajuste de funciones