



▼ PyPI (Python Package Index)

Python Package Index es el repositorio oficial de aplicaciones, módulos y paquetes de Python creados por terceros. Pensemos en él como en una gran librería donde podemos encontrar recursos y proyectos que podemos emplear libremente en nuestros propios programas.

Dale un vistazo a la página principal de PyPI [PyPI Python Package Index](https://pypi.org)

The screenshot shows the main page of the Python Package Index (PyPI). At the top, there's a navigation bar with icons for back, forward, refresh, and search, followed by the URL 'pypi.org'. On the right side of the header are links for 'Menú' and a user profile icon. The main title 'Encuentre, instale y publique paquetes de Python con el Índice de paquetes de Python' is displayed prominently in large white text against a blue background. Below the title is a search bar with the placeholder 'Buscar proyectos' and a magnifying glass icon. Underneath the search bar, there's a link 'O bien, explore los proyectos'. At the bottom of the main content area, there are four statistics: '436.243 proyectos', '4.226.996 versiones', '7.683.952 archivos', and '675.799 usuarios'. In the bottom left corner, the PyPI logo is visible. To the right of the logo, there's a block of text explaining what PyPI is and how it can be used, along with links for learning more about installation and packaging.

Encuentre, instale y publique paquetes de Python con el Índice de paquetes de Python

Buscar proyectos

O bien, explore los proyectos

436.243 proyectos 4.226.996 versiones 7.683.952 archivos 675.799 usuarios

python™
Package Index

El Índice de paquetes de Python (PyPI) es un repositorio de software para el lenguaje de programación Python.

PyPI le ayuda a encontrar e instalar programas desarrollados y compartidos por la comunidad de Python. [Aprenda a instalar paquetes](#).

Los autores de paquetes utilizan PyPI para distribuir sus programas. [Aprenda a empaquetar su código en Python para PyPI](#).

Puedes encontrar el listado de los paquetes más descargados de PyPI [Aqui](#)

Ahora, entra nuevamente a [PyPI Python Package Index](#) y busca la librería **contexto** Esta es una librería para el procesamiento y análisis de textos desarrollada por el Departamento Nacional de Planeación. Explora su documentación e identifica su utilidad.

The screenshot shows the PyPI project page for 'ConTexto 0.2.0'. At the top, there's a navigation bar with icons for back, forward, search, and user profile. The URL 'pypi.org/project/ConTexto/' is visible. On the right, there are links for 'Versión más reciente' (Latest version) and a green checkmark icon. Below the header, the title 'ConTexto 0.2.0' is displayed in large white text on a blue background. To its left is a button with the command 'pip install ConTexto'. To the right is a link to 'Publicación: 13 jul 2021'. A sub-header 'Librería para el procesamiento y análisis de texto con Python' follows. The main content area is divided into two columns: 'Navegación' on the left and 'Descripción de proyecto' on the right. The 'Descripción de proyecto' column contains the project summary: 'ConTexto - Librería de procesamiento y análisis de textos', the logo of the Colombian government (Escudo Nacional), the slogan 'El futuro es de todos', and the text 'DNP Departamento Nacional de Planeación'. It also features a large 'ConTexto' logo with the subtitle 'Librería de procesamiento y análisis de textos'. Below this are badges for 'pypi package 0.2.0', 'python 3.6 | 3.7 | 3.8 | 3.9', 'license MIT', 'downloads 14k', 'Fork 12', and 'Issues 1'. The 'Navegación' column includes links for 'Descripción de proyecto' (which is highlighted in blue), 'Histórico de versiones', and 'Archivos de descarga'. It also lists 'Enlaces del proyecto' with links to 'Homepage', 'Documentación', and 'Seguimiento de fallas'. The 'Estadísticas' section shows GitHub metrics: 'Estrellas: 43', 'Bifurcaciones: 12', 'Open issues: 9', and a 'Issues 1' badge. The 'Descripción' section provides a detailed overview of the library's purpose and functionality.

Navegación

- Descripción de proyecto
- Histórico de versiones
- Archivos de descarga

Enlaces del proyecto

- Homepage
- Documentación
- Seguimiento de fallas

Estadísticas

Estadísticas de GitHub:

- Estrellas: 43
- Bifurcaciones: 12
- Open issues: 9
- Issues 1

Descripción de proyecto

ConTexto - Librería de procesamiento y análisis de textos

El futuro es de todos DNP Departamento Nacional de Planeación

ConTexto

Librería de procesamiento y análisis de textos

pypi package 0.2.0 python 3.6 | 3.7 | 3.8 | 3.9 license MIT downloads 14k Fork 12 Issues 1

Descripción

La librería de procesamiento y análisis de texto, ConTexto, tiene como objetivo principal proporcionar herramientas que simplifiquen las tareas y proyectos que involucren procesamiento y análisis de texto. La librería fue desarrollada en el lenguaje de programación de *Python* y contiene un conjunto de funciones que permiten realizar transformaciones y análisis de textos de forma simple, utilizando diferentes técnicas para lectura y escritura de archivos de texto, incluyendo reconocimiento óptico de caracteres (OCR), limpieza de textos y remoción de palabras no deseadas.

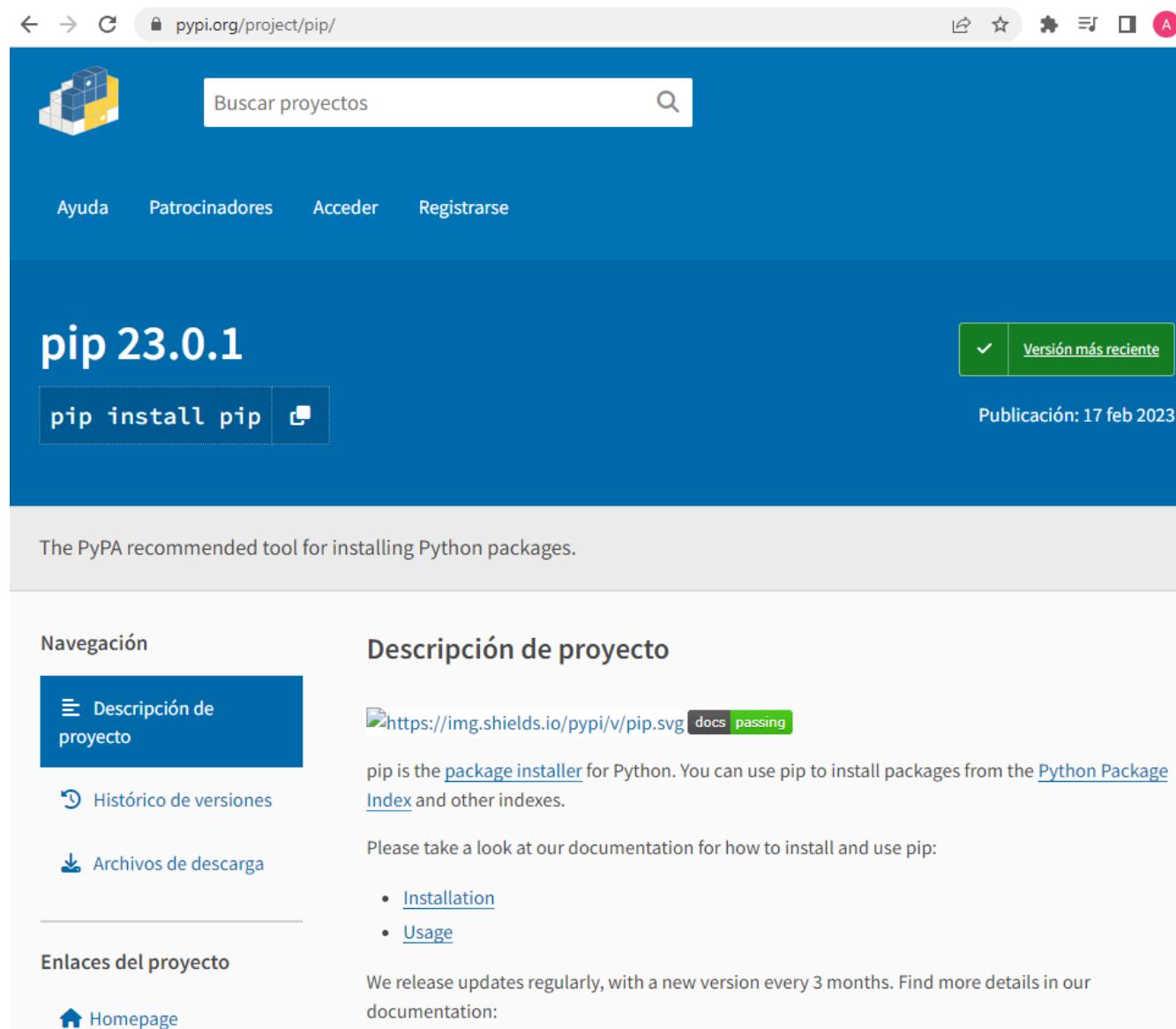
Como puedes ver, el repositorio PyPI es continuamente alimentado por la comunidad de desarrolladores Python al rededor del mundo. Esto hace que el lenguaje de programación Python sea cada vez más robusto y popular.

Pronto no solo utilizarás librerías desarrolladas por terceros, sino que podrás publicar tus propias librerías para el uso por parte de la comunidad.

PIP (PIP/Python Install Packages)

Para poder instalar algún paquete o librería de terceros debemos emplear una herramienta para la gestión de paquetes. En Python, la más popular herramienta es **PIP**

PIP en sí misma forma parte del repositorio PyPI. Con una búsqueda de PIP en el repositorio encontraremos esta herramienta a nuestra disposición.



The screenshot shows the PyPI project page for the package 'pip'. The title 'pip 23.0.1' is displayed prominently at the top, with a green button next to it labeled 'Versión más reciente' (Latest version). Below the title, there's a search bar with the placeholder 'Buscar proyectos' and a magnifying glass icon. A navigation bar at the top includes links for 'Ayuda', 'Patrocinadores', 'Acceder', and 'Registrarse'. On the left, a sidebar titled 'Navegación' contains links for 'Descripción de proyecto' (which is highlighted in blue), 'Histórico de versiones', and 'Archivos de descarga'. Below this is a section titled 'Enlaces del proyecto' with a 'Homepage' link. The main content area on the right is titled 'Descripción de proyecto' and contains the package's description, installation instructions, and update information. It also features a 'docs passing' badge from img.shields.io.

The PyPA recommended tool for installing Python packages.

Navegación

- Descripción de proyecto
- Histórico de versiones
- Archivos de descarga

Enlaces del proyecto

- Homepage

Descripción de proyecto

 <https://img.shields.io/pypi/v/pip.svg> docs passing

pip is the [package installer](#) for Python. You can use pip to install packages from the [Python Package Index](#) and other indexes.

Please take a look at our documentation for how to install and use pip:

- [Installation](#)
- [Usage](#)

We release updates regularly, with a new version every 3 months. Find more details in our documentation:

PIP al ser considerada la herramienta estándar de Python para la gestión de paquetes (instalación, eliminación, actualización....) viene instalada por defecto en las más recientes versiones de Python.

Para saber si PIP se encuentra instalado podemos escribir el siguiente comando en cualquier intérprete de Python:

```
pip --version
```

```
pip --version
```

Si queremos saber un poco más sobre PIP y sus comandos disponibles podemos preguntárselo a la misma herramienta mediante el comando

```
pip help
```

Commands:	
install	Install packages.
download	Download packages.
uninstall	Uninstall packages.
freeze	Output installed packages in requirements format.
list	List installed packages.
show	Show information about installed packages.
check	Verify installed packages have compatible dependencies.
config	Manage local and global configuration.
search	Search PyPI for packages.
cache	Inspect and manage pip's wheel cache.
index	Inspect information available from package indexes.
wheel	Build wheels from your requirements.
hash	Compute hashes of package archives.
completion	A helper command used for command completion.
debug	Show information useful for debugging.
help	Show help for commands.

```
pip help
```

Ahora que ya conocemos los comandos de **PIP** para la gestión de paquetes vamos a emplear algunos de ellos...

```
pip list
```

Nos muestra el listado de paquetes instalados en el entorno de ejecución actual. Intentalo en este cuaderno de colab y también en tu entorno local, podrás ver que hay diferencia en los paquetes instalados en ambos entornos

```
pip list
```

Si solo deseas verificar si un paquete en particular está instalado, puedes usar el siguiente comando:

```
pip show pandas
```

También podemos comprobar las dependencias entre paquetes que requieren ser atendidas con el comando **check** de pip

```
pip check
```

```
pip check
```

Ahora haremos un proceso de instalación, uso y desinstalación de una librería. Para nuestro ejemplo, usaremos una sencilla librería llamada **emoji** que nos permite incorporar emoticones en nuestros programas Python.



Buscar proyectos



Ayuda Patrocinadores Acceder Registrarse

emoji 2.2.0

`pip install emoji`

Versión más reciente

Publicación: 31 oct 2022

Emoji for Python

Navegación

Descripción de proyecto

Histórico de versiones

Archivos de descarga

Enlaces del proyecto

Homepage

Estadísticas

Descripción de proyecto

Emoji for Python. This project was inspired by [kyokomi](#).

Example

The entire set of Emoji codes as defined by the [Unicode consortium](#) is supported in addition to a bunch of [aliases](#). By default, only the official list is enabled but doing `emoji.emojize(language='alias')` enables both the full list and aliases.

```
>>> import emoji  
>>> print(emoji.emojize('Python is :thumbs_up:'))  
Python is   
>>> print(emoji.emojize('Python is :thumbsup:', language='alias'))  
Python is   
>>> print(emoji.demojize('Python is '))
```

Con el comando

`pip show emoji`

podemos comprobar si contamos con esta librería instalada

`pip show emoji`

Procederemos a instalarla con el comando pip apropiado según lo indica la página oficial

`pip install emoji`

`pip install emoji`

Una vez instalada podemos repetir el comando

```
pip show emoji
```

No solamente para comprobar que ya está instalada, sino también para obtener mayor información sobre este paquete

```
pip show emoji
```

```
pip list
```

Ahora ya podemos usar **emoji** en nuestras aplicaciones

```
import emoji  
#Mostrar todos los emojis disponibles  
all_emojis = emoji.EMOJI_DATA.keys()  
for em in all_emojis:  
    print(em)
```

```
import emoji  
  
print(emoji.emojize(':alien:'))  
print(emoji.emojize(':red_heart:'))  
print(emoji.emojize(':fire:'))  
print(emoji.emojize(':rocket:'))  
print(emoji.emojize(':sunflower:'))  
print(emoji.emojize(':cat:'))  
print(emoji.emojize(':casa:',language='es'))  
print(emoji.emojize(':estrella:', language='es'))
```

```
import emoji  
nota = float(input(emoji.emojize("Cuál fué tu nota :hear-no-evil_monkey: ?")))  
if nota >= 3:  
    print(emoji.emojize("Felicitaciones, aprobaste la materia :thumbs_up:"))  
else:  
    print(emoji.emojize("Lo siento, perdiste la materia :loudly_crying_face:"))  
  
print(emoji.emojize("python es :serpiente:", language ='es'))
```

Otra importante opción de PIP es el poder generar un archivo con todos los paquetes instalados en el entorno de ejecución actual. Para ello, utilizamos el comando

```
pip freeze
```

```
pip freeze > requerimientos.txt
```

Este archivo de requerimientos es de gran utilidad, pues al momento de ejecutar una aplicación en otro ambiente de ejecución, podemos conocer los paquetes utilizados para el desarrollo del programa y de esta manera recrear el ambiente para su correcto funcionamiento. PIP también nos permite instalar todos los módulos y paquetes requeridos a partir de un archivo de requerimientos con el siguiente comando:

```
pip install -r requerimientos.txt
```

Otra variación del comando `install` es la de permitirnos actualizar un paquete a través de la opción `--upgrade`

```
pip install --upgrade paquete
```

Hagamos la prueba con nuestro paquete de prueba emoji

```
pip install --upgrade emoji
```

Para finalizar nuestro recorrido por los comandos PIP vamos a desinstalar el paquete emoji instalado previamente. Para ello utilizamos el comando **uninstall**

```
pip uninstall paquete
```

```
pip uninstall emoji
```

```
pip show emoji
```

▼ Reto

Vamos a realizar un ejercicio donde carguemos datos desde un archivo CSV para realizar unas operaciones básicas y mostrar resultados.

Creemos una carpeta para este ejemplo con el nombre `ejemplo_panda` y allí guardemos un archivo CSV llamado `datos.csv` con la siguiente estructura:

```
nombre,edad,ciudad
Juan,25,Medellín
María,30,Bogotá
Luis,28,Cali
Ana,35,Barranquilla
```

Creemos otro archivo (`procesar.py`) en donde vamos a calcular el promedio de las edades y mostrar la información:

```
import pandas as pd

#leer el DataFrame
df = pd.read_csv('ruta/datos.csv')
print(df.head())

promedio=df['edad'].mean()
print(promedio)
```

Este código carga los datos del archivo CSV en un DataFrame utilizando `pd.read_csv('datos.csv')`, completa el ejercicio mostrando los registros, calcula el promedio de las edades con `df['edad'].mean()` y filtra las personas mayores de 30 años `{df[df['edad']>30]}`. Finalmente, muestra los resultados en la consola.

Apropiación

1. Utilizando el archivo ventas.csv lea el dataframe en un archivo llamado gestion_ventas.py y realice lo siguiente:

- Muestra las primeras 10 filas del data Frame
- Muestra el total de las ventas por producto

```
(ventas_por_producto = df.groupby('nombre_producto')['cantidad_vendida'].sum().reset_index()
```

- instala las librerías matplotlib y seaborn e investiga para que sirven.
- Elabora un gráfico de barras de las ventas totales por producto utilizando estas librerías

Configurar gráfico

```
plt.figure(figsize=(10,6)) sns.barplot(x='cantidad_vendida', y='nombre_producto', data=ventas_por_producto)  
plt.title('Ventas totales por producto') plt.xlabel('Cantidad vendida') plt.ylabel('Producto')
```

Mostrar gráfico

```
plt.show()
```