



✓ 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)

A screenshot of the PyPI (Python Package Index) homepage. The browser address bar shows 'pypi.org'. The page has a blue header with the PyPI logo and a 'Menú' dropdown. The main content area has a large blue background with the text 'Encuentre, instale y publique paquetes de Python con el Índice de paquetes de Python'. Below this is a search bar with the placeholder text 'Buscar proyectos' and a magnifying glass icon. Under the search bar, it says 'O bien, [explore los proyectos](#)'. A light gray bar contains statistics: '436.243 proyectos', '4.226.996 versiones', '7.683.952 archivos', and '675.799 usuarios'. The footer features the 'python Package Index' logo and text explaining that PyPI is a repository of software for the Python language, helping users find and install programs. It includes links to 'Aprenda a instalar paquetes' and 'Aprenda a empaquetar su código en Python para PyPI'.

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

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. The page has a blue header with the project name and version. Below the header, there's a navigation sidebar on the left and a main content area on the right. The sidebar includes links for project description, version history, download archives, project homepage, documentation, and issue tracking. The main content area features a project description, a banner image, and a list of statistics.

ConTexto 0.2.0 ✓ [Versión más reciente](#)

`pip install ConTexto`

Publicación: 13 jul 2021

Librería para el procesamiento y análisis de texto con Python

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

Descripción de proyecto

ConTexto - Librería de procesamiento y análisis de textos

The banner image features the text "El futuro es de todos" and "DNP Departamento Nacional de Planeación" above the project name "ConTexto" and its description "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

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.

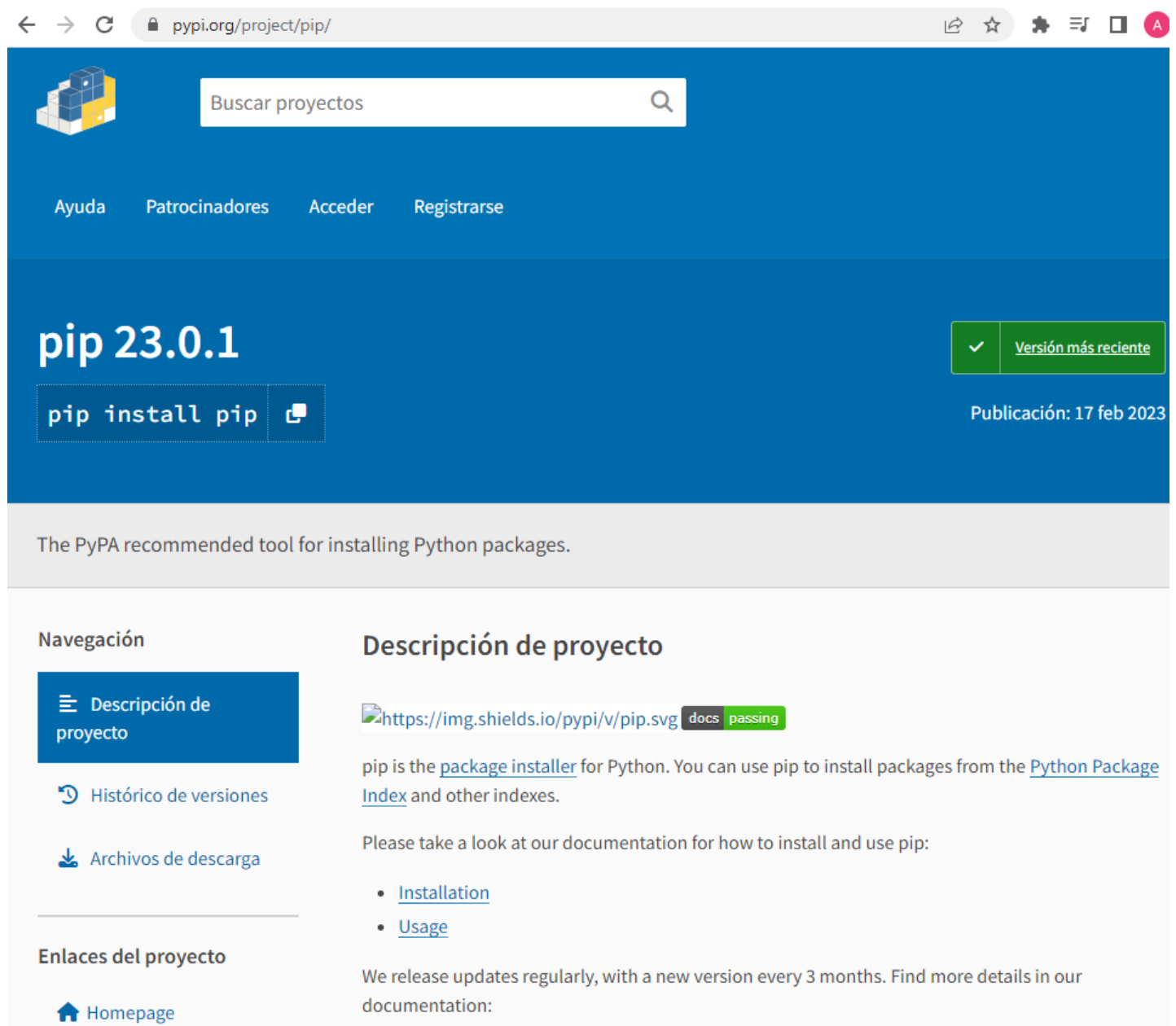
Como puedes ver, el repositorio PyPI es continuamente alimentado por la comunidad de desarrolladores Python al rededor del mundo. Esto hace que el lenguaje de programación Python sea cada vez más robusto y popular.

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 official PIP project page on the PyPI (Python Package Index) website. The browser address bar shows `https://pypi.org/project/pip/`. The page has a blue header with a search bar labeled "Buscar proyectos" and navigation links: "Ayuda", "Patrocinadores", "Acceder", and "Registrarse". The main content area features the text "pip 23.0.1" in large white font, with a green button labeled "Versión más reciente" (Latest version) and a green checkmark. Below this, there's a code snippet `pip install pip` and a copy icon. The release date "Publicación: 17 feb 2023" is shown on the right. A grey banner below the header states "The PyPA recommended tool for installing Python packages." The page is divided into two columns. The left column, titled "Navegación" (Navigation), contains links: "Descripción de proyecto" (Project description), "Histórico de versiones" (Version history), "Archivos de descarga" (Download files), and "Enlaces del proyecto" (Project links) with a "Homepage" link. The right column, titled "Descripción de proyecto" (Project description), includes a shield icon with a URL `https://img.shields.io/pypi/v/pip.svg` and labels "docs" and "passing". The description text says "pip is the package installer for Python. You can use pip to install packages from the Python Package Index and other indexes." It then says "Please take a look at our documentation for how to install and use pip:" followed by a bulleted list with links for "Installation" and "Usage". At the bottom, it states "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
```

```
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. Inténtalo 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:

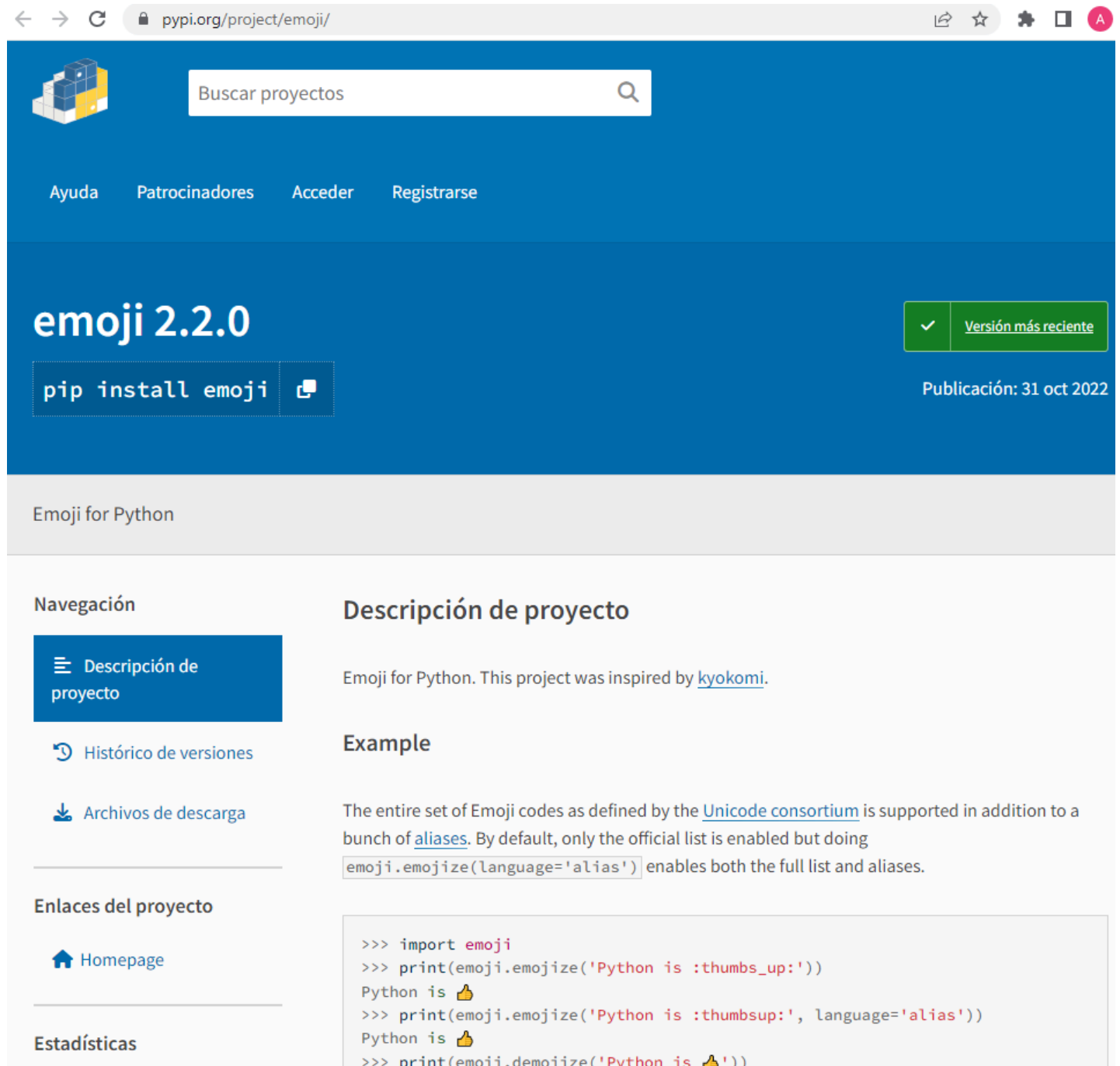
```
pip show pandas
```

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.



The screenshot shows the PyPI project page for the 'emoji' package. The browser address bar shows 'pypi.org/project/emoji/'. The page has a blue header with a search bar and navigation links: 'Ayuda', 'Patrocinadores', 'Acceder', and 'Registrarse'. The main section features the package name 'emoji 2.2.0' in large white text, a green button with a checkmark and 'Versión más reciente', and a green button with the command 'pip install emoji'. Below this, it says 'Publicación: 31 oct 2022'. The page is divided into two columns. The left column has a 'Navegación' sidebar with links: 'Descripción de proyecto' (highlighted), 'Histórico de versiones', and 'Archivos de descarga'. Below this is 'Enlaces del proyecto' with a link to 'Homepage', and 'Estadísticas'. The right column has a 'Descripción de proyecto' section stating 'Emoji for Python. This project was inspired by [kyokomi](#).' followed by an 'Example' section. The example text says: '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.emojiize(language='alias')` enables both the full list and aliases.' Below the text is a code block showing a Python session:

```
>>> import emoji
>>> print(emoji.emojiize('Python is :thumbs_up:'))
Python is 👍
>>> print(emoji.emojiize('Python is :thumbsup:', language='alias'))
Python is 👍
>>> print(emoji.demojiize('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
print(dir(emoji.emoji_list))
print(emoji.emoji_list)

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 :fuego:", 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 alli 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')
```

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. Finalmente, muestra los resultados en la consola.

✓ Apropiación

Consulte sobre algún paquete interesante en el PyPI instálelo y utilícelo en un programa. Presente el paquete usado y su utilidad y muestre su aplicación en el programa desarrollado.

No fue posible conectarse al servicio de reCAPTCHA. Comprueba tu conexión a Internet y vuelve a cargar la página para obtener un desafío de reCAPTCHA.