

Procesado de imagen y visión por computador

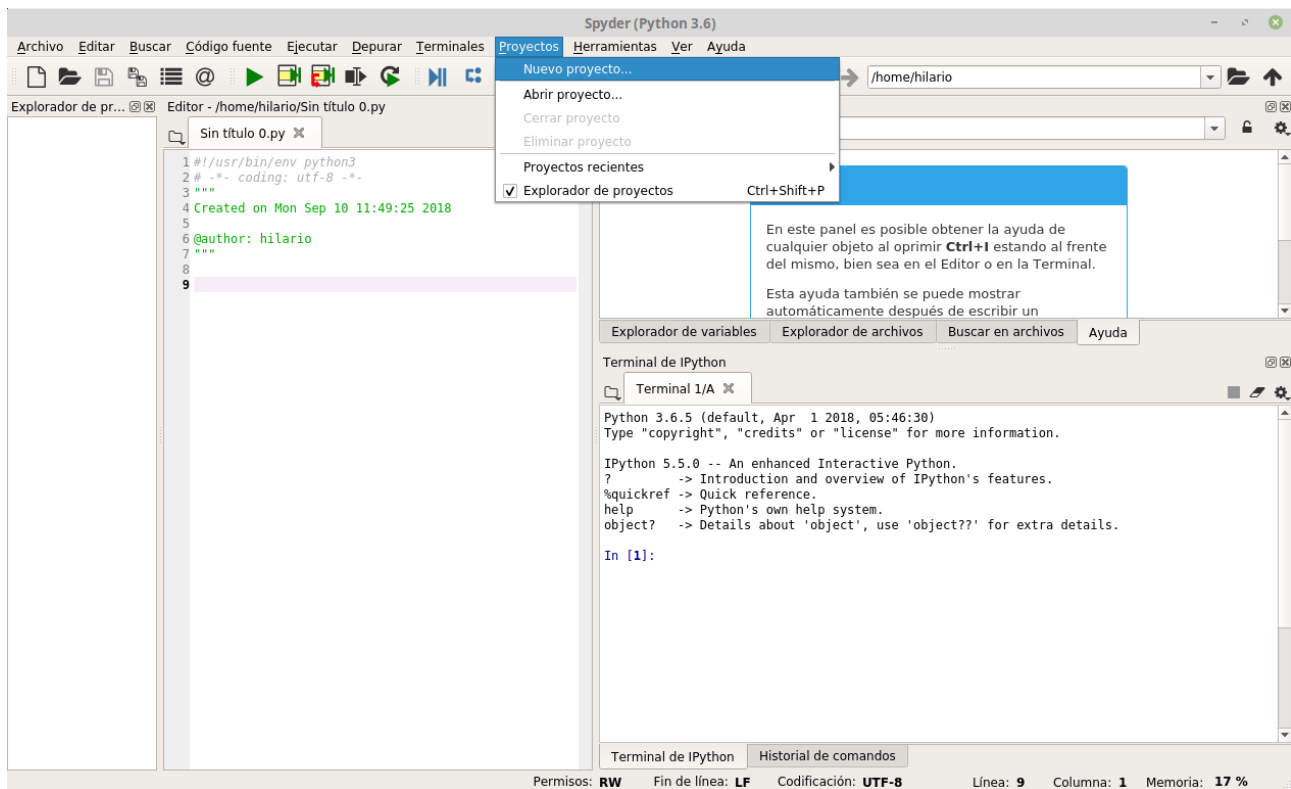
Curso 2018/2019

Programación de aplicaciones con Python y OpenCV (I)

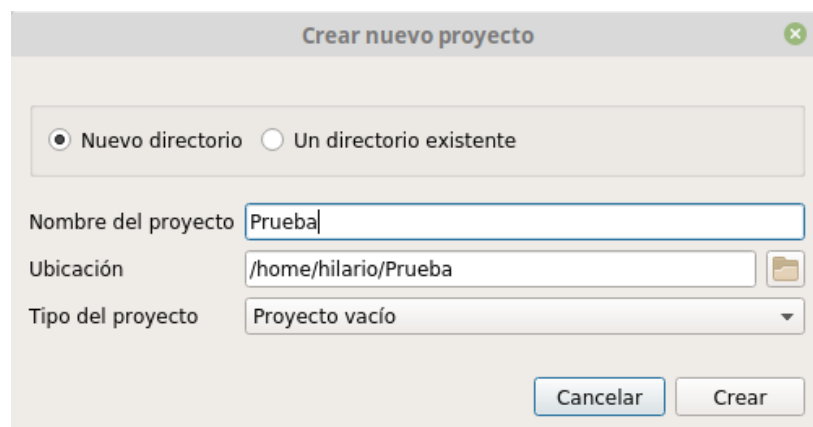
1. Creación de un proyecto

A lo largo del cuatrimestre iremos haciendo diversas actividades que implicarán la realización de diversos programas. Estos programas, muchas veces, serán distintos entre sí y conviene separarlos en Proyectos.

En este apartado veremos como crear un Proyecto en Spyder. Para ello sólo debemos ir al menú “Proyectos” y clicar en “Nuevo proyecto”.



Tras ello nos aparecerá un diálogo en el que elegiremos el nombre del Proyecto y su ubicación.



Procesado de imagen y visión por computador

Curso 2018/2019

Programación de aplicaciones con Python y OpenCV (I)


Una vez creado el Proyecto nos aparecerá a la izquierda el explorador de proyectos en el que podremos elegir el Proyecto que queramos usar en cada momento. Una vez elegido el proyecto nos aparecerán dentro de él los archivos Python que vayamos creando o que ya estuvieran en el directorio elegido.

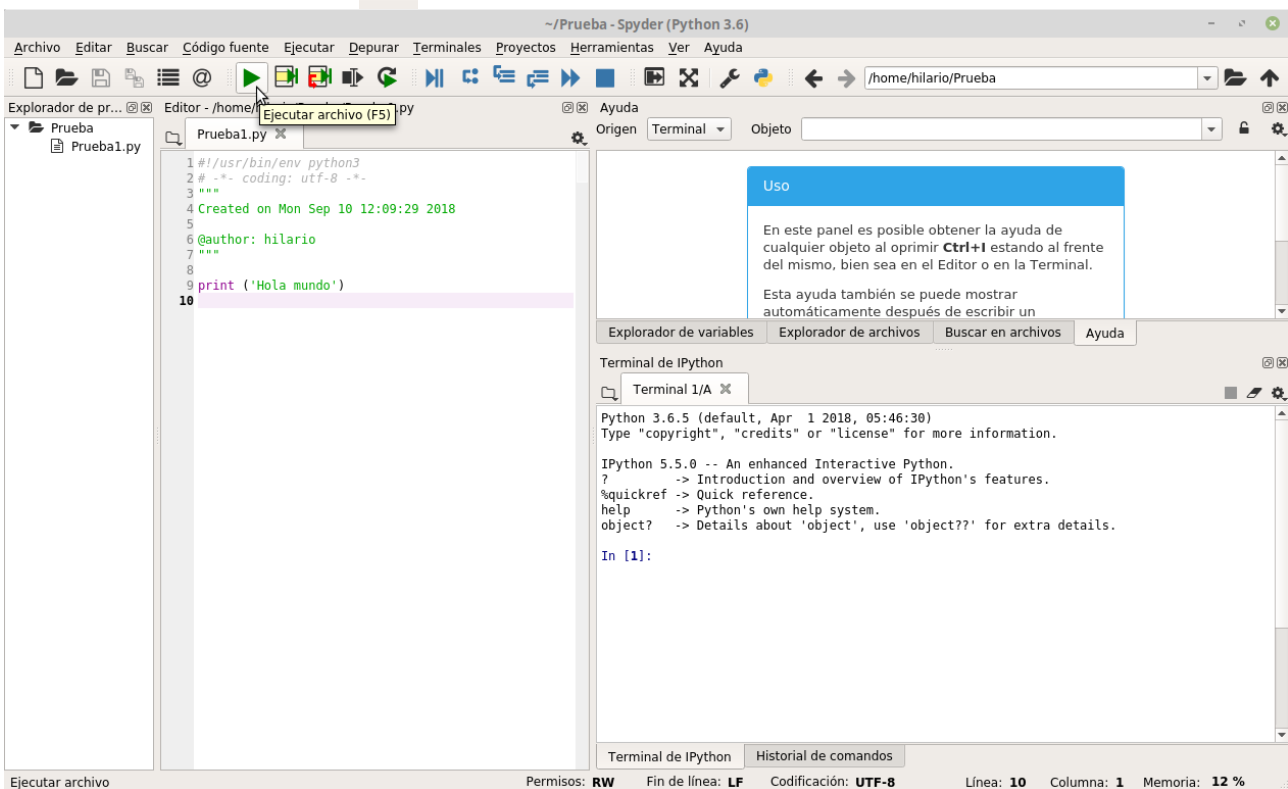
El siguiente paso sería editar el archivo. En este caso, vamos a crear un proyecto que imprima por pantalla el mensaje “*Hola mundo*”. El código sería el siguiente:

```
print ('Hola mundo')
```

Una vez editado, si hay errores sintácticos se marcarán en el propio editor.

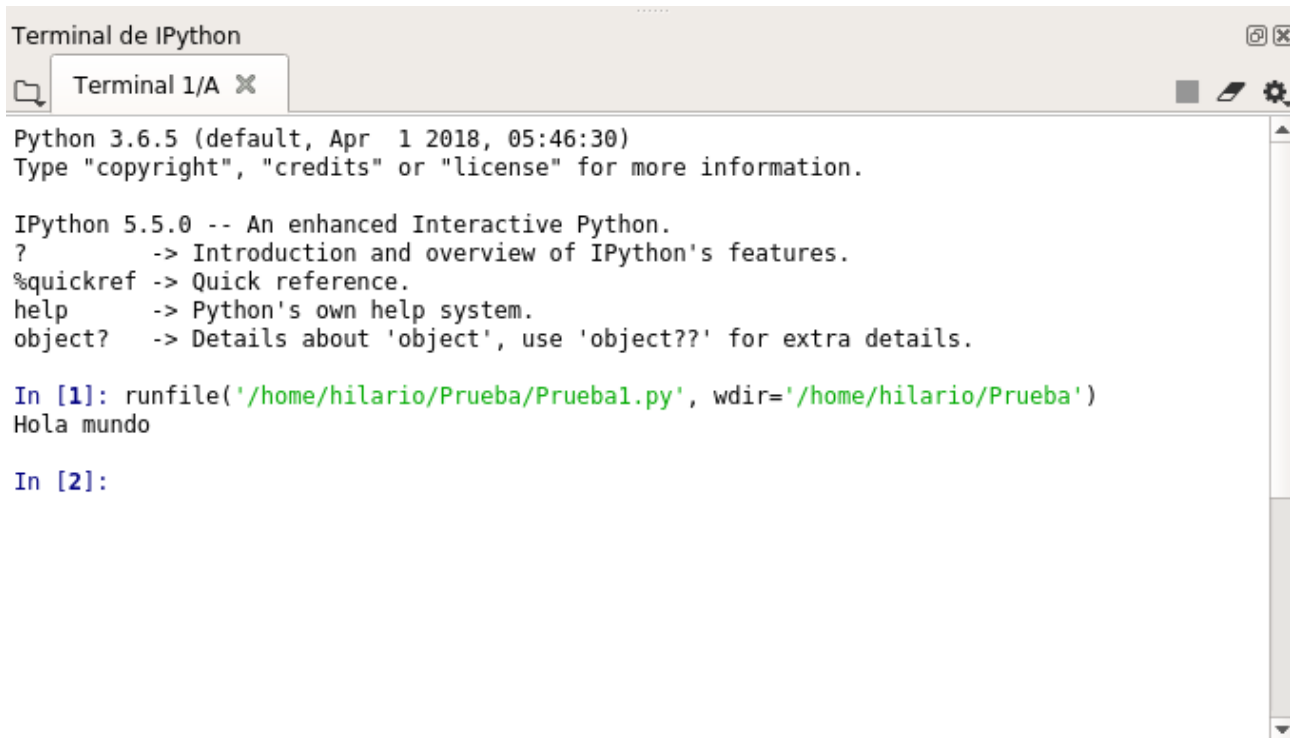
2. Ejecutar y depurar un proyecto

Para ejecutar un proyecto existen dos opciones. O bien usar el menú Ejecutar o el botón de ejecutar de la barra de herramientas  como se muestra en la siguiente imagen.



El resultado de la ejecución (si no es una aplicación de ventanas como es este caso) se puede ver en la pestaña *Terminal*.

Procesado de imagen y visión por computador
Curso 2018/2019
Programación de aplicaciones con Python y OpenCV (I)



```
Terminal de IPython
Terminal 1/A X

Python 3.6.5 (default, Apr 1 2018, 05:46:30)
Type "copyright", "credits" or "license" for more information.

IPython 5.5.0 -- An enhanced Interactive Python.
?          -> Introduction and overview of IPython's features.
%quickref  -> Quick reference.
help       -> Python's own help system.
object?    -> Details about 'object', use 'object??' for extra details.

In [1]: runfile('/home/hilario/Prueba/Prueba1.py', wdir='/home/hilario/Prueba')
Hola mundo

In [2]:
```

En caso de encontrar problemas en la ejecución es muy interesante el uso de las opciones de depuración del IDE. En el caso de Spyder tenemos las opciones de depuración en el menú “Depurar” o bien en la barra de herramientas.



Para depurar el programa paso a paso habiendo puesto algún *Breakpoint*. Para ejecutar paso a paso, están las opciones:

- Ctrl+F5 : Depurar el programa.
- Ctrl+F10 : Ejecutar línea.
- Ctrl+F11: Entrar dentro de las funciones.
- Ctrl+Shift+F11: Salir de las funciones.
- Ctrl+F12: Continuar.
- Ctrl+Shift+F12: Parar.
- F12: Añadir o quitar un punto de parada. (También con doble click del ratón)


En el menú *Ejecutar* tenemos además la posibilidad de ejecutar sólo una parte del código con la opción *Ejecutar Celda*.

En todo el proceso de depuración y ejecución del programa podemos comprobar los valores de la variables del programa en la pestaña “*Explorador de variables*”. El explorador de variables nos

Procesado de imagen y visión por computador

Curso 2018/2019

Programación de aplicaciones con Python y OpenCV (I)

permitirá además guardar una copia de las mismas si es necesario y, muy importante, borrar las variables de ejecuciones anteriores con el símbolo . Para no olvidar ese borrado podemos marcar la opción *Eliminar todas las variables antes de la ejecución*, del menú Ejecutar→Configuración por archivo.

3. Crear un proyecto de procesamiento de imágenes con OpenCV

En esta asignatura trabajaremos con la librería de procesamiento de imágenes *OpenCV*. Dentro del modulo *cv2* que importaremos si queremos usar *OpenCV* podemos encontrar funciones de todo tipo enfocadas al procesamiento de imagen y a la presentación de datos.

En este caso, vamos a crear un proyecto que permita abrir una imagen, y mostrarla en una ventana en la pantalla del ordenador. Para empezar, es necesario crear un nuevo proyecto (o reutilizar el creado en el ejemplo anterior). Seguidamente, escribimos el código que permite abrir una imagen almacenada en el disco duro, y mostrarla en la pantalla, como se indica a continuación:

```
# -*- coding: utf-8 -*-
import cv2 #Cargamos OpenCV
import sys #Cargamos las llamadas a sistema

#Leemos la imagen
imagen=cv2.imread(sys.argv[1], cv2.IMREAD_UNCHANGED)

#Creamos la ventana donde mostrar la imagen
cv2.namedWindow("Ejemplo1", cv2.WINDOW_AUTOSIZE);

if imagen is None: #Si está vacía es que no se ha leído
    print('Imagen no encontrada\n')
else:
    cv2.imshow('Ejemplo1', imagen) #Mostramos la imagen
    #Espera hasta presionar una tecla. Importante porque si no no visualizamos
    la imagen.
    cv2.waitKey(0)
#Liberamos la ventana creada
cv2.destroyAllWindows()
```

Para comprobar el funcionamiento del ejecutable, hay que tener en cuenta que el programa abre la imagen pasada como primer argumento de la línea de entrada. Para incluir argumentos en la línea de entrada ir al menú Ejecutar→Configuración por archivo. Y marcar la opción *Opciones de línea de comandos*. Escribir el nombre de la imagen, incluido el path completo.

Ejecutar y comprobar que el programa muestra en pantalla la imagen seleccionada. Pruebe con la imagen *Pez.jpg*. Si no funciona correctamente, probar a depurar el programa en busca del posible error. Finalmente, modificar el programa para que imprima por pantalla las dimensiones de la imagen. Las dimensiones de la imagen pueden obtenerse con la variable *shape* de la matriz *imagen* (*imagen.shape*) donde se ha abierto la imagen¹.

¹ http://docs.opencv.org/3.4.3/dc/d2e/tutorial_py_image_display.html

Procesado de imagen y visión por computador
Curso 2018/2019
Programación de aplicaciones con Python y OpenCV (I)

3.a.- Guardar una imagen con OpenCV:

Para finalizar, modificaremos el código anterior para que, una vez abierta la imagen, la guarde con un formato distinto, en este caso, vamos a guardarla con formato PNG. Para guardar la imagen, consulte, en la ayuda de *OpenCV*, la función *imwrite*. Ejecutamos el nuevo código, y guardamos la imagen con el mismo nombre que la imagen original, pero con extensión *.png*.

Vamos a comprobar ahora las posibilidades de cambio de espacios de color usando OpenCV². La función que vamos a usar es *cvtColor()* y con ella cambiaremos del espacio de color inicial a una escala de grises para después guardar esa imagen con otro nombre. El parámetro que usaremos será *cv2.COLOR_BGR2GRAY* (Las posibilidades de conversión pueden consultarse en este [enlace](#)).

4. Acceso a píxeles de la imagen y modificación

Una vez leída una imagen, el acceso a los píxeles individuales que la forman y su modificación es muy sencilla gracias a que las imágenes, cuando usamos Python y *OpenCV*, son matrices Numpy. Esto hace que su manejo sea muy similar a como accederíamos a una posición dentro de una matriz de Matlab, teniendo en cuenta que los índices empiezan en 0. Por ejemplo, si queremos acceder a la componente roja de la posición (1,2) de una imagen en color sólo tendríamos que poner:

valor = imagen[0,1,2]

Si queremos extraer una ROI (Region of interest) de una imagen sólo tenemos que poner unos índices que nos permitan acceder al rectángulo que la inscribe.

Por ejemplo, si queremos modificar una ROI de la imagen y pintarla en rojo lo podríamos hacer de la siguiente manera:

```
# -*- coding: utf-8 -*-
import cv2

img=cv2.imread('Smile.png') # Leemos la imagen

if img is None: #Si está vacía es que no se ha leído
    print('Imagen no encontrada\n')
    exit(0)

img[10:150,20:160]=(0,0,255)# Ponemos la ROI en rojo

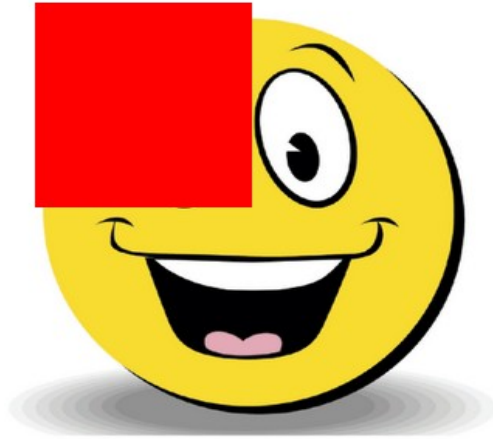
cv2.imshow('ROI',img)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

Con el siguiente resultado:

² https://docs.opencv.org/3.4.3/df/d9d/tutorial_py_colorspaces.html

Procesado de imagen y visión por computador
Curso 2018/2019
Programación de aplicaciones con Python y OpenCV (I)



Cómo ejercicio y basándonos en el código anterior vamos a extraer una ROI de la imagen *Smile.png* y la vamos a guardar en un fichero llamado *ROI.png*. Con la ROI pretendemos extraer el ojo izquierdo obteniendo algo parecido a:



Una vez conseguida la extracción de esa ROI haga lo mismo con el ojo derecho y la boca.

5. Copia y asignación de imágenes.

Al trabajar con la librería *OpenCV* hay que tener en cuenta que las asignaciones directas de imágenes pueden producir resultados inesperados ya que el signo “=” no hace copias de las imágenes sino que hace dos variables iguales y lo que hagamos en una y en otra tiene efectos sobre las dos. Para entender este efecto vamos a probar el código mostrado a continuación:

```
# -*- coding: utf-8 -*-
import cv2

img=cv2.imread('Smile.png') # Leemos la imagen

if img is None: #Si está vacía es que no se ha leído
    print('Imagen no encontrada\n')
    exit(0)

img2=img # Ahora img y img2 son iguales

img2[10:60,50:80]=(0,255,0)

cv2.imwrite("Copia.png",img)
cv2.imwrite("Modificada.png",img2)
```

Procesado de imagen y visión por computador

Curso 2018/2019

Programación de aplicaciones con Python y OpenCV (I)

Compruebe que, contra lo esperado, *Copia.png* y *Smile.png* no son iguales.

Si en el desarrollo de un programa necesitamos hacer una copia de una imagen (clonado) no debemos usar el signo de igualdad sino la función *copy()* propia de la librería Numpy y que permite copiar una matriz en otra:

```
x=np.array([1,2,3,5])
y=x.copy()
print y
[1 2 3 5]
```

Modifique el código anterior (que no funcionaba adecuadamente) para que funcione correctamente y se guarde la imagen original y la modificada.