Intro a Imagenes

Pillow

1. Crear 3 imagenes de $600x800$ pixeles, una roja, otra verde y otra amarilla. Guardarlas como *rojo.jpg*, *verde.jpg* y *amarillo.jpg*

2. Dada la imagen llamada *cuadrado.jpg (descargar con el práctico)* reemplazar el color negro por verde.

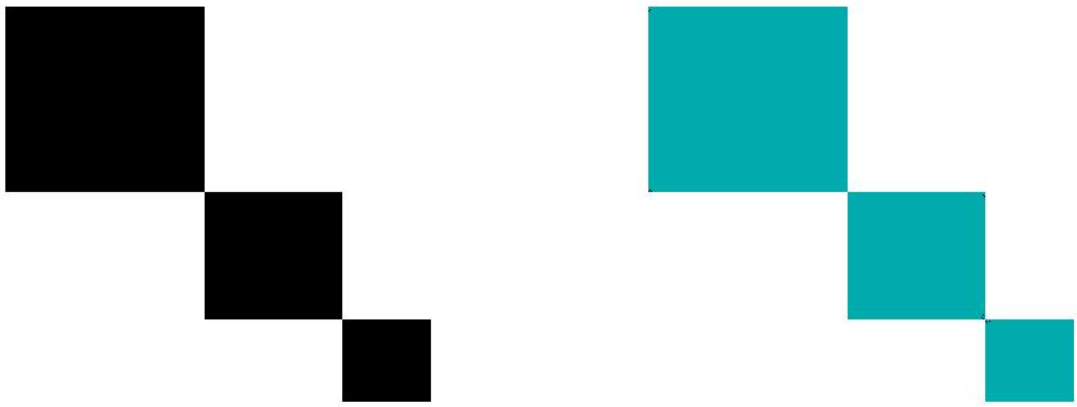

Figura 1: Convertir de negro a azulado los cuadrados.

3. Crear una imagen del tamaño que desee, que tenga un arcoiris de 6 colores, organizada en 6 grandes rayas horizontales.

4. Implementar una función llamada *saturar* que tome dos parámetros, la imagen a saturar y el porcentaje que se desea de saturación. Convirtiendo de RGB al espacio de color HSV. *(Hue, Saturation, Value = Tono/Matiz, Saturación, Brillo/Valor)*
**Ayuda: Una vez abierta la imagen, utilice el comando de Pillow imagen.convert('HSV') para realizar la conversión del espacio de color)*

OpenCV

5. Idem que en el ejercicio 1, pero esta vez con la librería OpenCV: Crear 3 imagenes de $600x800$ pixeles, una roja, otra verde y otra amarilla. Guardarlas como *rojo_cv.jpg*, *verde_cv.jpg* y *amarillo_cv.jpg*

6. Crear una imagen del tamaño que desee, que tenga un arcoiris de 6 colores, organizada en 6 grandes rayas verticales. Guardarla a disco con el nombre *verticales.jpg*

7. Crear una imagen de tamaño cuadrado desde cero, donde el usuario ingrese el color de fondo que desea, y el color para el cuadrado que se dibujará en el centro. Generar una imagen que contenga un pequeño cuadrado en el centro de la misma.

8. Dada la imagen adjuntada en el práctico, llamada *puros.jpg* realizar un programita que transforme el triángulo rojo a un triángulo amarillo. Observar que se tiene una imagen con colores planos puros.

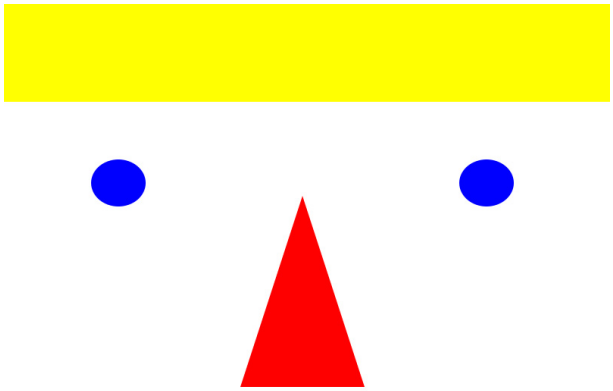

Figura 2: Convertir la nariz roja a una nariz amarilla.

Variado

Intente resolver la mayoría de los ejercicios a continuación, a través de realizar recorridos por los píxeles en vez de usar comandos pre fabricados que hagan toda la tarea por si mismos. Puede usar Pillow y/o OpenCV *(o realizar dos versiones de cada código, uno con cada una)*

9. Pedir una imagen al usuario y generar una nueva imagen que agregue un *padding* de $50px$ todo alrededor, del color que el usuario desee. Luego, generalizar en una función llamada *padding* que tome como parámetros una imagen y un valor de pixeles, y genere una nueva imagen que conste de la original más un borde de dicho grosor de pixeles todo alrededor.

10. Dada una imagen cualquiera, generar su imagen en negativo. Recordar que para los valores RGB de un pixel cualquiera, el valor RGB del pixel en negativo será restarle a 255 ese pixel.

11. Implementar 3 funciones, cada una se encargará de eliminar uno de los tonos de una imagen cualquiera. Las funciones llamadas *quitar_rojo*, *quitar_azul* y *quitar_verde* tomarán un solo parámetro, la imagen.

12. Implementar una función llamada *hamming_pal*, que calcule la distancia de Hamming entre dos palabras o cadenas de caracteres, de igual longitud.

13. Implementar la función distancia de hamming, pero para imágenes. Se llamará *hamming* y tomará dos imagenes como parámetros.

14. Generar dos imagenes del mismo tamaño, en blanco y negro, aleatorias *(imagine ruido de TV)*. Determinar si son iguales o no. Probar con imagenes pequeñas primero, por ejemplo de 6 x 6 y luego imagenes de 500x600 pixeles. Utilizar la función Distancia de Hamming implementada anteriormente.

15. Dada una imagen color cualquiera, mostrar en pantalla el siguiente collage de los 3 canales individuales de color de la misma.

Figura 3: Canal rojo, canal verde y canal azul (RGB).

16. Convertir una imagen a escala de grises, de modo artesanal. Dejarla implementada como una función propia que dado un nombre de archivo de imagen devuelva o la imagen lista, o grabe la imagen a disco, según un parámetro extra de la función, llamado *retorno* que tome uno de dos valores posibles ('disco', 'imagen').
    Realice diferentes implementaciones:

    a) Promedio.

    b) Corrección para el Ojo humano *(Luma por ejemplo)*.

    c) Descomposición/Desaturación por mínima y máxima.

    d) Un único canal.

17. Implementar una función que genere una versión en tonos sepia de una imagen pasada como argumento. Una posibilidad es utilizar los valores recomendados por Microsoft para generar un pixel sepia en base a un pixel RGB, dado por los siguientes valores:
    el nuevo rojo pasará a ser $(R * .393) + (G * .769) + (B * .189)$
    el nuevo verde será $(R * .349) + (G * .686) + (B * .168)$
    y el nuevo azul será $(R * .272) + (G * .534) + (B * .131)$