

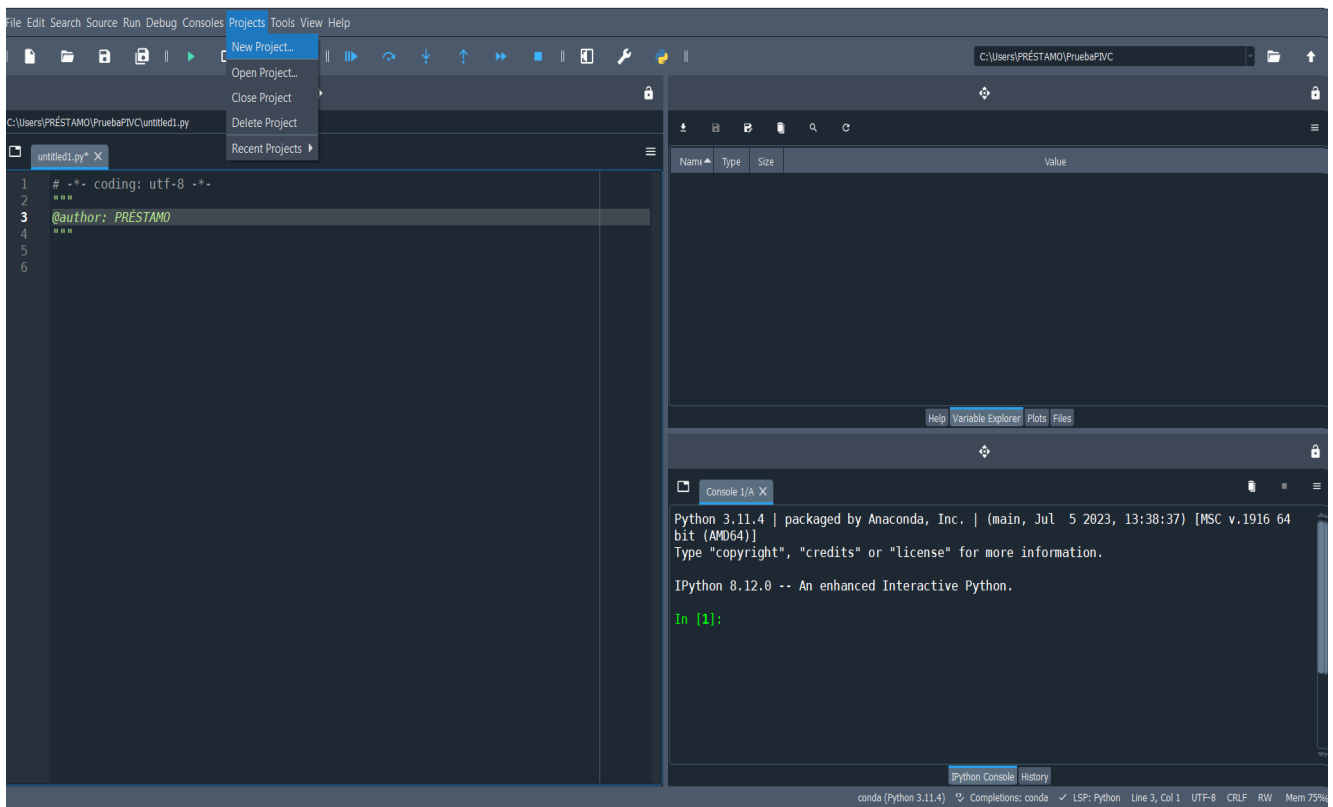
Procesado de imagen y visión por computador

Práctica 1.1. Programación de aplicaciones con Python y OpenCV (I)

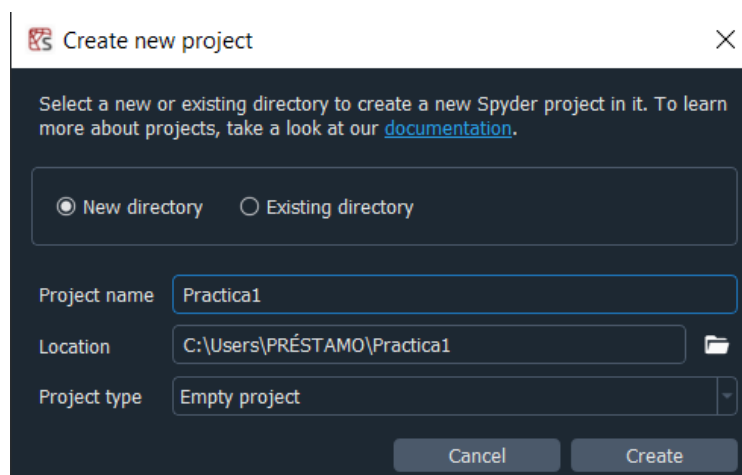
1. Creación de un proyecto

A lo largo del cuatrimestre se irán desarrollando en la asignatura numerosas sesiones 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 'Projects' y seleccionar 'New Project'.



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



Procesado de imagen y visión por computador

Práctica 1.1. 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 del mismo los archivos Python que vayamos creando o que ya estuvieran en el directorio elegido.

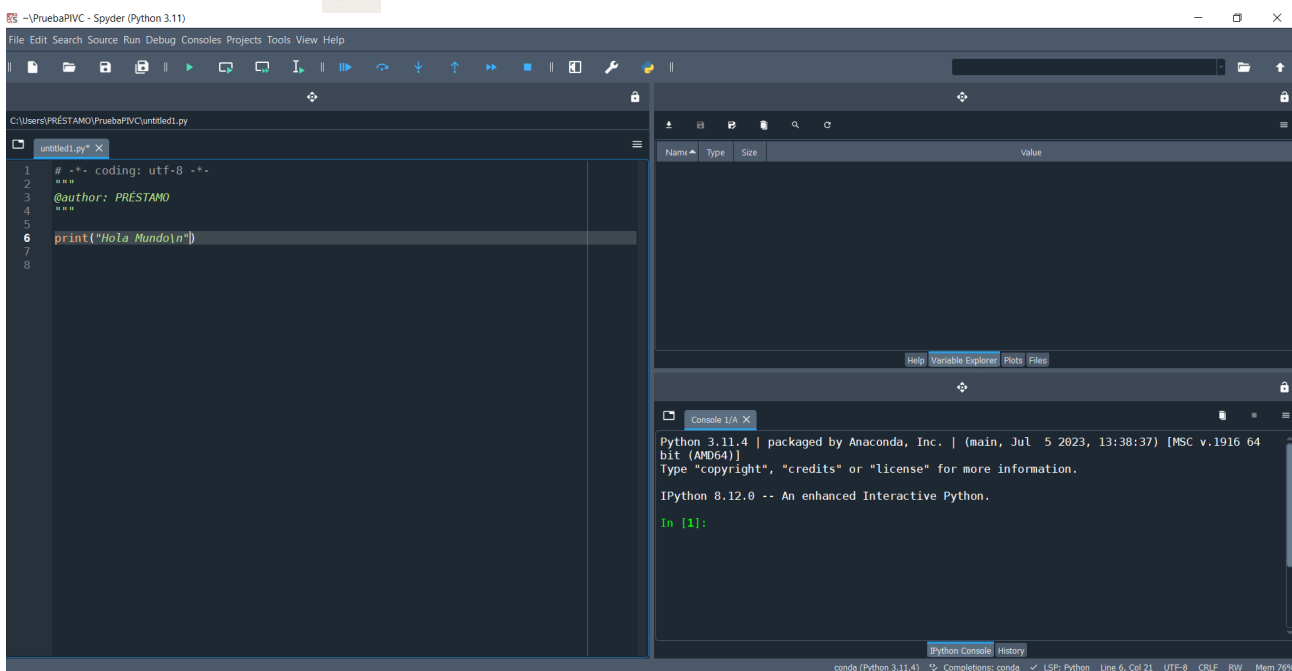
El siguiente paso sería editar el archivo *Prueba1.py*. En este caso, vamos a crear un proyecto que imprima por pantalla el mensaje '*Hola mundo*'. El código del script *Prueba1.py* sería el siguiente:

```
print ('Hola mundo\n')
```

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

2. Ejecución y depuración

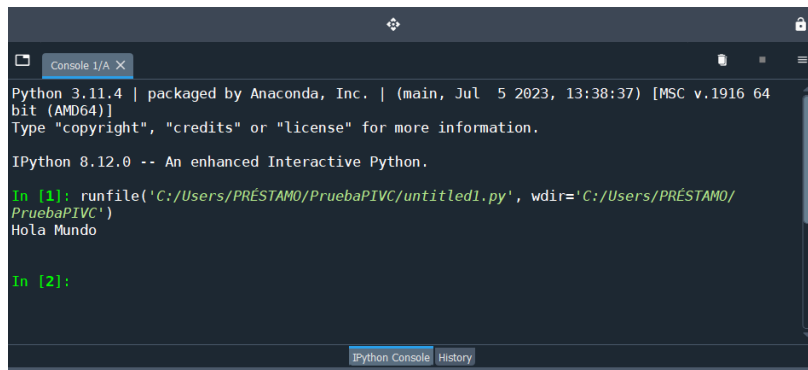
Para ejecutar un proyecto existen dos opciones. O bien usar el menú 'Run' o el botón de ejecutar de la barra de herramientas  como se muestra en la siguiente imagen.



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

Práctica 1.1. Programación de aplicaciones con Python y OpenCV (I)



```
Python 3.11.4 | packaged by Anaconda, Inc. | (main, Jul 5 2023, 13:38:37) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license()" for more information.

IPython 8.12.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/PRÉSTAMO/PruebaP1VC/untitled1.py', wdir='C:/Users/PRÉSTAMO/PruebaP1VC')
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 “*Debug*” o bien en la barra de herramientas.



Para depurar el programa paso a paso habiendo puesto algún *Breakpoint* y para ejecutar paso a paso, contamos con las siguientes 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 *Run* tenemos además la posibilidad de ejecutar sólo una parte del código con la opción ‘*Run cell*’.

En todo el proceso de depuración y ejecución del programa podemos comprobar los valores de la variables del programa en la pestaña “*Variable Explorer*”. El explorador de variables nos 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 ‘*Remove all variables before execution*’, del menú *Run*→*Configuration per file*.

Procesado de imagen y visión por computador

Práctica 1.1. Programación de aplicaciones con Python y OpenCV (I)

3. Procesado de imágenes con OpenCV

En esta asignatura trabajaremos con la librería de procesamiento de imágenes *OpenCV*. Dentro del módulo *cv2* que importaremos si queremos usar *OpenCV*¹ podemos encontrar funciones muy variadas enfocadas al procesamiento de imagen y a la presentación de datos.

En este apartado 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, tal como se indica a continuación:

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

import cv2
import sys

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

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

if image is None: #Si la imagen está vacía es que no se ha leído
    print("Imagen no encontrada\n")
else:
    cv2.imshow("Ejemplo1",image)
    #La visualización espera hasta que el usuario presione una tecla
    cv2.waitKey(0)

#Cerramos todas las ventanas creadas
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 *Run*→*Configuration per file*. Y marcar la opción '*Command line options*'. Escribir el nombre de la imagen incluyendo el *path* completo donde se encuentra.

Ejecutar y comprobar que el programa muestra en pantalla la imagen seleccionada. Pruebe con la imagen *politecnica.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, el número total de píxeles y el tipo de dato utilizado. Las dimensiones de la imagen y el número de canales de color pueden obtenerse accediendo a las propiedades de la imagen con la variable *shape* de la matriz *imagen* (*imagen.shape*)², mientras que para obtener el tipo de dato con

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

2 https://docs.opencv.org/master/d3/df2/tutorial_py_basic_ops.html

Procesado de imagen y visión por computador

Práctica 1.1. Programación de aplicaciones con Python y OpenCV (I)

que se almacena la imagen puede utilizar la variable *dtype* (*imagen.dtype*). También puede observar directamente la información en la ventana del explorador de variables.

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*.

Seguidamente vamos a utilizar el redimensionado de una imagen con la función *resize*³. Con ella intentaremos cambiar las dimensiones de la imagen (ancho y alto) a la mitad de las de la imagen original y guardaremos la nueva imagen en disco con otro nombre (*politecnica_resize.jpg*). Observe en el explorador de variables las dimensiones de la nueva imagen respecto a la imagen original.

Finalmente, vamos a comprobar las posibilidades de cambio de espacios de color usando *OpenCV*⁴. La función que vamos a usar es *cvtColor()* y con ella cambiaremos del espacio de color inicial de la imagen original a una escala de grises para después guardar esa imagen con otro nombre (*pajaro_gray.jpg*). El parámetro que usaremos será *cv2.COLOR_BGR2GRAY* (Las posibilidades de conversión pueden consultarse en este [enlace](#)). Observe en el explorador de variables las dimensiones de la nueva imagen respecto a la imagen original.

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, 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 tendríamos que escribir:

```
valor = imagen[0,1,2]
```

Es importante considerar que la función *imread* devuelve las imágenes de color en formato BGR (azul-verde-rojo), donde BGR representa el mismo espacio de color que RGB, pero con orden de byte inverso.

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

img=cv2.imread('Imagenes/Politecnica.jpg') # Leemos la imagen

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

3 <https://pythonexamples.org/python-opencv-cv2-resize-image/>

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

Procesado de imagen y visión por computador

Práctica 1.1. Programación de aplicaciones con Python y OpenCV (I)

```
sys.exit(0)

#Obtener el valor BGR del pixel en una posicion
b,g,r=img[300,200]
#Otra forma es: b,g,r=img[300,200,:]
print ('Pixel:',b,g,r)
```

De manera similar, podemos modificar el valor de un píxel de la imagen:

```
img[300,200]=[255,255,255]
```

En ocasiones deseamos trabajar con una porción de la imagen original, denominada comúnmente ROI (*region of interest*). En estos casos podemos usar el indexado de Numpy para seleccionar y copiar la región rectangular deseada. Así, por ejemplo, si queremos modificar una ROI de la imagen y reemplazarla por un rectángulo en rojo lo podríamos hacer del siguiente modo:

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

img=cv2.imread('Imagenes/Politecnica.jpg') # Leemos la imagen

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

img[105:245,170:293]=(0,0,255) # Ponemos la ROI en color rojo
cv2.imshow('ROI',img)

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

Como ejercicio y basándonos en el código anterior, intente extraer la ROI de la imagen *politecnica.jpg* que incluya aproximada la puerta de entrada al edificio y guárdela en un archivo *politecnica1.jpg*.

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 crea dos variables iguales y los cambios que hagamos en una de ellas también tienen efecto sobre la otra variable. Para entender este efecto vamos a probar el código mostrado a continuación:

Procesado de imagen y visión por computador

Práctica 1.1. Programación de aplicaciones con Python y OpenCV (I)

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

img=cv2.imread('Imagenes/Imagen1.jpg') # Leemos la imagen

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

img2=img # Ahora img y img2 son iguales

img2[105:245,170:293]=(0,255,0)

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

Compruebe que, contra lo esperado, *Copia.png* y *Modificada.png* son iguales, pues las modificaciones que hagamos, a posteriori de la asignación, en cualquiera de ellas afecta a ambas.

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 que permite copiar una matriz en otra. Para entenderlo mejor, veamos los siguientes dos ejemplos de código con un array numérico:

Ejemplo 1

```
>>> x=numpy.array([1,2,3])
>>> z=x
>>> z+=2
>>> print x
[3,4,5]
>>> print z
[3,4,5]
```

Ejemplo 2

```
>>> x=numpy.array([1,2,3])
>>> z=x.copy()
>>> z+=2
>>> print x
[1,2,3]
>>> print z
[3,4,5]
```

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

6. Manipulación de imágenes (Opcional)

Cada alumno trabajará sobre una imagen propia en la que aparezca correctamente identificado, ya sea a través de una captura de la cámara del portátil o de otra imagen cualquiera que pueda aportar.

Procesado de imagen y visión por computador

Práctica 1.1. Programación de aplicaciones con Python y OpenCV (I)

A partir de la misma, se deben realizar los tres siguientes scripts:

1. El primer script dibujará dos rectángulos de color diferente en dos ROIs del rostro (por ejemplo, ojo y boca).
2. El segundo script extraerá una imagen con la ROI de la cara del alumno.
3. El tercer script reemplazará el sol de la imagen sol.jpg por la cara del alumno extraída en el script 2. Muy posiblemente, el tamaño de la ROI de la cara extraída no se adecúe al tamaño del sol. Para resolver este problema puede probar a redimensionar la imagen de la ROI del rostro a través de la función de OpenCV *resize* (por ejemplo, *newImg=cv2.resize(img, (width, height))*).