



Jul. 2022

# Unidad 1: Entornos de desarrollo para programadores Python



- Algunas herramientas que mejoran la forma en que se puede
  - Experimentar con Python
  - Depurar los programas
  - Trabajar de manera efectiva

# ¿Cómo instalar Python?

- [python.org/downloads](https://python.org/downloads)
- Activar la opción “Add Python to PATH”

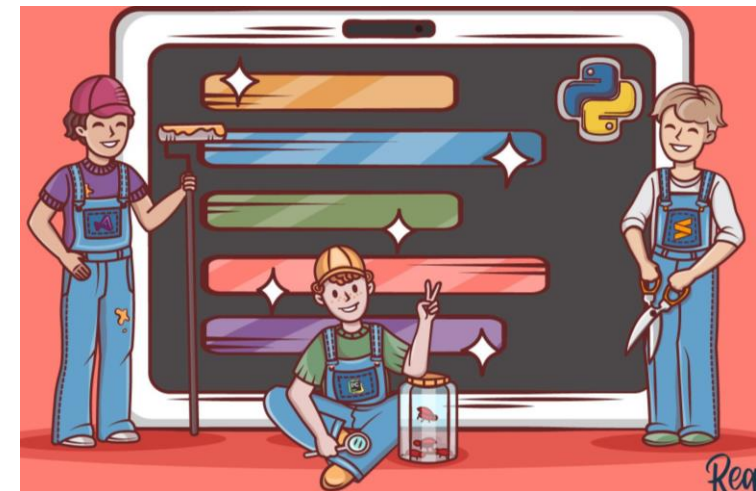


# Editores e IDEs generales con soporte Python

Editor	Web	Category	Escrito en
Eclipse + PyDev	<a href="http://pydev.org">pydev.org</a>	IDE	Java
Sublime Text	<a href="http://sublimetext.com">sublimetext.com</a>	IDE	Python
Atom	<a href="http://atom.io">atom.io</a>	IDE	JavaScript
GNU Emacs	<a href="http://gnu.org/software/emacs">gnu.org/software/emacs</a>	Editor de texto	Lisp
Vi / Vim	<a href="http://vim.org">vim.org</a>	Editor de código	C
Visual Studio	<a href="http://visualstudio.com/vs">visualstudio.com/vs</a>	IDE	C#, C++
Visual Studio Code	<a href="http://code.visualstudio.com">code.visualstudio.com</a>	Editor de código	TypeScript, JavaScript
PyCharm	<a href="http://jetbrains.com/pycharm">jetbrains.com/pycharm</a>	IDE específico	Java
Spyder	<a href="http://github.com/spyder-ide">github.com/spyder-ide</a>	IDE específico	Python
Thonny	<a href="http://thonny.org">thonny.org</a>	IDE específico	Python
Python IDLE	<a href="http://python.org">python.org</a>	IDE específico	Python
ipython (kernel Jupyter)	<a href="http://ipython.org">ipython.org</a>	Shell	Python

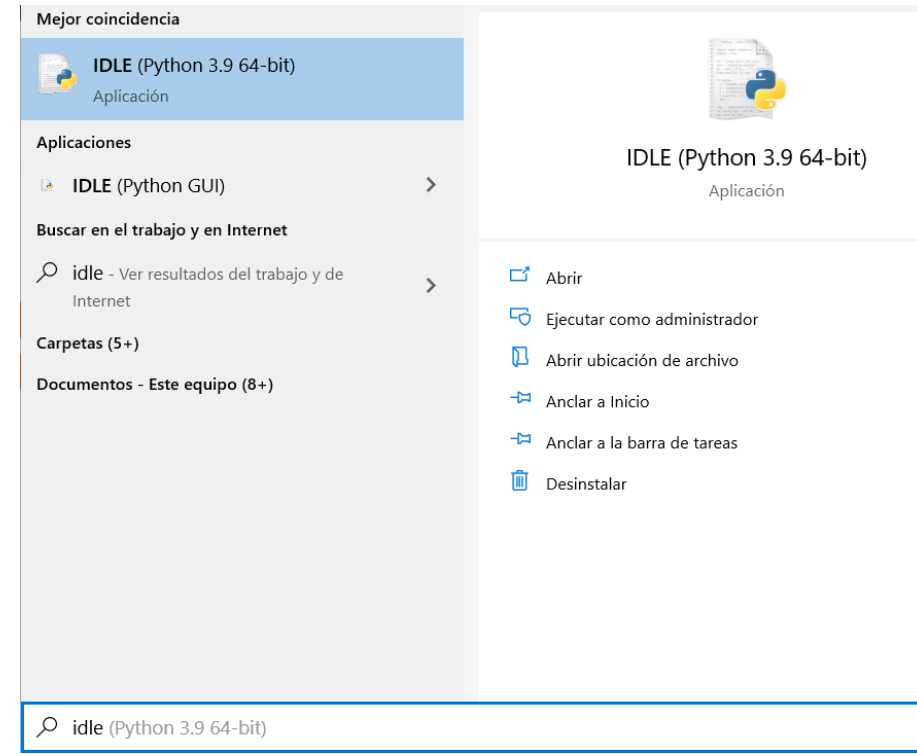
# ¿Qué IDE de Python es el adecuado para ti?

- Los nuevos desarrolladores de Python
  - deberían probar soluciones basadas en editores de texto para otras tareas
- Para desarrolladores de software
  - puede que te resulte más fácil añadir las capacidades de Python a tu conjunto de herramientas



# Python IDLE

- Cada instalación de Python viene con un Entorno Integrado de Desarrollo y Aprendizaje
- Se puede usar como
  - un intérprete interactivo
    - bucle básico de lectura-evaluación-impresión (REPL)
  - un editor de archivos
    - editar y guardar archivos de texto con la extensión .py

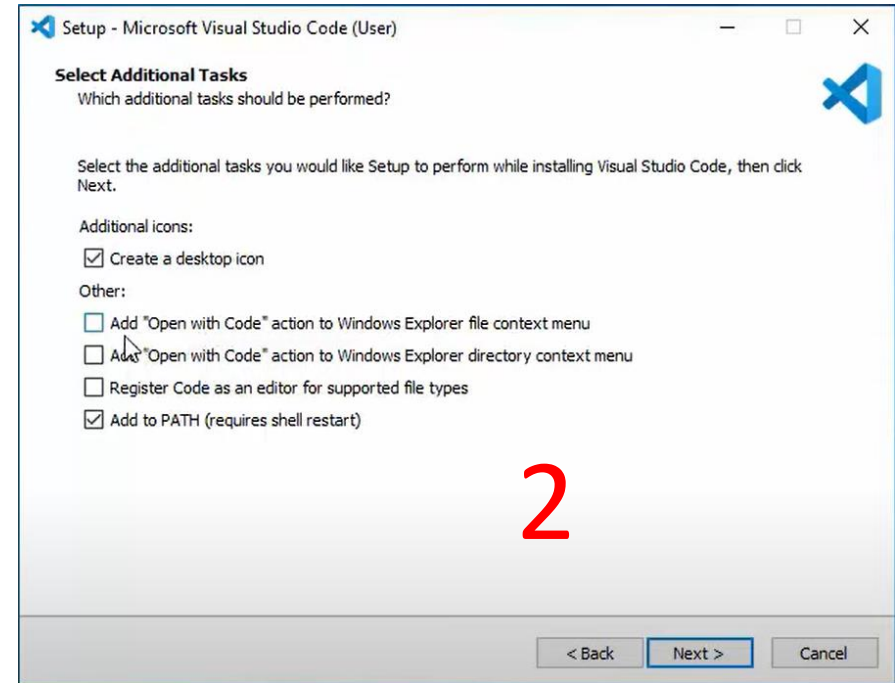
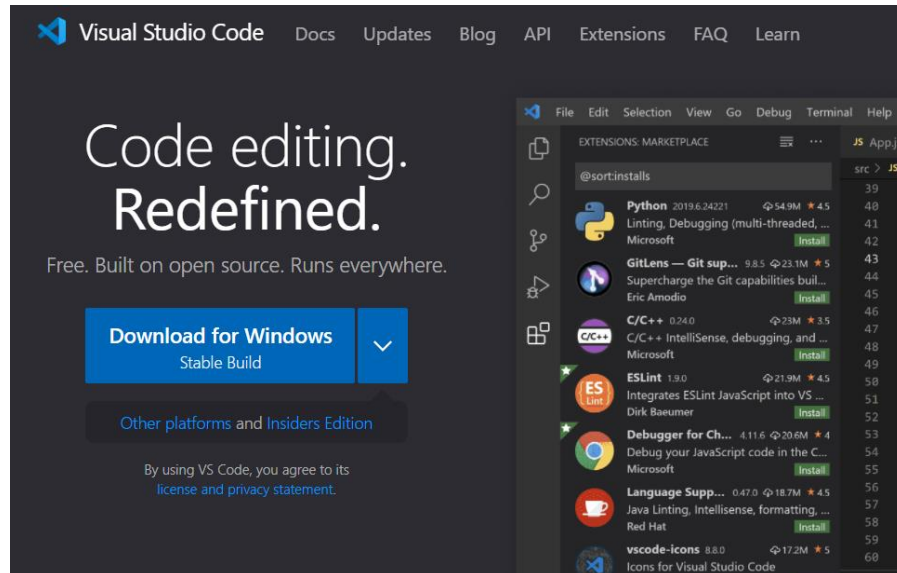




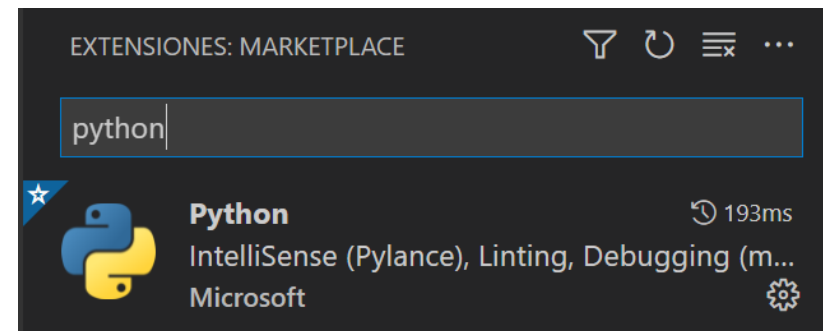
# VS code

1. [code.visualstudio.com](https://code.visualstudio.com)
2. Activar la opción "Add to PATH"
3. Añadir alguna extensión Python

1



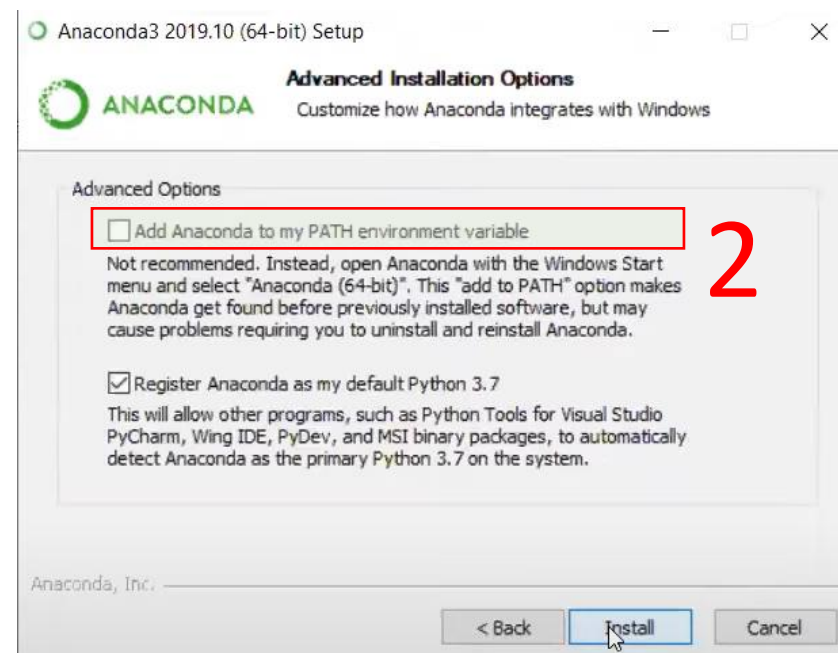
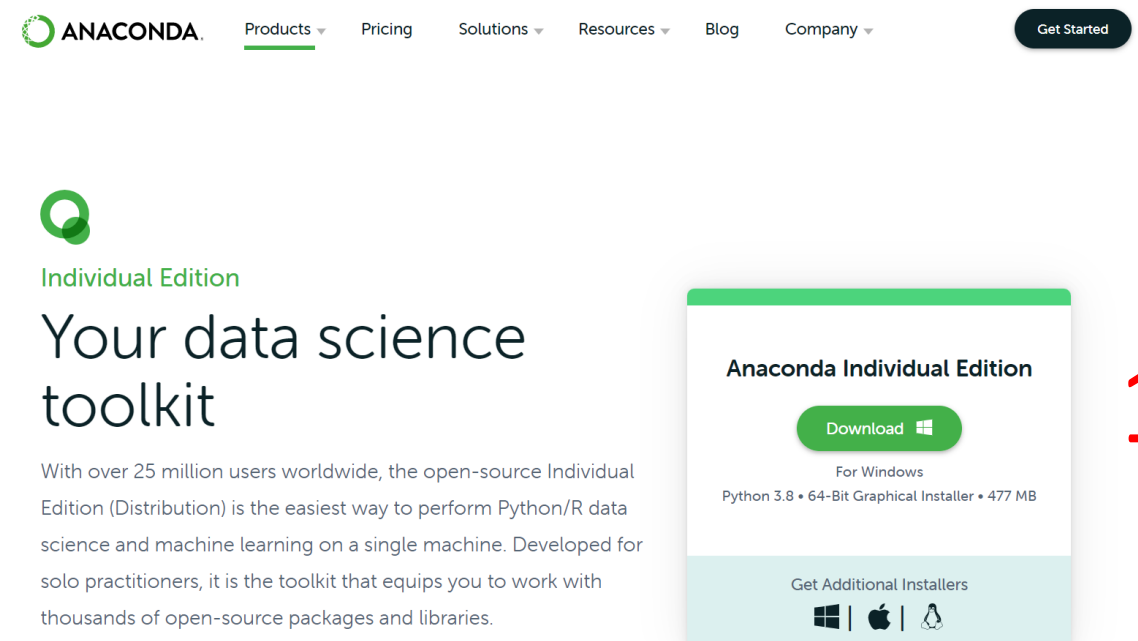
2



3

# Anaconda

1. Descargar desde [anaconda.com/products/individual](https://anaconda.com/products/individual)
2. Activar la opción “Add Anaconda to my PATH environment variable”





# ipython (ipython.org)

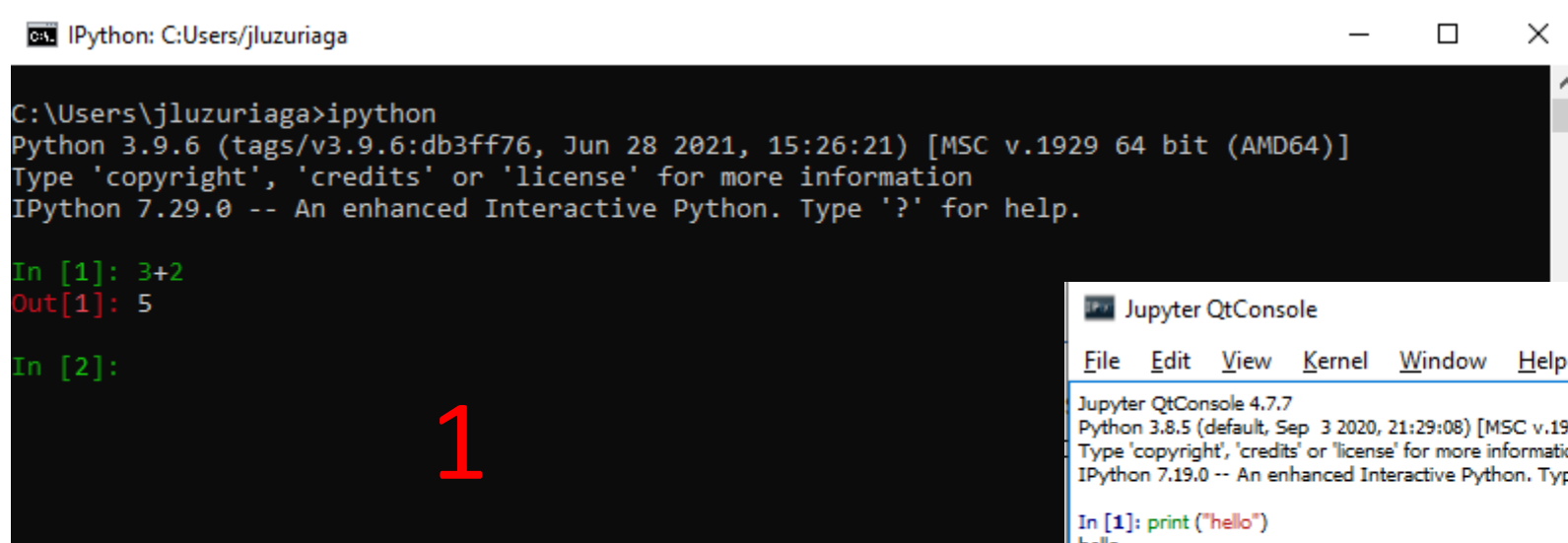
Nos permite probar ideas rápidamente en lugar de pasar por la creación de archivos y su ejecución.

## Características:

- Historial de entrada disponible entre sesiones (reutilizar comandos que se escribieron anteriormente)
- Autocompletado de comandos y variables (uso del tabulador)
- Resaltado de sintaxis

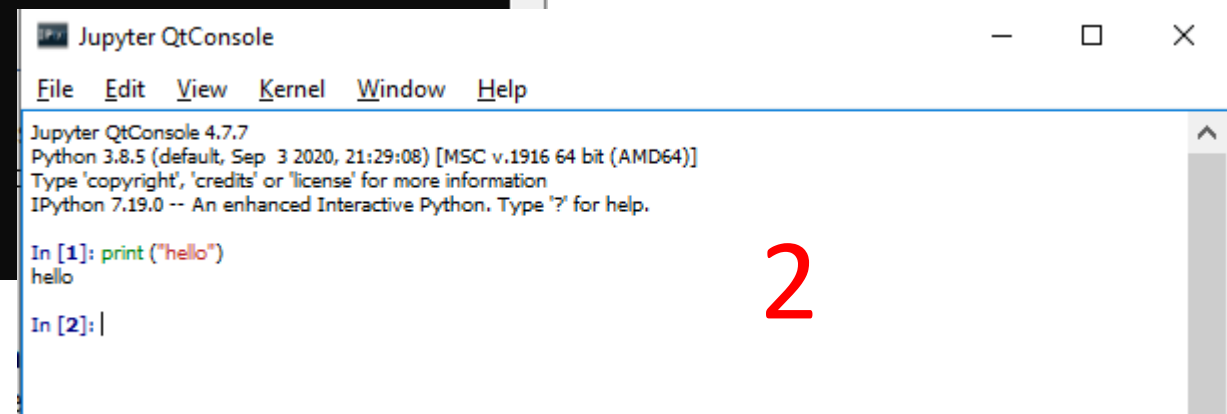
# ipython (*cont.*)

1. pip install ipython
2. conda update ipython



```
IPython: C:\Users\jluzuriaga  
C:\Users\jluzuriaga>ipython  
Python 3.9.6 (tags/v3.9.6:db3ff76, Jun 28 2021, 15:26:21) [MSC v.1929 64 bit (AMD64)]  
Type 'copyright', 'credits' or 'license' for more information  
IPython 7.29.0 -- An enhanced Interactive Python. Type '?' for help.  
  
In [1]: 3+2  
Out[1]: 5  
  
In [2]:
```

1



```
Jupyter QtConsole  
File Edit View Kernel Window Help  
Jupyter QtConsole 4.7.7  
Python 3.8.5 (default, Sep 3 2020, 21:29:08) [MSC v.1916 64 bit (AMD64)]  
Type 'copyright', 'credits' or 'license' for more information  
IPython 7.19.0 -- An enhanced Interactive Python. Type '?' for help.  
  
In [1]: print("hello")  
hello  
  
In [2]:
```

2

# Aislamiento del entorno para los proyectos

- Dos niveles diferentes de aislamiento de entornos
  - aplicación
  - sistema
- Aislamiento **a nivel de aplicación** a través de entornos virtuales
  - El enfoque más fácil y ligero
  - Creación de entornos virtuales (utilizando el módulo *venv*)
  - Se centra en aislar el intérprete de Python y los paquetes disponibles en él
  - No son portables (se basan en rutas absolutas del sistema)
  - Ideal para escribir código independientes del sistema operativo o proyectos de baja complejidad que no tengan demasiadas dependencias externas

# Aislamiento del entorno para los proyectos

- **Aislamiento a nivel de sistema**

- Si escribe el software en un ordenador diferente al que lo va a ejecutar
- No todos los paquetes se comportan igual en todos los sistemas operativos
- Para asegurar la suficiente consistencia y reproducibilidad
  - paquetes que dependen de bibliotecas compartidas de terceros
  - uso intensivo de extensiones compiladas de Python
- Herramientas clásicas de virtualización de sistemas operativos
  - VMware, Parallels y VirtualBox
- Herramientas sistemas de contenedores
  - Docker y Rocket

# Aislamiento del entorno a nivel de aplicación

- Python tiene soporte incorporado para crear entornos virtuales
- Invocando al módulo **venv** directamente desde el shell del sistema

```
C:\> python -m venv <nombre-del-entorno>
```

Crearé un nuevo directorio con el nombre del entorno en la ruta del directorio de trabajo actual, y dentro unos cuantos subdirectorios

**bin/, lib/ e include/**

Una vez creado, es necesario activarlo usando el script:

```
C:\> [nombre-del-entorno]\Scripts\activate.bat
```

Para que el usuario sepa que ha activado el entorno virtual el prompt cambiará añadiendo la cadena (nombre-del-entorno) al principio

# Aislamiento del entorno a nivel de aplicación

- Un ejemplo de sesión:

```
>python -m venv ejemplo
```

```
C:\> <venv>\Scripts\activate.bat      (cmd.exe)  
PS C:\> <venv>\Scripts\Activate.ps1   (PowerShell)  
$source ejemplo/bin/activate          (bash)
```

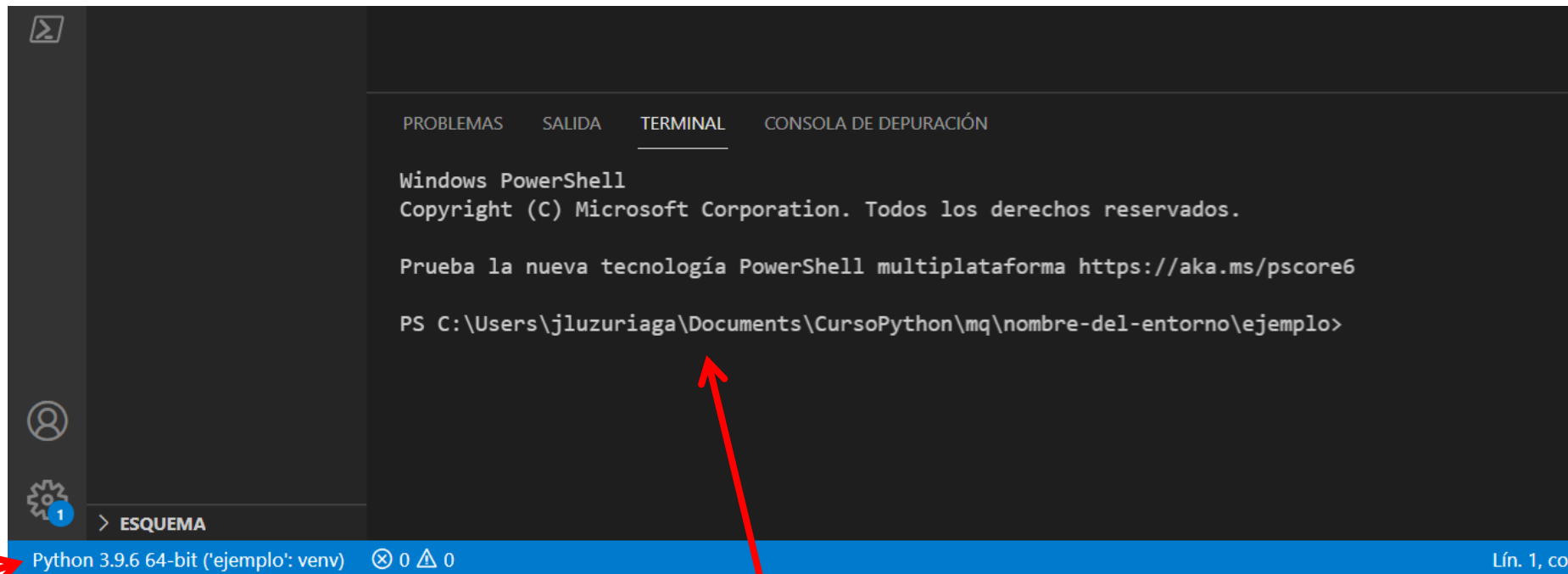
```
(ejemplo) C:\>whereis python  
C:\Users\jluzuriaga\Documents\CursoPython\ejemplo\Scripts\python.exe
```

```
(ejemplo) C:\>deactivate  
> whereis python  
C:\Users\jluzuriaga\AppData\Local\Programs\Python\Python39\python.exe
```



# Configurar el intérprete de VS Code para que funcione con el nuevo entorno virtual

>code .



1

2

# El ecosistema de paquetes de Python

- El núcleo del ecosistema de empaquetado de Python es el Python Packaging Index (PyPI).
- PyPI es el repositorio público de proyectos Python de uso gratuito que alberga casi **tres millones distribuciones de más de 330 mil proyectos**.
- Las aplicaciones modernas se construyen utilizando múltiples paquetes de PyPI que
  - Tienen sus propias dependencias
    - Dependencias también pueden tener sus propias dependencias
    - Cadenas de dependencias pueden ser interminables
  - Paquetes requieren versiones específicas de otros paquetes
- Conocer herramientas para trabajar con los paquetes disponibles en PyPI

# Instalación de paquetes Python con pip

- La Autoridad de Empaquetado de Python (PyPA) recomienda **pip para instalar paquetes**.

pip: una herramienta de línea de comandos  
permite instalar paquetes directamente desde PyPI  
`$ pip install <nombre-del-paquete>`

- Ejem: paquete llamado django `$ pip install django`

`pip install <nombre-del-paquete>==<versión>`

`pip install --upgrade <nombre-del-paquete>`

`pip --help`

# Instalación de paquetes Python con pip (*cont*)

- Con pip install

los paquetes se instalarán en el directorio site-packages del usuario o en el global

dependiendo de varias condiciones:

1. user site-packages: si se especifica el parámetro --user
2. global site-packages: si el directorio global site-packages tiene permisos de escritura para el usuario que invoca pip
3. user site-packages: en caso contrario

# Mejores prácticas de los usuarios de pip

- Almacenar la definición de todas las dependencias del proyecto en un solo lugar (creando un archivo de requerimientos <requirements.txt>)
- Con el archivo de requerimientos se pueden instalar fácilmente todas las dependencias en un solo paso
- El comando pip install entiende el formato del archivo, se especifica la ruta al archivo y se utiliza la bandera -r

```
$ pip install -r requirements.txt
```

# Mejores prácticas de los usuarios de pip

- Ejemplo de archivo de requerimientos <requirements.txt>:

# los especificadores de la versión son los mejores para la reproducibilidad

eventlet==0.17.4

graceful==0.1.1

# para proyectos que están bien probados con diferentes

# versiones de dependencias con rangos de versiones aceptables

falcon>=0.3.0,<0.5.0

# los paquetes sin versiones deben evitarse a menos que

# la última versión sea siempre necesaria/deseada

pytz