

-----  
**Manual de Instalación y configuración de la librería OpenCV - Python en Windows**  
-----

<http://umh1782.edu.umh.es/python>

**Requisitos:** Windows 7 - Windows 11 - (64bits)

Más información en <http://opencv.org>

### **1) Instalación de Python:**

**Python** es un lenguaje de programación interpretado y orientado a objetos que puede usarse en cualquier tipo de equipo aunque disponga de recursos hardware limitados. Permite utilizar módulos o paquetes para ampliar las capacidades del lenguaje base, lo que lo ha hecho muy popular en la programación de aplicaciones que precise el acceso y transferencia de información en sistemas remotos o aplicaciones web. Se trata asimismo, de un lenguaje popular en la implementación de aplicaciones de aprendizaje máquina (“*machine learning*”) y el análisis de *big data*.

Para utilizar **Python** debemos instalar el interprete y adicionalmente un gestor de paquetes o módulos. Actualmente existen dos gestores de paquetes disponibles: **pip** y **conda**:

- **pip** es el gestor de paquetes nativo de Python y viene integrado dentro del mismo, aunque también es posible ejecutar el gestor **pip** como un aplicación externa (precisa de instalación adicional).
- **conda** es un gestor de paquetes más avanzado y permite adicionalmente crear entornos virtuales con configuraciones de paquetes específicas para ejecutar nuestro código.

En ambos casos se trata de aplicaciones en línea de comandos, por lo que para simplificar su uso, lo más común es utilizarlas a través de un entorno gráfico que incorpore algún tipo de editor para programación y depuración del código.

Los dos entornos más utilizados son **Pycharm** y **Anaconda**:

- **PyCharm** es un entorno de desarrollo Python desarrollado por *Jetbrains* y que dispone de una versión libre de código abierto denominada “*Community Edition*”. Es ligero e incorpora un editor de código, un gestor de paquetes y entornos virtuales, y una consola de depuración.
- **Anaconda** es un entorno grafico para **conda**, que permite no solo crear y manejar entornos virtuales con diferentes configuraciones de paquetes o módulos python, sino que adicionalmente incorpora acceso directo a múltiples herramientas de desarrollo como editores de código (Spyder, VS Code,...), frameworks de desarrollo específicos y

entornos interactivos como **Jupyter**. Es un aplicación mucho mas pesada y su ejecución puede ralentizarse en equipos con hardware limitado.

En este curso vamos a utilizar el entorno de trabajo **PyCharm**, al ser más ligero e incorporar el gestor de paquetes y entornos virtuales perfectamente integrado en una sola aplicación. En internet puedes encontrar tutoriales para realizar estas tareas en **Anaconda**.

En la página web del curso (<http://umh1782.edu.umh.es/python>) dispones del instalador de **PyCharm** y de **Python**. Deberás seguir las instrucciones que se muestran y completar la instalación. Si no dispones del interprete Python previamente instalado, deberás descargar e instalar la última versión. Al trabajar en este curso con una librería específica (**OpenCV**) pueden presentarse problemas de compatibilidad, por lo que es recomendable instalar una versión de *python* compatible con OpenCV, en la página web del curso tienes disponibles versiones previas (3.7, 3.8, 3.9, 3.10).

**Nota:** en Windows 7 la última versión compatible de python es la 3.8

En todo caso **PyCharm** permite gestionar proyectos con diferentes versiones de *Python* por lo que no supone ningún problema tener instaladas varias versiones. Es recomendable instalar las versiones de *Python* necesarias previamente a la instalación de *PyCharm* para que las reconozca y las integre directamente en el entorno. En todo caso, es posible incorporarlas posteriormente.

En la **Figura 1** se muestra la configuración de la instalación de *Python 3.10* (marcaremos la opción de 'Add Python to PATH').

En la **Figura 2** se muestra la configuración de la instalación de *PyCharm* (Marcaremos la pestaña 'Create Associations .py' y 'Add bin folder to PATH').



Figura 1. Instalación Python 3

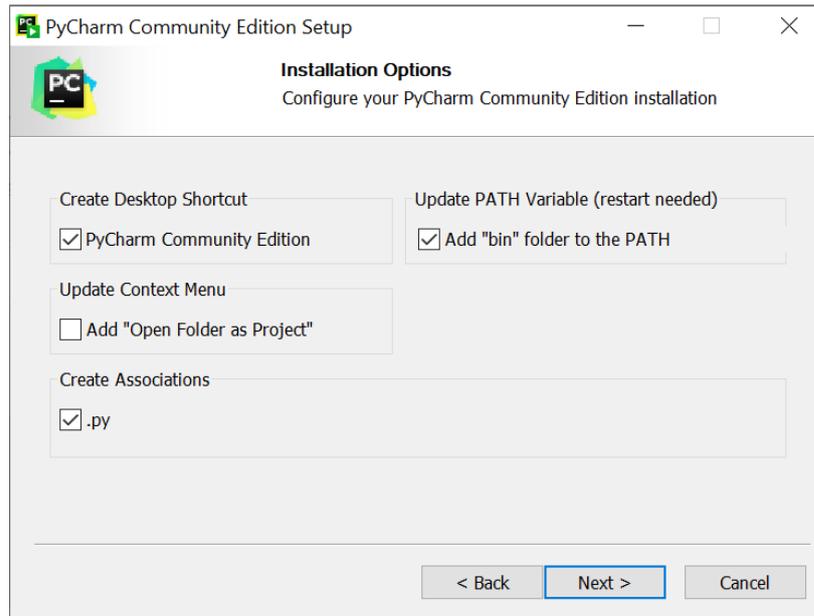


Figura 2. Instalador PyCharm

## **2) Instalar Fuentes librería OpenCV:**

Este proceso es opcional, ya que los paquetes de la librería se descargarán desde *PyCharm*, pero nos permite disponer del código fuente de la librería con los programas de ejemplo.

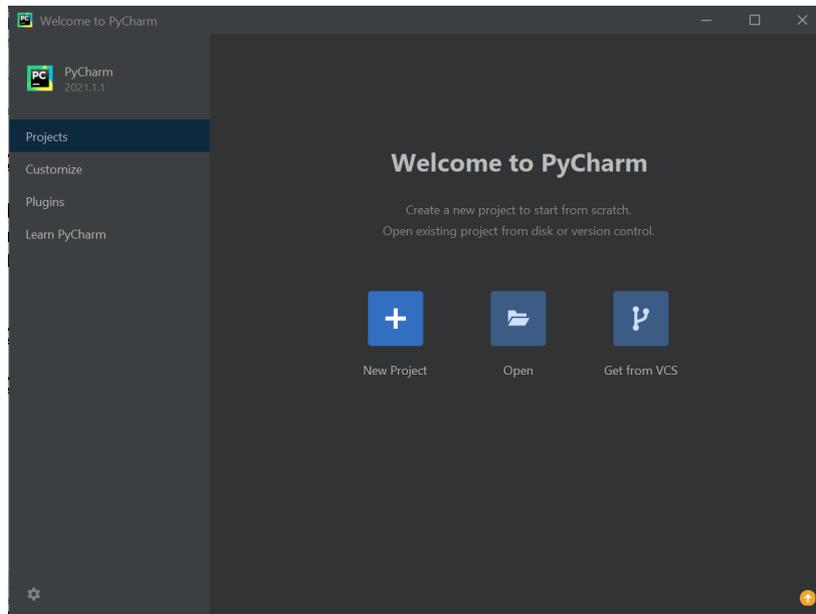
- Descargaremos la última versión. Podemos elegir descargar el fichero comprimido con el código fuente (actualmente ***opencv-4.5.5.zip***), o bien la versión compilada para para Windows (***opencv-4.5.5-vc14-vc15.exe***) que incluye también las librerías para programar en C++ :

<http://opencv.org/downloads.html>

- Si descargamos sólo el código fuente, descomprimiremos el fichero en una carpeta del disco (por ejemplo ***c:\***). El contenido de la librería queda ubicado de este modo en el directorio (***c:\opencv-4.5.5\***). En la carpeta '***c:\opencv-4.5.5\samples\python***' disponemos de los ejemplos de código *python* de la librería OpenCV.
- Si hemos descargado la versión compilada, ejecutaremos el fichero descargado que descomprimirá el contenido de la librería. Nos pedirá el directorio donde deseamos ubicarla (***c:\***) (el propio descompresor creará una carpeta en la ruta que le indiquemos denominada *opencv*). El contenido de la librería queda ubicado de este modo en el directorio (***c:\opencv\***). En la carpeta '***c:\opencv\sources\samples\python***', disponemos de los ejemplos de código *python* de la librería OpenCV.

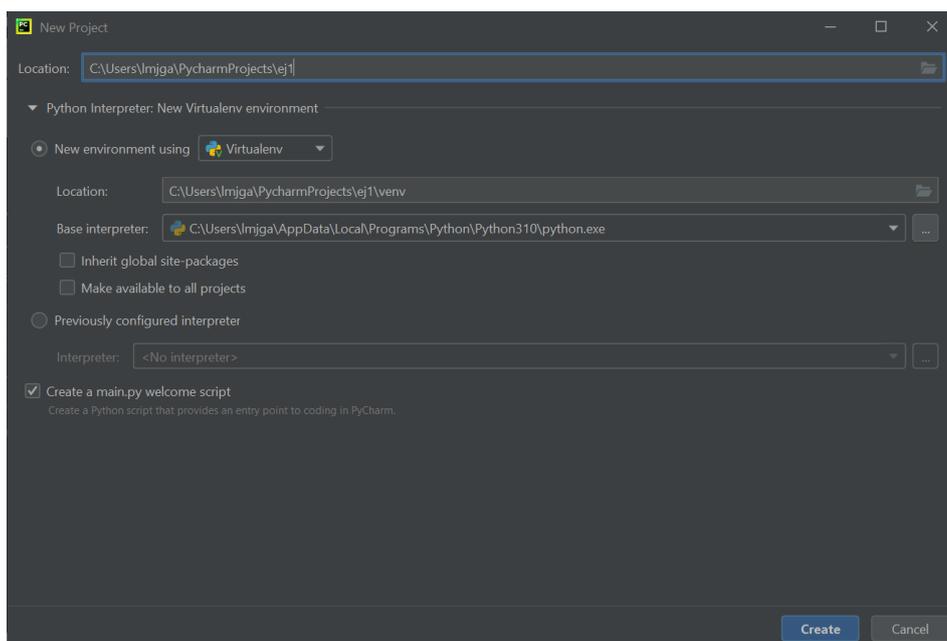
### **3) Crear un proyecto en PyCharm:**

Abriremos la aplicación *PyCharm*. En la **Figura 3** podemos observar el panel principal, que está en inglés. En el lateral izquierdo disponemos opciones para configurar la aplicación '*Customize*', instalar complementos '*Plugins*' y gestionar los proyectos '*Projects*'. Procederemos a crear un nuevo proyecto '*New Project*':



*Figura 3. Panel inicio de PyCharm*

En la **Figura 4** se muestra el panel de opciones para la configuración inicial del proyecto. Seleccionaremos la carpeta y el nombre del proyecto *ej1* ('*Location*'), el entorno virtual ('*Virtualenv*') y el interprete de Python en caso de tener varias versiones instaladas. Adicionalmente dejaremos seleccionada la opción para que cree un script '*main.py*' básico.



*Figura 4. Creación de un nuevo proyecto en Pycharm*

En la **Figura 5** se muestra la ventana de trabajo de PyCharm con los diferentes paneles con el editor de código Python. Si pulsamos el botón **Play**, disponible en la esquina superior derecha, o la opción del menú '**Run/run main**', podremos ver en el panel inferior la salida por consola.

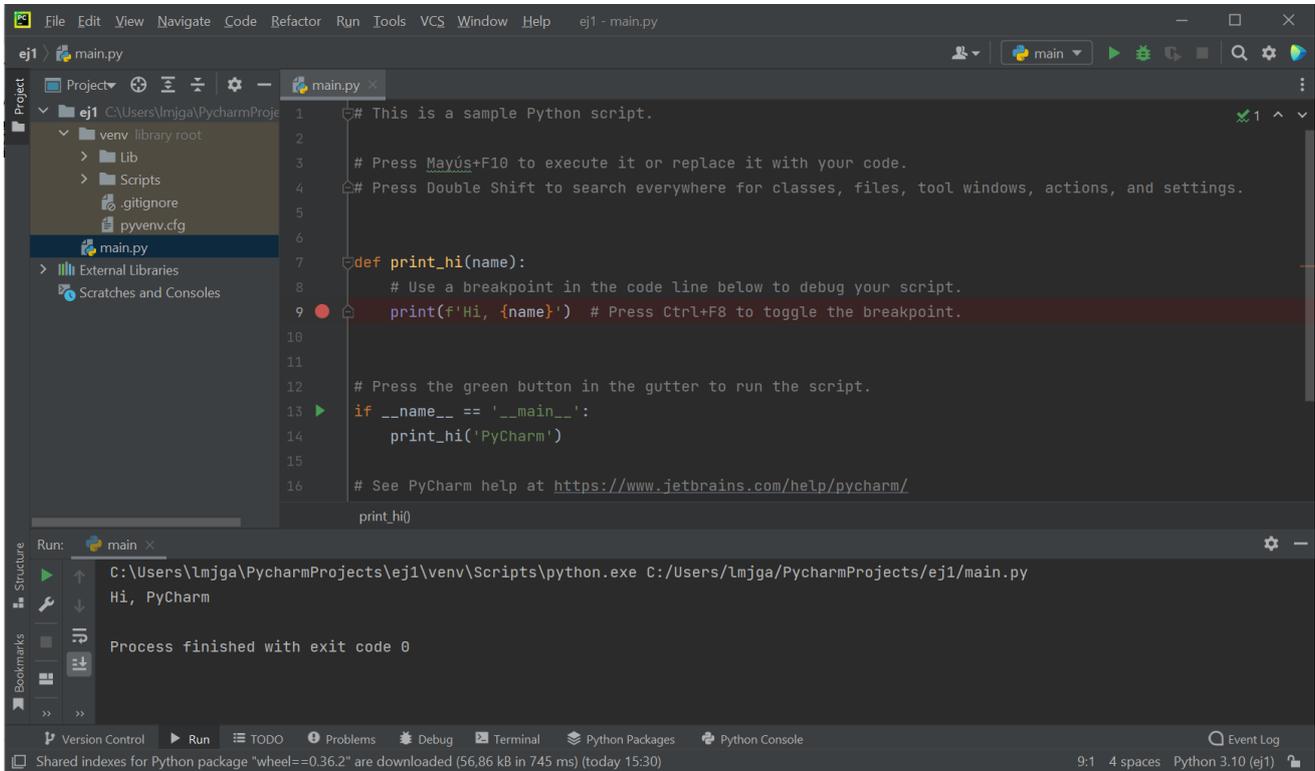


Figura 5. Proyecto ejemplo python. Editor Pycharm

En el proyecto de ejemplo podemos ver la estructura básica de un programa en *Python* y probar las opciones de depuración (**Debug**) de *PyCharm*.

#### **4) Configurar el proyecto con la librería OpenCV:**

Vamos ahora a crear un nuevo proyecto vacío y configurar diferentes paquetes o módulos, entre ellos la librería *OpenCV*.

En el Menú '**File**' seleccionamos '**New Project**' y creamos un nuevo proyecto '*ej2*', pero esta vez desmarcamos la opción '*Create a main.py welcome script*' para crear un proyecto vacío (**Figura 6**).

Adicionalmente, en la opción marcada '**New enviroment using**' (**Figura 6**), vamos a ubicar la carpeta del entono virtual **venv** (parámetro **Location**) en la carpeta raíz de nuestros proyectos para poder reutilizarla en los diferentes programas que realicemos en este curso.

Esto es importante ya que cuando instalemos los paquetes adicionales ocupará bastante espacio en el disco por lo que no querremos duplicarlo en cada proyecto.

Lo etiquetaremos con el nombre '**opencv-venv**', ya que lo vamos a configurar para usar la librería *OpenCV*.

Marcaremos adicionalmente la casilla: '**Make available to all projects**'

**Nota:** En futuros proyectos seleccionaremos la opción '**Previously configured Interpreter**', indicando el entono virtual que hemos creado (**opencv-venv**).

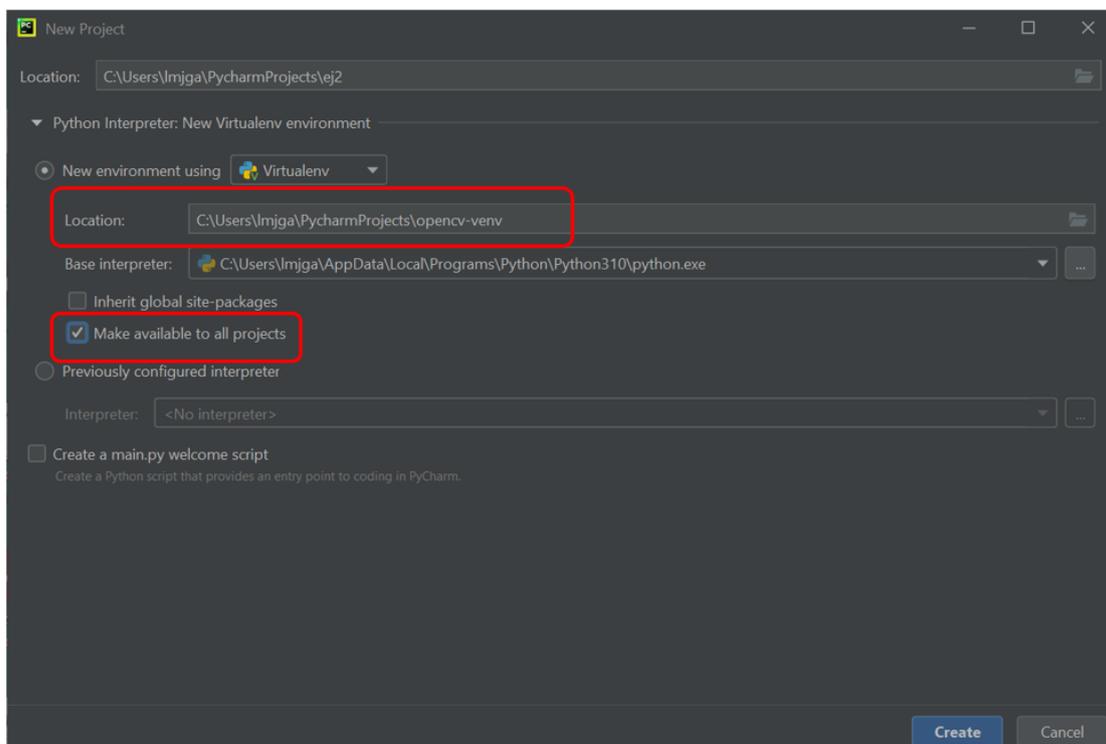


Figura 6. Crear Proyecto vacío en PyCharm. Entono virtual común

Como tenemos ya un proyecto abierto, nos solicita (**Figura 7**) si queremos abrir el nuevo proyecto en una nueva ventana o sustituyendo al proyecto previo.

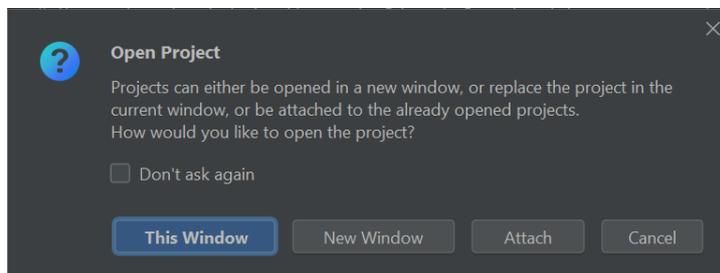


Figura 7. Aviso selección de ventana para nuevo proyecto

Una vez creado, si pulsamos sobre el panel izquierdo '**Project**', podemos añadir un fichero de código *python*, para lo cual abriremos el menú contextual (botón derecho del ratón) sobre el nombre del proyecto y seleccionando '**New / Python file**' (**Figura 8**) y crearemos el fichero *ej2.py*.

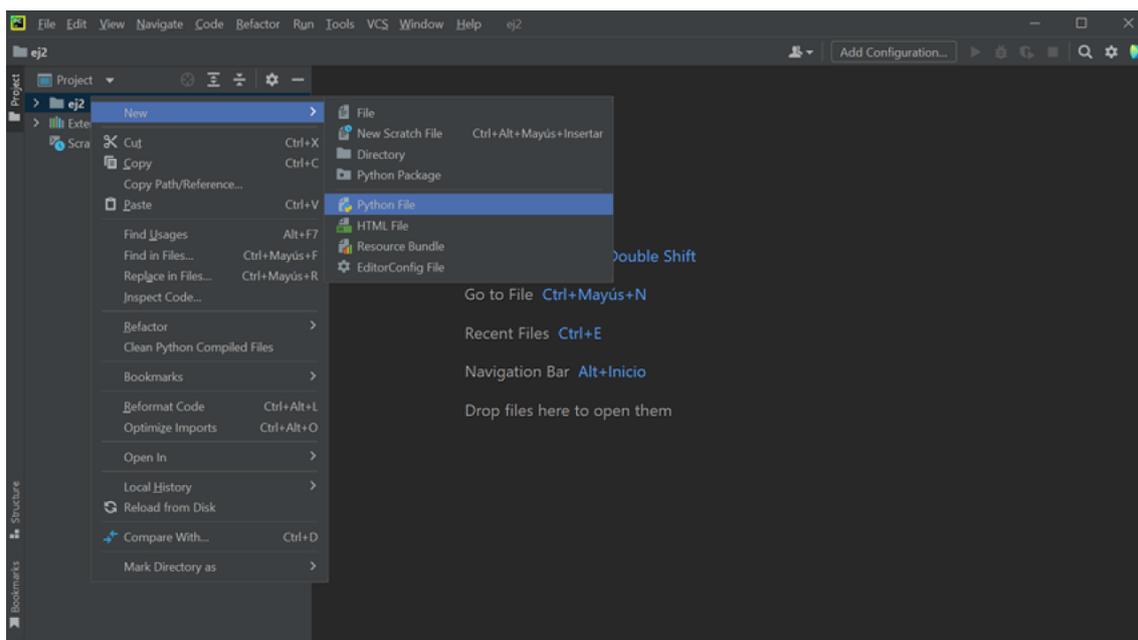


Figura 8. Añadir fichero .py al proyecto

Nos queda por configurar la ejecución de código Python en nuestro proyecto, para ello pulsaremos sobre el botón '**Add Configurations**' en el panel superior derecho. Nos muestra una ventana de configuración donde pulsaremos el botón '**Add New Configuration**' y seleccionaremos '**Python**' (**Figura 9**).

En la ventana de configuración indicaremos el nombre '*ej2*' en el recuadro '**Name**' y añadiremos el script a ejecutar (entrada *main*) '*ej2.py*' en el recuadro '**Script path**' (**Figura 10**).

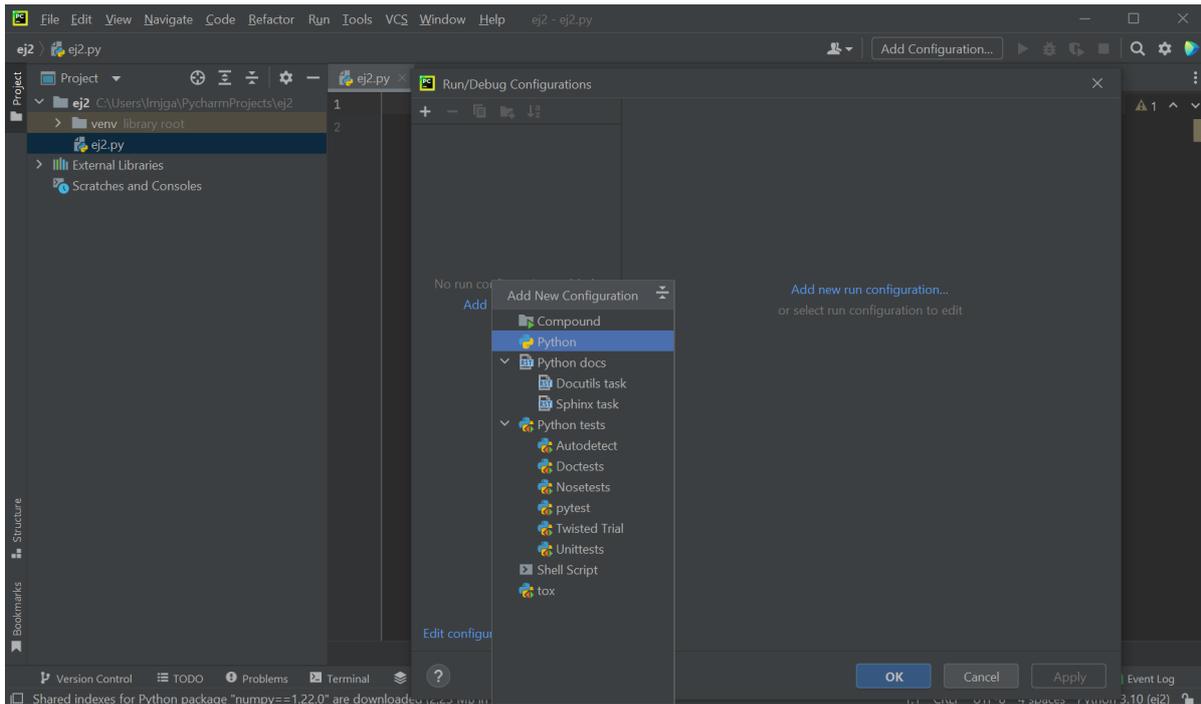


Figura 9. Añadir configuración de código python al proyecto

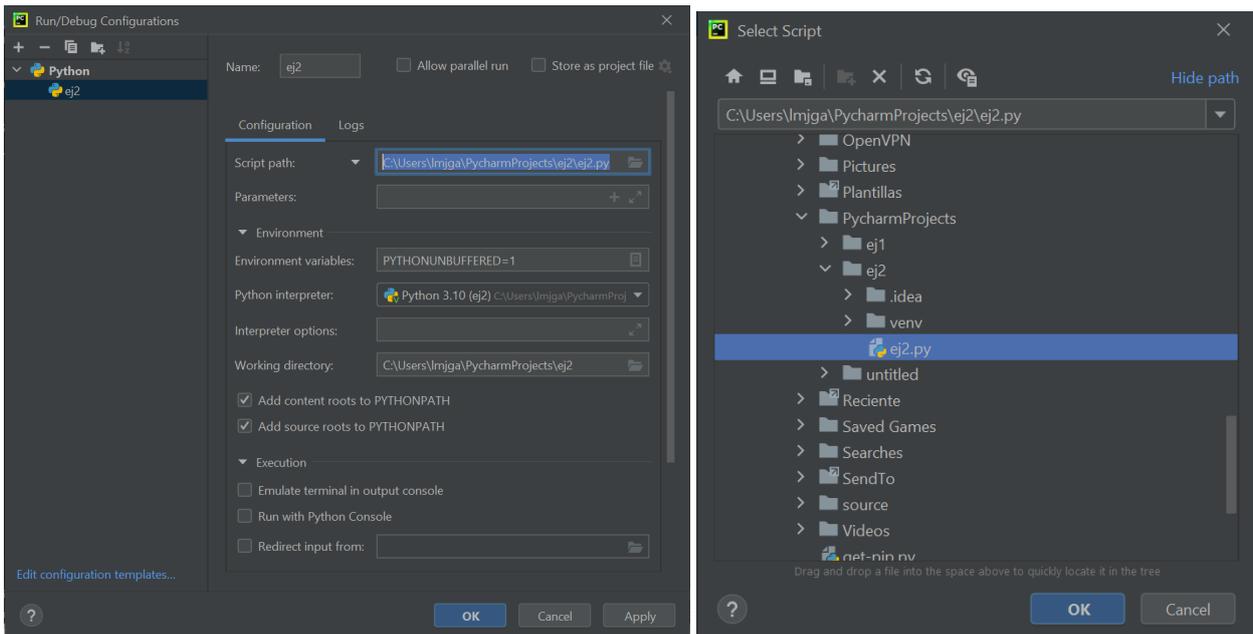


Figura 10. Configuración de ejecución del script

A continuación, vamos a configurar los paquetes adicionales para crear la aplicación usando la librería OpenCV. PyCharm nos proporciona dos métodos:

- Ventana de configuración del Proyecto: Menú '**File / Settings**' pestaña '**Project ej2 / Python Interpreter**' (**Figura 11**).
- Panel '**Python Packages**' en la barra inferior de la ventana principal (**Figura 12**)

Utilizaremos esta última alternativa al ser más sencilla y directa desde la ventana principal.

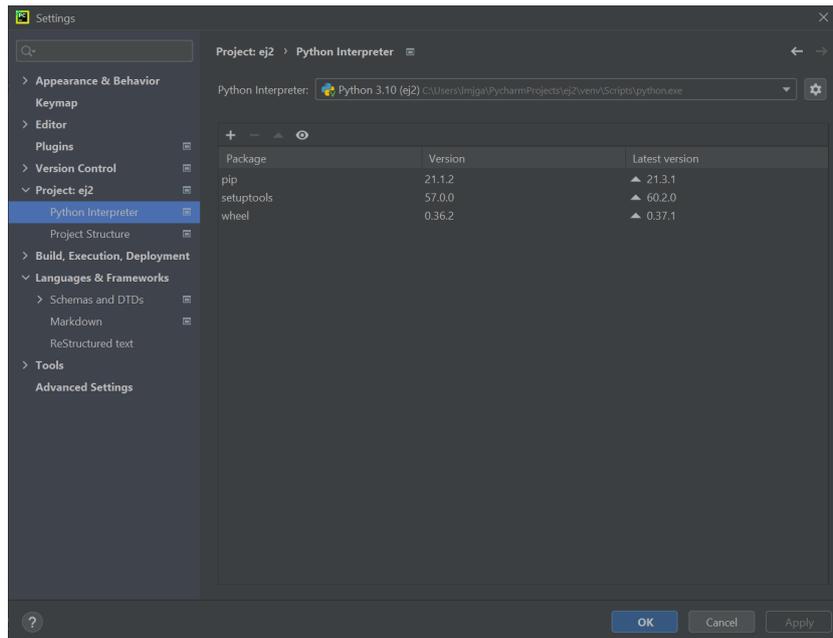


Figura 11. Configuración de paquetes desde el Panel de Configuración de PyCharm

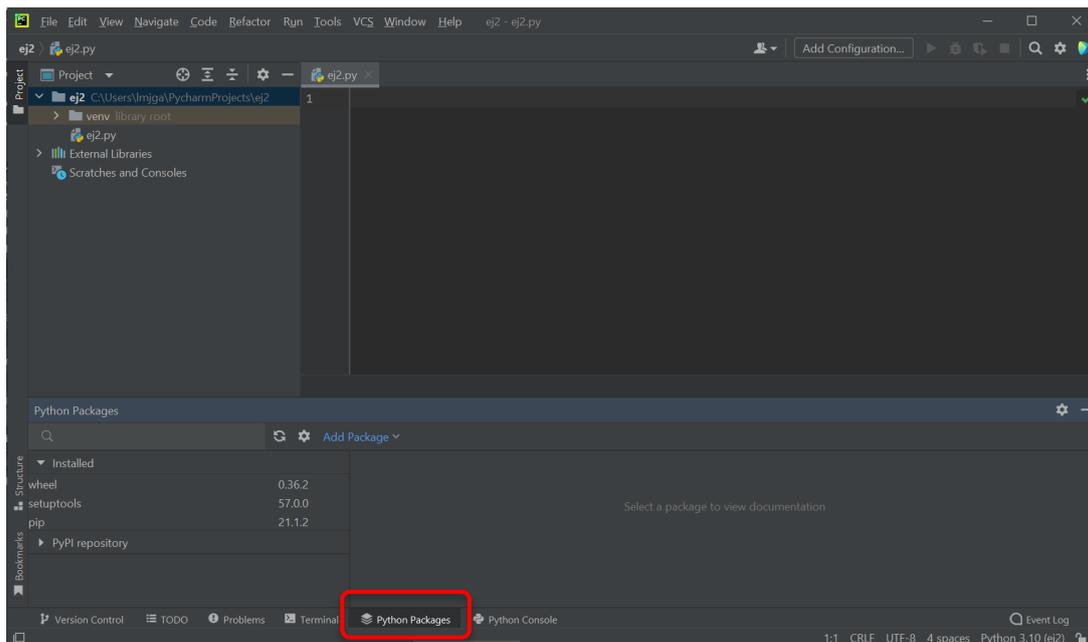


Figura 12. Configuración de paquetes desde el panel 'Python Packages'

Buscaremos en el recuadro de búsqueda aquellos paquetes que queramos instalar, en nuestro caso:

- **opencv**: librería de captura y procesamiento de imágenes
- **numpy**: librería de álgebra vectorial y matricial
- **scipy**: librería de álgebra lineal, optimización y procesamiento de señal.
- **matplotlib**: visualización de datos 2D similar a Matlab

El instalador de paquetes se encarga de tratar las dependencias entre los mismos.

Cuando introducimos el nombre del paquete a buscar (*opencv*) nos muestra todos los paquetes disponibles (**Figura 13**):

- **opencv-python**: paquete principal
- **opencv-contrib-python**: paquete principal + módulos con algoritmos adicionales
- **opencv-python-headless/opencv-contrib-python-headless**: paquetes sin las funciones de visualización de imágenes y de interfaz gráfica (GUI).

En nuestro caso instalaremos el paquete base (**opencv-python**). En la ventana derecha nos muestra información del paquete, indicando como instalarla manualmente con *pip* y como importarla a nuestro programa. Podemos seleccionar una versión específica o dejar que busque la última (*latest*).

Pulsando el botón '**Install**' procede a instalar el paquete y todas sus dependencias. Una vez terminada la instalación, podemos verificarla en el panel izquierdo dentro del apartado '**Installed**' (**Figura 14**). Procedemos del mismo modo con el resto de paquetes.

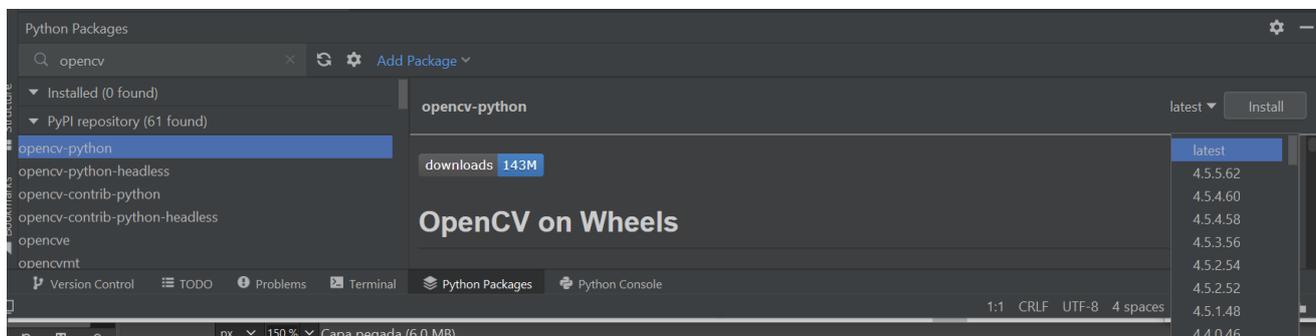


Figura 13. Instalar paquete opencv-python

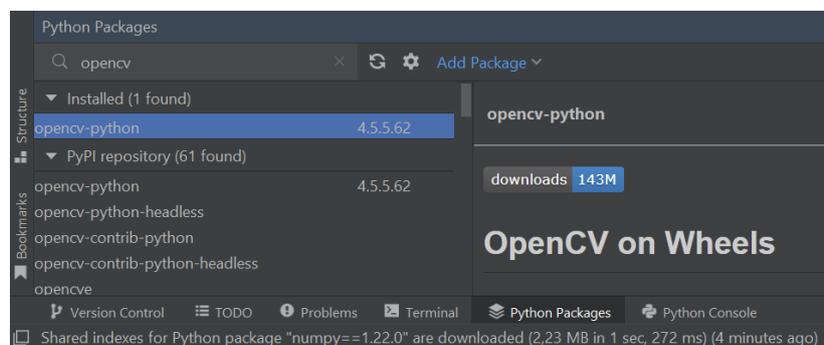


Figura 14. Paquetes instalados

## 5) Prueba de la librería OpenCV:

Vamos a implementar un programa sencillo en *python* que verifique que está instalada la librería OpenCV mostrando la versión, cargaremos una imagen de ejemplo y la visualizaremos.

En primer lugar, copiaremos una de las imágenes de ejemplo (*'building.jpg'*) desde la carpeta de la instalación del código fuente de OpenCV a la carpeta del proyecto:

– *'c:\opencv-4.5.5\samples\data\'* ó *'c:\opencv\sources\samples\data\'*

**Nota:** en la página web del curso puedes descargarte también la imagen de ejemplo.

En el editor introducimos el siguiente programa:

```
# OpenCV example

import cv2 as cv
import sys

# Show OpenCV Version
print('Python ver.: ' + sys.version)
print('OpenCV ver.: ' + cv.getVersionString())

img = cv.imread('building.jpg')
if img is not None:
    cv.imshow("Display window", img)
    k = cv.waitKey(0) # wait for a key press
```

La ejecución nos debe mostrar la imagen cargada:

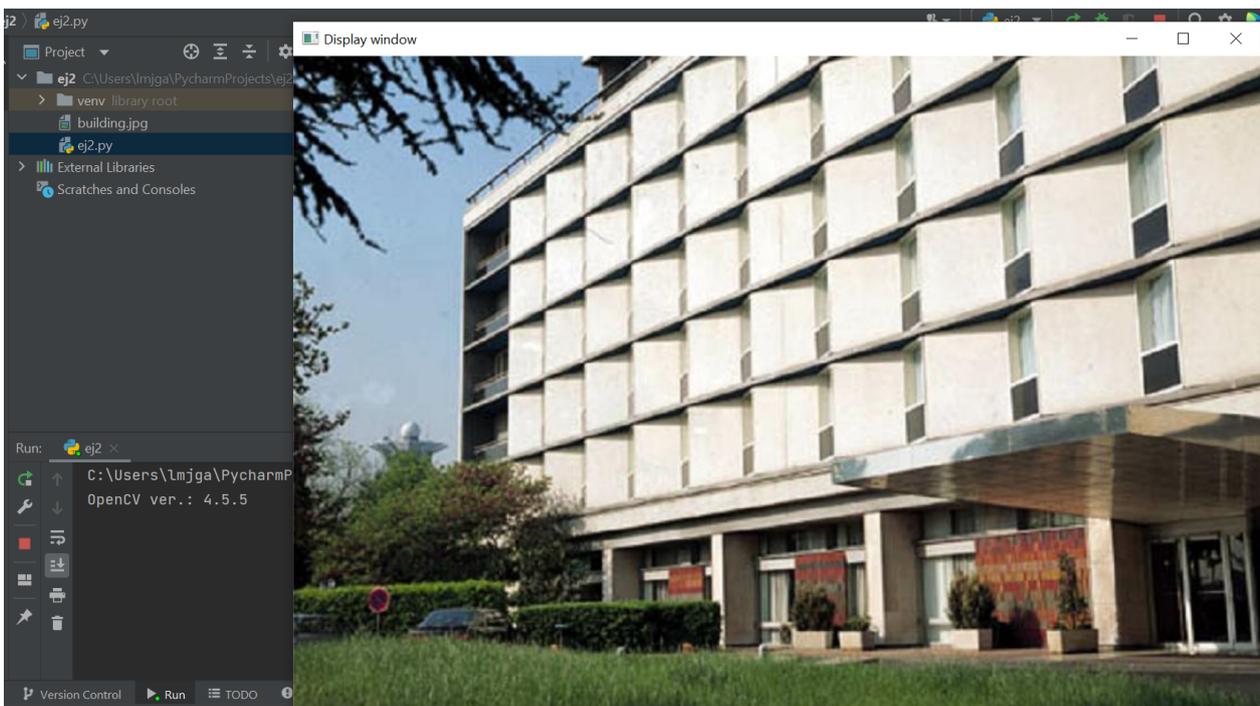


Figura 15. Programa de ejemplo

## **6) Ejemplos:**

- En la siguiente carpeta tenemos ejemplos de código de las diferentes funciones de la librería OpenCV, si hemos instalado el código fuente opcional:

```
c:\opencv\sources\samples\python\  
ó  
c:\opencv-4.5.5\samples\python\
```

## **7) Instalación manual de paquetes:**

La ejecución del código *python* desde *Pycharm* utiliza la configuración del entorno virtual con todos los paquetes configurados, pero en caso de que queramos ejecutar nuestro script en otra máquina precisaremos instalar los paquetes necesarios usando **pip** desde una consola (**CMD**).

**Nota:** Si tenemos la versión 2 de python instalada, deberemos especificar la versión 3 tanto de pip como de python, añadiendo un 3 al final del nombre del comando.

El comando para instalar los paquetes es el siguiente:

```
pip install opencv-python matplotlib numpy scipy  
ó  
python -m pip install opencv-python matplotlib numpy scipy
```

Si queremos instalar *pip* como aplicación debemos descargar y ejecutar este script desde python3:

```
curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py  
  
python get-pip.py
```

Actualizar *pip*:

```
python -m pip install --upgrade pip
```

Desinstalar paquetes en *pip*:

```
pip uninstall opencv-python
```

## **8) Ejecución del script desde VirtualEnv:**

La ejecución del código *python* puede realizarse desde línea de comandos (consola) sin necesidad de instalar los paquetes manualmente, usando la configuración de **VirtualEnv**. Para ello, abriremos una consola (**CMD**) y cambiaremos el directorio de trabajo a la ubicación del proyecto. Ejecutaremos el siguiente código:

```
..\opencv-venv\Scripts\activate  
  
python ej2.py
```