



python

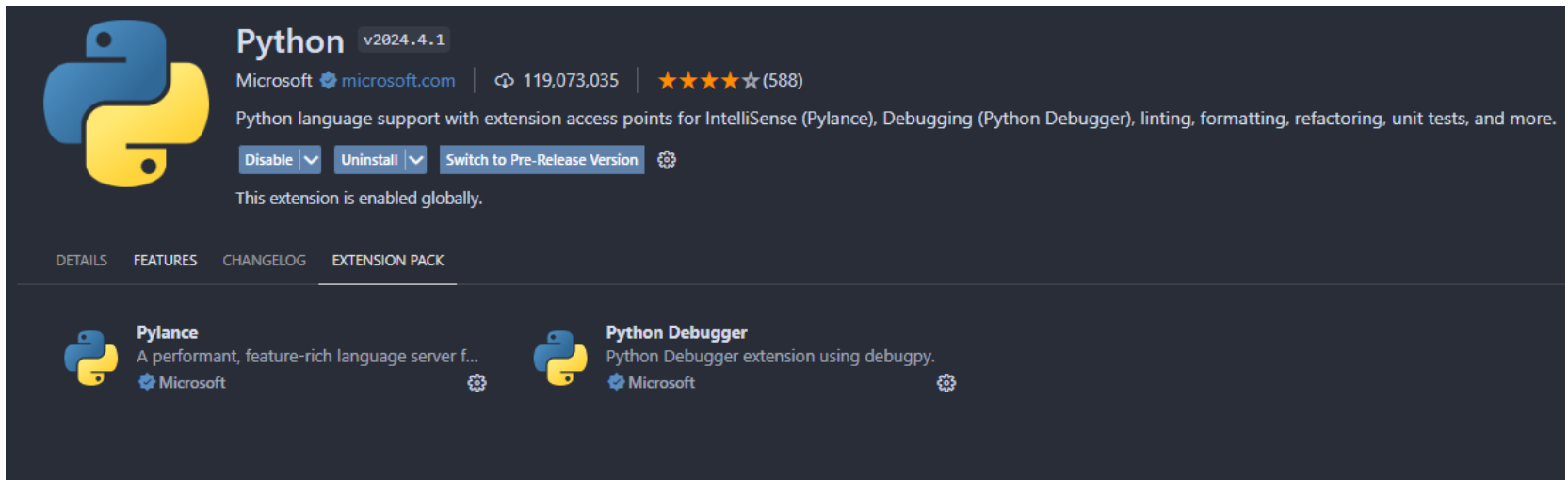
# https://www.python.org/

The image shows a screenshot of the Python.org website and a Windows PowerShell terminal window. The website has a dark blue header with the Python logo and the word "python" in white. Navigation links include "Python", "PSF", "Docs", "PyPI", and "Jobs". A "Donate" button and a search bar are also visible. Below the header, there are links for "About", "Downloads", "Documentation", "Community", "Success Stories", and "News". The "Downloads" link is highlighted, showing a dropdown menu with options: "All releases", "Source code", and "Windows". The "Download for Windows" section is also visible, showing a button for "Python 3.12.3".



The Windows PowerShell terminal window is open, showing the command prompt. The user has entered the command `python`, and the output is:

```
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr 9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> |
```


# En VS code



The screenshot shows the Python extension page in the Visual Studio Code marketplace. At the top left is the Python logo. To its right, the extension name "Python" is displayed with the version "v2024.4.1" in a small box. Below the name, it says "Microsoft" with a Microsoft logo and "microsoft.com". To the right of this, there is a download icon, the number "119,073,035", and a star rating of four stars with "(588)" reviews. A description follows: "Python language support with extension access points for IntelliSense (Pylance), Debugging (Python Debugger), linting, formatting, refactoring, unit tests, and more." Below the description are three buttons: "Disable" with a dropdown arrow, "Uninstall" with a dropdown arrow, and "Switch to Pre-Release Version" with a gear icon. Below these buttons, it says "This extension is enabled globally." At the bottom, there are four tabs: "DETAILS", "FEATURES", "CHANGELOG", and "EXTENSION PACK", with "EXTENSION PACK" being the active tab. Under the "EXTENSION PACK" tab, there are two items: "Pylance" and "Python Debugger". Each item has a small Python logo, a title, a description, and a Microsoft logo with a gear icon.




**Python** v2024.4.1  
Microsoft  microsoft.com |  119,073,035 | ★★★★★ (588)




Python language support with extension access points for IntelliSense (Pylance), Debugging (Python Debugger), linting, formatting, refactoring, unit tests, and more.

[Disable](#) [Uninstall](#) [Switch to Pre-Release Version](#) 

This extension is enabled globally.

DETAILS FEATURES CHANGELOG **EXTENSION PACK**

 **Pylance**  
A performant, feature-rich language server f...  
 Microsoft 

 **Python Debugger**  
Python Debugger extension using debugpy.  
 Microsoft 

# Python

- Python es uno de los lenguajes de programación más populares y con un crecimiento más rápido del mundo. Se usa para todo tipo de tareas, como las de programación web y análisis de datos, y se ha convertido en el lenguaje que hay que conocer para el aprendizaje automático.
- A los investigadores, matemáticos y, en particular, a los científicos de datos les gusta Python gracias a su sintaxis completa y fácil de entender y a la amplia gama de paquetes de código abierto disponibles. Los paquetes son bibliotecas de código compartido que están disponibles de forma gratuita para todos los usuarios.

# Python

- Python es un lenguaje interpretado, lo que reduce el ciclo editar-probar-depurar porque no se requiere ningún paso de compilación. A fin de ejecutar aplicaciones de Python, necesita un entorno o un intérprete de runtime para ejecutar el código.



# Implementaciones de Python

- CPython, la implementación de referencia: La más popular. Se usa habitualmente para desarrollo web, desarrollo de aplicaciones y creación de scripts.
- Anaconda: Es una distribución especializada de Python adaptada para tareas de programación científicas, como la ciencia de datos y el aprendizaje automático.
- IronPython: Es una implementación de código abierto de Python compilada en el runtime de .NET.

# Python

- Python es compatible con una experiencia de consola interactiva que permite escribir en comandos y ver los resultados inmediatamente. Esto se conoce a veces como "read-eval-print loop" o REPL.
- Para usar REPL, escriba python en la consola. Recibirá un mensaje similar a la salida siguiente que, después, espera a que escriba comandos:

```
▶ python
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr 9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hola Mundo!")
Hola Mundo!
>>> 1+2
3
>>> |
```

# Python {Variables}

- En Python, se declara una variable y se le asigna un valor mediante el operador de asignación `=`. La variable está en el lado izquierdo del operador y el valor asignado (que puede ser una expresión como `2 + 2` e incluso puede incluir otras variables) se encuentra en el lado derecho.

```
x = 1          # assign variable x the value 1
y = x + 5      # assign variable y the value of x plus 5
z = y          # assign variable z the value of y
```

```
x = True
print(type(x)) # outputs: <class 'bool'>
```

```
x = 'This is a string'
print(x) # outputs: This is a string
print(type(x)) # outputs: <class 'str'>
y = "This is also a string"
```

```
x = 'Hello' + ' ' + 'World!'
print(x) # outputs: Hello World!
```



# Python {Entrada de usuario}

- Para leer la entrada hecha desde el teclado, Python proporciona la función `input()`. `input()` lee lo que el usuario escribe en el teclado y lo devuelve como una cadena.

```
name = input('Enter your name:')  
print(name)
```

```
print('What is your name?')  
name = input()  
print(name)
```

```
x = input('Enter a number: ')  
print(type(x))
```

```
x = 5  
print('The number is ' + str(x))
```

# Leer archivos en Python

- Con la función `open()` y como argumento el nombre del fichero que queremos abrir.

```
fichero = open('ejemplo.txt')
```

- y podemos imprimir su contenido con `read()`.

```
print(fichero.read())
```

- Es posible también leer un número de líneas determinado y no todo el fichero de golpe. Para ello hacemos uso de la función `readline()`.

# Leer archivos en Python

```
fichero = open('ejemplo.txt')
caracter = fichero.readline(1)
while caracter != "":
    #print(caracter)
    caracter = fichero.readline(1)
```

```
fichero = open('ejemplo.txt')
lineas = fichero.readlines()
for linea in lineas:
    print(linea)
```

## Argumentos de open()

- 'r': Por defecto, para leer el fichero.
- 'w': Para escribir en el fichero.
- 'x': Para la creación, fallando si ya existe.
- 'a': Para añadir contenido a un fichero existente.
- 'b': Para abrir en modo binario.

```
fichero = open('ejemplo.txt', 'r')
# Usar la variable fichero
# Cerrar el fichero
fichero.close()
```

# Escribir archivos en Python

- Con el método `write()` podemos añadir contenido.

```
fichero = open("datos_guardados.txt", 'w')  
fichero.write("Contenido a escribir")  
fichero.close()
```

- También podemos usar el método `writelines()` y pasarle una lista. Dicho método se encargará de guardar todos los elementos de la lista en el fichero.

```
fichero = open("datos_guardados.txt", 'w')  
lista = ["Manzana", "Pera", "Plátano"]  
  
fichero.writelines(lista)  
fichero.close()
```

```
fichero = open("datos_guardados.txt", 'w')  
lista = ["Manzana\n", "Pera\n", "Plátano\n"]  
  
fichero.writelines(lista)  
fichero.close()
```

# Escribir archivos en Python

```
# Escribe un mensaje en un fichero
def escribe_fichero(mensaje):
    with open('fichero_comunicacion.txt', 'w') as fichero:
        fichero.write(mensaje)

# Leer el mensaje del fichero
def lee_fichero():
    mensaje = ""
    with open('fichero_comunicacion.txt', 'r') as fichero:
        mensaje = fichero.read()
    # Borra el contenido del fichero para dejarlo vacío
    f = open('fichero_comunicacion.txt', 'w')
    f.close()
    return mensaje

escribe_fichero("Esto es un mensaje")
print(lee_fichero())
```

# Módulos en Python

- Un módulo en Python es un fichero .py que alberga un conjunto de funciones, variables o clases y que puede ser usado por otros módulos. Nos permiten reutilizar código y organizarlo mejor en namespaces.

```
# mimodulo.py
def suma(a, b):
    return a + b

def resta(a, b):
    return a - b
```

```
# otromodulo.py
import mimodulo

print(mimodulo.suma(4, 3)) # 7
print(mimodulo.resta(10, 9)) # 1
```

```
from mimodulo import suma, resta

print(suma(4, 3)) # 7
print(resta(10, 9)) # 1
```

```
from mimodulo import *

print(suma(4, 3)) # 7
print(resta(10, 9)) # 1
```

# Módulos en Python

- Rutas y Uso de sys.path. Normalmente los módulos que importamos están en la misma carpeta, pero es posible acceder también a módulos ubicados en una subcarpeta.

```
.  
├── ejemplo.py  
├── carpeta  
│   └── modulo.py
```

```
# modulo.py  
def hola():  
    print("Hola")
```

```
from carpeta.modulo import *  
print(hola())  
# Hola
```

```
# moduloconnombrrelargo.py  
hola = "hola"
```

```
import moduloconnombrrelargo  
print(moduloconnombrrelargo.hola)
```

```
import moduloconnombrrelargo as m  
print(m.hola)
```

# Módulos en Python

- **Función Main**

```
# modulo.py

def suma(a, b):
    return a + b

c = suma(1, 2)
print("La suma es:", c)
```

```
# otromodulo.py
import modulo

# Salida: La suma es: 3
```

```
# modulo.py
def suma(a, b):
    return a + b

if (__name__ == '__main__'):
    c = suma(1, 2)
    print("La suma es:", c)
```

```
import mimodulo
import importlib
importlib.reload(mimodulo)
importlib.reload(mimodulo)
```



# Python Package Installer (PIP)

- Es una herramienta que permite instalar, desinstalar y actualizar paquetes Python.
- PIP es una forma simple de instalar una librería, solo requiere el uso de una ventana “símbolo de sistema” y la conexión a internet. En la ventana se escribe «pip install libreria».

# Python Package Installer (PIP)

```
[kralos]
[~]
▶ pip list
Package      Version
-----
colorama     0.4.6
distlib      0.3.8
filelock     3.13.1
iniconfig    2.0.0
packaging    23.2
pip          24.0
platformdirs 4.1.0
pluggy       1.3.0
pytest       7.4.4
virtualenv   20.25.0
[kralos]
[~]
▶ |
```

# Cómo crear un entorno virtual con virtualenv en Windows

- Virtualenv es una herramienta para crear entornos Python aislados.

`virtualenv --versión`

- Crear el entorno virtual

`virtualenv NombreMiEntorno`

```
[kralos]
[D:\code\Python\entorno]
> dir
[kralos]
[D:\code\Python\entorno]
> python -m virtualenv --version
virtualenv 20.25.0 from C:\Python312\Lib\site-packages\virtualenv\__init__.py
[kralos]
[D:\code\Python\entorno]
> python -m virtualenv venv
created virtual environment CPython3.12.3.final.0-64 in 11401ms
creator CPython3Windows(dest=D:\code\Python\entorno\venv, clear=False, no_vcs_ignore=False, global=False)
seeder FromAppData(download=False, pip=bundle, via=copy, app_data_dir=C:\Users\kralos\AppData\Local\pypa\virtualenv)
added seed packages: pip==23.3.2
activators BashActivator,BatchActivator,FishActivator,NushellActivator,PowerShellActivator,PythonActivator
[kralos]
[D:\code\Python\entorno]
> dir

Directory: D:\code\Python\entorno

Mode                LastWriteTime         Length Name
----                -
d-----          4/23/2024 12:00 PM                venv
```

# Cómo crear un entorno virtual con virtualenv en Windows

- Activar el entorno virtual

```
[kralos]  
[D:\code\Python\entorno]  
▶ .\venv\Scripts\activate  
(venv) [kralos]  
[D:\code\Python\entorno]  
▶ |
```

- Revisar las bibliotecas instaladas en un entorno virtual

```
(venv) [kralos]  
[D:\code\Python\entorno]  
▶ pip freeze  
(venv) [kralos]  
[D:\code\Python\entorno]  
▶ |
```

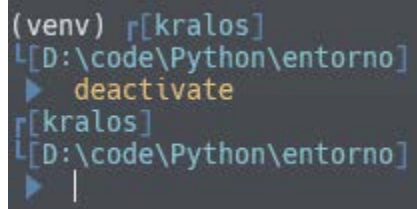
# Cómo crear un entorno virtual con virtualenv en Windows

- Migrar un entorno virtual a otro:

```
pip freeze > requirements.txt
```


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

- Desactivar un entorno



```
(venv) [kralos]  
[D:\code\Python\entorno]  
▶ deactivate  
[kralos]  
[D:\code\Python\entorno]  
▶ |
```

# Referencias

- *Prefacio — Python para Ingenieros.* (n.d.). Github.io. Retrieved April 16, 2024, from <https://jorgedelossantos.github.io/apuntes-python/intro.html>
-  *inicio.* (n.d.). El Libro De Python. Retrieved April 22, 2024, from <https://ellibrodepython.com/>
- Moisset, D. (n.d.). *SQLite : Base de datos desde Python.* Tutorialesprogramacionya.com. Retrieved April 23, 2024, from <https://www.tutorialesprogramacionya.com/pythonya/detalleconcepto.php?punto=87&codigo=88&inicio=75>