

INTRODUCCIÓN A LA VISIÓN ARTIFICIAL OPENCV PYTHON



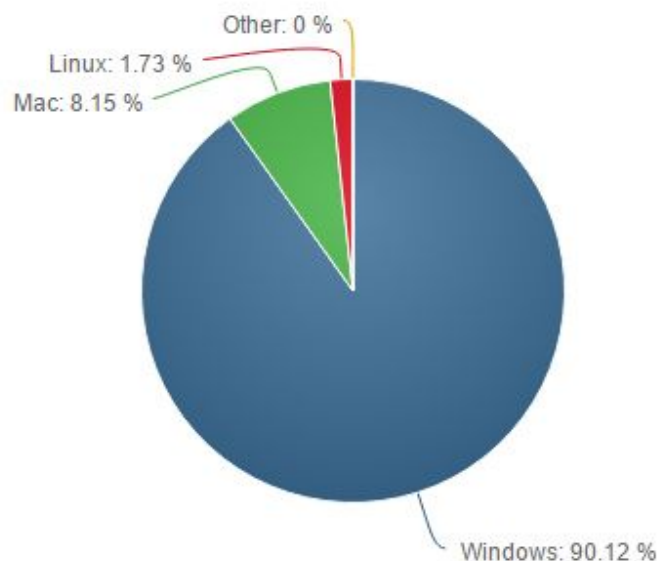
1 Instalación de OpenCV en Windows	3
1.1 Instalar Python con Anaconda 3	4
1.2 Instalar OpenCV para Python 3	16
1.2.1 Descargar el archivo	16
1.2.2 Instalando OpenCV	17
1.3 Instalar Sublime Text 3 y el paquete para Python Anaconda	19
2 Trabajando con imágenes con OpenCV y Python	26
2.1 Probar instalación OpenCV y Python	26
2.2 Cargar una imagen desde el disco duro	27
2.3 Mostrar la imagen por pantalla	29
2.4 Guardar la imagen en el disco duro	30
3 Fundamentos de imágenes digitales	31
3.1 Píxeles en una imagen	31
3.2 Sistema de coordenadas	33
3.3 Acceder y manipular los píxeles de una imagen	34
3.4 Accediendo y manipulando una región de la imagen	37

1 Instalación de OpenCV en Windows

Tanto en el Campus como en el blog de Programarfacil, te hablo de diferentes tecnologías que en su mayoría son *plug and play*, Arduino, Scratch, Processing, Snap!, Raspberry Pi, etc... Cuando empezamos a profundizar en otras materias como el entorno web, la programación del backend, la cosa cambia.

Encontramos multitud de frameworks, librerías y herramientas que, aunque nos facilitan la vida a la hora de programar y desarrollar aplicaciones, no deja de ser un inconveniente a la hora de comenzar. Lo mismo ocurre con la visión artificial o visión por computador, necesitamos preparar el entorno para empezar a desarrollar.

En esta lección vamos a dar los primeros pasos, comenzaremos por la **instalación de OpenCV en Windows**.



Por lo tanto es una obligación empezar por aquí, ya que la gran mayoría de usuarios tienen este sistema operativo.

Esto no quiere decir que con sistemas operativos como Linux y OS X no se pueda o se deba programar aplicaciones de visión artificial, al contrario. Si realmente queremos profundizar en esta materia debemos migrar hacia el entorno Linux.

Lo primero es su rendimiento, supera con creces al que nos pueda ofrecer Windows. Lo segundo es que, el objetivo es poder hacer dispositivos autónomos como un robot o un vehículo.

Para conseguir este objetivo la mejor manera es migrar todo nuestro código a Linux y que nuestra aplicación funcione en una Raspberry Pi, por su bajo coste y por sus prestaciones.

El lenguaje de programación lo es todo. Gracias a que **el software es infinito**, podemos crear todo aquello que nos permita la imaginación. La idea es poder desarrollar aplicaciones de una forma fácil, que nos sintamos a gusto con el entorno (IDE, sistema operativo, lenguaje de programación etc...) y luego poder migrar a un dispositivo como Raspberry Pi.

La gran ventaja que tiene **OpenCV es que podemos programar en diferentes lenguajes de programación entre ellos Python.**

Python es un lenguaje muy sencillo de aprender y con mucha potencia. Es multiplataforma ya que lo podemos ejecutar en diferentes sistemas operativos. Una vez que hayamos adquirido la destreza suficiente con OpenCV y Python, no nos resultará difícil migrar de un sistema a otro nuestro código, esa es la idea.

Que todo esto no te lleve a engaño, la visión artificial es un tema tedioso y complejo. Desde que instalamos los paquetes y herramientas necesarios para comenzar a trabajar hasta que realmente hemos creado nuestra primera aplicación.

Con este curso irás viendo progresivamente los diferentes conceptos matemáticos y de programación, paso a paso. Yo estaré siempre a tu lado, para que cuando tengas dudas, no te quedes atascado y así puedas avanzar en este maravilloso mundo.

Vamos a empezar por el principio. Instalando todo lo necesario para empezar a programar. Estos son los pasos que vamos a seguir.

- Paso 1: Instalar Python con Anaconda 3
- Paso 2: Instalar OpenCV para Python 3
- Paso 3: Instalar Sublime Text 3 y el paquete para Python Anaconda

1.1 Instalar Python con Anaconda 3

Para trabajar con OpenCV sobre Python, es necesario utilizar ciertos paquetes que nos harán la vida más fácil. Lo entenderás según vayamos avanzando en el curso.

Estos paquetes son:

- [NumPy](#) es una librería de código abierto que da soporte a arrays y vectores para Python. Imagínate una imagen como si fuera un panel de las abejas. Cada imagen tendrá un número de celdas dependiendo de su ancho, alto y canales de color. Numpy nos ayuda a gestionar este tipo de estructura de datos de una forma muy sencilla.
- [SciPy](#) es una biblioteca de código abierto de herramientas y algoritmos matemáticos para Python. Contiene módulos para optimización, álgebra lineal, integración, interpolación, procesamiento de imágenes y otras tareas para la ciencia e ingeniería.
- [Matplotlib](#) es una biblioteca para la generación de gráficos a partir de datos contenidos en listas de arrays y vectores. Esta optimizada especialmente para Python y NumPy.

La versión recomendada de Python es la 3.5 o superiores. Existen diferentes maneras de instalar Python y los paquetes necesarios.

Por un lado podemos ir uno por uno descargando e instalando. Debemos de tener cuidado con el sistema operativo, las versiones, los procesadores y todo aquello que pueda generar algún tipo de error.

Pero también existen plataformas orientadas a la minería de datos y la ciencia que nos ofrecen estos y otros muchos paquetes. Es muy fácil de instalar. El que vamos a ver en esta lección se llama [Anaconda](#). En esta plataforma vienen todos los paquetes anteriormente citados y además tenemos versiones para Windows, Linux y OS X.

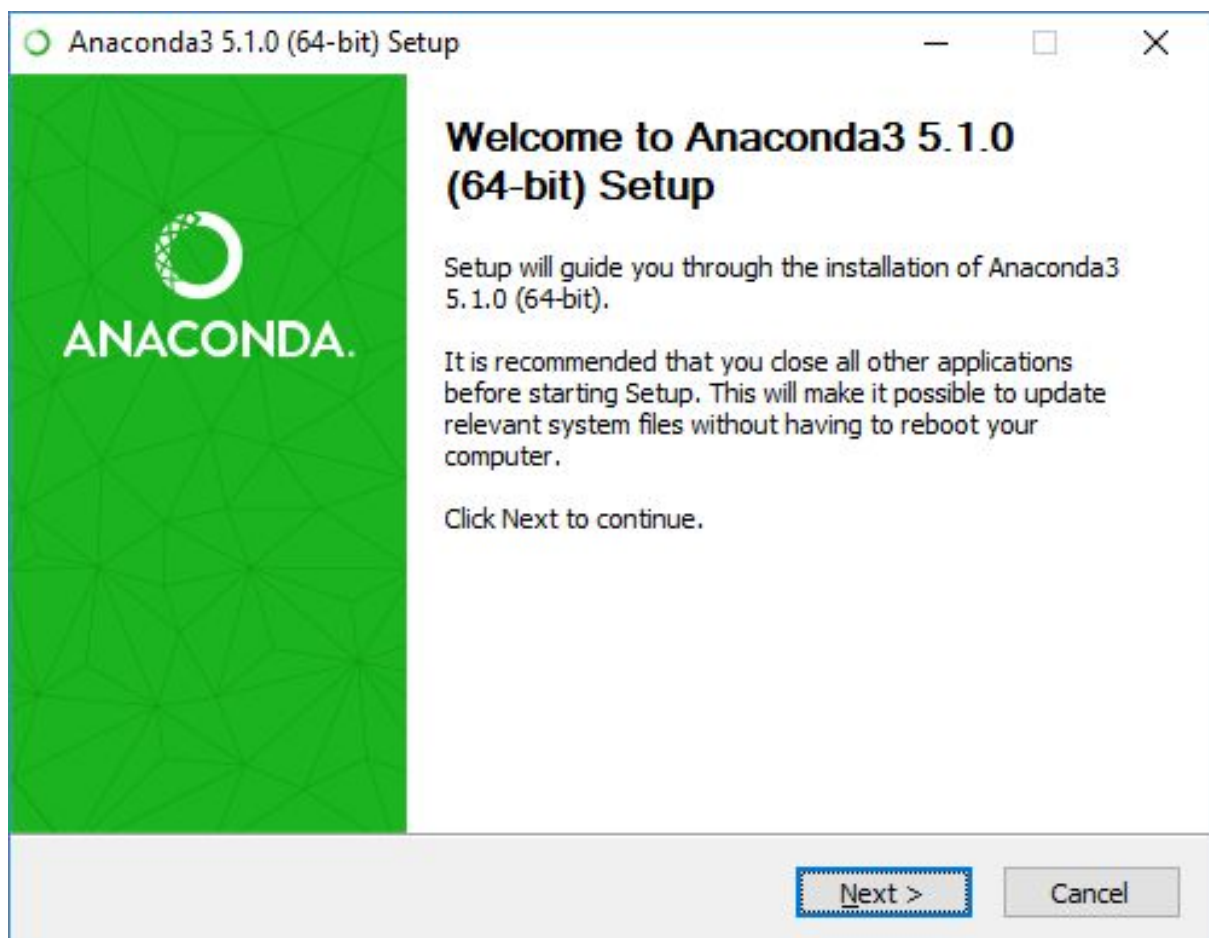
Lo que pretendo es que esto no sea un dolor de cabeza y que empecemos a programar lo antes posible.

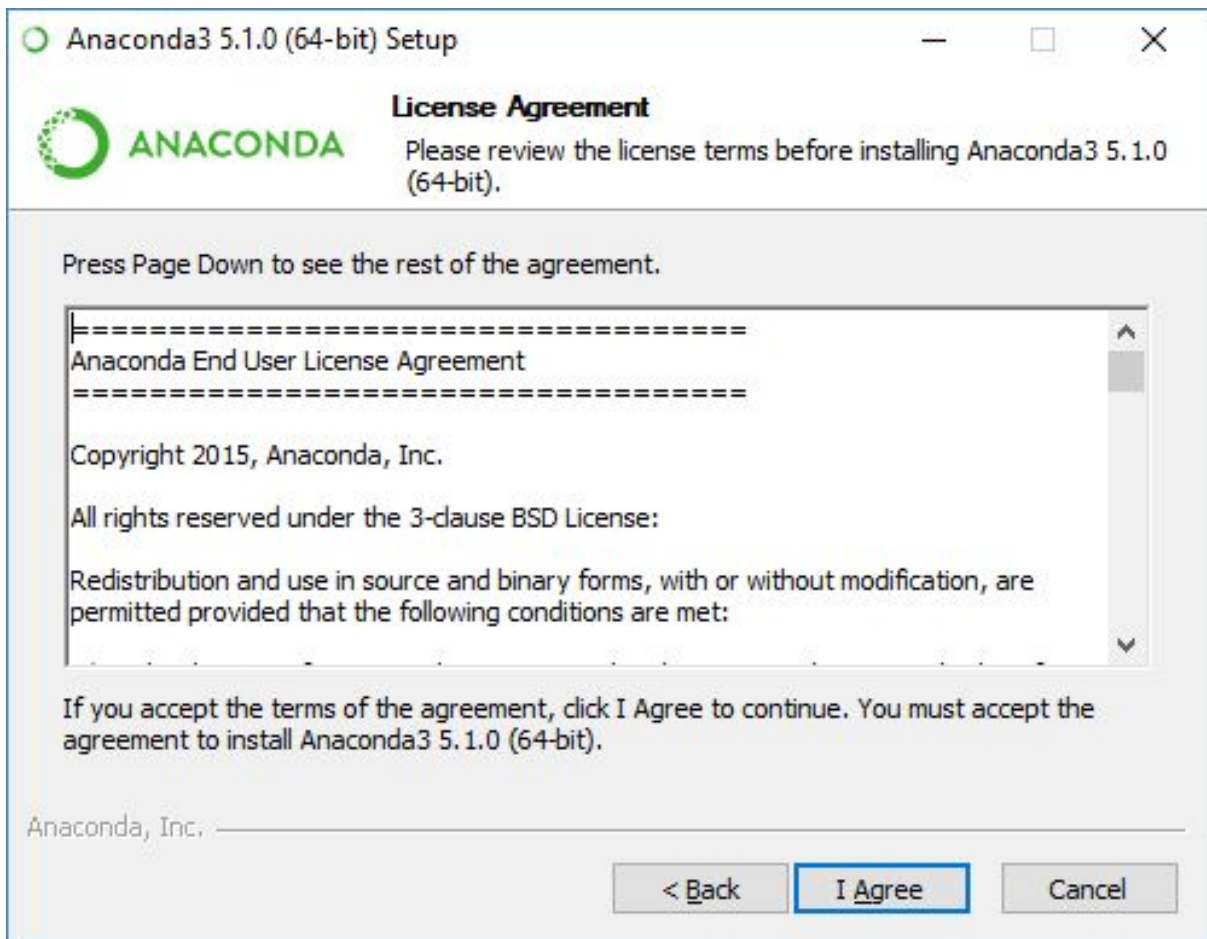
Accedemos a la [zona de descarga de Anaconda](#) y descargamos la versión para nuestro sistema operativo. Asegúrate de escoger la versión Python 3.5 y la versión 32-bit o 64-bit según sea tu sistema operativo.

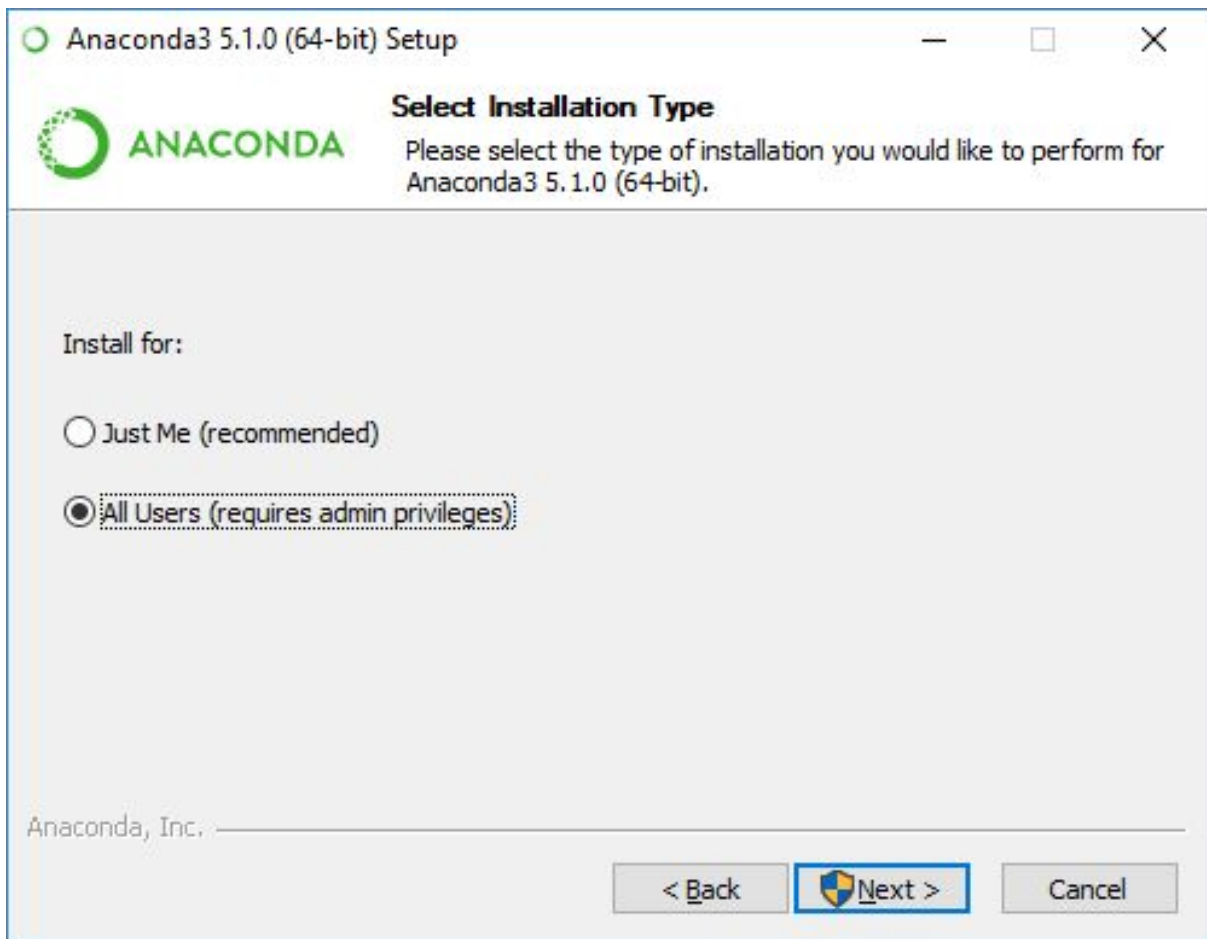


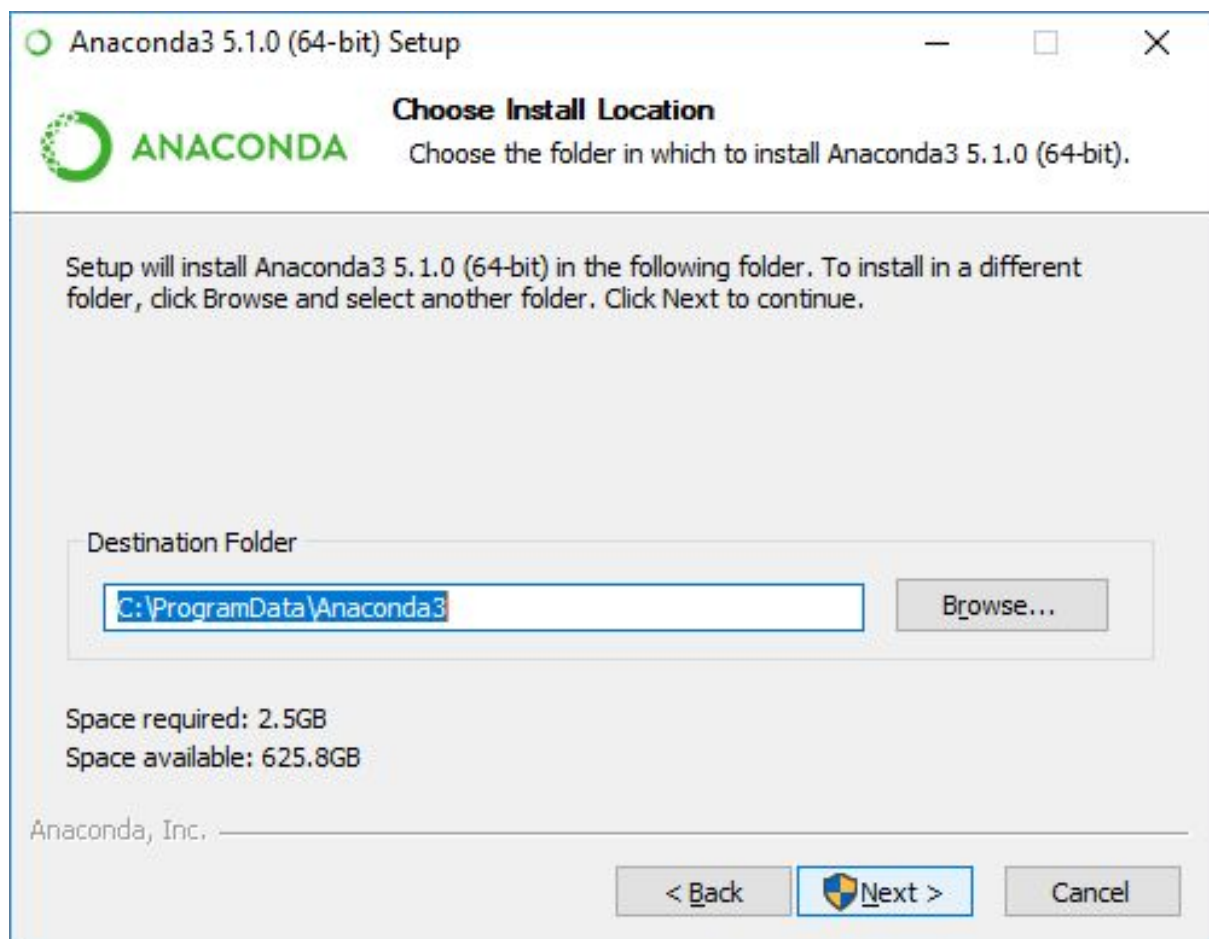
Una vez descargado, **si utilizas Windows ejecútalo como administrador**, así no tendrás ningún problema en la instalación.

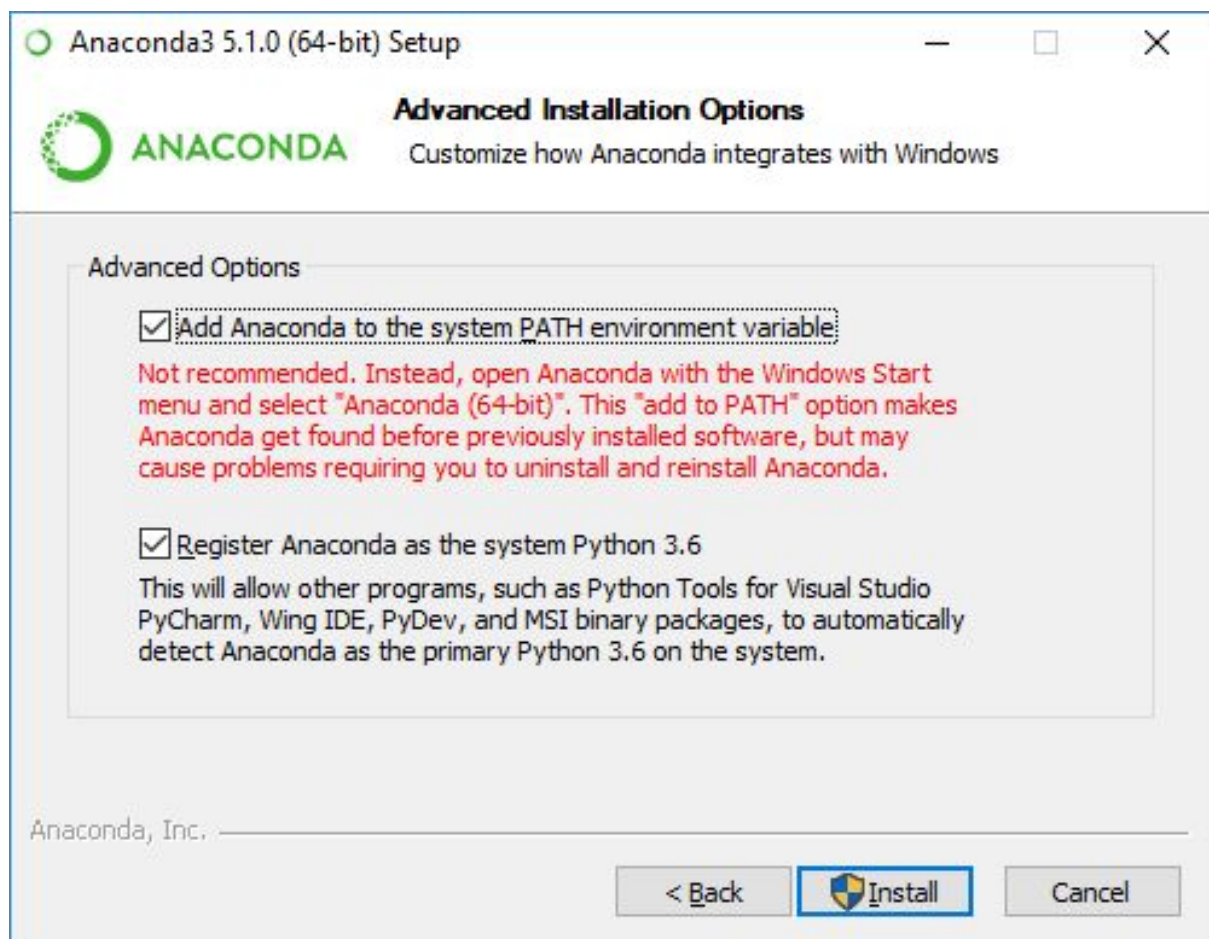
Sigue la siguiente secuencia de instalación.



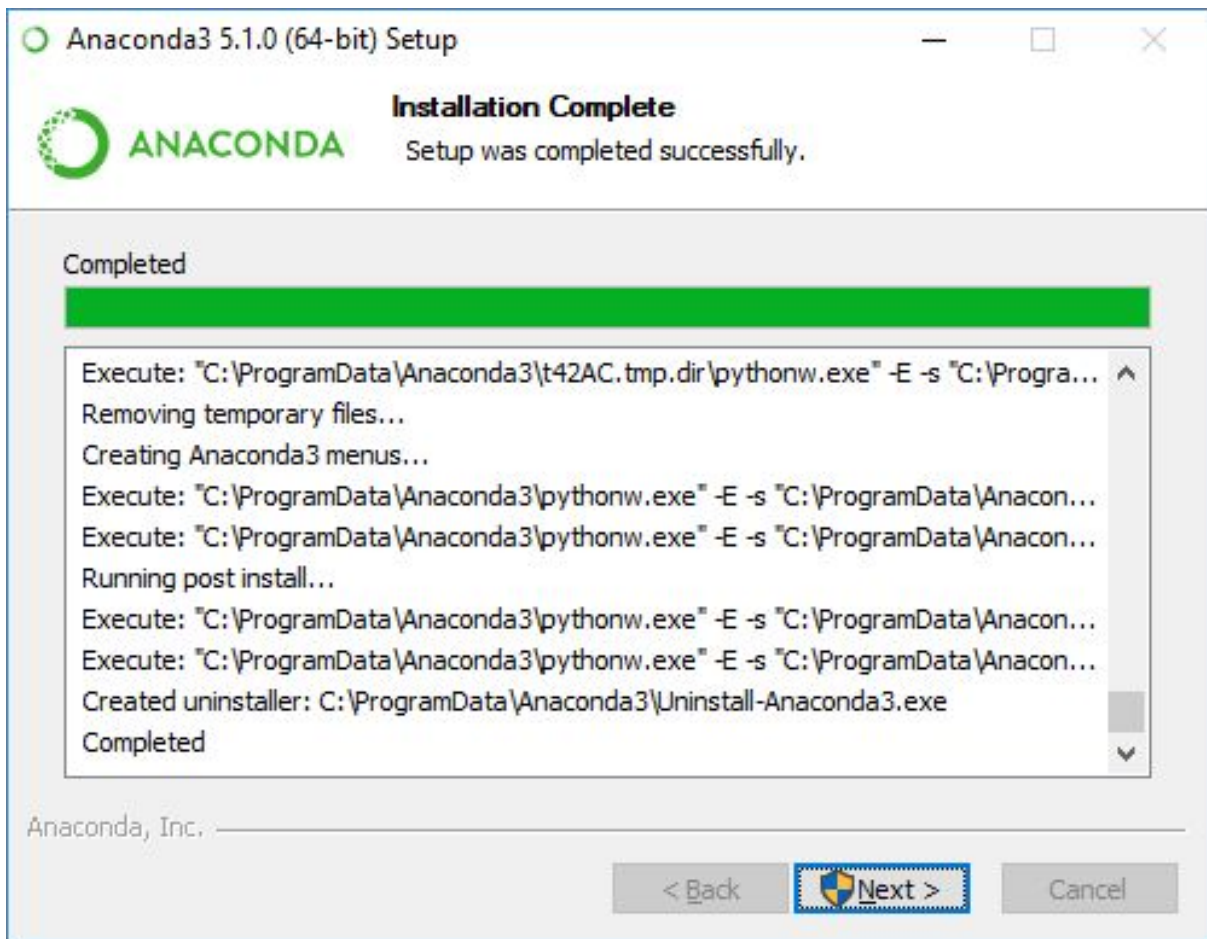


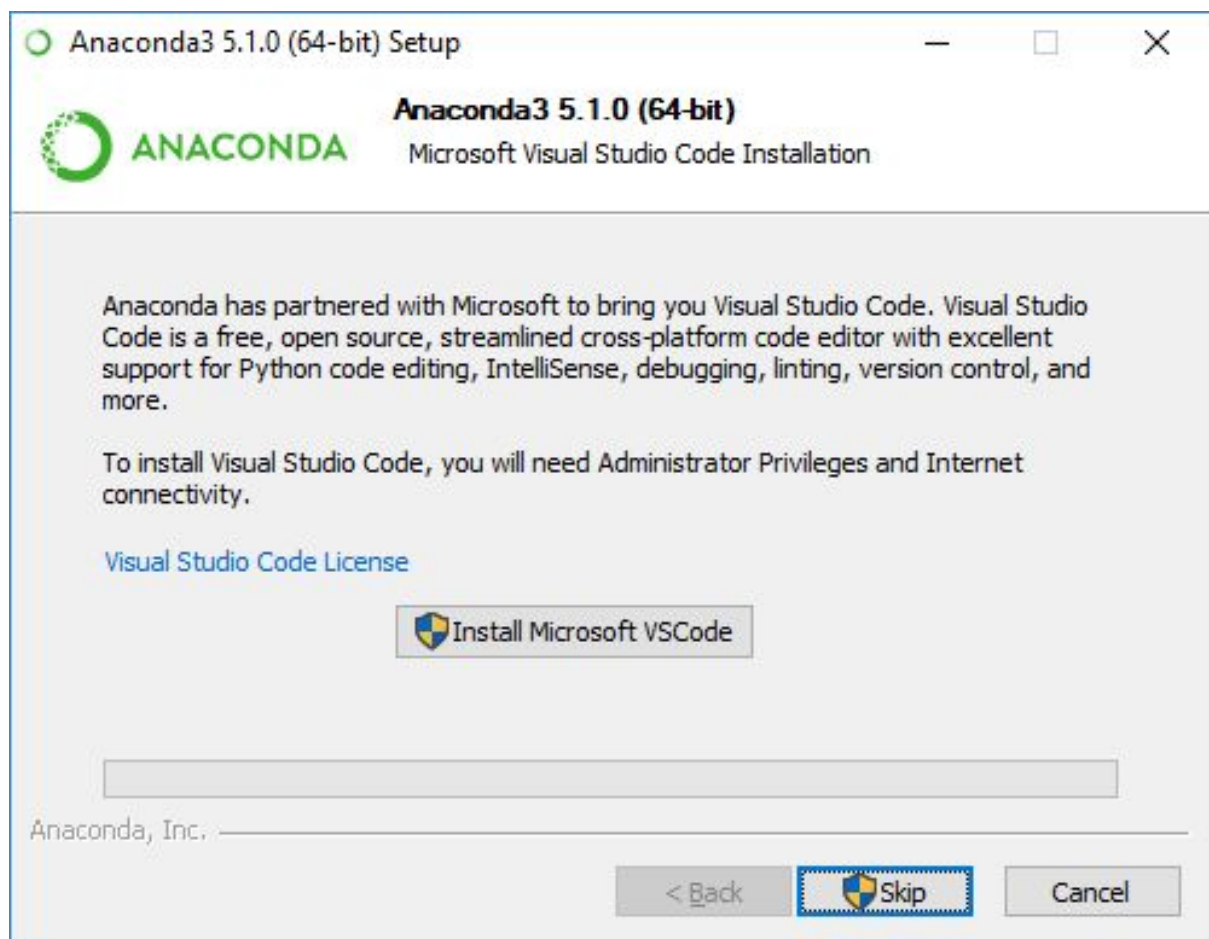




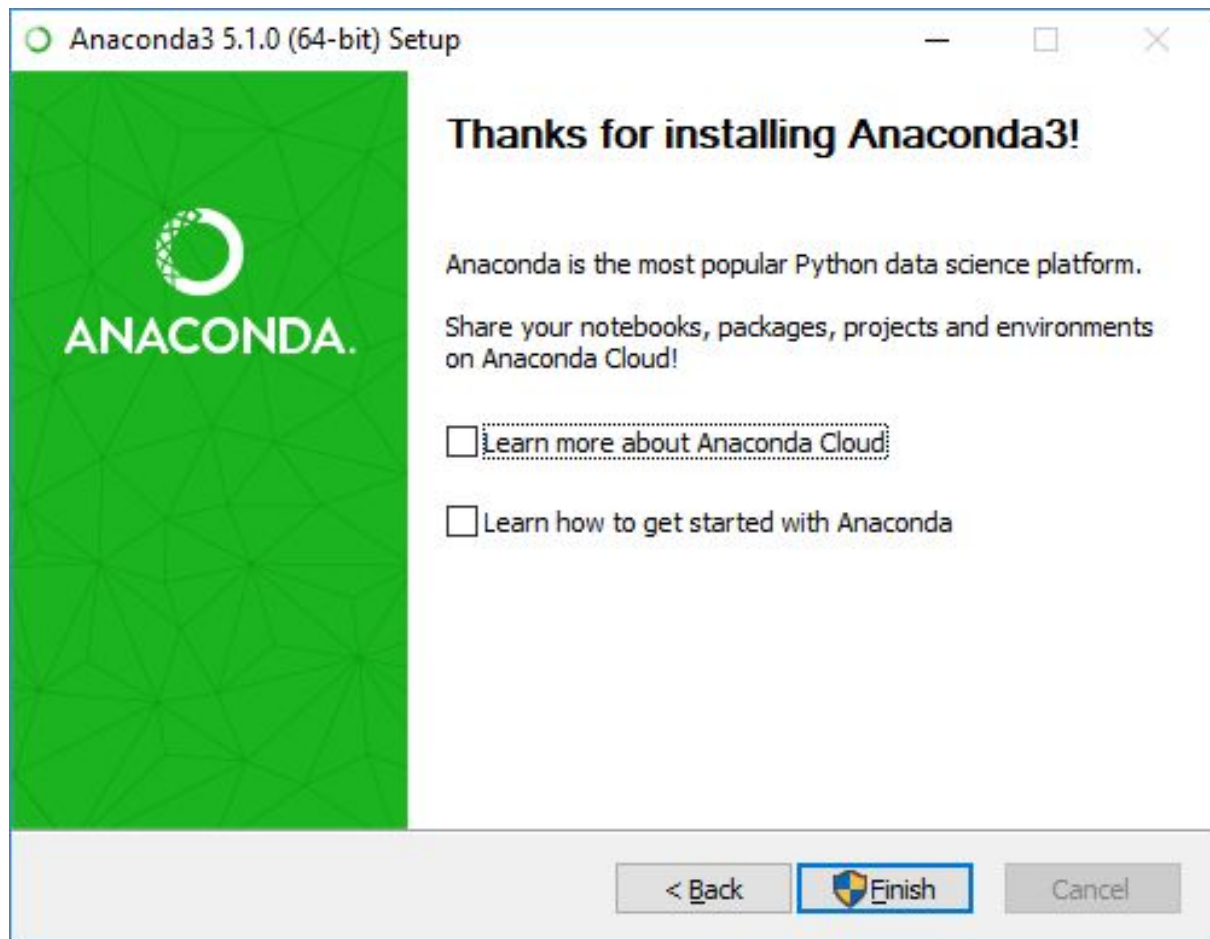


Importante este punto, marca los dos aunque aparezca en rojo, esto nos permitirá luego acceder a Python desde cualquier terminal.





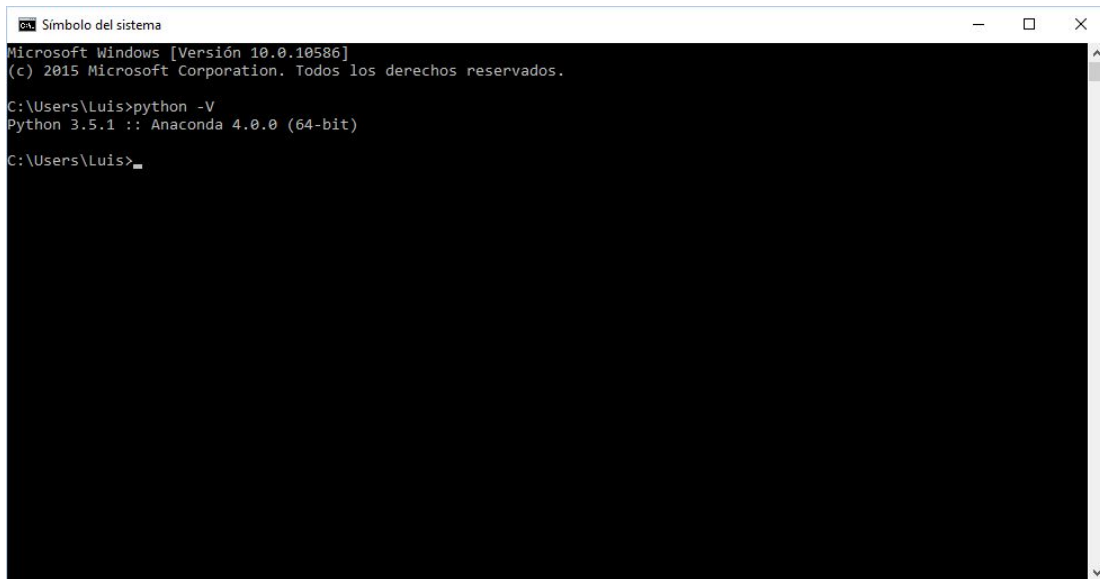
*Aquí te dice si quieres instalar Visual Studio Code. Te recomiendo que lo hagas si no lo tienes instalado.
Es un IDE muy ligero multiplataforma.*



Con esto ya tenemos la primera parte terminada. Para comprobar que todo ha ido bien podemos escribir el siguiente comando en la línea de comandos de Windows.

```
python -V
```

El resultado sería el siguiente.



```
Símbolo del sistema
Microsoft Windows [Versión 10.0.10586]
(c) 2015 Microsoft Corporation. Todos los derechos reservados.

C:\Users\Luis>python -V
Python 3.5.1 :: Anaconda 4.0.0 (64-bit)

C:\Users\Luis>
```

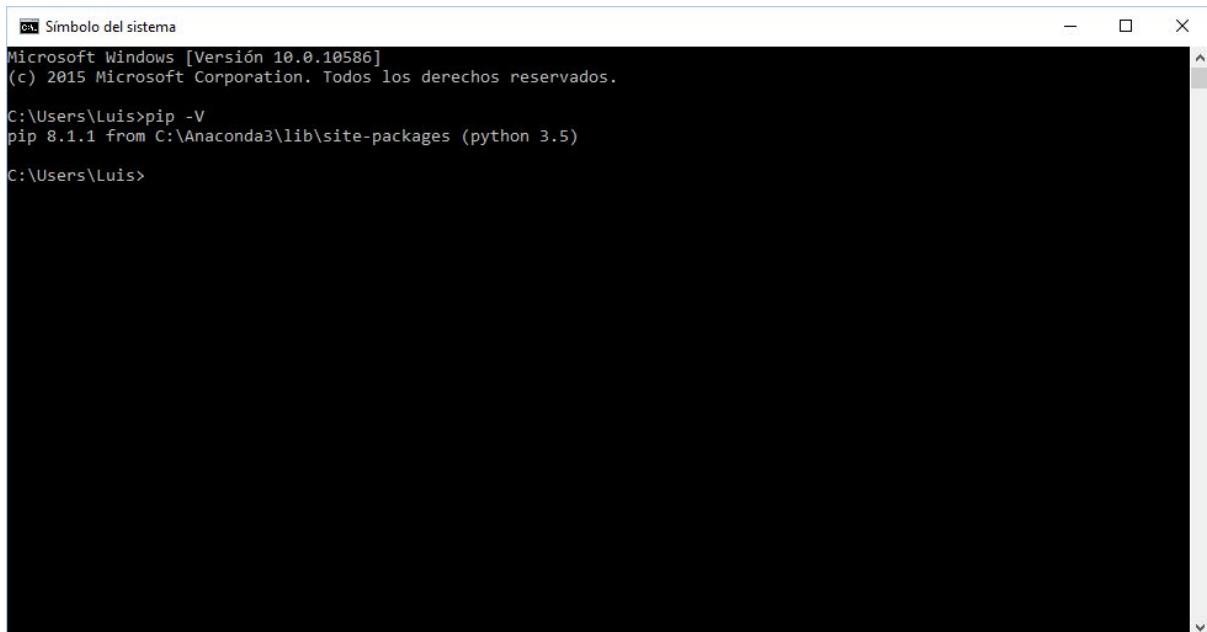
Acostúmbrate a utilizar la línea de comandos, vamos a utilizarla muy a menudo. Te iré enseñando poco a poco los diferentes comandos básicos que vamos a utilizar.

Junto con las bibliotecas ya mencionadas, **Anaconda instala en tu sistema un gestor de paquetes para Python, [Pip](#)**. Se trata de una **herramienta para la instalación de paquetes en Python de una forma muy sencilla**.

La necesitaremos más adelante para instalar OpenCV. Lo primero asegurarnos que está funcionando bien, abre una línea de comandos y escribe el siguiente comando.

```
pip -V
```

El resultado debe ser algo parecido a esto.



```
Símbolo del sistema
Microsoft Windows [Versión 10.0.10586]
(c) 2015 Microsoft Corporation. Todos los derechos reservados.

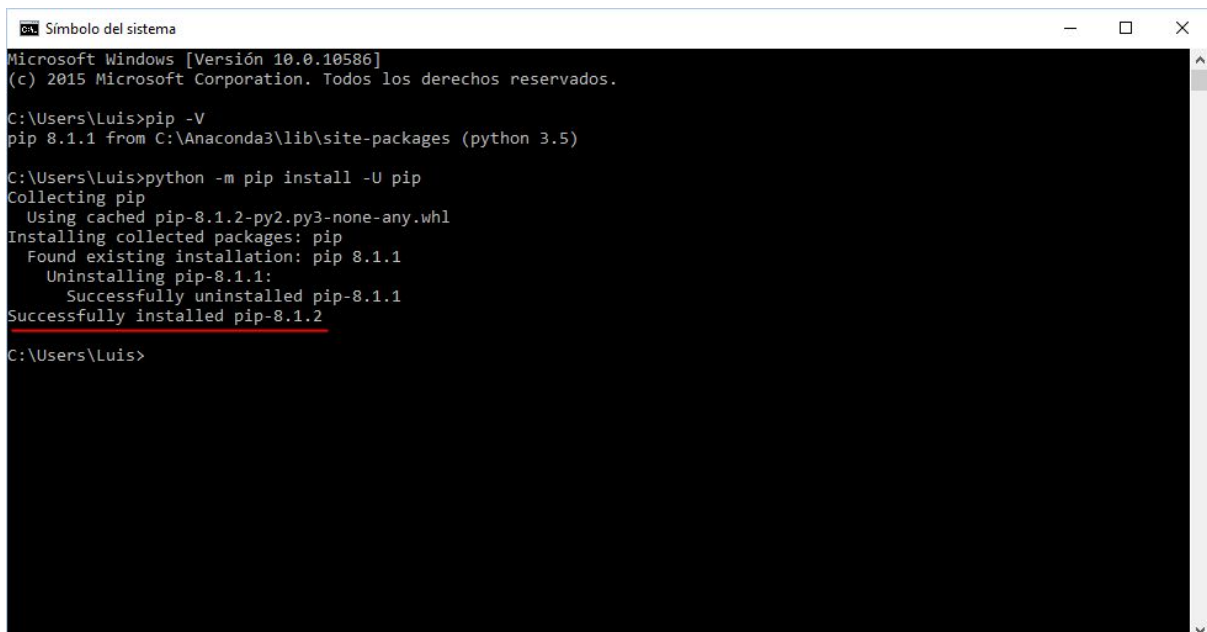
C:\Users\Luis>pip -V
pip 8.1.1 from C:\Anaconda3\lib\site-packages (python 3.5)

C:\Users\Luis>
```

Ahora lo que haremos es comprobar si hay alguna versión nueva. En la misma línea de comandos ejecuta este otro comando.

```
python -m pip install -U pip
```

El resultado que se obtiene en mi caso es el siguiente.



```
Símbolo del sistema
Microsoft Windows [Versión 10.0.10586]
(c) 2015 Microsoft Corporation. Todos los derechos reservados.

C:\Users\Luis>pip -V
pip 8.1.1 from C:\Anaconda3\lib\site-packages (python 3.5)

C:\Users\Luis>python -m pip install -U pip
Collecting pip
  Using cached pip-8.1.2-py2.py3-none-any.whl
Installing collected packages: pip
  Found existing installation: pip 8.1.1
    Uninstalling pip-8.1.1:
      Successfully uninstalled pip-8.1.1
Successfully installed pip-8.1.2

C:\Users\Luis>
```

Como ves, se ha actualizado de la versión 8.1.1 a la versión 8.1.2 y sin bajar ningún instalador, simplemente ejecutando un comando. Así lo haremos con el paquete más importante, OpenCV.

1.2 Instalar OpenCV para Python 3

Ahora vamos a instalar el paquete principal de esta lección, OpenCV. Como ya tenemos todo preparado, no va a resultar muy difícil aunque dependerá del sistema operativo que estés utilizando y la versión (32-bit o 64bit).

1.2.1 Descargar el archivo

[Accede a la web donde podrás descargar un archivo whl](#). En esta web aparecerán diferentes versiones de OpenCV según la versión de Python que tengas instalada.

```
opencv_python-3.1.0-cp34-cp34m-win32.whl
opencv_python-3.1.0-cp34-cp34m-win_amd64.whl
opencv_python-3.4.1+contrib-cp35-cp35m-win32.whl
opencv_python-3.4.1+contrib-cp35-cp35m-win_amd64.whl
opencv_python-3.4.1+contrib-cp36-cp36m-win32.whl
opencv_python-3.4.1+contrib-cp36-cp36m-win_amd64.whl
opencv_python-3.4.1+contrib-cp37-cp37m-win32.whl
opencv_python-3.4.1+contrib-cp37-cp37m-win_amd64.whl
opencv_python-3.4.1-cp35-cp35m-win32.whl
opencv_python-3.4.1-cp35-cp35m-win_amd64.whl
```

Debemos de centrarnos en las que pone la versión 3.4.1 de OpenCV que es la última. Dentro de las diferentes versiones elegimos la que pone *+contrib* y la parte del nombre del archivo donde pone cpXX, indica la versión de Python.

Por ejemplo, *cp35* es la versión Python 3.5, *cp36* es la versión Python 3.6 y así sucesivamente.

La última parte indica si el sistema operativo es 32-bit (win32) o 64-bit (amd64).

<u>opencv_python-3.4.1+contrib-cp36-cp36m-win_amd64.whl</u>		
Versión OpenCV	Versión Python	Versión Windows

Conociendo la versión de Python (recuerda el comando `python -V`) y sabiendo la arquitectura de nuestro ordenador, ya podemos elegir el archivo a descargar.

Por ejemplo, yo voy a descargar el siguiente archivo
`opencv_python-3.4.1+contrib-cp36-cp36m-win_amd64.whl`

Es la versión 3.4.1 de OpenCV, utiliza Python 3.6 y la versión de 64-bit.
Descárgalo a tu máquina y en el siguiente paso vamos a instalarlo.

1.2.2 Instalando OpenCV

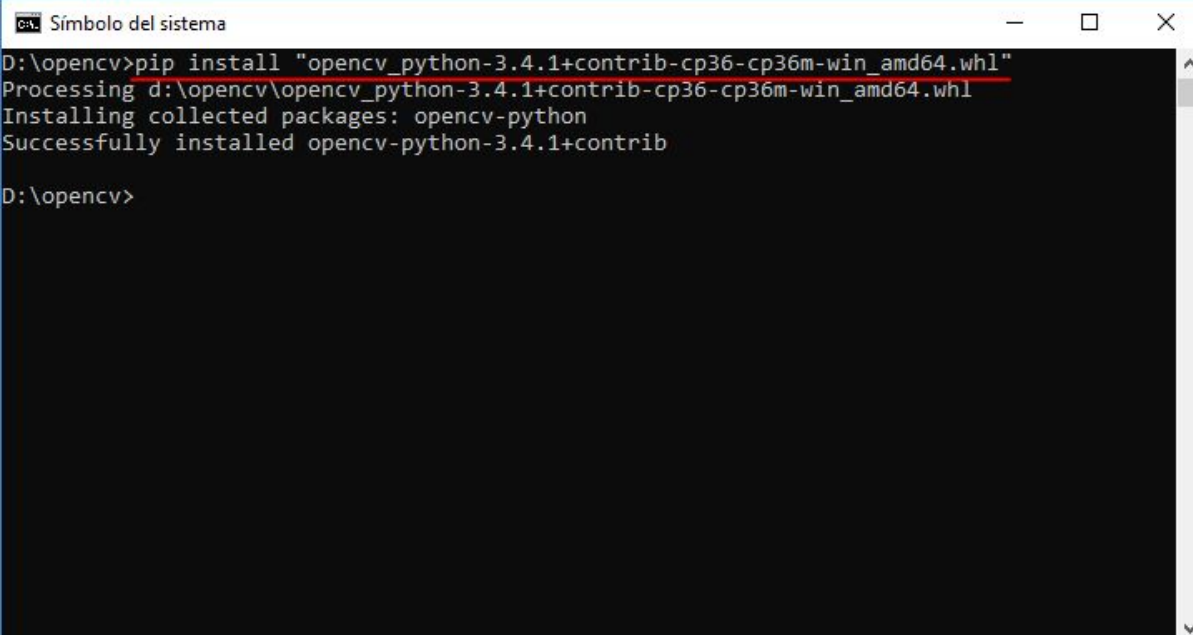
Lo siguiente es instalar el paquete de instalación que nos acabamos de bajar a nuestra máquina.

Abre una línea de comandos y vete a la carpeta donde lo has descargado. Utiliza el comando `cd` para moverte entre carpetas.

Una vez estés en la misma carpeta del archivo escribe el siguiente comando

```
pip install opencv_python-3.4.1+contrib-cp36-cp36m-win_amd64.whl
```

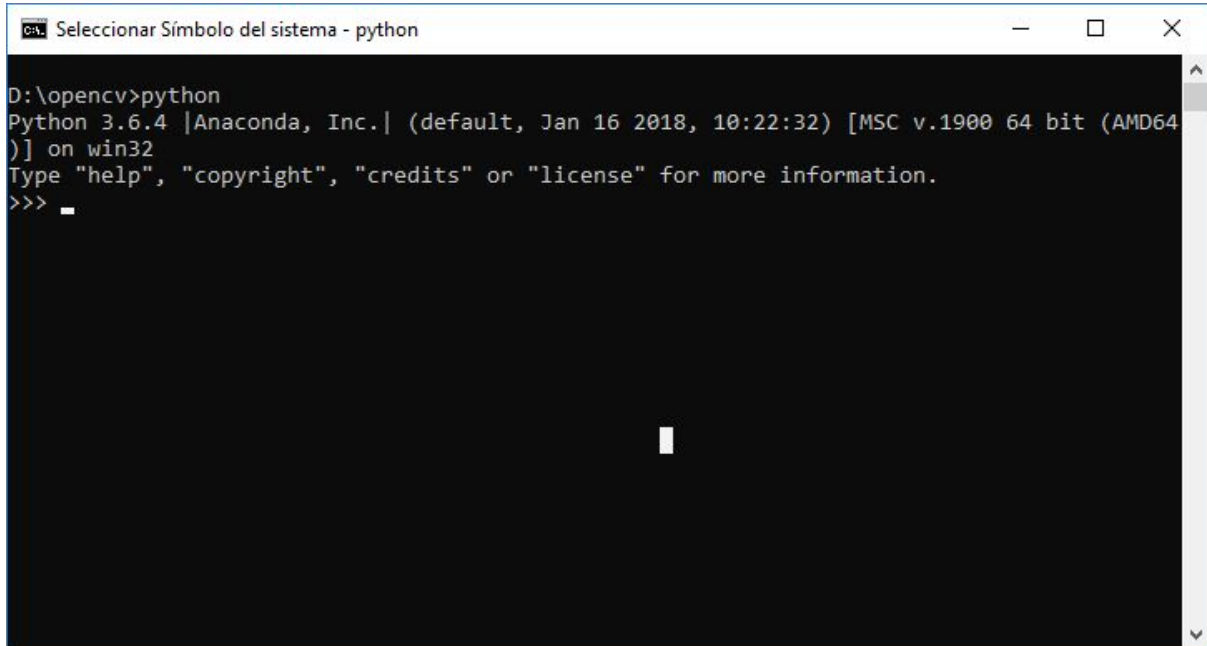
Debes sustituir el nombre del archivo por el que te has descargado tu. Recuerda que tienes que estar en la misma carpeta donde has descargado el archivo.



```
C:\> Símbolo del sistema
D:\opencv>pip install "opencv_python-3.4.1+contrib-cp36-cp36m-win_amd64.whl"
Processing d:\opencv\opencv_python-3.4.1+contrib-cp36-cp36m-win_amd64.whl
Installing collected packages: opencv-python
Successfully installed opencv-python-3.4.1+contrib
D:\opencv>
```

Con esto ya tendríamos instalado OpenCV. Podemos hacer la primera prueba. Abre una nueva línea de comandos y escribe el siguiente comando.

```
python
```

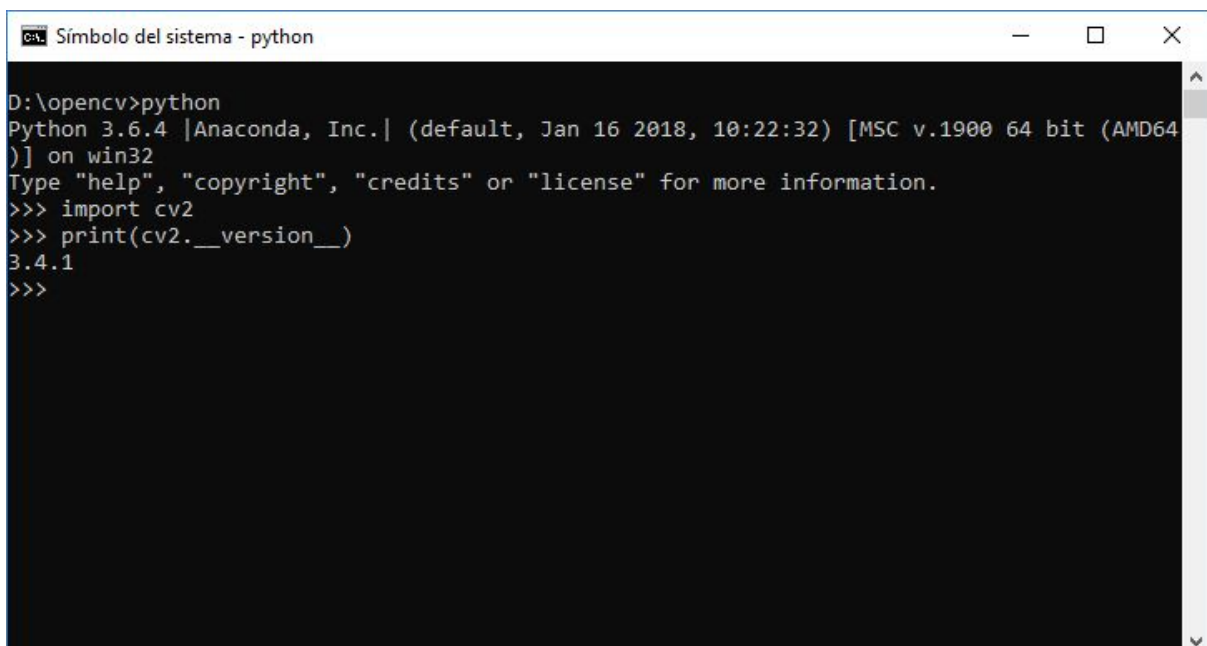


```
Selecc... Símbolo del sistema - python
D:\opencv>python
Python 3.6.4 |Anaconda, Inc.| (default, Jan 16 2018, 10:22:32) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> _
```

Esto abre el intérprete de Python. Ahora vamos a probar el primer programa con OpenCV. Escribe lo siguiente

```
import cv2
print(cv2.__version__)
```

Obtendrás un resultado parecido a la siguiente imagen donde te dice la versión de OpenCV.



```
Símbolo del sistema - python
D:\opencv>python
Python 3.6.4 |Anaconda, Inc.| (default, Jan 16 2018, 10:22:32) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import cv2
>>> print(cv2.__version__)
3.4.1
>>>
```


Todo funciona correctamente así que ya podemos pasar al último paso, donde instalaremos Sublime Text 3 para programar con OpenCV a través de este IDE.

1.3 Instalar Sublime Text 3 y el paquete para Python Anaconda

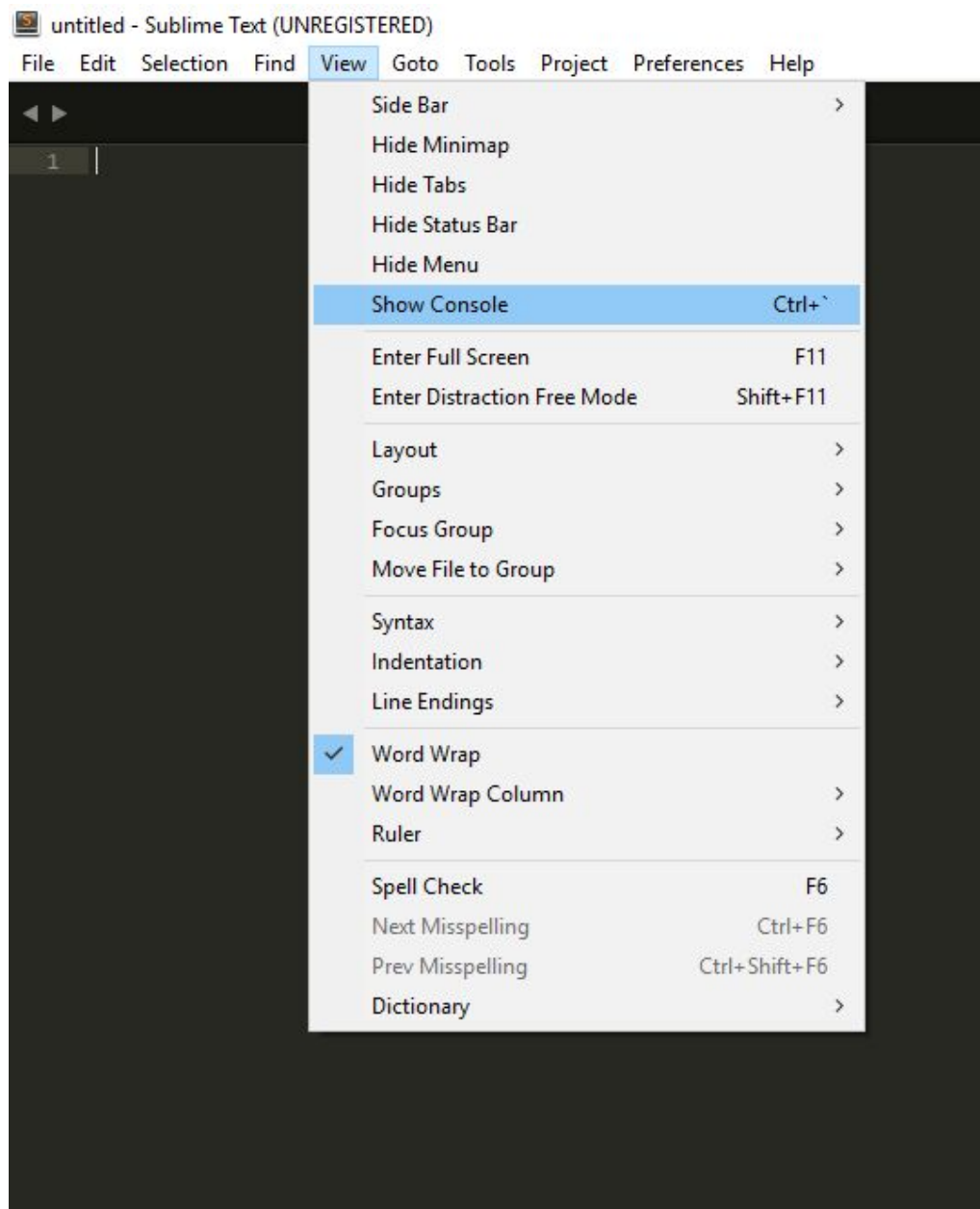
Ya hemos llegado al último paso, instalar el entorno de desarrollo. Este paso es opcional, no necesitamos instalar nada para empezar a programar, con el bloc de notas de Windows podemos empezar.

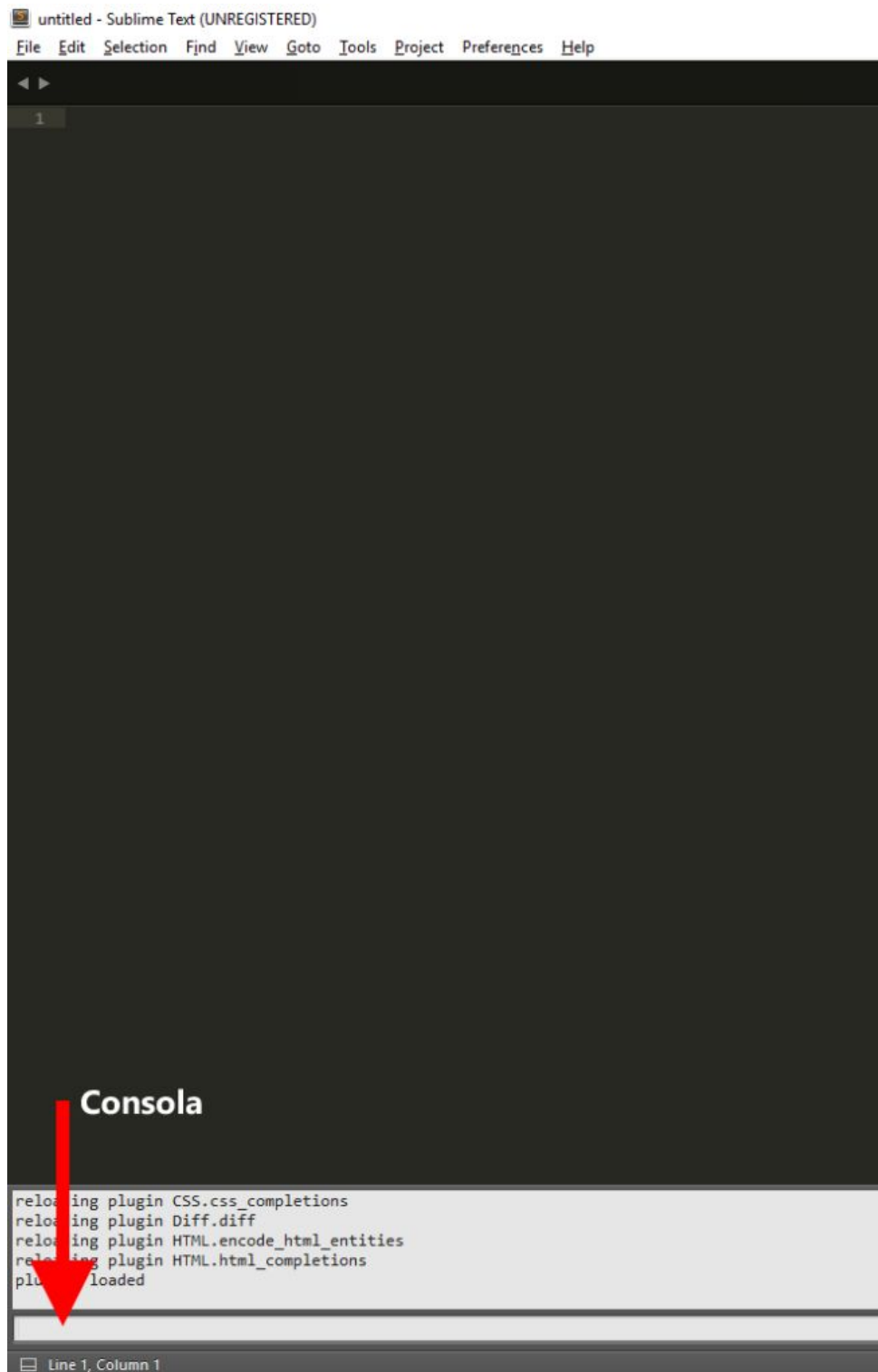
Si que es recomendable instalar un entorno de desarrollo y el que yo recomiendo es Sublime Text 3 con el plugin Anaconda, que convierte a este entorno en un IDE para Python, con todas las funcionalidades.

Descarga la [versión 3 de Sublime Text](#) que, aunque sea una versión beta, es bastante estable.

Una vez descargado hay que habilitar el gestor de paquetes, sigue las siguientes instrucciones.

Abre la consola en *View>Show Console*.





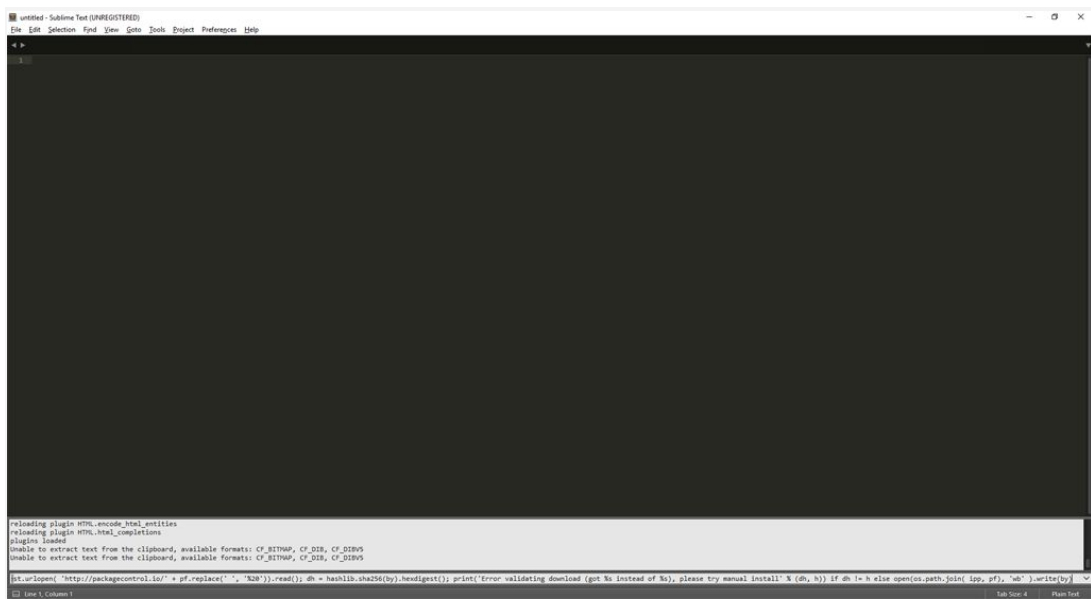
Pega uno de los siguientes código y pulsa el enter.

```
import urllib.request,os,hashlib; h = 'df21e130d211cfc94d9b0905775a7c0f' +  
'1e3d39e33b79698005270310898eea76'; pf = 'Package Control.sublime-package'; ipp =  
sublime.installed_packages_path(); urllib.request.install_opener( urllib.request.build_opener(  
urllib.request.ProxyHandler()) ); by = urllib.request.urlopen( 'http://packagecontrol.io/' + pf.replace(' ',
```

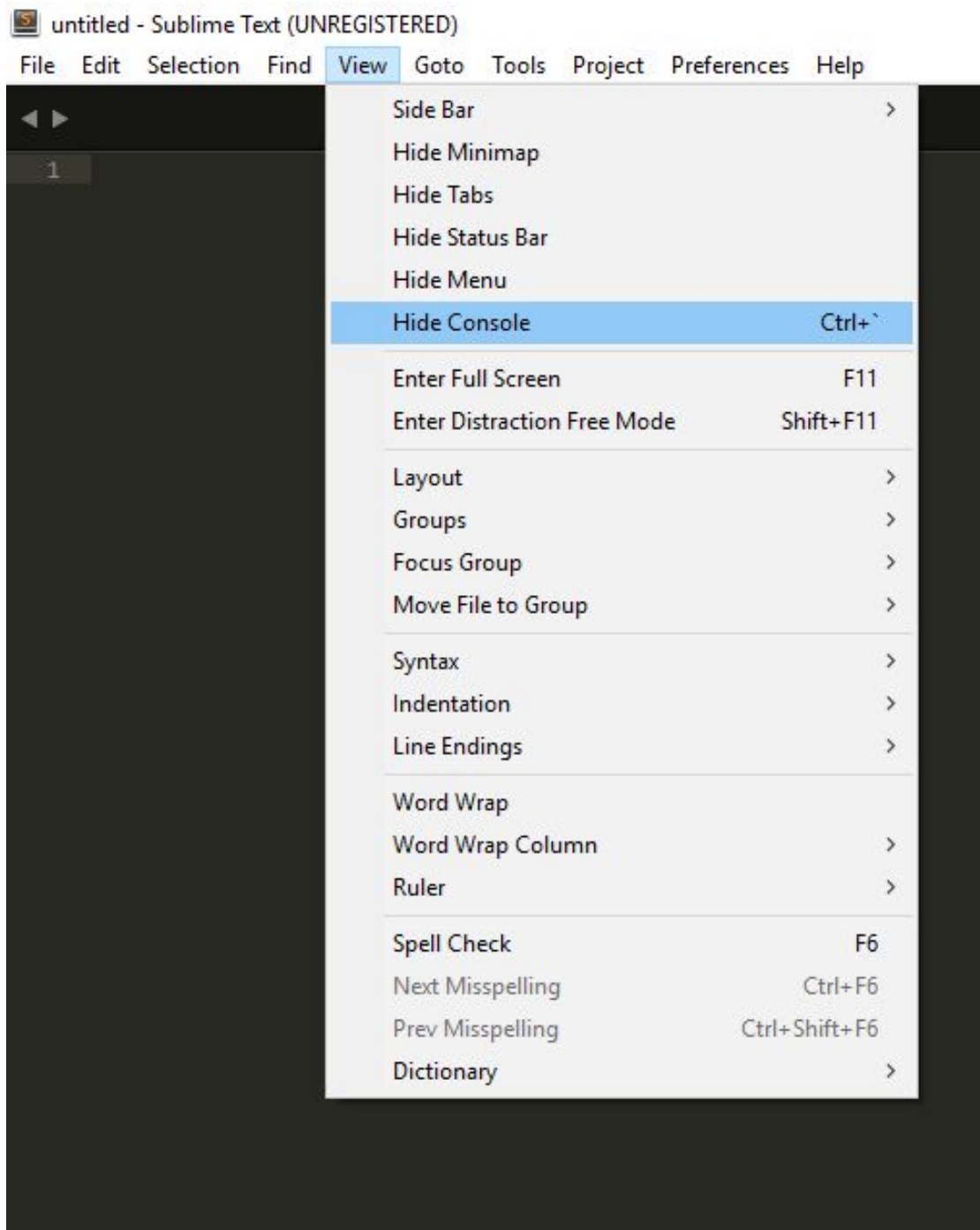
```
'%20')).read(); dh = hashlib.sha256(by).hexdigest(); print('Error validating download (got %s instead of %s), please try manual install' % (dh, h)) if dh != h else open(os.path.join( ipp, pf), 'wb' ).write(by)
```

Código antiguo

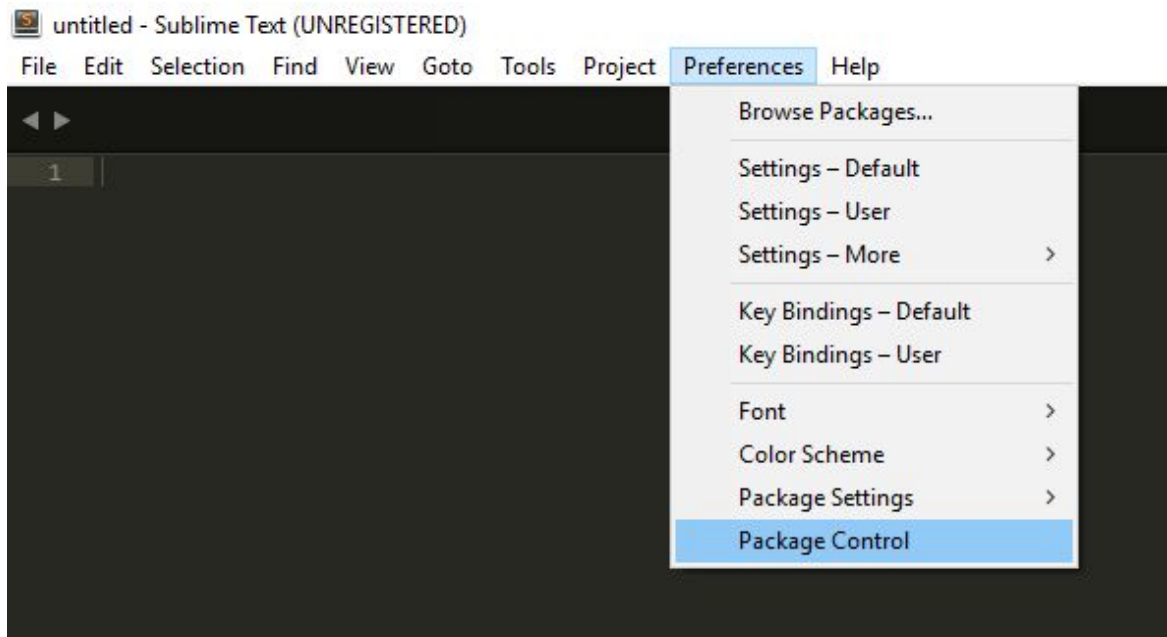
```
import urllib.request,os,hashlib; h = '2915d1851351e5ee549c20394736b442' +  
'8bc59f460fa1548d1514676163dafc88'; pf = 'Package Control.sublime-package'; ipp =  
sublime.installed_packages_path(); urllib.request.install_opener( urllib.request.build_opener(  
urllib.request.ProxyHandler()) ); by = urllib.request.urlopen( 'http://packagecontrol.io/' + pf.replace(' ',  
'%20')).read(); dh = hashlib.sha256(by).hexdigest(); print('Error validating download (got %s instead of %s), please try manual install' % (dh, h)) if dh != h else open(os.path.join( ipp, pf), 'wb' ).write(by)
```



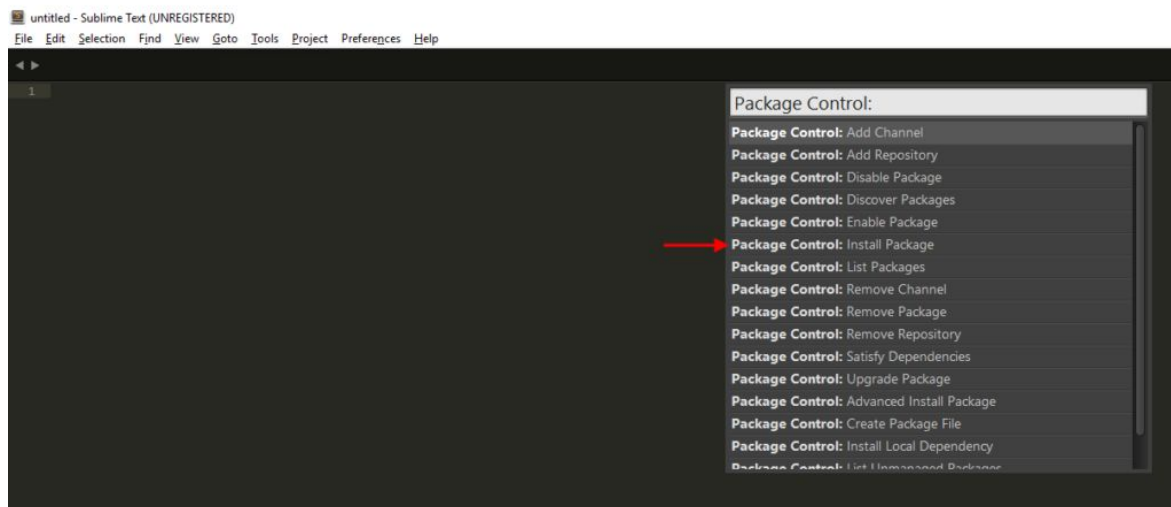
Una vez instalado el gestor de paquetes puedes ocultar la consola en *View>Hide Console*.



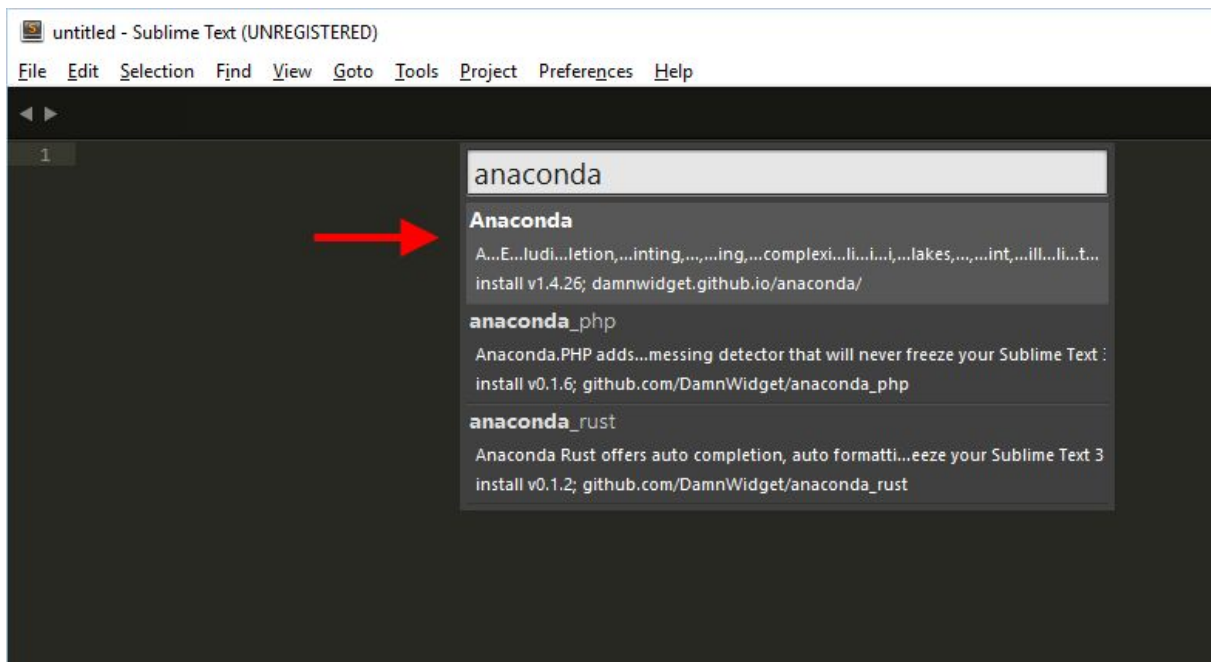
Abre el control de paquetes en *Preferences>Package Control*.



Y haz click en *Install Package*.



En el buscador que aparece escribe Anaconda y selecciona el primer paquete que aparece haciendo click.



Con esto ya tenemos instalado todo lo necesario para utilizar Sublime Text como nuestro editor de Python. Te aconsejo que reinicies el IDE.

Solo nos quedaría probar toda la instalación pero esto lo veremos en el siguiente tema donde empezaremos a programar con Python y la visión artificial.

2 Trabajando con imágenes con OpenCV y Python

En la lección anterior hemos dejado preparado el entorno de desarrollo para empezar a programar y trabajar. Lo primero que haremos en este tema será comprobar que todo está bien y luego veremos las acciones básicas **cargar, ver y guardar imágenes con OpenCV**.

Este tutorial es una manera práctica de introducirse en la visión artificial con OpenCV y Python. Por lo tanto no vamos a perder el tiempo en explicaciones teóricas así que vamos a empezar escribiendo el código necesario para poder cargar una imagen, mostrarla por la pantalla y guardarla en un archivo con diferente formato.

Todo el código necesario para seguir el tema te lo iré dejando según lo vayamos necesitando y te daré las explicaciones necesarias de lo que hace cada línea.

Los pasos que vamos a seguir son los siguientes:

- Probar que la instalación de OpenCV y Python ha salido bien.
- Cargar una imagen desde el disco duro del ordenador.
- Mostrar la imagen por pantalla.
- Guardar la imagen en un formato diferente al original de la imagen.

Antes de empezar un simple recordatorio, vamos a trabajar con el Sublime Text 3 así que si no lo tienes instalado hazlo, no es imprescindible pero sí que es muy recomendable.

2.1 Probar instalación OpenCV y Python

Comenzaremos haciendo un programa muy sencillo para comprobar que todo está en bien instalado.

Abre un nuevo documento en Sublime Text 3 y copia el siguiente código.

```
import cv2
import numpy as np
import scipy as sc
from matplotlib import pyplot as plt
```

Guárdalo con extensión .py por ejemplo con el nombre, ***mi-primer-programa.py***.

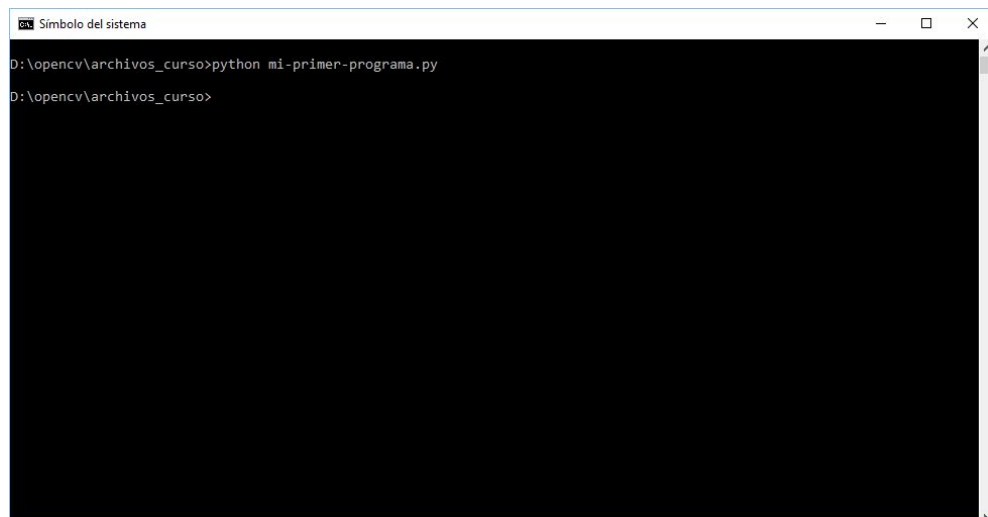
Las cuatro líneas de código anteriores permiten importar las librerías de OpenCV (cv2), NumPy (numpy), SciPy (scipy) y Matplotlib (matplotlib).

La que más vamos a utilizar es OpenCV (cv2).

Abre una línea de comandos, accede a la carpeta donde lo has guardado y ejecuta el siguiente comando.

```
python mi-primer-programa.py
```

Si todo ha ido bien en la instalación, no aparecerá ningún error. Esto te garantiza que todo está correcto.



2.2 Cargar una imagen desde el disco duro

Para poder trabajar con imágenes, tenemos que cargarlas desde el disco duro. Eso es lo primero que haremos.

Ten preparada alguna imagen. Por ejemplo, yo voy a utilizar esta.



Puedes utilizar cualquier otra. La debes guardar en una carpeta en el mismo directorio donde tienes el archivo con extensión .py.

Yo lo que hago es crearme una carpeta de *imagenes* y dentro voy guardando todas las imágenes.

Ahora crea un nuevo documento en Sublime Text 3 y copia el siguiente código.

```
import cv2

# Cargamos la imagen del disco duro
imagen = cv2.imread("imagenes/primera-imagen.jpg")

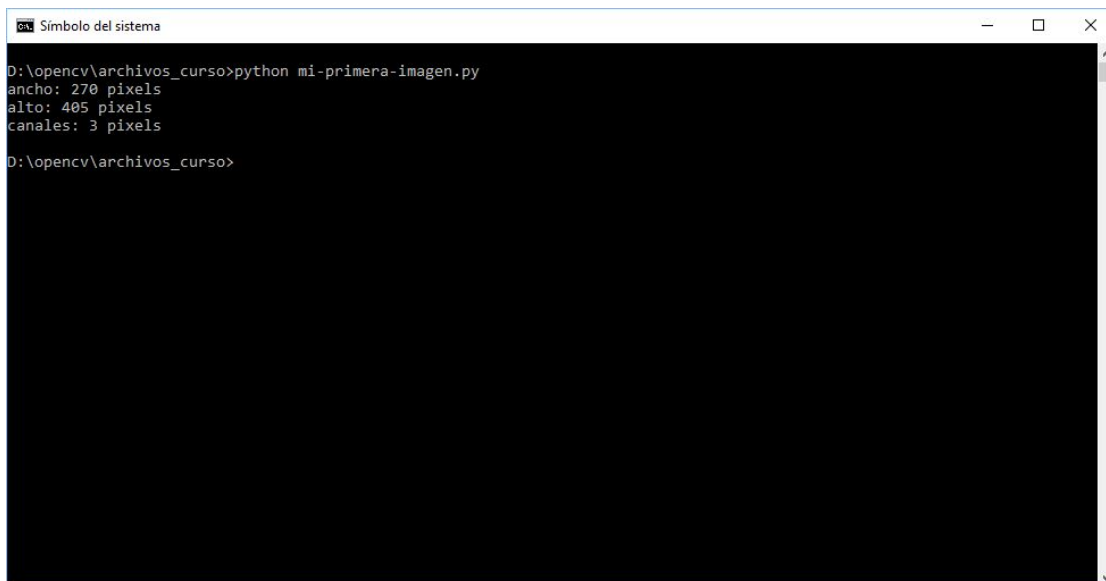
# Mostramos ancho, alto y canales
print("ancho: {} pixels".format(imagen.shape[1]))
print("alto: {} pixels".format(imagen.shape[0]))
print("canales: {} pixels".format(imagen.shape[2]))
```

Lo puedes guardar con el nombre que quieras por ejemplo ***mi-primera-imagen.py***. Es muy importante la extensión .py.

Ejecuta en la línea de comandos el siguiente código

```
python mi-primera-imagen.py
```

El resultado variará según el tamaño de tu imagen.



```
Símbolo del sistema
D:\opencv\archivos_curso>python mi-primer-a-imagen.py
ancho: 270 pixels
alto: 405 pixels
canales: 3 pixels
D:\opencv\archivos_curso>
```

En mi caso se trata de una imagen de 270 píxeles de ancho, 405 de alto y 3 canales de color.

Para cargar la imagen utilizamos el comando `cv2.imread("archivo")` donde archivo es la ruta donde está guardada la imagen.

Esto devuelve una variable del tipo imagen que tiene un array (shape) con tres elementos. El primer elemento es el alto, el segundo el elemento es el ancho y el tercer elemento son los canales de color.

Este último varía por ejemplo, para un espacio de color RGB hay 3 canales pero para una imagen en escala de grises hay 1 canal.

Para acceder a estos datos lo único que tenemos que hacer es `imagen.shape[elemento]` donde elemento puede tomar un valor del 0 al 2.

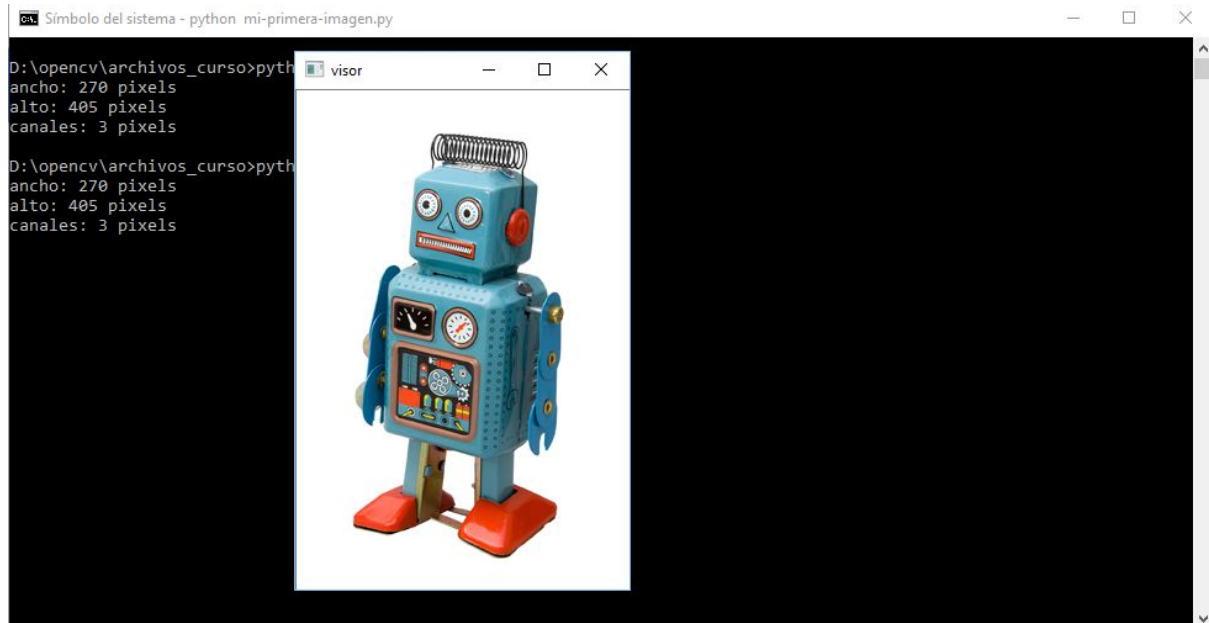
2.3 Mostrar la imagen por pantalla

Para mostrar la imagen en la pantalla, solo debe añadir el siguiente código al programa anterior.

```
# Mostramos la imagen en la pantalla
cv2.imshow("visor", imagen)
cv2.waitKey(0)
```

El método `cv2.imshow("visor", imagen)` muestra la imagen a través de una ventana. La última sentencia permite que la ventana se mantenga abierta hasta que pulsemos una tecla.

Vuelve a ejecutar el archivo **mi-primer-a-imagen.py** a través de la línea de comandos y verás el resultado.



Tachán !!!! la imagen aparece en una ventana. Sencillo ¿no?

2.4 Guardar la imagen en el disco duro

Por último vamos a guardar una copia de la imagen en el disco duro. Copia el siguiente código después de mostrar la imagen.

```
# Guardar la imagen con otro nombre y otro formato
# Solo hay que poner la extensión del formato que queramos guardar
cv2.imwrite("imagenes/nueva-imagen.png", imagen)
```

El método `cv2.imwrite(param1, param2)` admite dos parámetros. El primero es la ruta donde quieres guardar la nueva imagen. Si pones un nombre de imagen que exista la sobrescribirá.

El segundo parámetro es la variable que contiene la imagen.

Debes fijarte en la extensión. Hemos abierto un archivo JPG y vamos a guardar una archivo PNG. Con solo cambiar la extensión automáticamente se hace la conversión entre formatos.

3 Fundamentos de imágenes digitales

En este tema vamos a entrar en materia, vamos a ver los **fundamentos de la imagen digital**. El componente básico con el que vamos a trabajar es el píxel.

Seguro que has oído hablar un montón de veces de ellos, sobre todo cuando [se habla de dispositivos de adquisición de imágenes](#) como las cámaras de los móviles o las fotográficas.

Además veremos el sistema de coordenadas que nos ayudará a localizar la posición de cada píxel en la imagen y por último veremos cómo acceder a los píxeles para obtener o modificar su contenido.

3.1 Píxeles en una imagen

¿Qué es un píxel? Un píxel no es más que el elemento más pequeño, **la unidad mínima de una imagen**. Cada imagen consta de un conjunto de píxeles. Cada uno de ellos **nos dará información sobre el color o la intensidad** del sitio donde esté colocado.

Imagina una gran rejilla de celdas cuadradas, cada celda representaría un píxel. Por lo tanto, si tenemos una imagen de 300 píxeles de ancho y 300 píxeles de alto, **esto sería la resolución**, tendríamos una rejilla con 300 filas y 300 columnas.

Para saber el número de píxeles solo tendremos que multiplicar el ancho por el alto.
 $300 \times 300 = 90.000$ píxeles tendrá nuestra imagen.

Si ampliamos una imagen en algún software de edición, podremos ver con claridad los píxeles y la información contenida en cada uno de ellos.



La mayoría de píxeles **se representan de dos formas, en escala de grises o en componentes de color.**

En **escala de grises**, cada píxel tiene **un valor entre 0 y 255**. El valor 0 representa el negro y el valor 255 representa el blanco. Los valores que hay entre medias representan diferentes tonalidades de gris. Cuanto más cerca del cero, más oscuro será el gris y cuanto más cerca del 255 más claro será el gris.

Los píxeles de color normalmente se representan en el **espacio de color RGB** (Red Green Blue - Rojo Verde Azul). Esto quiere decir que **cada píxel tendrá 3 valores**, un valor para la componente roja, un valor para la componente verde y un valor para la componente azul.

Existen otros espacios de color, pero empezaremos por el básico e iremos avanzando a partir de ahí.

Cada componente de color está representado por un entero entre el 0 y el 255 lo que nos indica cuánta cantidad de color tiene ese componente. Dado que los píxeles solo necesitan un rango entre 0 y 255, solo será necesario utilizar variables del tipo entero (int) de 8-bit sin signo para representar la intensidad de cada color.

Para obtener el **color final**, debemos combinar los tres valores RGB es decir, por separado cada componente nos dirá qué cantidad tiene de rojo, verde y azul. Si combinamos estos tres colores obtendremos el color final. Por lo tanto, **cada píxel tendrá asociado tres valores (rojo, verde, azul).**

Vamos a ver dos ejemplos muy simples, obtener el color blanco y el negro.

Para obtener el color blanco debemos tener el máximo valor en cada componente RGB, 255. El blanco se representaría con (255, 255, 255).

Para obtener el color negro debemos tener el mínimo valor en cada componente RGB, 0. El negro se representaría con (0, 0, 0).

Si queremos obtener un rojo puro solo tenemos que poner la componente roja al máximo y las otras dos al mínimo, (255, 0, 0).

Veamos la representación que tendrían unos cuantos colores comunes en el espacio de color RGB:

- Negro (0, 0, 0)

- Blanco (255, 255, 255)
- Rojo (255, 0, 0)
- Verde (0, 255, 0)
- Azul (0, 0, 255)
- Agua (0, 255, 255)
- Fucsia (255, 0, 255)
- Marrón (128, 0, 0)
- Navy (0, 0, 128)
- Oliva (128, 128, 0)
- Amarillo (255, 255, 0)

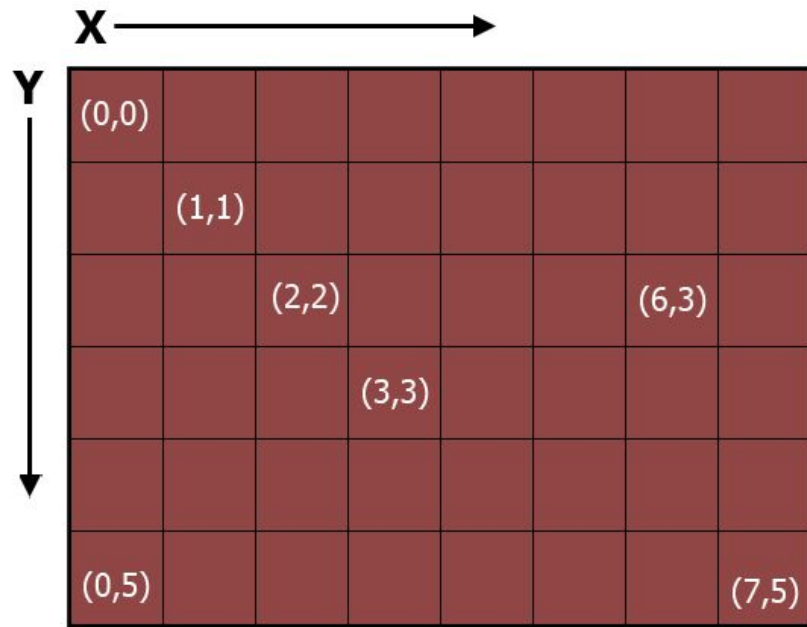
Ahora que conocemos el componente más básico que tiene una imagen, el píxel, vamos a ver el sistema de coordenadas para localizar cada píxel en una imagen.

3.2 Sistema de coordenadas

Como ya he mencionado con anterioridad, una imagen se representa como una rejilla de píxeles donde tendremos filas y columnas.

Al contrario de la geometría a la que estamos acostumbrados, donde el origen de coordenadas, el punto (0, 0), está situado en la parte inferior izquierda, en las imágenes este origen está situado en la parte superior izquierda. El eje horizontal es el eje de las X y el eje vertical es el eje de las Y.

Si nos desplazamos en diagonal hacia la esquina inferior derecha, aumentará la coordenada X y la coordenada Y. Para que quede más claro, mira la siguiente imagen.



Si queremos acceder a cualquier píxel de una imagen, debemos conocer su posición. Por ejemplo el píxel (0, 5) está situado en la columna 0 (X) y en la fila 5 (Y).

Por lo tanto **estamos accediendo a los valores contenidos dentro de una matriz**. Como ya te he comentado, una imagen no es más que una rejilla o **matriz donde se encuentran todos los píxeles y a los que podemos acceder con la coordenada X e Y correspondientes**.

Una última aclaración con respecto al sistema de coordenadas y el acceso desde **Python**. Este **lenguaje de programación es indexado a cero** es decir, siempre empezará a contar desde cero. El **primer elemento de un array o de una matriz será el cero**. Es importante que lo recuerdes así te evitarás muchas confusiones después.

3.3 Acceder y manipular los píxeles de una imagen

Ahora ha llegado la hora de la verdad, todo parte de aquí. En el anterior tema hemos visto un ejemplo muy sencillo y poco emocionante. En el siguiente ejemplo de código vamos a ver como acceder y manipular una imagen.

Antes de meternos de lleno en el código hay que tener en cuenta unas consideraciones.

Lo primero es recordarte que **OpenCV representa las imágenes como arrays de NumPy**. Para que quede claro, podemos pensar en una representación de una matriz como hemos visto anteriormente.

Para acceder a un píxel solo tenemos que saber la coordenada X e Y donde está situado. **Como resultado tendremos un trío de valores que representan las componentes de color RGB (Rojo, Verde y Azul).**

Algo muy importante es que **OpenCV almacena las componentes RGB en orden inverso es decir, BGR (Azul, Verde y Rojo).** Es muy importante recordar esto para que no nos genere confusión más tarde.

Vamos a partir del siguiente código que ya conoces del tema anterior donde cargamos y mostramos una imagen.

```
import cv2

# Cargamos la imagen del disco duro
imagen = cv2.imread("imagenes/robot-vision-artificial.jpg")

# Mostramos ancho, alto y canales
print("ancho: {} pixels".format(imagen.shape[1]))
print("alto: {} pixels".format(imagen.shape[0]))
print("canales: {} pixels".format(imagen.shape[2]))

# Mostramos la imagen en la pantalla
cv2.imshow("visor", imagen)
cv2.waitKey(0)
```

Para obtener el píxel de una determinada posición por ejemplo la (0,0), debemos escribir la siguiente sentencia.

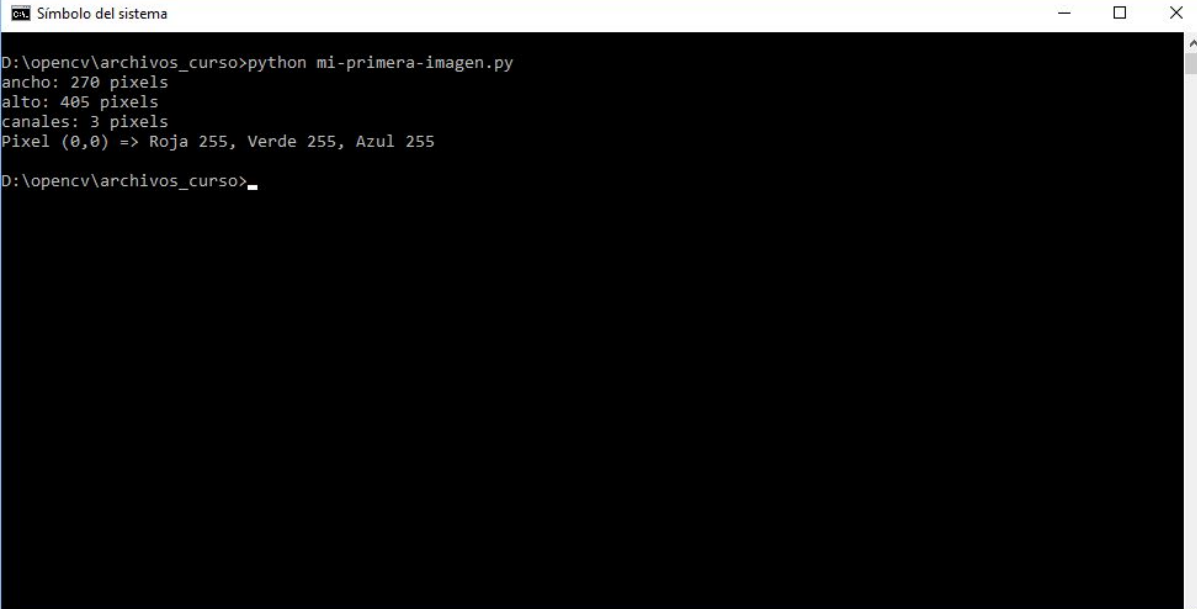
```
# Obtener el pixel (0,0)
(b, g, r) = imagen[0,0]
print("Pixel (0,0) => Roja {}, Verde {}, Azul {}".format(r,g,b))
```

Como la imagen es a color, esto nos devolverá un array con 3 valores uno para cada componente de color.

Si vuelves a ejecutar el programa con el comando en la línea de comandos.

```
python mi-primer-a-imagen.py
```

Obtendrás el siguiente resultado



```
Símbolo del sistema
D:\opencv\archivos_curso>python mi-primer-a-imagen.py
ancho: 270 pixels
alto: 405 pixels
canales: 3 pixels
Pixel (0,0) => Roja 255, Verde 255, Azul 255
D:\opencv\archivos_curso>
```

Al tratarse de un píxel blanco, las 3 componentes de color tienen el valor 255. Así de sencillo es acceder a un píxel.

Por el contrario si lo que queremos es cambiar el valor del píxel, tendremos que igualar ese píxel a un array con 3 componentes de color.

```
# Cambiamos el color del pixel (0,0)
imagen[0,0] = (0,0,255)
(b, g, r) = imagen[0,0]
print("Pixel (0,0) => Roja {}, Verde {}, Azul {}".format(r,g,b))
```

El resultado tras su ejecución sería el siguiente.

```
Símbolo del sistema
D:\opencv\archivos_curso>python mi-primer-a-imagen.py
ancho: 270 pixels
alto: 405 pixels
canales: 3 pixels
Pixel (0,0) => Roja 255, Verde 255, Azul 255
Pixel (0,0) => Roja 255, Verde 0, Azul 0
D:\opencv\archivos_curso>
```

En este caso puedes comprobar como el valor del píxel ha pasado de ser blanco (255, 255, 255) a rojo puro (255, 0, 0).

Si muestras la imagen y haces zoom verás como ha cambiado de color.

3.4 Accediendo y manipulando una región de la imagen

Acceder a un píxel está bien pero no es nada práctico. Imagínate que quieres cambiar el color de una región de la imagen.

Gracias a la librería NumPy que viene dentro del paquete Anaconda, esto es muy sencillo.

Vamos a comenzar con un nuevo archivo. Lo vamos a llamar **region-imagen.py**.

Copia el siguiente código.

```
import cv2

# Cargamos la imagen
imagen = cv2.imread("imagenes/robot-vision-artificial.jpg")

# Obtener region cuadrada
region = imagen[0:150,0:150]
cv2.imshow("region", region)
cv2.waitKey(0)
```

La primera parte ya la conoces, importar la librería OpenCV (cv2) y cargar la imagen con el método cv2.imread().

Para obtener una región sólo tenemos que acceder a través de unos índices.

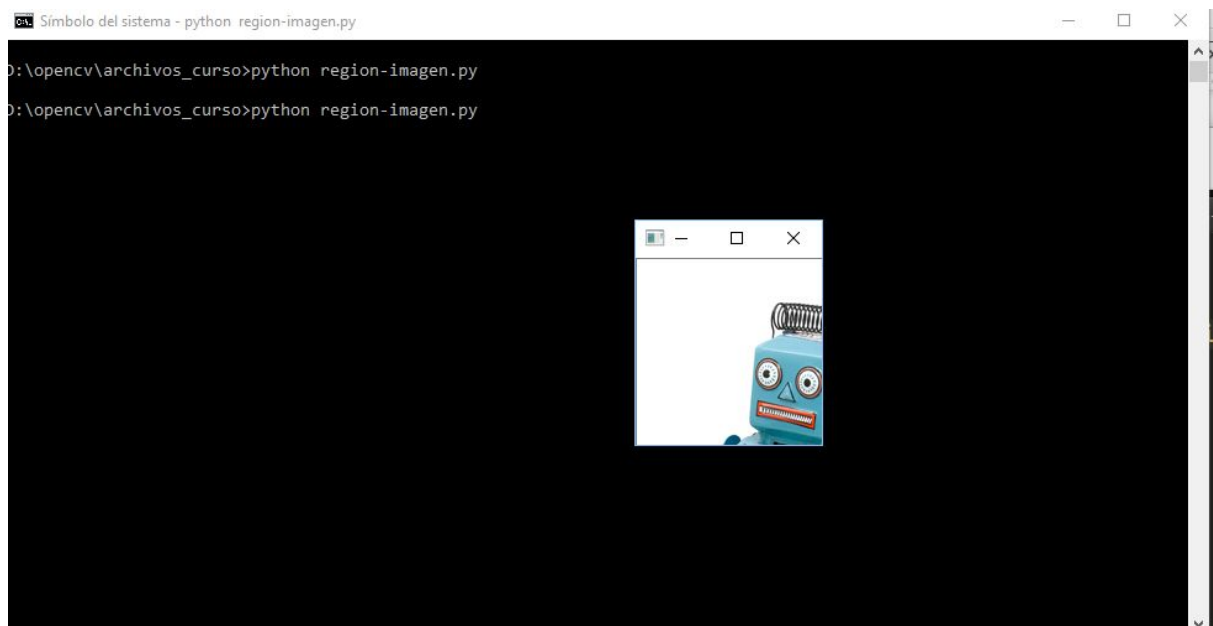
```
imagen[min:max, min:max]
```

Los dos primeros parámetros min:max indican la coordenada x mínima y máxima.

Los dos siguientes parámetros min:max indican la coordenada y mínima y máxima.

Esto nos devuelve una región en forma de array, donde cada elemento tiene 3 componentes, las componentes de color RGB.

Por lo tanto es una imagen obtenida de la imagen original. Si ejecutas el código obtendrás un resultado como este.

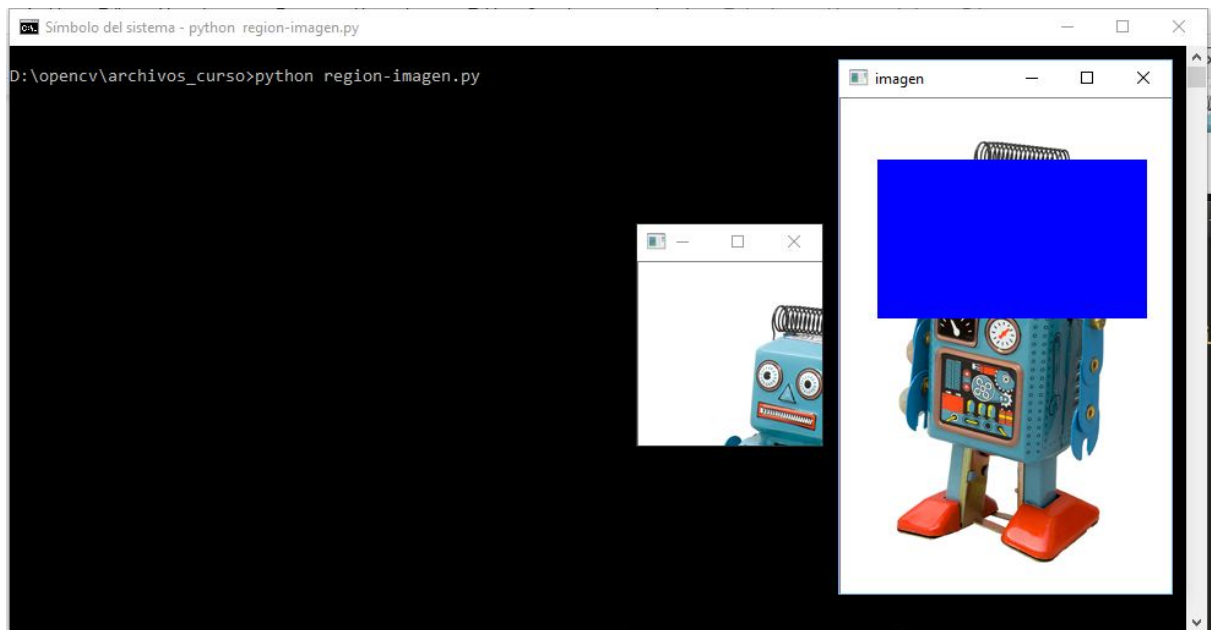


Si ahora lo que quieres es cambiar de color una región, lo único que tienes que hacer es igualar la región a un array con las 3 componentes de color.

Copia el siguiente código a continuación.

```
# Cambiar el color de una región
imagen[50:180,30:250]=(255,0,0)
cv2.imshow("imagen", imagen)
cv2.waitKey(0)
```

El resultado es el siguiente.



Y con esto hemos terminado esta Introducción a la Visión Artificial con OpenCV y Python. Esta es la base para comenzar a crear tus propios algoritmos y programas de Visión Artificial.

Todavía queda mucho camino por recorrer a través de diferentes técnicas y métodos del tratamiento digital de imágenes.

Cualquier duda ponte en contacto conmigo en ldelvalleh@programarfacil.com.