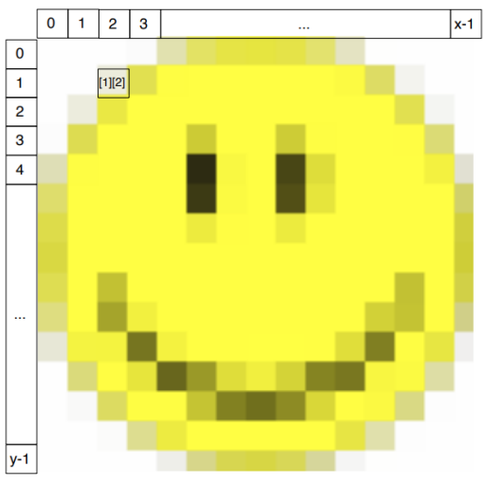
Departamento de Matemática

TAREA 2 – PCPictures

Pensamiento computacional y programación

¡Otro gran proyecto que te desafía como programador(a)! Deberás poner aprueba todo lo que sabes sobre Python, como variables, operadores, control de flujo, funciones, importación de módulos y listas. Tu desafío será programar un procesador de imágenes que permita aplicar filtros, podríamos tratarlo como una suerte de Instagram en pañales.

Una imagen digital no es más que una matriz (lista de listas) de pixeles, como la imagen más abajo. Cada pixel es representado por una tupla de tres números enteros entre 0 y 255. El primero de estos números representa la intensidad de rojo, el segundo la intensidad verde y el tercero la intensidad de azul. Por ejemplo, la tupla (0, 150, 0) representa un pixel de color verde opaco, mientras que la tupla (0, 255, 255) representa un pixel de color cyan (color que resulta de mezclar el verde con el azul).



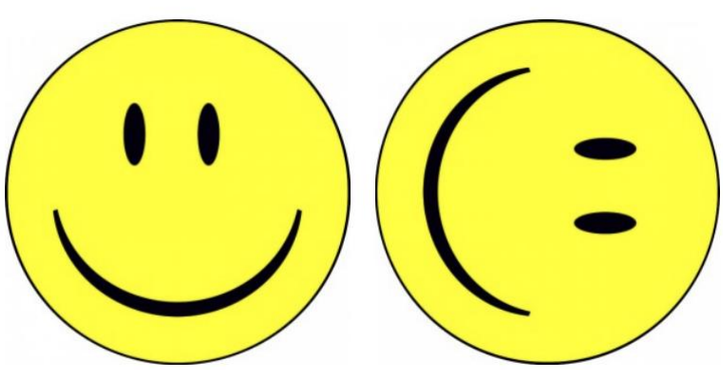
Dada una imagen A en su representación como matriz, consideraremos que el elemento A[0][0] corresponde al pixel de la esquina superior izquierda. La primera coordenada corresponderá al número de fila (de arriba hacia abajo) y la segunda al número de columna (de izquierda a derecha).

Enunciado

En esta tarea deberás escribir un programa que permita al usuario abrir una imagen en formato gif y aplicarle las siguientes operaciones:

- **Girar**

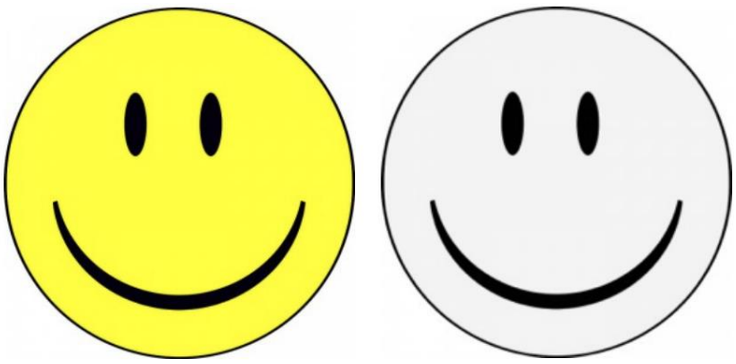
Gira la imagen 90 grados en el sentido de las manecillas del reloj



(a) Imagen original (b) Imagen girada 90 grados

- **Escala de grises**

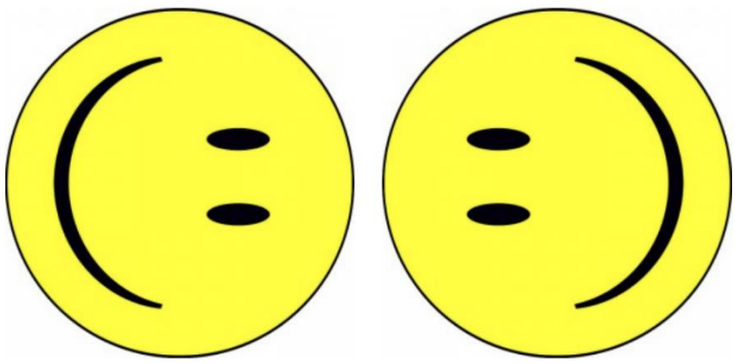
Convierte una imagen de color a otra en escala de grises. Para que una imagen sea vea en tonos de gris se requiere que los tres componentes básicos del color (rojo, verde, azul) tengan más o menos la misma intensidad. Por lo tanto, podemos decir que si queremos convertir un pixel a su equivalente en escala de grises bastará con cambiar la intensidad de cada color del pixel por el promedio de los tres colores básicos. Por ejemplo, el pixel (255, 0, 0) será convertido al (85, 85, 85).



(a) Imagen original (b) Imagen en escala de grises

- **Invertir**

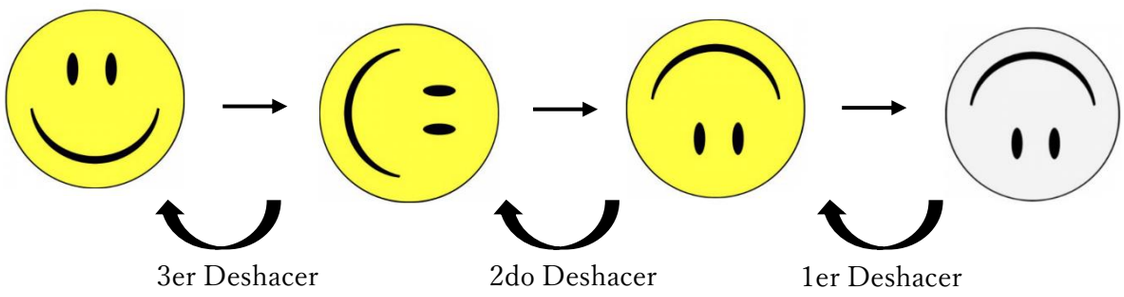
Invierte la imagen horizontalmente. Es el resultado de aplicar una simetría respecto al eje vertical que pasa por el centro de la imagen



(a) Imagen Original (b) Imagen invertida

- **Deshacer**

Deshace el último cambio aplicado a la imagen y retorna la imagen a su estado anterior. Debe ser posible deshacer todos los cambios que se le hayan hecho a la imagen.

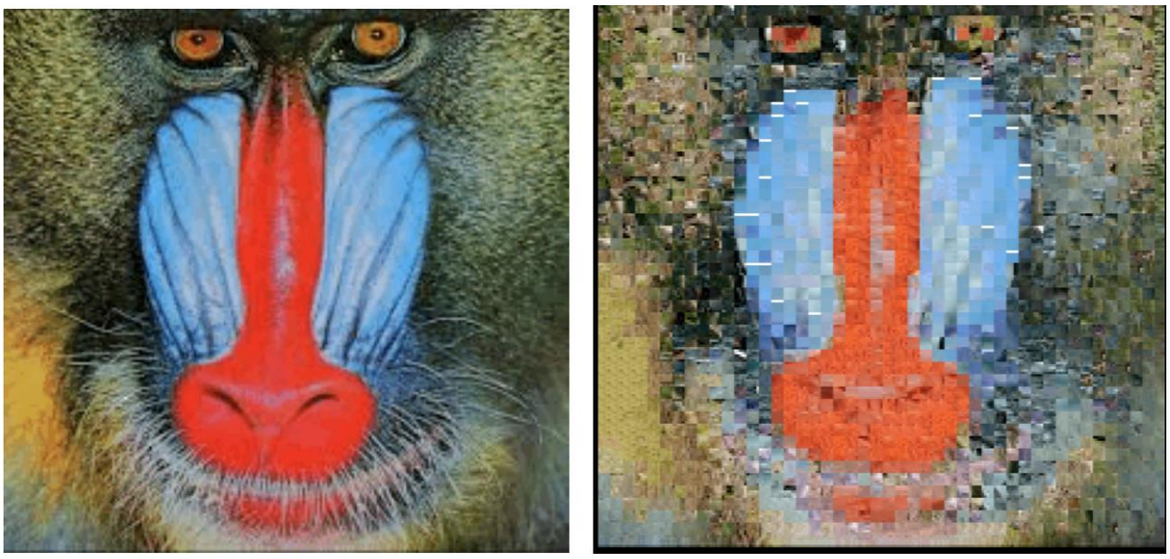


- **Mosaico**

La carpeta de tu tarea deberá incluir una subcarpeta llamada mosaicos que contiene un buen número de imágenes de tamaño 5 x 5 pixeles. El operador Mosaico consiste en generar una nueva imagen reemplazando porciones de la imagen original por la imagen de 5 x 5 que más se parece a dicha porción.

Más específicamente, dada una imagen base (de tamaño arbitrario), deberás dividirla en porciones cuadradas de 5 x 5 pixeles (puedes asumir que las dimensiones de tu imagen son múltiplos de 5). Posteriormente, para cada una de esas porciones deberás elegir la imagen de la carpeta mosaicos que más se parece a la porción, y reemplazar esa porción por la imagen elegida.

Para calcular la similitud entre una imagen I_1 y otra imagen I_2 se considera la suma de las diferencias entre cada pixel de I_1 y el pixel que corresponde a la misma posición en I_2. La diferencia entre el pixel $p1 = (r_1, g_1, b_1)$ y $p2 = (r_2, g_2, b_2)$ es el resultado de $(r_1 - r_2)^2 + (g_1 - g_2)^2 + (b_1 - b)^2$.



(a) Imagen original (b) Imagen como un mosaico

- **(Bonus 1) Bordes**

Al implementar esta funcionalidad de modo satisfactorio se te otorgarán 25 pp.

Genera una imagen en escala de grises con líneas blancas que representan los bordes de la imagen original.

(a) Imagen original (b) Bordes de la imagen

Para calcular esta imagen, primero se transforma la imagen original a su representación en escala de grises, que denominaremos $\boldsymbol{A}$. Luego se generan dos matrices: una para detectar los bordes horizontales y otra para detectar los bordes verticales. Estas dos matrices se denotan G_x y G_y, y se definen como:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * \boldsymbol{A} \quad y \quad G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \boldsymbol{A},$$

Donde * es el operador de *convolución*. Este operador toma una matriz $\boldsymbol{B}$ de 3×3 y una matriz de pixeles $\boldsymbol{C}$, y genera una nueva matríz de pixeles $\boldsymbol{C'}$, donde el pixel $\boldsymbol{C'}[i][j]$ es definido como:

$$\boldsymbol{C'}[i][j] = \sum_{k=0}^{2} \sum_{l=0}^{2} \boldsymbol{B}[k][l] \cdot \boldsymbol{C}[i+k-1][j+l-1]$$

Cabe notar que cada entrada de $\boldsymbol{C}$ es una tupla y cada entrada de $\boldsymbol{B}$ es un número, por lo que la multiplicación de una estrada de $\boldsymbol{C}$ por una de $\boldsymbol{B}$ es la multiplicación usual entre un vector y un escalar.

Para calcular el operador convolución sobre los pixeles en los extremos, no se toman en cuenta las coordenadas que se "salen" de la imagen. Por ejemplo, $\boldsymbol{C'}[0][0]$ se calcularía sin considerar la primera fila ni la primera columna de $\boldsymbol{B}$:

$$\boldsymbol{C'}[0][0] = \boldsymbol{B}[1][1] \cdot \boldsymbol{C}[0][0] + \boldsymbol{B}[1][2] \cdot \boldsymbol{C}[0][1] + \boldsymbol{B}[2][1] \cdot \boldsymbol{C}[1][0] + \boldsymbol{B}[2][2] \cdot \boldsymbol{C}[1][1]$$

Dado un pixel $p = (r, g, b)$, definimos p^2 como (r^2, g^2, b^2) y $\sqrt{p}$ como $\left(\lfloor\sqrt{r}\rfloor, \lfloor\sqrt{g}\rfloor, \lfloor\sqrt{b}\rfloor\right)$. A partir de las imágenes G_x y G_y se genera una matriz $\boldsymbol{G}$ cuyo elemento en la posición $[i][j]$ se calcula como

$$\boldsymbol{G}[i][j] = \sqrt{G_x[i][j]^2 + G_y[i][j]^2}$$

De acuerdo a lo anterior, la matriz $\boldsymbol{G}$ contiene en cada entrada una tupla de tres números, los cuales podrías ser mayores que 255. Para transformar esta matriz a una imagen debemos 'normalizar' las tuplas que contienen números mayores a 255. Dada una tupla $t = (t_1, t_2, t_3)$ definimos el factor de normalización de dicha tupla como

$$n_t = \frac{max\{t_1, t_2, t_3, 255\}}{255}$$

Teniendo el factor de normalización, se define el pixel asociado a la tupla t como

$$p(t) = \left(\frac{t_1}{n_t}, \frac{t_2}{n_t}, \frac{t_3}{n_t}\right)$$

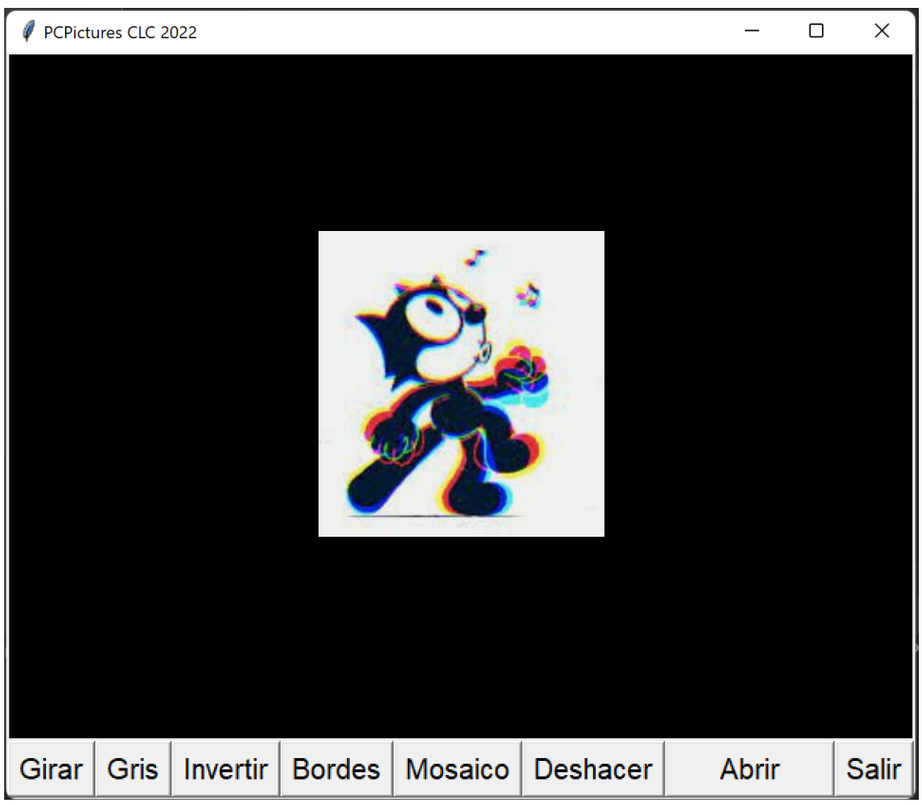
Finalmente, generamos una imagen $\boldsymbol{H}$ a partir de la normalización de tuplas de la matriz $\boldsymbol{G}$. Más precisamente, el pixel en la posición $[i][j]$ de la imagen $\boldsymbol{H}$ (imagen que muestra los bordes de $\boldsymbol{A}$) se calcula como

$$\boldsymbol{H}[i][j] = p(\boldsymbol{G}[i][j])$$

Librería

Al iniciar tu programa se abrirá una interfaz gráfica, como en la imagen más abajo, que contiene todos los elementos necesarios para que funcione el sistema de filtros. Para completar tu tarea, dispones de las siguientes funciones de la librería PCPictures_gui:

- `obtener_pixeles()`: Retorna la matriz de pixeles de la imagen que fue cargada usando el botón Abrir.
- `actualizar_imagen(pixeles)`: Actualiza la imagen que se muestra en pantalla usando la matriz de pixeles que recibe como parámetro.
- `esperar_click()`:Al llamar a esta instrucción el programa se detiene hasta que el usuario haga click en alguno de los botones de la interfaz que no sea el botón Abrir. Luego retorna el nombre en minúsculas del botón que se presionó. Por ejemplo, si el usuario presionó el botón Gris entonces esta función retornará la cadena de texto `gris`.
- `obtener_imagenes_mosaico()`: Al llamar a esta instrucción el programa procesará aquellas imágenes de tipo GIF (.gif) que estén contenidas en la carpeta mosaico. Luego de esto, la función retornará la lista de imágenes procesadas. Ten en cuenta que cada imagen es una matriz de pixeles.
- `salir()`: Ofrece al usuario salir del programa y, en caso de que el usuario acepte, cierra el programa.



- (Bonus 2) Guardar

Al implementar esta funcionalidad de modo satisfactorio se te otorgarán 10 pp.

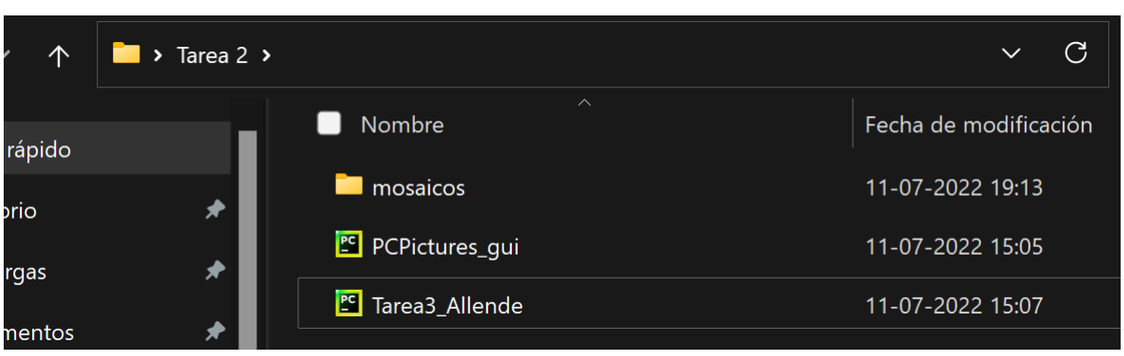
Agrega un botón nuevo a la interfaz gráfica que permita guardar en tu computador la imagen que se muestre en pantalla en formato gif, luego de aplicarle cuantos filtros se estimen convenientes. Los contenidos necesarios para abarcar este bonus no serán vistos en el curso, aun así, es completamente factible hacerlo, solo basta leer el código escrito en la librería PCPictures_gui e indagar sobre algunos métodos utilizados en ella.

Ten en cuenta que para lograr implementar esta funcionalidad deberás editar tu programa y la librería.

Una vez implementado el botón, la interfaz debería verse así:

Set up

Para que tu proyecto funcione correctamente, debes tener en una misma carpeta tu proyecto junto con los archivos que se encuentran junto a este enunciado en Github. La carpeta que contiene tu tarea debería verse así:



Indicaciones generales

1. La entrega de la tarea es a través del correo institucional (jallende@colegiolacruz.cl). Debes enviar un archivo de nombre Tarea2_Apellido.py, donde debes reemplazar Apellido por tu apellido.

2. Esta tarea es individual. Has firmado un código de honor y te has comprometido a trabajar de forma honesta. El plagio es castigado con nota 1.1. Se utilizará software de detección de plagio al revisar las tareas, si este indica un alto porcentaje de similitud a otro programa, se considerará que hubo plagio.

3. La fecha de entrega es el viernes 16 de Septiembre hasta las 23:59 pm. Tienes un mes para hacer entrega de tu tarea, no esperes al último minuto. No se revisarán las tareas que sean entregadas después de la fecha y hora indicadas y serán calificadas con nota 1.1.

4. Para desarrollar esta tarea no puedes importar ninguna librería a excepción de PCPictures_gui.

5. Revisa muy bien antes de enviar tu archivo, se corregirá tu primer envío y ningún otro.