



Instalar y empezar

1. Instalación del entorno de aprendizaje Anaconda
2. Como funciona el intérprete de Python : Jupyter
3. Trabajando con números
4. Trabajando con variables
5. Práctica
6. Información adicional

1. Instalación del entorno Anaconda



Anaconda: distribución libre de Python utilizada en ciencia de datos y machine learning, muy útil para aprender.



Jupyter Notebook: herramienta que permite crear y compartir documentos con código fuente y explicaciones.

Data science technology for human sensemaking.

A movement that brings together millions of data science practitioners,
data-driven enterprises, and the open source community.

Get Started



[Products](#) ▼[Pricing](#)[Solutions](#) ▼

Individual Edition
Open Source Distribution



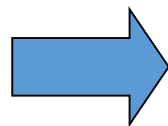
Team Edition
Package Manager



Enterprise Edition
Full Data Science Platform



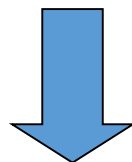
Professional Services
Data Experts Work Together



Individual Edition

Your data science toolkit

With over 20 million users worldwide, the open-source Individual Edition (Distribution) is the easiest way to perform Python/R data science and machine learning on a single machine. Developed for solo practitioners, it is the toolkit that equips you to work with thousands of open-source packages and libraries.

[Download](#)

Anaconda Installers

Windows

MacOS

Linux

Python 3.8

64-Bit Graphical Installer (466 MB)

32-Bit Graphical Installer (397 MB)

Python 3.8

64-Bit Graphical Installer (462 MB)

64-Bit Command Line Installer (454 MB)

Python 3.8

64-Bit (x86) Installer (550 MB)

64-Bit (Power8 and Power9) Installer (290 MB)



Anaconda3-2020.07....exe
1,9/467 MB, 33 min restants

Getting Started

Welcome to Anaconda!

Here are some important resources to help you get started.

Create your free [Anaconda Cloud](#) today to get access to a package management service that makes it easy to find, access, store, and share public notebooks, environments, and conda and PyPi packages.

Create an Account

Individual Edition Tutorial

This quick 15-minute tutorial provides an introduction to help you get started using this powerful tool.

[Watch Tutorial](#) ↻

Quick Start Guide

Learn how to use Anaconda Individual Edition, Anaconda Navigator and conda with cheat sheets, FAQs, and more.

[Learn More](#) ↻

Anaconda Individual Edition Tutorial

More from Anaconda



Team Edition

Thousands of curated data science packages in an enterprise-grade repository. Ideal as you scale the use of Python and R across the data science discipline.

[Learn More](#) 

Professional Services

Give your data science team a boost and help you get the most from Anaconda Enterprise (AE). Solve heavy-weight data science and machine learning challenges with Anaconda's experts.

[Learn More](#) 

Enterprise Edition

Our enterprise platform is a comprehensive foundation for any organization that wants to use data science and machine learning to make better decisions and build differentiating products.

[Learn More](#)



By data scientists, for
data scientists

PRODUCTS

Individual Edition

Team Edition

Enterprise Edition

Professional Services

Pricing

SOLUTIONS

Use Cases

Industries

RESOURCES

Blog

Open Source

Library

Help & Training

Events



Resources ▼

Blog

C

Open Source

Technologies for data science

Library

Videos, Datasheets and
Whitepapers

Help & Training

Training, Documentation and
Support

Events

Meetups, Webinars and
Conferences

<https://www.anaconda.com/open-source#the-fundamentals>

The Fundamentals



Jupyter és un projecte de codi obert creat per donar suport a la ciència de dades interactiva i la informàtica científica a través de llenguatges de programació. Jupyter ofereix un entorn basat en web per treballar amb quaderns que contenen codi, dades i text. Els quaderns Jupyter són l'espai de treball estàndard per a la majoria de científics de dades de Python.



Una biblioteca per a estructures de dades tabulars, anàlisi de dades i eines de modelatge de dades, inclosos els gràfics integrats mitjançant Matplotlib. pandas pretén ser el bloc fonamental d'alt nivell per fer anàlisis de dades pràctiques i reals a Python



Matplotlib és l'eina de visualització de dades Python més consolidada, que se centra principalment en gràfics bidimensionals (gràfics de línies, gràfics de barres, gràfics de dispersió, histogrames i molts altres). Funciona amb moltes interfícies GUI i formats de fitxer, però té un suport interactiu relativament limitat als navegadors web.



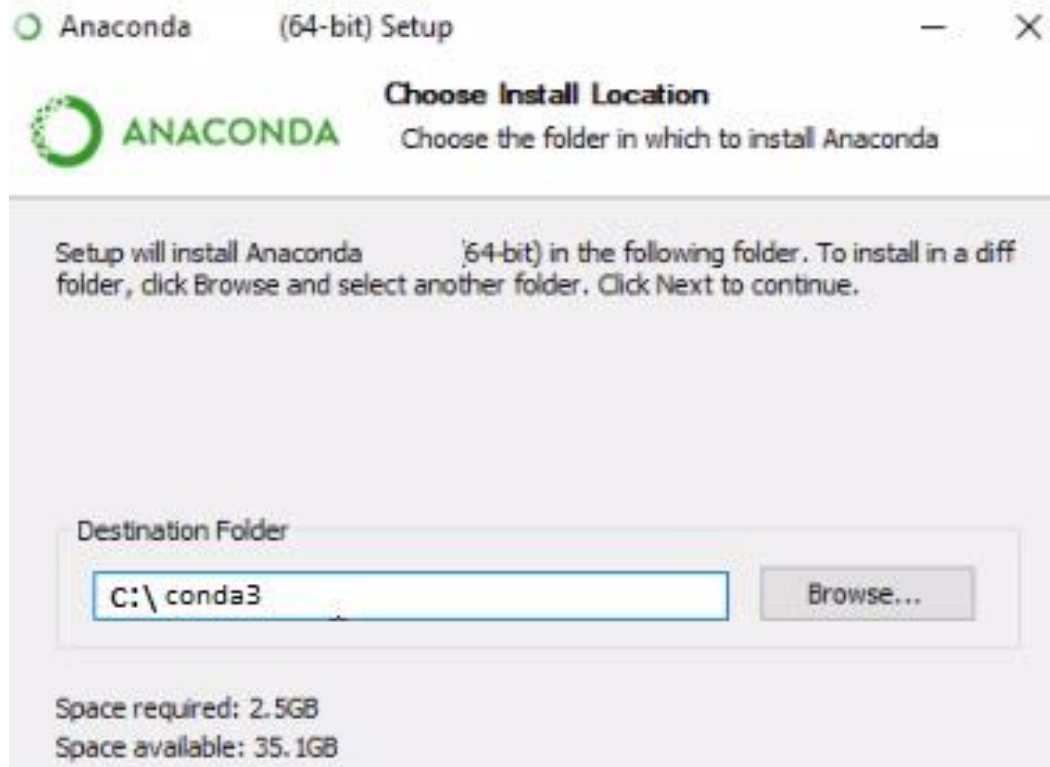
Un paquet bàsic per a la informàtica científica amb Python. Numpy permet la formació de matrius i operacions bàsiques amb matrius. Numpy s'utilitza per indexar i ordenar, però també es pot utilitzar per a l'àlgebra lineal i altres operacions. Moltes altres biblioteques de ciències de dades per a Python es construeixen a NumPy internament, inclosos Pandas i SciPy.



La biblioteca de gràfics Python de Plotly fa gràfics interactius de qualitat de publicació. És una popular i potent biblioteca de visualització basada en navegadors que us permet crear parcel·les interactives basades en JavaScript des de Python.



Ejecutar como
administrador !!!



Cambiar la carpeta
por defecto, poner
c:\conda3



Papelera de reciclaje



Filtros ▾



Mejor coincidencia



Anaconda Navigator

Aplicación de escritorio



Aplicaciones



Anaconda Prompt



Anaconda3-5.1.0-Windows-x86_64.exe

Sugerencias de búsqueda



anaconda - Ver resultados web



anaconda|Navigator



13:22

25/04/2018



Applications on

base (root)

Channels

Refresh



jupyterlab

0.31.4

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

Launch



notebook

5.4.0

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

Launch



qtconsole

4.3.1

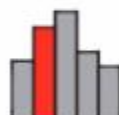
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

Launch



spyder

3.2.6



glueviz

0.12.0



orange3

3.4.1

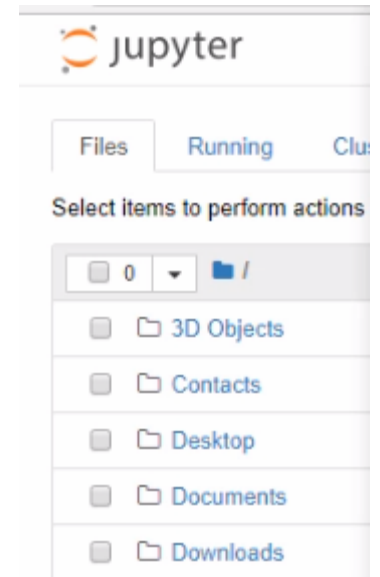
2. Cómo funciona el intérprete Jupyter

Es un entorno de trabajo interactivo que permite desarrollar código en Python de manera dinámica. Integra en un mismo documento tanto bloques de código como texto, gráficas o imágenes

4. Creando un cuaderno

Como arrancamos **Jupyter** desde la **terminal**, se abrirá el **workspace** en la carpeta en la que estemos (que generalmente será nuestra carpeta de usuario). Es conveniente saber que no permite navegar a carpetas superiores a la inicial. Con el botón **New**, podemos crear:

- Carpetas (recomendable para organizar todo).
- Archivos de texto (con un editor bastante simple).
- Terminales.
- Cuadernos, que es la opción que vamos a estudiar en este tutorial.



Cuando entramos con el navegador se abre nuestra carpeta de usuario.

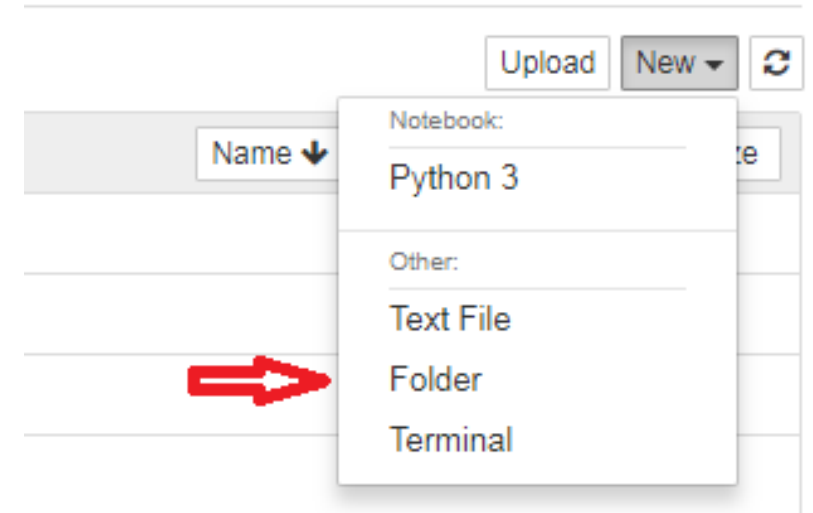
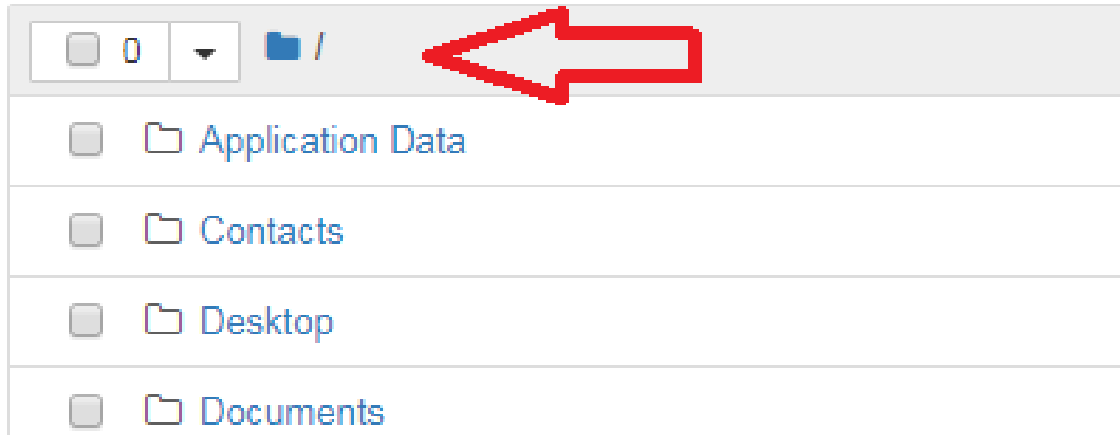
Files

Running

Clusters

Select items to perform actions on them.

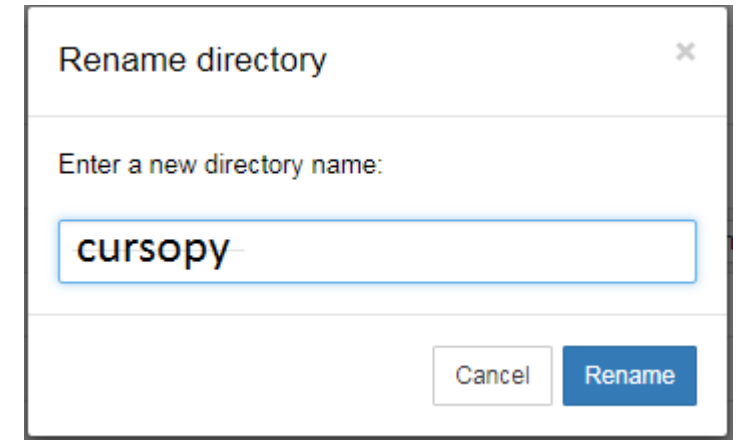
Vamos a asegurarnos que estamos en la raíz. Y vamos a crear una carpeta nueva (New->Folder)

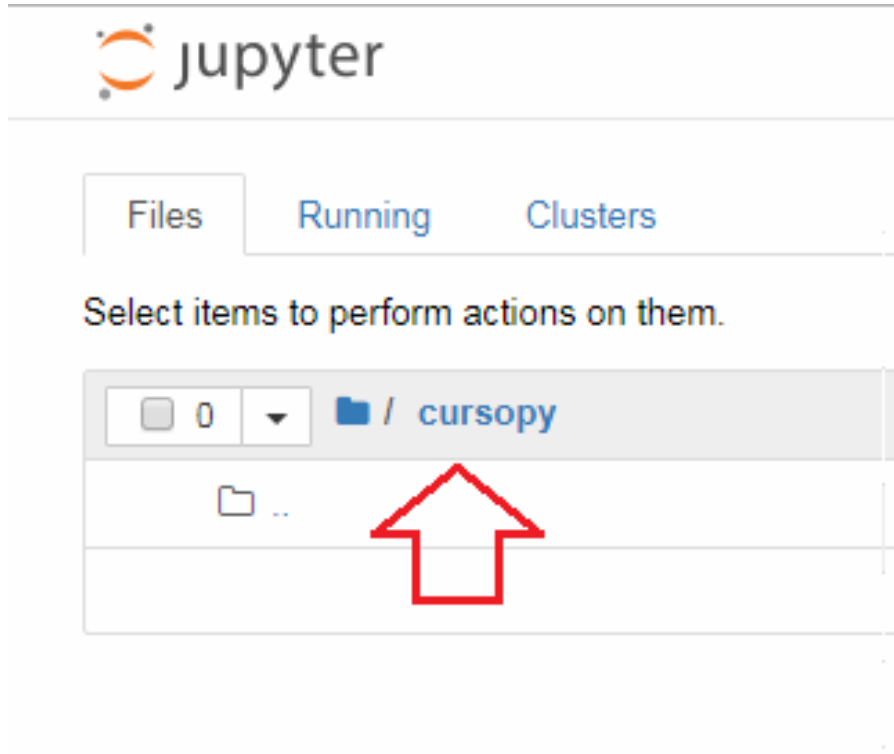


Veremos que nos ha creado una carpeta nueva

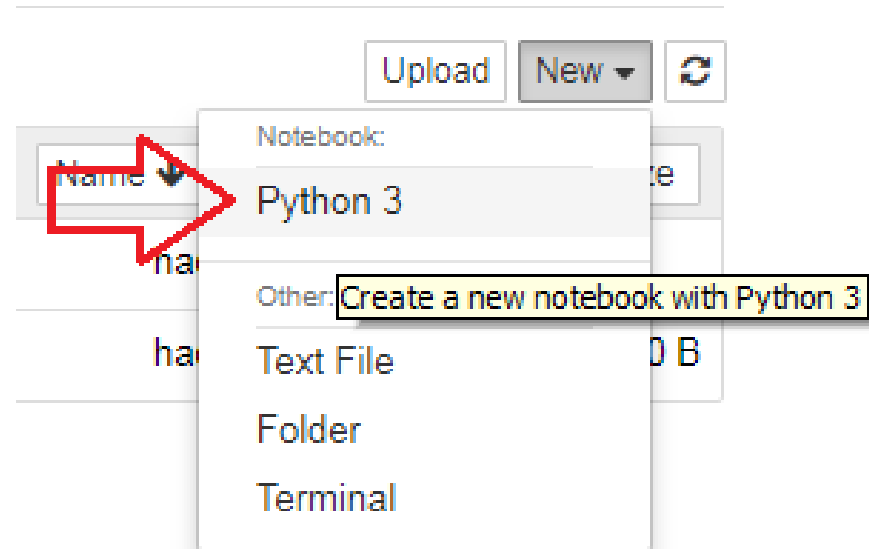


marcar el check e ir arriba, buscando el botón de renombrar





Esta carpeta **cursopy** será en adelante nuestra carpeta de trabajo. En ella crearemos los distintos ejemplos y ejercicios.

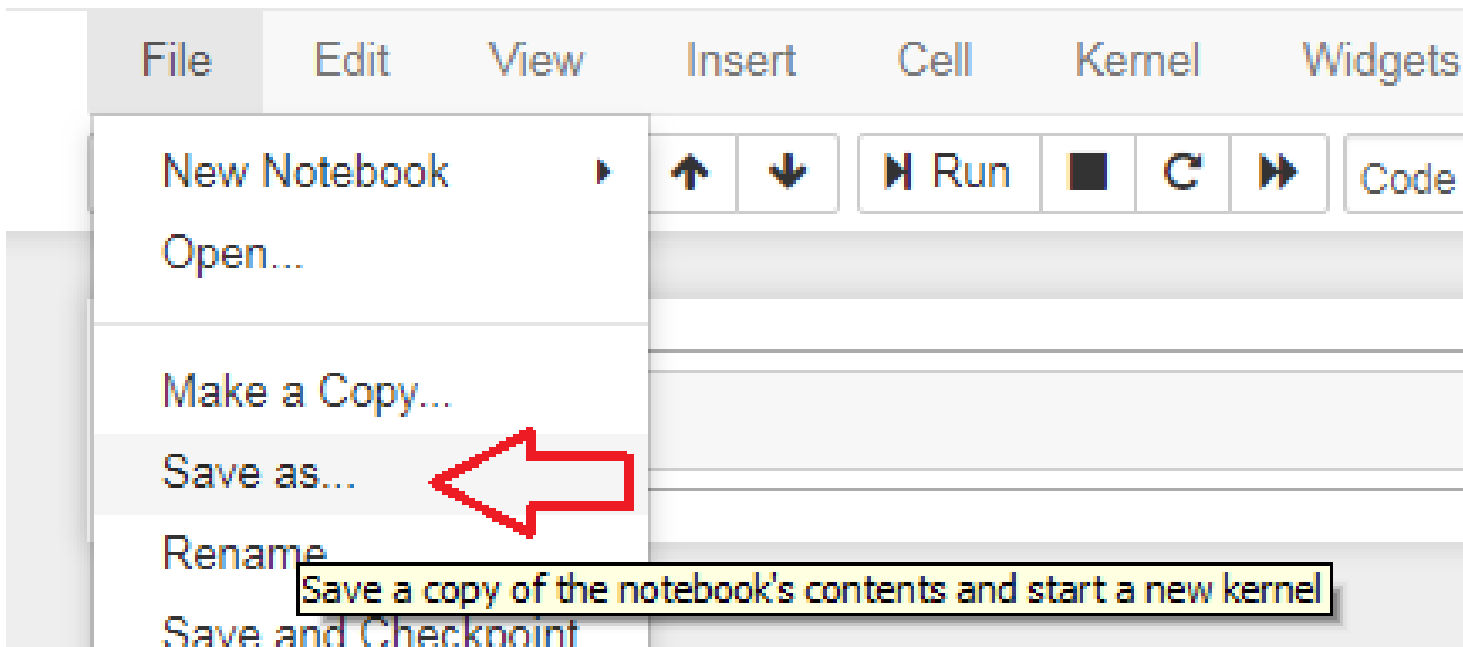


Vamos a crear un primer cuaderno de ejercicios :

[New->Python3](#)

Esto nos creará un cuaderno y lo abrirá para trabajar. Notad que no le hemos asignado nombre todavía

jupyter Untitled Last Checkpoint: hace unos segundos



Vamos a guardarlo con un nombre. Para organizarnos, crearemos un sistema.

Pondremos como prefijo el número de la sesión y luego la temática.

Save As

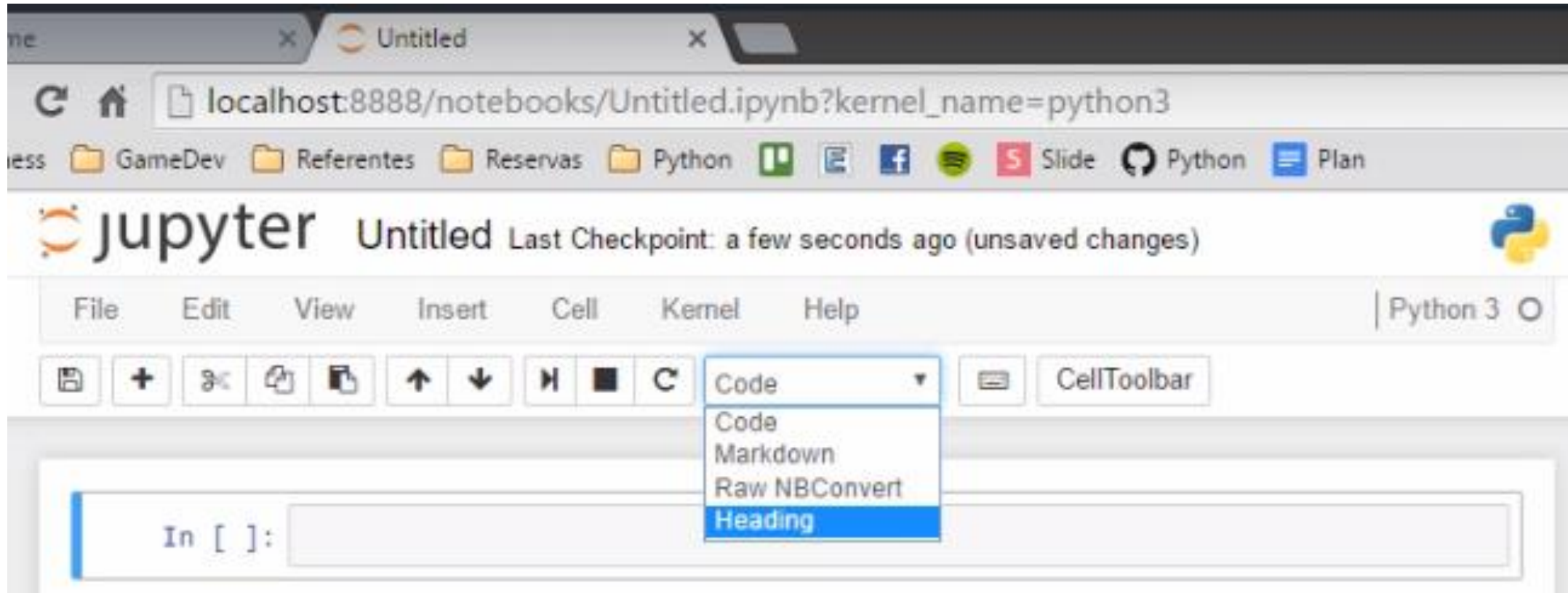
Enter a notebook path relative to notebook dir

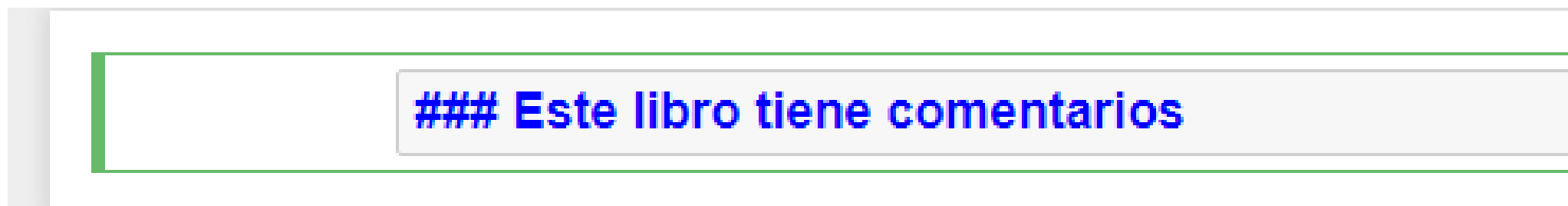
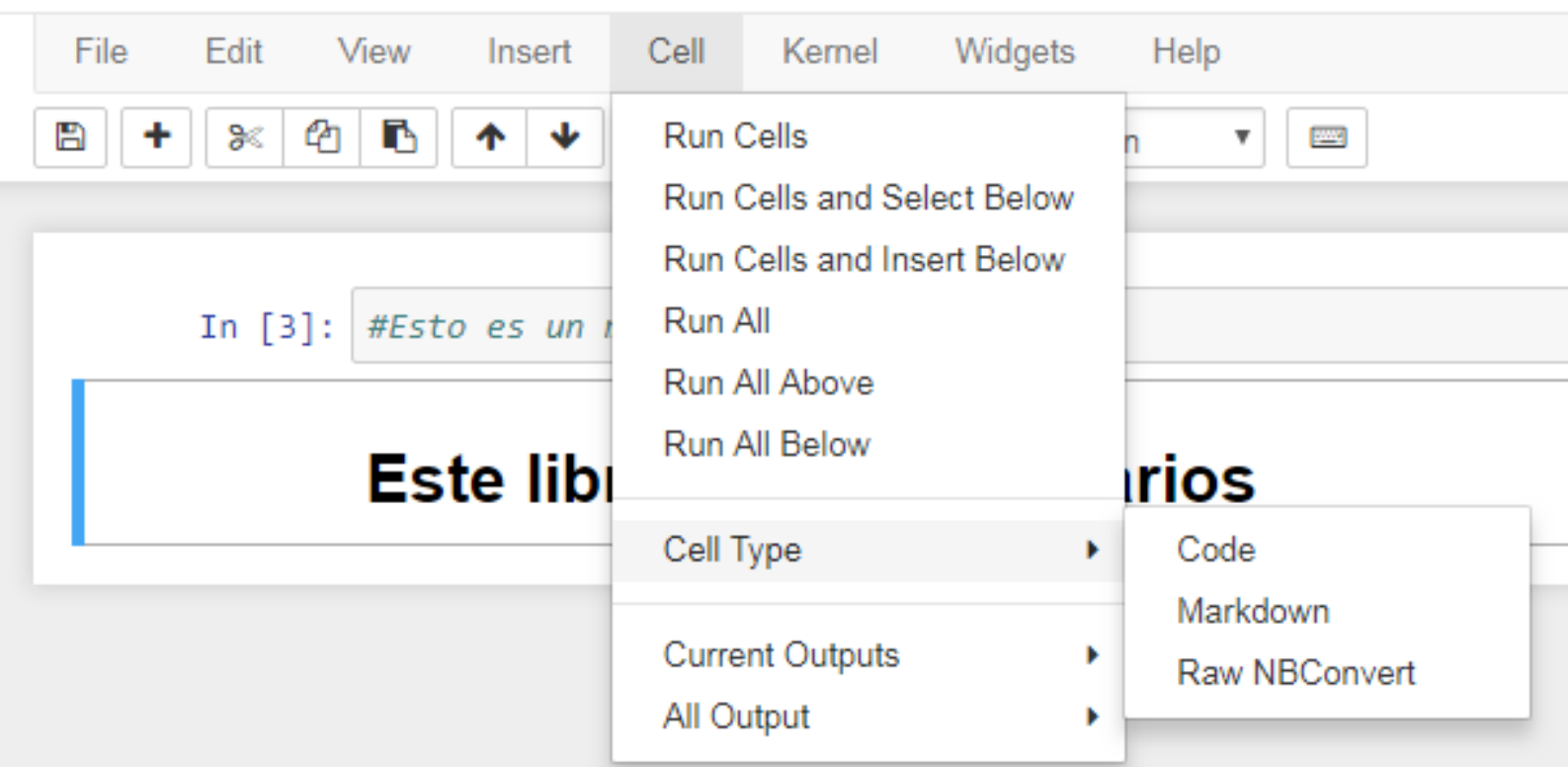
curscopy/ s02 _numeros

Cancel

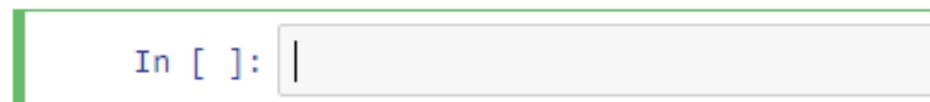
Save

Vemos los tipos de Casillas o de líneas que podemos crear.





Este libro tiene comentarios



Este libro tiene comentarios

```
# hola  
## cabecera 2  
##cabecera 3
```

 Run

Usar tipo de celda Markdown

Este libro tiene comentarios

hola

cabecera 2

##cabecera 3



```
In [ ]: print("Hola mundo")
```

Mayusculas



Entrar

- Edición, que permite modificar el contenido de las celdas, como si fuera un editor de texto. La celda que tengamos seleccionada se muestra en verde. Podemos entrar en el mismo seleccionando una celda y pulsando *Enter*.

```
In [ ]: Celda en modo Edición
```

- Mando, que nos permite ejecutar celdas o modificar el cuaderno y su estructura. La celda seleccionada se muestra en azul. Podemos volver al modo de mando pulsando *Esc*.

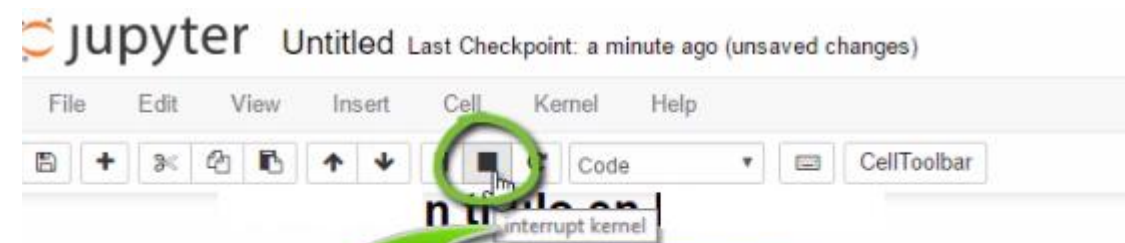
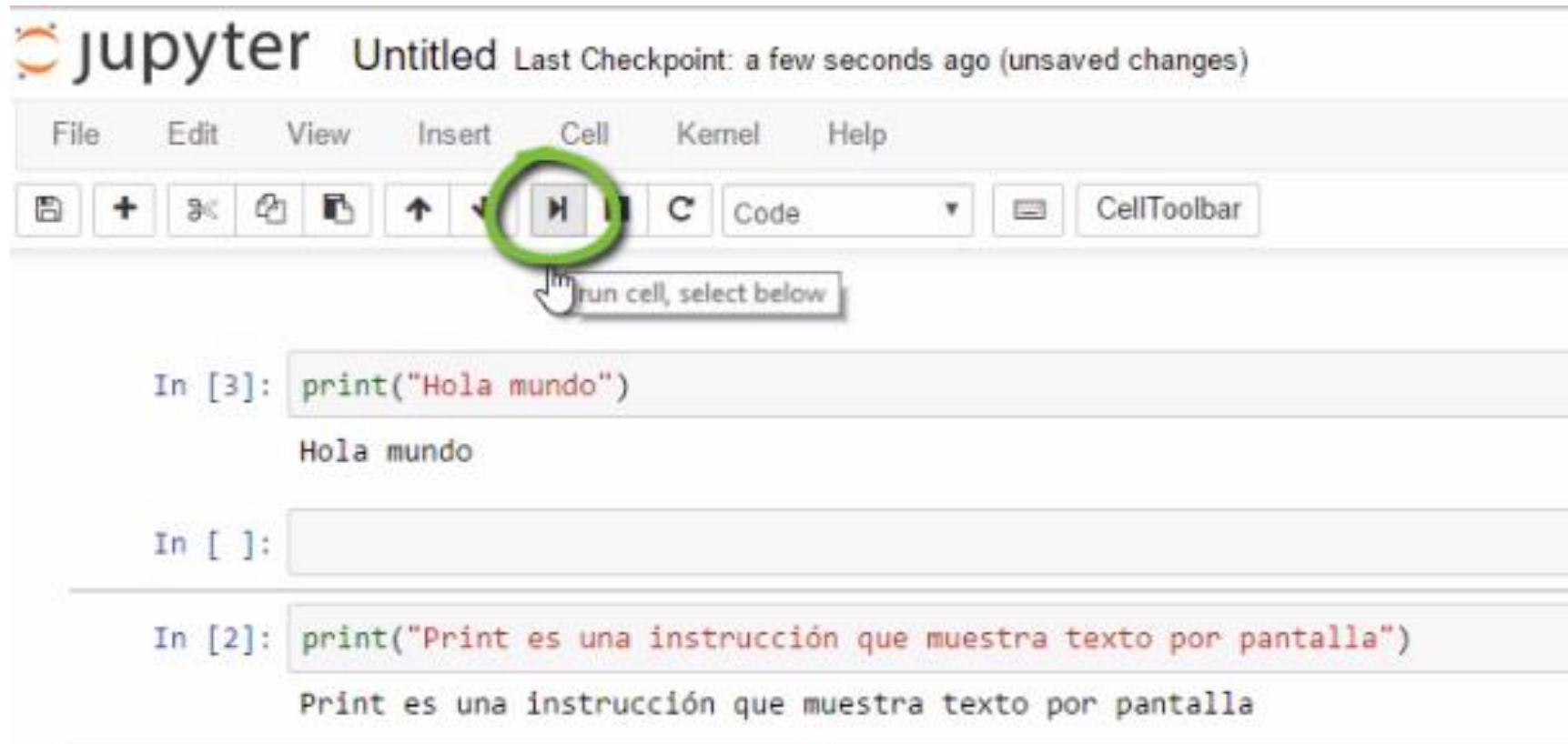
```
In [ ]: Celda en modo Mando
```

In [1]: `print("Hola mundo")`

Hola mundo

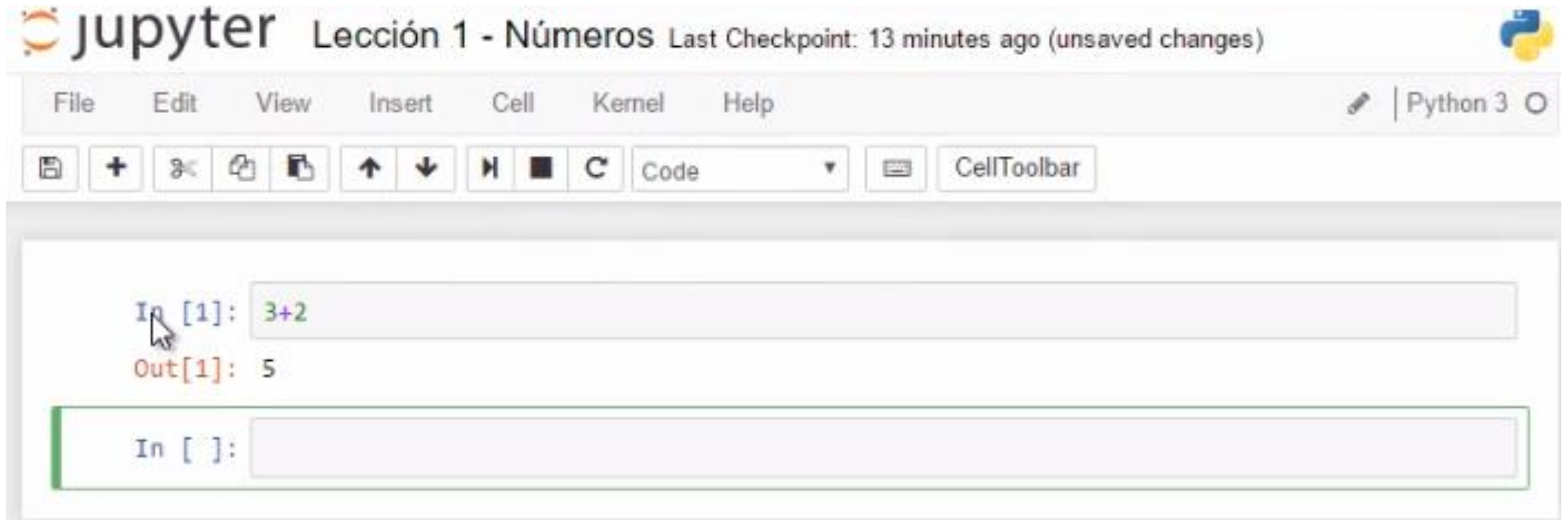
In [2]: `print("Print es una instrucción que muestra texto por pantalla")`

Print es una instrucción que muestra texto por pantalla



Si en algún momento se bloquea una celda [*]
probad a interrumpir o reiniciar el kernel

3. Trabajando con números (constantes)



The image shows a Jupyter Notebook interface. At the top, the title bar reads "jupyter Lección 1 - Números" followed by "Last Checkpoint: 13 minutes ago (unsaved changes)". Below the title bar is a menu bar with "File", "Edit", "View", "Insert", "Cell", "Kernel", and "Help". To the right of the menu bar is a "Python 3" selector. Below the menu bar is a toolbar with icons for saving, adding cells, undo, redo, and other actions. The main area of the notebook contains two input/output pairs. The first pair shows "In [1]: 3+2" and "Out[1]: 5". The second pair shows "In []:" followed by an empty input box.

```
In [1]: 3+2
Out[1]: 5

In [ ]:
```

Los tipos de datos numéricos almacenan valores numéricos.

In [1]: 3+2

Out[1]: 5

In [2]: 3-2

Out[2]: 1

In [3]: 3*2

Out[3]: 6

In [4]: # División
3/2

Out[4]: 1.5

In []:

In [5]: # Módulo
3%2

Out[5]: 1

In [6]: # Potencia
3**2

Out[6]: 9

In [7]: 1

Out[7]: 1

In [8]: 323239829389.238273283

Out[8]: 323239829389.2383

In [9]: 3-2+4*10

Out[9]: 41

Expresión con operador

Operación

A + B

Suma

A - B

Resta

A * B

Multiplicación

A % B

Resto

A / B

División real

A // B

División entera

A ** B

Potencia

enteros

flotantes

Leyes de precedencia

Leyes de precedencia (operadores matemáticos)

Operador	Descripción
**	Exponenciación
*, @, /, //, %	Multiplicación, multiplicación de matrices, división, división entera y resto
+, -	Adición y sustracción

Pràctica P01:

Quina d'aquestes expressions dona el resultat

Exercici 1 a 4

- Resultat 81, **opcions:** a) -3^{**4} b) (-3^{**4})
- Resultat 2396 **opcions:** a) $-5+7^{**4}$ b) $(-5+7)^{**4}$
- Resultat 70 **opcions:** a) $27*10 \% 100$ b) $27*10 / 100$
- Resultat -2697.4 **opcions:** a) $\text{round}(67435 * (-0.03 / 0.7), 2)$
b) $67435 * \text{round}(-0.03 / 0.7, 2)$

4. Trabajando con variables

```
In [10]: # Asignación de un valor  
# a una variable  
n = 3  
n
```

Out[10]: 3

```
In [11]: n+3
```

Out[11]: 6

```
In [12]: n*2
```

Out[12]: 6

```
In [13]: n*n
```

Out[13]: 9

```
In [14]: m=10
```

```
In [15]: n+m
```

Out[15]: 13

```
In [16]: n*m+10
```

Out[16]: 40

```
In [17]: n=10  
m=15  
n+m ]
```

Out[17]: 25

```

In [19]: n
Out[19]: 15

In [20]: m
Out[20]: 15

In [21]: n=m+10

In [22]: n
Out[22]: 25

In [23]: n=n+25

In [24]: n
Out[24]: 50

```

Nombre de una variable

- No puede comenzar por un número
- No puede contener espacios
- Tampoco símbolos especiales (áéíñç¿?)
- Podemos utilizar _ para los espacios

Exercici 5. Quin resultat donen aquestes operacions :

- $n=3$; $m=5$; Resultat de : $m-n**n$
- $m = n - 7$; Resultat de : $m-n**n$
- Després de b. Resultat de $m**2$
- Després de b. Resultat de $n**2$

- Los objetos número son creados al ser asignado un valor a los mismos. Ejemplo:

var1 = 1

var2 = 10

También puedes eliminar o referenciar un objeto

usando la sentencia del.

- La sintaxis de la sentencia es:

del var1[,var2[,var3[....,varN]]]]

Puedes borrar un solo objeto o múltiples objetos usando la sentencia del. Por ejemplo:

del var

del var_a, var_b


```
In [25]: nota_1 = 5  
nota_2 = 10  
nota_media = nota_1 + nota_2 / 2  
nota_media
```

Out[25]: 10.0

```
In [26]: nota_1 = 5  
nota_2 = 10  
nota_media = (nota_1 + nota_2) / 2  
nota_media
```

Out[26]: 7.5

```
In [28]: nota_1 = 2  
nota_2 = 5  
nota_media = (nota_1 + nota_2) / 2  
nota_media
```

Out[28]: 3.5

Python soporta 4 distintos tipos numéricos:

- **Int (Entero con signo).**- A menudo simplemente llamados enteros o ints, son números enteros positivos o negativos sin punto decimal.
- **Long(Entero largo) o largos.** Son números enteros de tamaño ilimitado.

Escritos como enteros y seguidos por una coma.

Int	long
10	51924361L
100	-0x19323L
-786	0122L
080	0xDEFA BCE CBD AECBFBAEI

En Python se pueden representar mediante el tipo int (de integer, entero) o el tipo long (largo). La única diferencia es que el tipo long permite almacenar números más grandes. Es aconsejable no utilizar el tipo long a menos que sea necesario, para no malgastar memoria.

El tipo int de Python se implementa a bajo nivel mediante un tipo long de C. Y dado que Python utiliza C por debajo, como C, y a diferencia de Java, el rango de los valores que puede representar depende de la plataforma. En la mayor parte de las máquinas el long de C se almacena utilizando 32 bits, es decir, mediante el uso de una variable de tipo int de Python puede almacenar números de -2^{31} a $2^{31} - 1$, o lo que es lo mismo, de -2.147.483.648 a 2.147.483.647. En plataformas de 64 bits, el rango es de -9.223.372.036.854.775.808 hasta 9.223.372.036.854.775.807.

- **Float(valor real de punto flotante) o flotantes.** Representan números reales y se escriben con un punto decimal dividiendo la parte entera y fraccional. Pueden estar en notación científica con **E** o **e** indicando la potencia de 10 ($2.5e2 = 2.5 \times 10^2 = 250$)

float	complex
0.0	3.14j
15.20	45.j
-21.9	9.322e-36j
32.3+e18	.876j

- **Complex(números complejos).** Son de la forma **a+bJ**, donde a y b son flotantes y **J** (o **j**), representa la raíz cuadrada de -1 (que es un número imaginario). **a** es la parte real del número y **b** es la parte imaginaria. No son muy usados en Python.

- Python convierte los números internamente en una expresión que contiene los tipos mixtos, a un tipo común para su evaluación. Pero algunas veces necesitarás forzar explícitamente un número a otro tipo para satisfacer los requerimientos de un operador o parámetros de una función.

```
num1 = input("Introduzca un entero: ")

resultado = num1 * 3.5

print (resultado)
```

Introduzca un entero: 7

```
-----
TypeError                                Traceback (most recent call last)
<ipython-input-8-aeeb7099f818> in <module>
      1 num1 = input("Introduzca un entero: ")
      2
----> 3 resultado = num1 * 3.5
      4
      5 print (resultado)
```

TypeError: can't multiply sequence by non-int of type 'float'

Ejercicio : Corrige este problema.

Ejemplos de conversión de tipos numéricos

- Tipo **int(x)** para convertir x a un entero simple.
- Tipo **float(x)** para convertir x a un número de punto flotante.
- Tipo **complex(x)** para convertir x a un número complex con una parte real x y una parte imaginaria zero.
- Tipo **complex(x, y)** convierte x e y a un número complejo con la parte real x y la parte imaginaria y. x e y son expresiones numéricas.

Ejemplo :

```
mientero = int(4557.89990)
print (mientero)
```

4557

Ejercicio 6: Que resultado dan las siguientes operaciones ?

- int(123456789012345)
- float (3 / 4)
- complex(7,9)

5. Algunas funciones matemáticas incluidas

- *abs(number)* : devuelve el valor absoluto de un número (siempre positivo)
- *min(val1, val2,...)* : devuelve el valor mínimo de una lista de de valores
- *max(val1, val2,...)* : devuelve el máximo de una lista de valores
- *pow(x,y)* : devuelve x elevado a la y
- *round(number, digits)* : redondea un número a los dígitos indicados
- *type(var)* : devuelve el tipo de la variable

Ejercicio 7: Que devuelve el siguiente programa ?

```
x =1
y =35656222554887711
z =-3255522
zz =-35.59
print(type(x))
print(type(y))
print(type(z))
print(type(zz))
```

Ejercicio 8: Que valores crees que pueden dar las siguientes instrucciones ?

```
abs(-8.7)
round(5.76543, 2)
divmod(8, 3)
max(12, -12.3, 18.9, -9.8)
min(12, -12.3, 18.9, -9.8)
```


Pràctica P02:

Programa els càlculs al quadern, copiant els enunciats com a comentari

1. Calcula a) el cub de 3, més el quadrat de 2 més 1.
b) el cub de 6, més el quadrat de 7 més 8.
2. Calcula l'àrea base i el volum dels armaris:
 - a) Alt 25 cms, ample 80 cms, fondo 15 cms.
 - b) Alt 75 cms, ample 30 cms, fondo 40 cms.
3. Calcula els 6 primers termes de la sèrie de Fibonacci (sense bucles i amb variables), a la diapo següent tens l'algorisme.



FORMULA MATEMÀTICA

$$F_0 = 0, \quad F_1 = 1,$$

and for $n > 1$

$$F_n = F_{n-1} + F_{n-2},$$

Exemples de la fórmula quan $n > 1$

Si $n=2$ llavors $n-1=1$ $n-2=0$

$$F_2 = F_1 + F_0$$

Si $n=3$ llavors $n-1=2$ $n-2=1$

$$F_3 = F_2 + F_1$$

Si $n=4$ llavors $n-1=3$ $n-2=2$

$$F_4 = F_3 + F_2$$

ALGORISME FIBONACCI(6) SENSE BUCLES

Inici

$$f_0 = 0$$

$$f_1 = 1$$

$$\text{CALCULAR : } f_2 = f_1 + f_0$$

$$\text{CALCULAR : } f_3 = f_2 + f_1$$

$$\text{CALCULAR : } f_4 = f_3 + f_2$$

$$\text{CALCULAR : } f_5 = f_4 + f_3$$

$$\text{CALCULAR : } f_6 = f_5 + f_4$$

IMPRIMIR ($f_0, f_1, f_2, f_3, f_4, f_5$)

Fin

6. Repaso de las funciones incluidas en Python (built-in)

<https://www.programiz.com/python-programming/methods/built-in>

Si tienes dudas sobre alguna de las funciones marcadas con asterisco, accede a la url indicada para obtener explicaciones y ejemplos.

Built-in Functions				
<code>abs()</code> *	<code>delattr()</code>	<code>hash()</code>	<code>memoryview()</code>	<code>set()</code>
<code>all()</code>	<code>dict()</code>	<code>help()</code>	<code>min()</code> *	<code>setattr()</code>
<code>any()</code>	<code>dir()</code>	<code>hex()</code>	<code>next()</code>	<code>slice()</code>
<code>ascii()</code>	<code>divmod()</code> *	<code>id()</code>	<code>object()</code>	<code>sorted()</code>
<code>bin()</code>	<code>enumerate()</code>		<code>oct()</code>	<code>staticmethod()</code>
<code>bool()</code>	<code>eval()</code>	<code>int()</code>	<code>open()</code>	<code>str()</code>
<code>breakpoint()</code>	<code>exec()</code>	<code>isinstance()</code>	<code>ord()</code>	<code>sum()</code>
<code>bytearray()</code>	<code>filter()</code>	<code>issubclass()</code>	<code>pow()</code> *	<code>super()</code>
<code>bytes()</code>	<code>float()</code> *	<code>iter()</code>		<code>tuple()</code>
<code>callable()</code>	<code>format()</code> *	<code>len()</code>	<code>property()</code>	<code>type()</code> *
<code>chr()</code>	<code>frozenset()</code>	<code>list()</code>	<code>range()</code>	<code>vars()</code>
<code>classmethod()</code>	<code>getattr()</code>	<code>locals()</code>	<code>repr()</code>	<code>zip()</code>
<code>compile()</code>	<code>globals()</code>	<code>map()</code>	<code>reversed()</code>	<code>__import__()</code>
<code>complex()</code> *	<code>hasattr()</code>	<code>max()</code> *	<code>round()</code> *	

Construcción de funciones numéricas

- Funciones matemáticas. Python incluye las siguientes funciones que realizan cálculos matemáticos:

Función	Devuelve (descripción)
<u>abs(x)</u>	El valor absoluto de x: la distancia (positiva) entre x y cero.
<u>ceil(x)</u>	El redondeo de x: El entero mas pequeño no menor que x
<u>cmp(x, y)</u>	-1 if x < y, 0 if x == y, or 1 if x > y
<u>exp(x)</u>	El exponencial de x: e^x
<u>fabs(x)</u>	El valor absoluto x.
<u>floor(x)</u>	El redondeo de x: el entero mas grande no mayor que

Ejemplo :

```
: positivo = abs(-33)  
print(positivo)
```

33

<u>log10(x)</u>	Logaritmo base-10 de x para $x > 0$.
<u>max(x1, x2,...)</u>	El mayor de sus argumentos: El valor mas cercano al infinito positivo
<u>min(x1, x2,...)</u>	El mas pequeño de sus argumentos: El valor más cercano al infinito negativo
<u>modf(x)</u>	Las partes fraccional y entera de x en una tupla de 2 elementos. Ambas partes tienen el mismo signo como x. La parte entera es devuelta como un flotante.
<u>pow(x, y)</u>	El valor de $x^{**}y$.
<u>round(x [,n])</u>	x redondeada a n digitos desde el punto decimal. Python redondea la parte decimal a 1 o -1 pero nunca a 0: round(0.5) es 1.0 y round(-0.5) es -1.0.
<u>sqrt(x)</u>	La raíz cuadrada de x para $x > 0$

Práctica P03:

1. Programa els càlculs al quadern, copiant els enunciats com a comentari

Demana 4 xifres (poden ser senceres o amb decimals, positives o negatives).

Mostra per pantalla :

- el màxim i el mínim.
- la mitja arrodonida a 3 decimals
- la suma dels valors absoluts, arrodonida a 2 decimals.

Librería math

Constantes matemáticas

El módulo define también dos constantes matemáticas

Constante	Descripción
pi	La constante matemática pi.
e	La constante matemática e.

Ejemplo :

```
: import math  
print(math.pi)
```

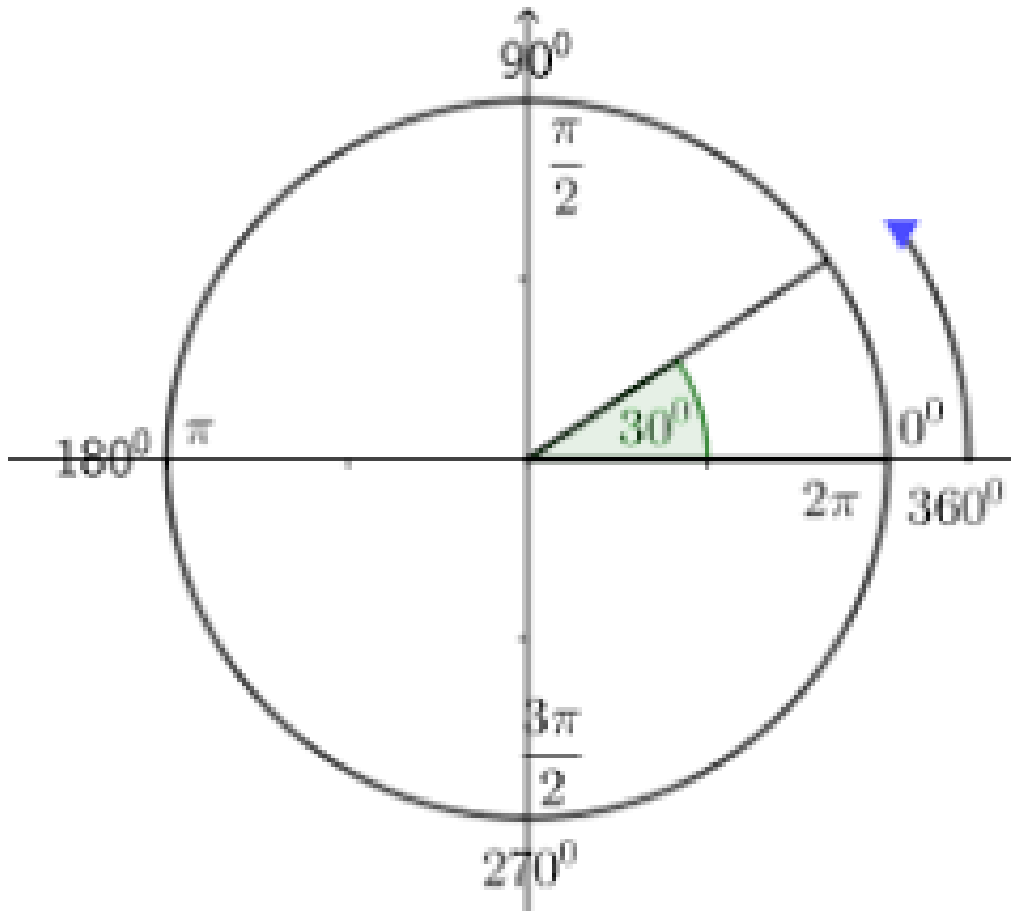
3.141592653589793

Funciones trigonométricas

Función	Descripción
<u>acos(x)</u>	Return the arc cosine of x, in radians.
<u>asin(x)</u>	Return the arc sine of x, in radians.
<u>atan(x)</u>	Return the arc tangent of x, in radians.
<u>atan2(y, x)</u>	Return atan(y / x), in radians.
<u>cos(x)</u>	Return the cosine of x radians.
<u>hypot(x, y)</u>	Return the Euclidean norm, $\sqrt{x^2 + y^2}$.
<u>sin(x)</u>	Return the sine of x radians.
<u>tan(x)</u>	Return the tangent of x radians.
<u>degrees(x)</u>	Converts angle x from radians to degrees.
<u>radians(x)</u>	Converts angle x from degrees to radians.

```
import math
rad = math.radians(90)
print ("el seno es : ", math.sin(rad))
print ("el coseno es : ", round(math.cos(rad),2))
```

```
el seno es : 1.0
el coseno es : 0.0
```

```
math.sin(math.pi / 2)  
math.sin(math.radians(90))
```

Práctica P03

Ejercicio 2 : Calcula el seno, coseno y tangente de :

45 grados

73 grados

Ejercicios Avanzados

Práctica P04

- Sea $x=-3$, $y=8$, $z=4$.

1. $(3x^2+y^2+5)/2z^2$

2. $\text{raiz}(x^2+y^3)+\text{raiz}(3y^2+2z)$

3. $(x^4+2y+z)/3yz$

4. $\text{Raiz}(5x^2/(2y+z))+\text{raiz}(3y^2/(2z+y))$

5. Pide un número por pantalla, conviértelo a entero positivo, y calcula :

Ejemplo :

1. su logaritmo en base 10

2. su seno y su tangente (consideralo grados)

redondea los resultados a 4 decimales.

```
numero = input("Introduce un número")
```

Librería random

Funciones numéricas aleatorias

- Los números aleatorios son usados para juegos, simulaciones, pruebas, seguridad y aplicaciones privadas. Python Incluye las siguientes funciones que son comúnmente usadas.

Ejemplo :

```
: import random  
print (random.random())
```

0.5165282228290028

Función	Retorna (descripción)
<u>choice(seq)</u>	Un item aleatorio de una lista, tupla, o cadena.
<u>randrange</u> <u>([start,] stop</u> <u>[,step])</u>	Un elemento seleccionado aleatoriamente de un rango(start, stop, step)
<u>random()</u>	A random float r, such that 0 is less than or equal to r and r is less than 1
<u>seed([x])</u>	Sets the integer starting value used in generating random numbers. Call this function before calling any other random module function. Returns None.
<u>shuffle(lst)</u>	Randomizes the items of a list in place. Returns None.
<u>uniform(x, y)</u>	A random float r, such that x is less than or equal to r and r is less than y