



# Programación Python

José Javier Galán Hernández :



# Programación Python

## 1. Introducción a la programación

# 1 Introducción a la programación

Tradicionalmente podemos definir la programación como la automatización de tareas, especialmente las que suponen un mayor esfuerzo para el ser humano o aquellas que por ser repetitivas no aportan valor y se pueden realizar desatendida.

En la ciencia de datos la programación adquiere un valor extra, además del automatismo implican la codificación del código necesario para representar de forma visual el resultado de aplicar IA a un conjunto enorme de datos, hablamos de big data y machine learning.



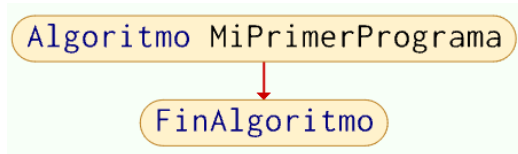
Con este objeto surgen lenguajes como R y Python muy enfocados a obtener estos resultados pero que en esencia son lenguajes de programación, por lo tanto es necesario tener al menos unas destrezas mínimas programando.



# 1 Introducción a la programación

## Pseudocódigo

El pseudocódigo es independiente del lenguaje de programación, representaremos mediante símbolos lo que queremos conseguir, en la programación estructurada el pseudocódigo seguirá un orden secuencial y lógico. A continuación vemos la composición básica de un programa, todo programa debe tener un comienzo y un final. Dicho programa representa la solución de un problema y esto es lo que se conoce como algoritmo.



**Algoritmo** MiPrimerPrograma

**FinAlgoritmo**

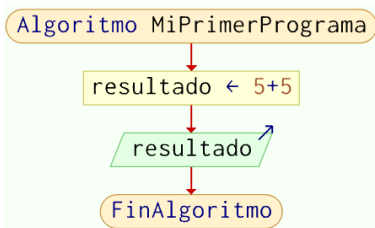


# 1 Introducción a la programación

## Pseudocódigo

Un algoritmo puede tener tantas acciones, conocidas como instrucciones, dentro de codificación como sean necesarias. Para conseguir el éxito del objetivo propuesto deberemos utilizar todas las instrucciones a nuestro alcance en el orden secuencial correcto con la mayor eficiencia posible.

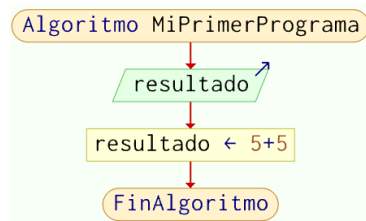
Orden correcto:



```
Algoritmo MiPrimerPrograma
    resultado ← 5+5
    Escribir resultado
FinAlgoritmo
```

Resultado de la ejecución: 10

Orden incorrecto:



```
Algoritmo MiPrimerPrograma
    Escribir resultado
    resultado ← 5+5
FinAlgoritmo
```

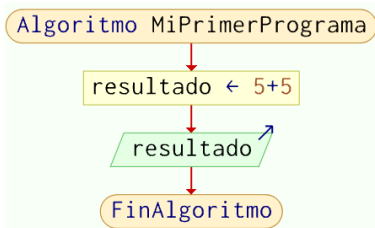
Resultado de la ejecución: 0

# 1 Introducción a la programación

## Pseudocódigo

Un algoritmo puede tener tantas acciones, conocidas como instrucciones, dentro de codificación como sean necesarias. Para conseguir el éxito del objetivo propuesto deberemos utilizar todas las instrucciones a nuestro alcance en el orden secuencial correcto con la mayor eficiencia posible.

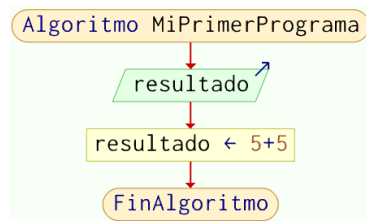
Orden correcto:



```
Algoritmo MiPrimerPrograma
    resultado ← 5+5
    Escribir resultado
FinAlgoritmo
```

Resultado de la ejecución: 10

Orden incorrecto:



```
Algoritmo MiPrimerPrograma
    Escribir resultado
    resultado ← 5+5
FinAlgoritmo
```

Resultado de la ejecución: 0



# Programación Python

## 2. PYTHON

## 2 Python



**PYTHON**

Python es un lenguaje de programación multiparadigma. Esto significa que más que forzar a los programadores a adoptar un estilo particular de programación, permite varios estilos: programación orientada a objetos, programación imperativa y programación funcional.

Es un lenguaje interpretado, de alto nivel y enfocado principalmente a la legibilidad y facilidad de uso

Los lenguajes interpretados se pueden ejecutar sin compilar usando un intérprete, no tienen código binario y su código se llama script. Al disponer de intérprete no dependen de la plataforma y suelen ser más lentos que los compilados.



# 2 Python

Python destaca por el elevado uso de bibliotecas y una gran cantidad de paquetes (pilas incluidas), ser software libre (licencia “Python Software Foundation License”) distribuyéndose gratuitamente, sin pagar licencias.

Su creador, Guido van Rossum lo define como código limpio y legible. Simple sin ser limitado.



# 2 Python

Filosofía Python

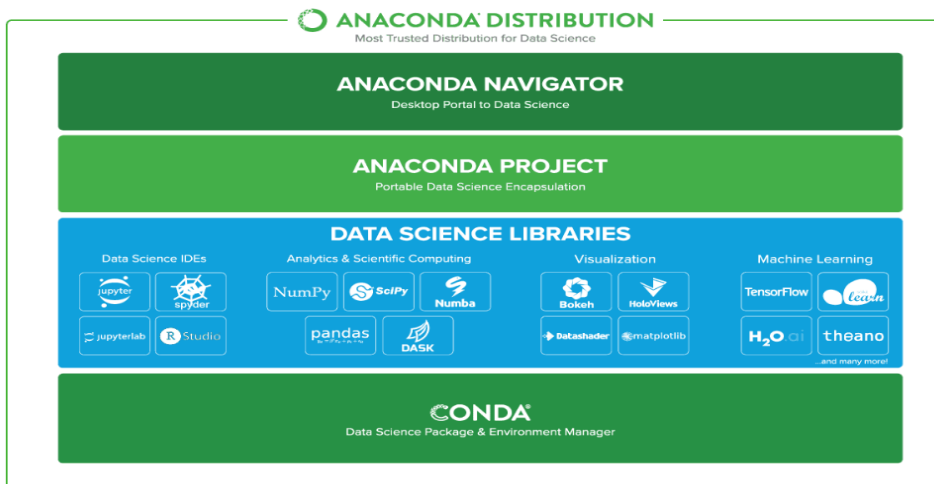
- Hermoso es mejor que feo.
- Explícito es mejor que implícito.
- Simple es mejor que complejo.
- Complejo es mejor que complicado.
- Plano es mejor que anidado.
- Escaso es mejor que denso.
- La legibilidad cuenta.
- Los casos especiales no son lo suficientemente especiales como para romper las reglas.
- Aunque la practicidad supera la pureza.
- Los errores nunca deben pasar en silencio.
- A menos que sea silenciado explícitamente.
- Ante la ambigüedad, rechaza la tentación de adivinar.
- Debe haber una, y preferiblemente solo una, forma obvia de hacerlo.
- Aunque esa manera puede no ser obvia al principio a menos que seas holandés.
- Ahora es mejor que nunca.
- Aunque nunca es mejor que el ahora correcto.
- Si la implementación es difícil de explicar, es una mala idea.
- Si la implementación es fácil de explicar, puede ser una buena idea.
- Los espacios de nombres son una gran idea, ¡hagamos más de eso!

# Programación Python

## 3. Instalación y primer contacto con el entorno de trabajo: Anaconda y Jupyter

# 3. Instalación y primer contacto con el entorno de trabajo: Anaconda y Jupyter

**Anaconda** es una Suite de código abierto que abarca una serie de aplicaciones, librerías y conceptos diseñados para el desarrollo de la Ciencia de datos como Python. En líneas generales Anaconda Distribution es una distribución que funciona como un gestor de entorno, un gestor de paquetes y que posee una colección de más de 720 paquetes de código abierto.



# 3. Instalación y primer contacto con el entorno de trabajo: Anaconda y Jupyter

Accedemos a la url <https://www.anaconda.com/download> y rellenamos los datos de registro o pulsamos “Skip registration”



Products ▾

Solutions ▾

Resources ▾

Company ▾

[↓ Free Download](#)

[Sign In](#)

[Get Demo >](#)

## Distribution

Register to get everything you need to get started on your workstation including Cloud Notebooks, Navigator, AI Assistant, Learning and more.

- ✓ Easily search and install thousands of data science, machine learning, and AI packages
- ✓ Manage packages and environments from a desktop application or work from the command line
- ✓ Deploy across hardware and software platforms
- ✓ Distribution installation on Windows, MacOS, or Linux

### Free Download

Get access in 30 seconds. Completely free.\*

[Get Started >](#)

[Returning Users >](#)

\*Subject to our [Terms of Service](#). Use of Anaconda's offerings at an organization of more than 200 employees/contractors requires a paid business license unless your organization is eligible for discounted or free use. [See Pricing](#).

[Skip registration](#)

# 3. Instalación y primer contacto con el entorno de trabajo: Anaconda y Jupyter

Descargamos la versión correspondiente a nuestro Sistema Operativo.

## Download Now

Download Anaconda Distribution or Miniconda by choosing the proper installer for your machine. Learn the difference from our [Documentation](#).

### Distribution Installers

Download

For installation assistance, refer to [troubleshooting](#).

**Windows** ^

Python 3.13

📎 64-Bit Graphical Installer (914M)

### Miniconda Installers

Download


For installation assistance, refer to [troubleshooting](#).

**Windows** v

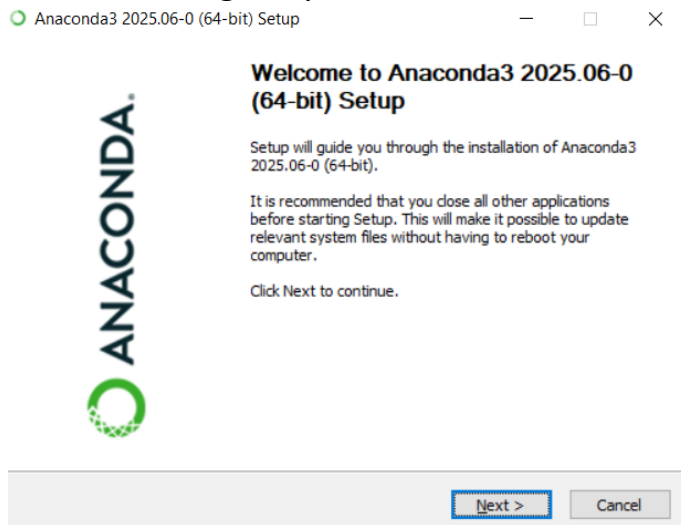
**Mac** v

# 3. Instalación y primer contacto con el entorno de trabajo: Anaconda y Jupyter

Se ha descargado el ejecutable de instalación:

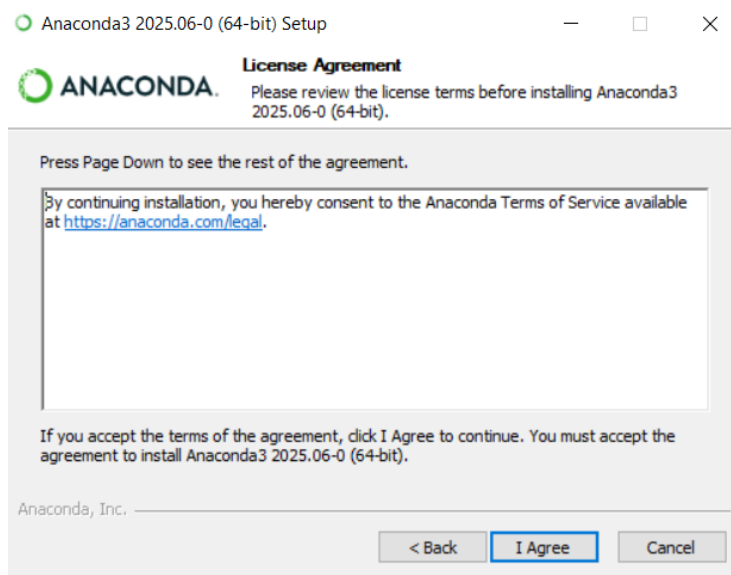
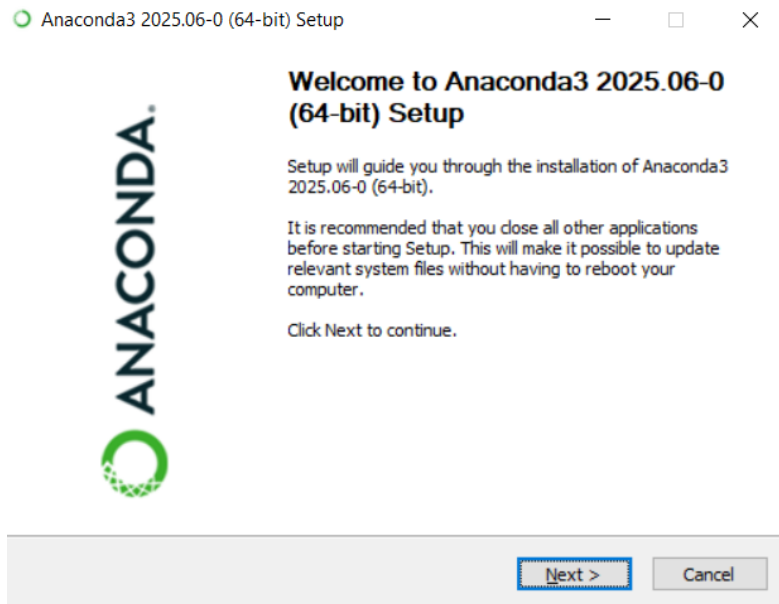
 Anaconda3-2025.06-0-Windows-x86\_64.exe

Hacemos doble click en el fichero descargado y hacemos click en **Next**.



# 3. Instalación y primer contacto con el entorno de trabajo: Anaconda y Jupyter

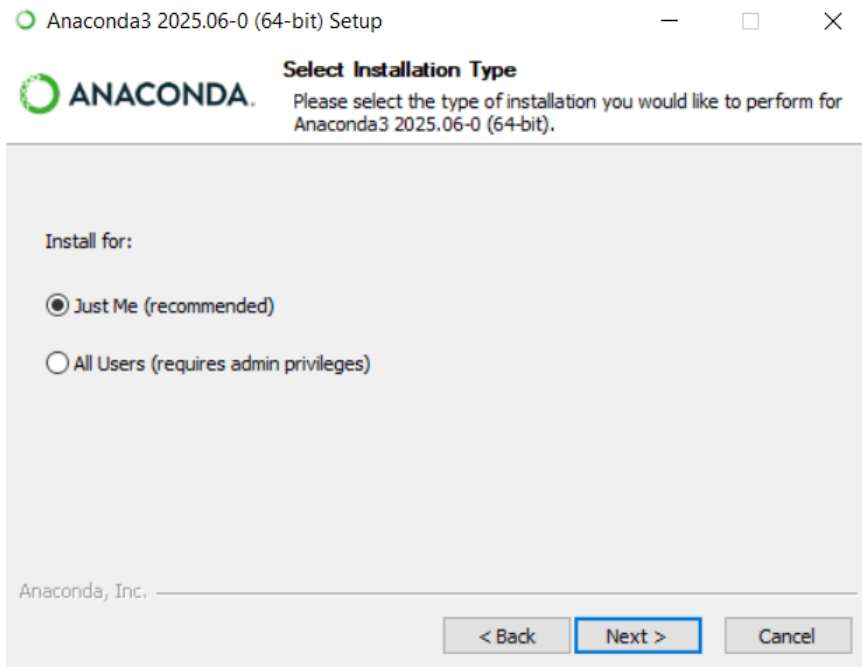
Si estamos de acuerdo con el acuerdo de licencia hacemos click en **I agree**





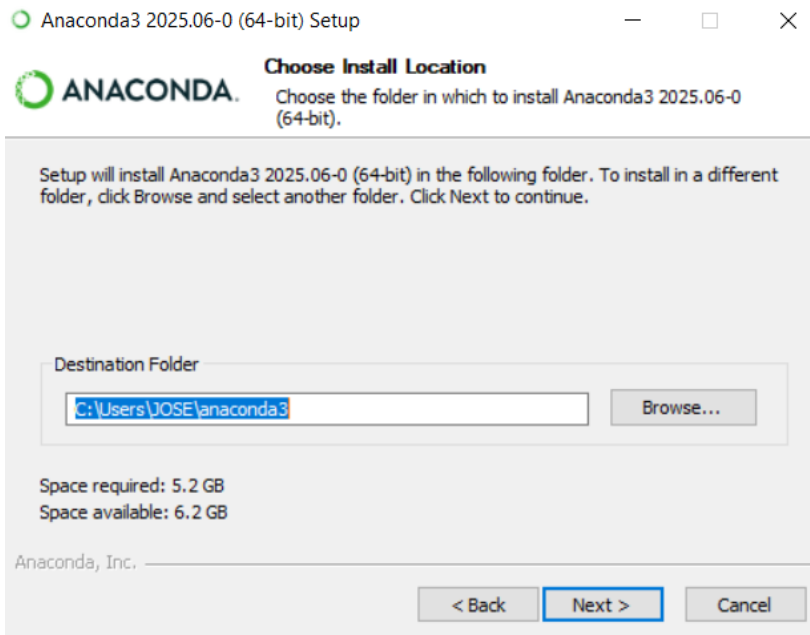
# 3. Instalación y primer contacto con el entorno de trabajo: Anaconda y Jupyter

Seleccionamos la opción solo para nuestro usuario y pulsamos **Next**



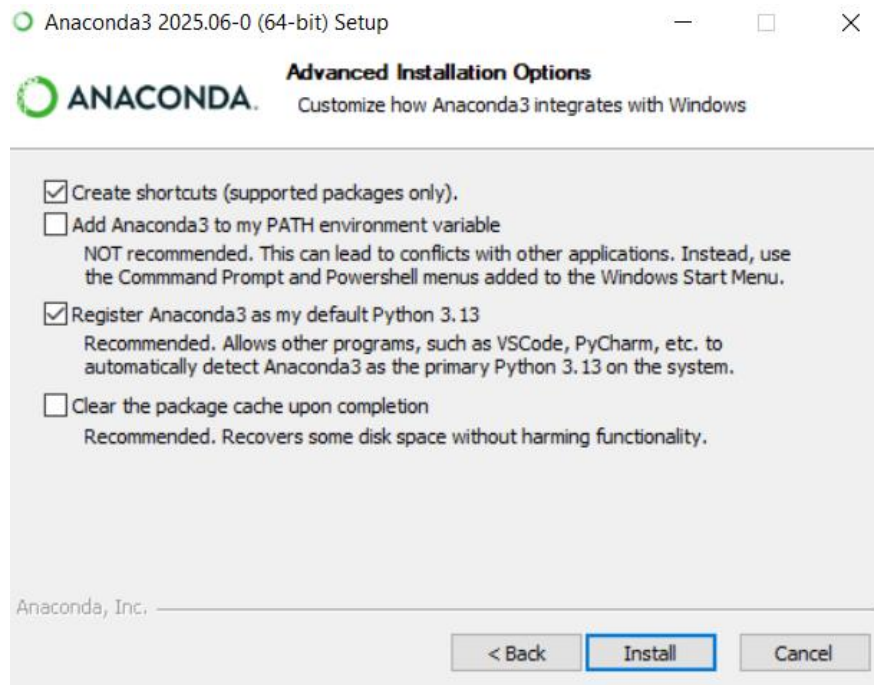
# 3. Instalación y primer contacto con el entorno de trabajo: Anaconda y Jupyter

Ahora indicamos la ruta donde se copiarán los archivos de instalación, podemos dejar la ruta por defecto y pulsar **Next**



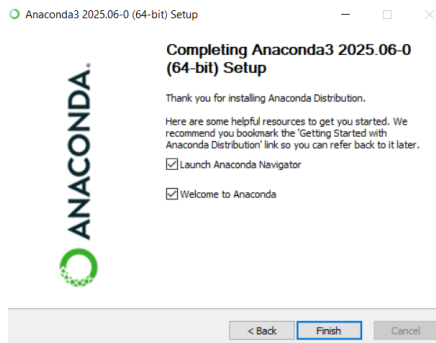
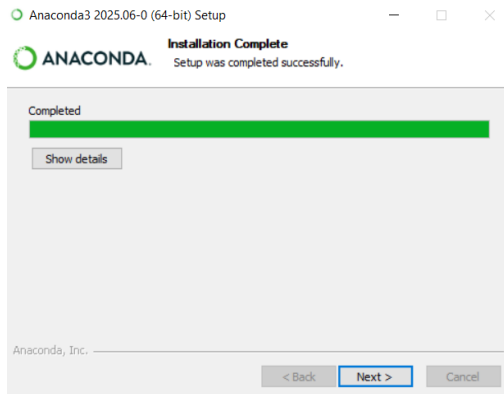
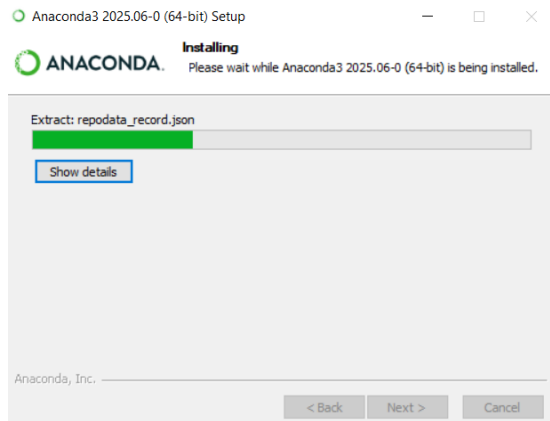
# 3. Instalación y primer contacto con el entorno de trabajo: Anaconda y Jupyter

Finalmente pulsamos **Install**



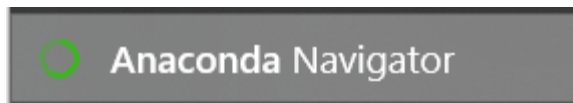
# 3. Instalación y primer contacto con el entorno de trabajo: Anaconda y Jupyter

Tras unos minutos la instalación termina. Una vez instalado, pulsamos Finish.

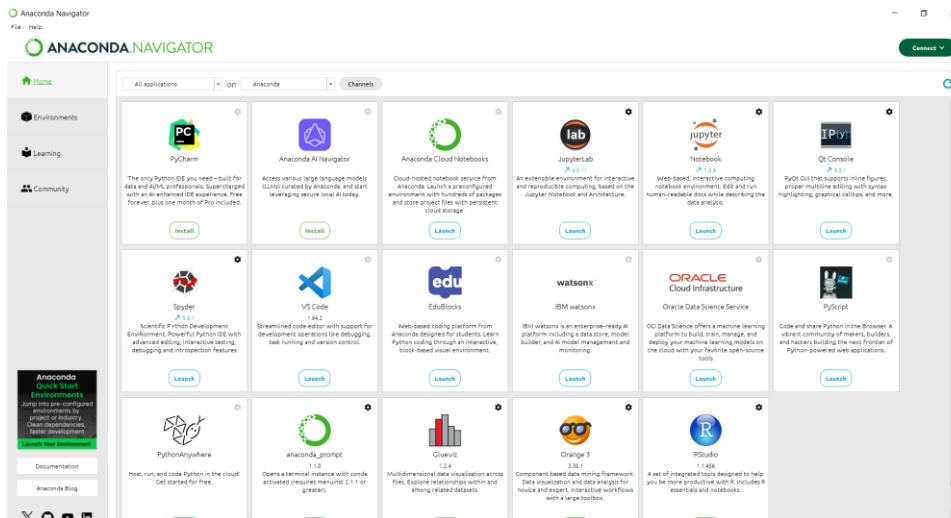


# 3. Instalación y primer contacto con el entorno de trabajo: Anaconda y Jupyter

Podemos abrirlo haciendo click en el acceso directo



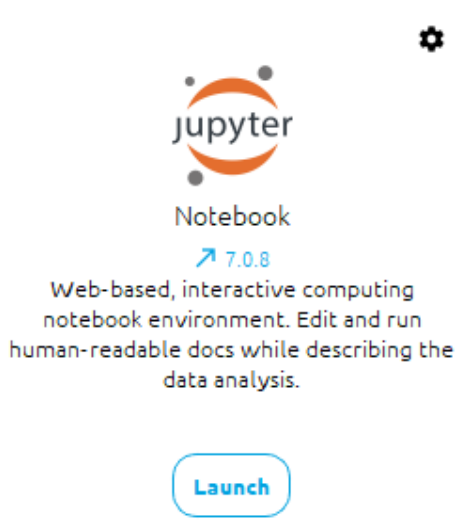
Viendo la pantalla principal



# 3. Instalación y primer contacto con el entorno de trabajo: Anaconda y Jupyter

**Jupyter** es una aplicación web que se puede abrir desde cualquier navegador permitiendo crear documentos que contienen código y pueden ser interpretados como el lenguaje de programación Python.

**ABRIMOS JUPYTER.** En el menú principal de Anaconda hacemos click en **Launch** de la aplicación Jupyter



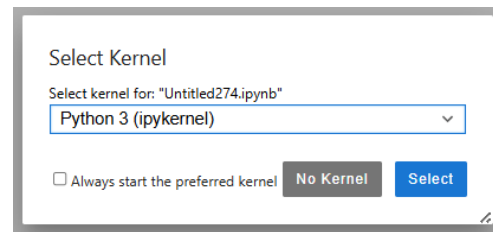
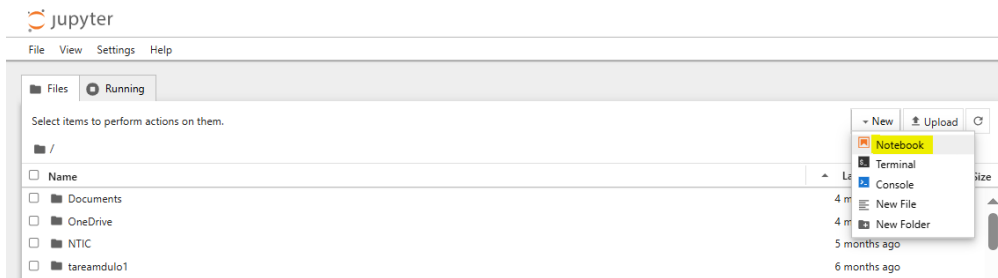
# 3. Instalación y primer contacto con el entorno de trabajo: Anaconda y Jupyter

Esto abre en el navegador el entorno Jupyter donde podemos desarrollar nuestros proyectos.

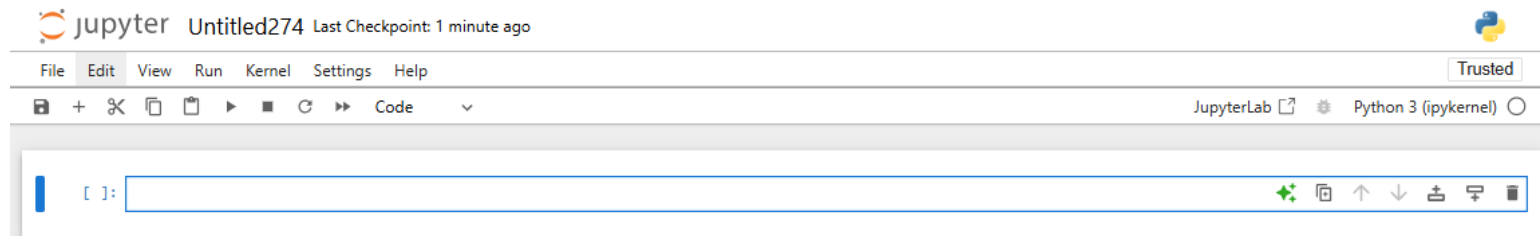


# 3. Instalación y primer contacto con el entorno de trabajo: Anaconda y Jupyter

Para comenzar hacemos click en **New / Notebook**. Seleccionamos Python 3.



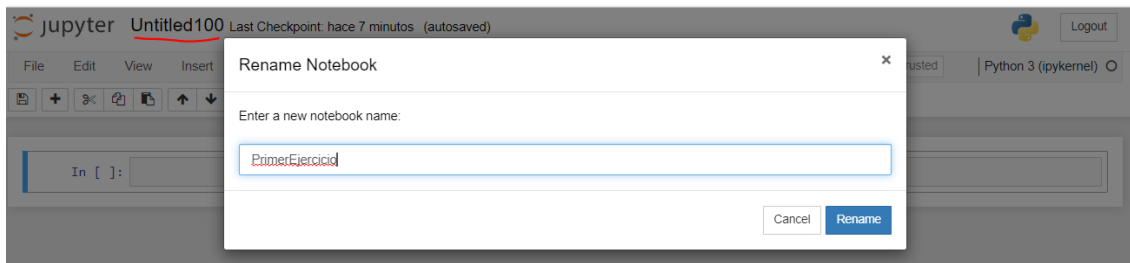
Con esto hemos creado nuestro primer Notebook y podemos empezar a desarrollar código.





# 3. Instalación y primer contacto con el entorno de trabajo: Anaconda y Jupyter

Para establecer un nombre al notebook que estamos desarrollando debemos hacer click en Untitled100 y escribir el nombre deseado.



Cada celda puede ser de 4 tipos distintos:

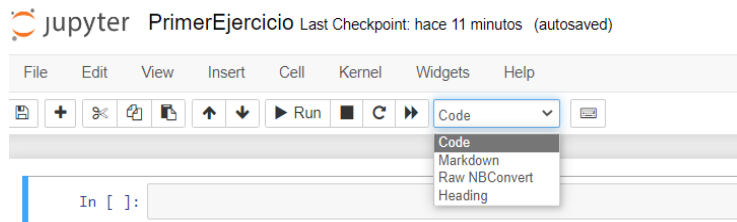
Code: sirve para escribir código ejecutable

Markdown: sirve para escribir texto

Raw NBConvert: escribe código que no se ejecutara

Heading: se utiliza para escribir titulos

Se seleccionan desde el menú desplegable:

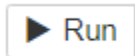


# 3. Instalación y primer contacto con el entorno de trabajo: Anaconda y Jupyter

Para crear nuevas celdas se utiliza el símbolo +



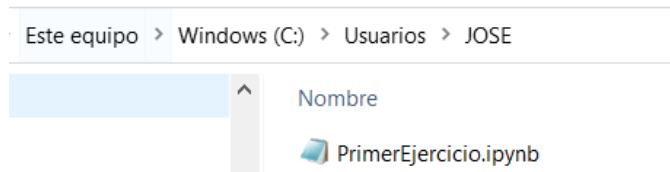
Para ejecutar la celda se utiliza el botón Run



Existen comandos especiales precedidos de % como %pwd que nos permite ejecutar comandos típicos de Linux. Este en este caso nos muestra el directorio de trabajo actual.

```
%pwd  
'C:\\Users\\JOSE'
```

Podemos ver que efectivamente se ha guardado aquí nuestro notebook.



# ...ahora...comencemos!

***Instalemos Anaconda!***

***Abramos Jupyter y creemos un proyecto llamado MiPrimerPrograma!***



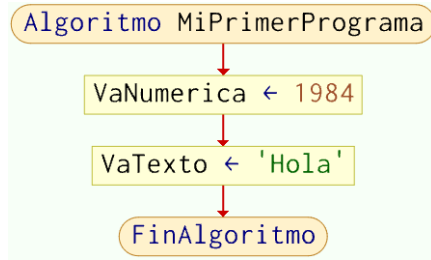


# Programación Python

## 4. Variables

# 4. Variables

Los datos que manejamos durante la ejecución de un programa deben ser almacenados para su tratamiento, este almacenamiento se hará por defecto de forma temporal en la memoria principal del ordenador hasta que se decida almacenar en la memoria física. Estos datos sufrirán cambios a lo largo de la ejecución del programa, su contenido variara y por eso lo llamaremos variables.



```
Algoritmo MiPrimerPrograma
    VaNumerica ← 1984
    VaTexto ← 'Hola'
FinAlgoritmo
```

El tipo de dato que podemos almacenar en cada variable será distinto dependiendo del uso que queramos darle, no es lo mismo disponer de variables para operaciones aritméticas que contendrán números que variables cuyo contenido es texto y que por lo tanto las letras que contienen no se podrán sumar entre sí.

# 4. Variables

Los tipos de datos primitivos más comunes en Python son:

Tipo	Descripción
bool	Booleano
int	Numérico
str	Texto

Python es un lenguaje de tipos (llamado tipado), esto quiere decir que el tipo de dato que almacena puede cambiar a lo largo de la ejecución. Por ello aunque usaremos en Python el término variable, por motivos didácticos, sería más correcto hablar de tipos.

En Python no es necesario crear una variable ni definir su tipo. Directamente podemos asignarle valor y utilizarla.

# 4. Variables

Ejemplo:

```
Valor1=1
Valor2=2
Resultado=Valor1+Valor2
Resultado=True
Resultado="Hola"
Resultado
```

'Hola'

```
Resultado=Resultado+Valor1
```

-----  
**TypeError**

Traceback (most recent call last)

Input In [7], in <cell line: 1>()

----> 1 Resultado=Resultado+Valor1

**TypeError:** can only concatenate str (not "int") to str



# 4. Variables

Usamos **None** para indicar la ausencia de valor y **type** para conocer de qué tipo es una variable.

```
Valor1=None  
Valor1
```

Indicamos con None que la variable Valor1 no tiene valor, por ello al intentar mostrarla por pantalla no muestra nada.

```
type(Valor1)
```

```
NoneType
```

Continuando con el ejemplo anterior usamos type sobre la variable Valor1. Como no tiene valor el tipo mostrado por pantalla es NoneType.

```
Valor1=2  
type(Valor1)
```

```
int
```

A la variable Valor1 le asignamos el valor 2, por ello al preguntar cuál es su tipo mediante el comando type obtenemos como resultado int.

```
Valor1="Hola"  
type(Valor1)
```

```
str
```

A la variable Valor1 le asignamos el valor "Hola", por ello al preguntar cuál es su tipo mediante el comando type obtenemos como resultado str.

```
Valor1=False  
type(Valor1)
```

```
bool
```

A la variable Valor1 le asignamos el valor False, por ello al preguntar cuál es su tipo mediante el comando type obtenemos como resultado bool.



# Programación Python

## 5. Tipos y operadores

# 5. Tipos y operadores. Conversión de tipo

Con el comando **int()** podemos convertir texto en número

```
Valor1=20
Valor2="2"
Resultado=Valor1+Valor2
Resultado
```

```
-----
TypeError                                Traceback (most recent call last)
Input In [22], in <cell line: 3>()
      1 Valor1=20
      2 Valor2="2"
----> 3 Resultado=Valor1+Valor2
      4 Resultado

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

```
Valor1=20
Valor2="2"
Resultado=Valor1+int(Valor2)
Resultado
```

22

# 5. Tipos y operadores. Conversión de tipo

Con el comando **str()** podemos convertir números en texto siempre que sea posible

```
Valor1="El numero elegido es el: "  
Valor2=2  
Frase=Valor1+Valor2  
Frase
```

```
-----  
TypeError                                Traceback (most recent call last)  
Input In [24], in <cell line: 3>()  
      1 Valor1="El numero elegido es el: "  
      2 Valor2=2  
----> 3 Frase=Valor1+Valor2  
      4 Frase
```

**TypeError:** can only concatenate str (not "int") to str

```
Valor1="El numero elegido es el: "  
Valor2=2  
Frase=Valor1+str(Valor2)  
Frase
```

'El numero elegido es el: 2'

# 5. Tipos y operadores. Operadores Lógicos

Operador lógico	Descripción
And	Deben cumplirse todas las condiciones
Or	Debe cumplirse alguna condición
Not	Negación de la condición

```
#Usamos == porque no es una asignacion, es pregunta.  
Valor1=2  
Valor2=5  
(Valor1==2 and Valor2==5)
```

True

```
Valor1=2  
Valor2=5  
(Valor1==2 and Valor2==2)
```

False

```
Valor1=2  
Valor2=5  
(Valor1==2 or Valor2==2)
```

True

```
Valor1=2  
Valor2=5  
not(Valor1==2 or Valor2==2)
```

False

Como se cumplen todas las condiciones, usando **AND**, el resultado es verdadero

Como no se cumplen todas las condiciones, usando **AND**, la evaluación es falsa

Con **OR** solo es necesario que se cumpla alguna condición, por lo tanto a diferencia del ejemplo anterior este caso es verdadero.

Con **NOT** conseguimos el valor lógico contrario porque lo negamos, como es verdadero lo convertimos en falso.

## 5. Tipos y operadores. Operadores Comparativos

Operador comparativo	Descripción
==	Igual a
!=	Distinto a
>	Mayor que
<	Menor que
>=	Mayor o igual que
<=	Menor o igual que

```
Valor1=2  
Valor2=5  
  
Valor1==Valor2
```

False

Valor1 no es igual que Valor2, es falso.

```
Valor1=2  
Valor2=5  
  
Valor1!= Valor2
```

True

En este caso como no son iguales, pero es lo que preguntamos, la respuesta es verdadero.

```
Valor1=2  
Valor2=5  
  
Valor1>=Valor2
```

False

Valor1 no es mayor ni igual que Valor2, no se cumple, siendo falso.

## 5. Tipos y operadores. Operadores Aritméticos

Operador aritmético	Descripción
+	Suma
-	Resta
*	Multiplicación
/	División
**	Exponente
//	División entera
%	Modulo

```
print("Introduce el primer valor: ")
Valor1=int(input()) #Convertimos el texto introducido en numero, error si se introduce texto
print("Introduce el segundo valor: ")
Valor2=int(input())

resultado=Valor1+Valor2
print("El resultado de la suma es: " + str(resultado))
```

```
Introduce el primer valor:
2
Introduce el segundo valor:
4
El resultado de la suma es: 6
```

## 6. Entrada y Salida de datos.

**input()** obtiene el texto escrito por teclado.

**print()** puede recibir argumentos como `end=""` que sustituye el salto de línea por el valor entre comillas, o valores que mostrar.

```
In [7]: print("Dame un numero")
numero = input()
print ("El numero introducido es: " + numero)
```

```
Dame un numero
4
El numero introducido es: 4
```

```
In [11]: print("Dame un numero ", end="")
numero = input()
print ("El numero introducido es: " + numero)
```

```
Dame un numero 4
El numero introducido es: 4
```

```
In [15]: print("Dame un numero ", end="")
numero = input()
print ("El numero introducido es: ", numero)
```

```
Dame un numero 4
El numero introducido es: 4
```



# ...ahora...comencemos!

***Instalemos Anaconda!***

***Abramos Jupyter y creemos un proyecto llamado MiPrimerPrograma!***

<b><i>Ejercicio 1</i></b>	<b>Crea dos variables:</b> variable1 con el valor numérico 5 y variable2 con el valor de texto "Hola".
<b><i>Ejercicio 2</i></b>	<b>Crea dos variables:</b> variable1 y variable2 que reciban valores por teclado. Después, muestra el resultado de su <b>multiplicación</b> .
<b><i>Ejercicio 3</i></b>	<b>Pide tu nombre</b> por teclado y muestra un <b>mensaje personalizado</b> deseándote un buen día.
<b><i>Ejercicio 4</i></b>	<b>Pide tres variables:</b> variable1, variable2 y variable3. Introduce en ellas valores numéricos por teclado. Si los tres son iguales, el programa mostrará True; en caso contrario, False. ( <i>Todavía no podemos usar if</i> ).
<b><i>Ejercicio 5</i></b>	<b>Pide tres variables:</b> variable1, variable2 y variable3. Introduce en ellas valores numéricos por teclado. Si alguna de ellas es mayor que 100, el programa mostrará True; en caso contrario, False. ( <i>Todavía no podemos usar if</i> ).

