 Institución Universitaria	INFORME FINAL DE TRABAJO DE GRADO	Código	FDE 089
		Versión	03
		Fecha	2015-01-27

## PROCESAMIENTO DE IMÁGENES USANDO PYTHON

Brayan Esteban Álvarez Echeverri

Victor Mora Campiño

Ingeniería de sistemas

María Constanza Torres Madroñero

**INSTITUTO TECNOLÓGICO METROPOLITANO**

**Abril/2018**

	INFORME FINAL DE TRABAJO DE GRADO	Código	FDE 089
		Versión	03
		Fecha	2015-01-22

## RESUMEN

Desde la aparición de los ordenadores en la década de los setenta, se puso claramente en evidencia su alto potencial para el tratamiento de información espacial, en campos de aplicación directamente relacionados con el estudio de propiedades del sistema de visión humana. A pesar de la enorme complejidad mostrada por el sistema de visión humana, gracias a los avances en tecnología y a la disminución en los precios del hardware necesario para su implementación, el progreso de las capacidades de visión por computadora ha estado ocurriendo a una velocidad vertiginosa. Las diversas aplicaciones han impulsado el uso de nuevas plataformas para el desarrollo de algoritmos para el análisis y procesamiento de imágenes. Uno de los lenguajes de programación que ha tomado fuerza en este campo es Python, muy similar a Matlab en su robustez y potencia, pero desarrollado bajo una licencia completamente Open Source y compatible con la librería OpenCV para el procesamiento y adquisición de imágenes. OpenCV a través de Python facilita enormemente el procesamiento análisis de imágenes en escala de grises y a color. En este producto de laboratorio se realiza una exploración de técnicas de transformación de intensidades, ecualización del histograma, modelos de color, filtrado y operaciones morfológicas usando OpenCV. Como entregables se proporciona una guía completa para el manejo de Python y OpenCV en Windows, así como una librería de funciones que permite el procesamiento de imágenes.

*Palabras clave:* Python, Matlab, RGB, histograma, operaciones morfológicas.

 Institución Universitaria	INFORME FINAL DE TRABAJO DE GRADO	Código	FDE 089
		Versión	03
		Fecha	2015-01-22

## RECONOCIMIENTOS

---

Nuestros más sinceros agradecimientos a la PHD María Constanza Torres Madroñero, por su acompañamiento e incansable actitud de docente y guía a lo largo no solo del proyecto, si no de nuestra formación profesional.

 Institución Universitaria	INFORME FINAL DE TRABAJO DE GRADO	Código	FDE 089
		Versión	03
		Fecha	2015-01-22

## ACRÓNIMOS

---

- *BSD: Berkeley Software Distribution*
- *RGB: Red, Green, Blue.*
- *HSI: Hue, Matiz, Intensity.*
- *CMY: Cian, Magenta, Yellow.*
- *MATLAB: Matrix Laboratory.*
- *OpenCV: Open Source Computer Vision.*
- *IDE: Integrated Developed Enviroment.*
- *PEP8: Python Enhancement Proposals.*

	<b>INFORME FINAL DE TRABAJO DE GRADO</b>	Código	FDE 089
		Versión	03
		Fecha	2015-01-22

# TABLA DE CONTENIDO

---

2.	MARCO TEÓRICO.....
3.	METODOLOGÍA .....
4.	CONCLUSIONES, RECOMENDACIONES Y TRABAJO FUTURO .....
	REFERENCIAS .....

	INFORME FINAL DE TRABAJO DE GRADO	Código	FDE 089
		Versión	03
		Fecha	2015-01-22

## 1. INTRODUCCIÓN

Hoy en día, el procesamiento de imágenes y la visión artificial es uno de los campos de mayor desarrollo e interés en el área de las ciencias computacionales. Las diversas aplicaciones han impulsado el uso de nuevas plataformas para el desarrollo de algoritmos para el análisis y procesamiento de imágenes. Uno de los lenguajes de programación que ha tomado fuerza en este campo es Python, muy similar a Matlab; un lenguaje de programación de alto nivel que busca el desarrollo de código legible de fácil portabilidad. Python permite el desarrollo de código abierto, siendo esta una gran ventaja sobre plataformas como Matlab. El objetivo de este trabajo de grado en el laboratorio es desarrollar una librería de funciones en Python que permita el procesamiento básico de imágenes. Para el desarrollo de este objetivo los estudiantes desarrollarán las siguientes actividades: 1. Elaboración de tutorial básico para el uso de Python (creación de variables, operaciones con arreglos y matrices, y funciones) 2. Programación de librería de funciones para la visualización de imágenes RGB y en escala de grises, transformación de intensidades, procesamiento del histograma, filtrado lineal y operaciones morfológicas. 3. Validación de las funciones usando imágenes reales.

Los objetivos de este producto son: 1. Explorar el uso de Python para el procesamiento de imágenes 2. Desarrollar una librería de funciones para el procesamiento básico de imágenes

Al finalizar el producto se realizará la entrega de:

1. Informe técnico que incluya el tutorial para el uso de Python y la documentación de las funciones.
2. Funciones en Python para visualización de imágenes a color y escala de grises, transformación de intensidades, procesamiento del histograma y filtrado lineal.

 Institución Universitaria	INFORME FINAL DE TRABAJO DE GRADO	Código	FDE 089
		Versión	03
		Fecha	2015-01-22

## 2. MARCO TEÓRICO

---

Python es un lenguaje de programación **interpretado** creado a finales de los 80's por Guido van Rossum en el Centro para las Matemáticas y la Informática (CWI, Centrum Wiskunde & Informatica), en los Países Bajos, como un sucesor del lenguaje de programación ABC, capaz de manejar excepciones e interactuar con el sistema operativo **Amoeba**. (PythonSoftwareFoundation, s.f.)

Python es un lenguaje de programación claro y poderoso orientado a objetos, comparable a Perl, Ruby, Scheme o Java. Algunas de las características notables de Python son:

- Sintaxis elegante, lo que facilita la lectura de los programas.
- Fácil de usar lo que simplifica el funcionamiento de su programa. Python es ideal para el desarrollo de prototipos y otras tareas de programación ad-hoc, sin comprometer el mantenimiento.
- Incluye una gran biblioteca estándar que admite muchas tareas de programación comunes, como conectarse a servidores web, buscar texto con expresiones regulares, leer y modificar archivos.
- El modo interactivo de Python hace que sea fácil probar pequeños fragmentos de código. También hay un entorno de desarrollo incluido llamado IDLE.
- Se extiende fácilmente agregando nuevos módulos implementados en un lenguaje compilado como C o C++.
- También se puede incrustar en una aplicación para proporcionar una interfaz programable.
- Se ejecuta en cualquier sistema operativo, incluidos Mac OS X, Windows, Linux y Unix, con versiones no oficiales también disponibles para Android y iOS.

	INFORME FINAL DE TRABAJO DE GRADO	Código	FDE 089
		Versión	03
		Fecha	2015-01-22

- Es software libre en dos sentidos. No cuesta nada descargar o usar Python, o incluirlo en su aplicación. Python también se puede modificar y redistribuir libremente, porque si bien el idioma está protegido por derechos de autor, está disponible bajo una licencia de código abierto (Open Source Initiative, s.f.).

Debido a que Python se ejecuta en su propia “máquina virtual”, hay varias formas de instalarlo. La primera, es instalar el intérprete de forma nativa y trabajarlo directamente desde la línea de comandos de nuestro o equipo, o el IDE de nuestra elección. (Esta opción es un poco más avanzada, y recomendada para usuarios con experiencia previa en Python).

La segunda (y con la que trabajaremos en este curso), es instalar un IDE que incluye el intérprete de Python, una consola de depuración, y un gestor de soluciones. El escogido para este manual y para el curso se llama “**Anaconda**”, debido a que es una distribución de código abierto de los lenguajes de programación Python y R. Se utiliza para procesamiento de datos a gran escala, análisis predictivo y computación científica. Tiene como objetivo simplificar la administración y el despliegue de paquetes. A continuación, se describen algunas de sus características:

- Provisión de la consola IPython (Qt) como un mensaje interactivo, que puede mostrar gráficos en línea
- Capacidad de ejecutar fragmentos de código desde el editor de la consola
- Análisis continuo de archivos en el editor y provisión de advertencias visuales sobre posibles errores
- Ejecución paso a paso
- Explorador de variables

Todo el código fuente desarrollado en el presente trabajo, conserva el estilo de notación definido en la **PEP8** (Cologne, s.f.).



	<b>INFORME FINAL DE TRABAJO DE GRADO</b>	Código	FDE 089
		Versión	03
		Fecha	2015-01-22

Como en la mayoría de los lenguajes de programación de alto nivel, en Python se compone de una serie de elementos que alimentan su estructura. Entre ellos, podremos encontrar los siguientes:

**Variables:** Una variable es un espacio para almacenar datos modificables, en la memoria de un ordenador. En Python, una variable se define con la sintaxis:

```
nombre_de_la_variable = valor_de_la_variable
```

Cada variable, tiene un nombre y un valor, el cual define a la vez, el tipo de datos de la variable.

Existe un tipo de “variable”, denominada constante, la cual se utiliza para definir valores fijos, que no requieran ser modificados.

Python nos provee de reglas de estilos, con el fin de poder escribir código fuente más legible y de manera estandarizada. Estas reglas de estilo son definidas a través de la Python Enhancement Proposal N° 8 , las cuales serán mencionadas en este tutorial (PythonSoftwareFoundation, s.f.).

### **PEP 8: variables**

Utilizar nombres descriptivos y en minúsculas. Para nombres compuestos, separar las palabras por guiones bajos. Antes y después del signo =, debe haber uno (y solo un) espacio en blanco.

Correcto: `mi_variable = 12`

Incorrecto: `MiVariable = 12` | `mivariable = 12` | `mi_variable=12` | `mi_variable = 12`

### **PEP 8: constantes**

Utilizar nombres descriptivos y en mayúsculas separando palabras por guiones bajos.

Ejemplo: `MI_CONSTANTE = 12`

	<b>INFORME FINAL DE TRABAJO DE GRADO</b>	Código	FDE 089
		Versión	03
		Fecha	2015-01-22

Para imprimir un valor en pantalla, en Python, se utiliza la palabra clave print:

```
mi_variable = 15
```

```
print mi_variable
```

Lo anterior, imprimirá el valor de la variable mi\_variable en pantalla.

### **Tipos de datos**

Una variable (o constante) puede contener valores de diversos tipos. De una manera general, se observan 5 tipos de datos: enteros, de coma flotante, cadenas, booleanos y de tipo objeto. En la imagen siguiente se observan algunos ejemplos de los tipos de datos básicos.

Lo primero que se debe hacer es declarar y asignar un valor inicial a las variables en el editor.

### **Procesamiento de Imágenes:**

En Python es posible trabajar con el procesamiento de imágenes, utilizando las librerías nativas del lenguaje diseñadas para tal fin, como lo son “Matplotlib” y “numpy”. Ambas vienen instaladas por defecto con el paquete de Anaconda. Sin embargo, para fines más sofisticados, como lo son el filtrado de imágenes, escalamiento de colores, detección de formas, bordes y fines prácticos de la visión artificial, se hace necesario utilizar paquetes de terceros que cuentan con implementaciones más robustas. Es el caso de librerías como “Scikit-image”, creado por Scipy.org (una organización dedicada al análisis de datos matemáticos, ciencias e ingenierías), “Pillow” una variante PIL, desarrollado por Alex Clark a finales del año 2009 (y sostenida actualmente por varios colaboradores) y la librería la cual será nuestra fuente de datos para este curso “OpenCV”, del gigante de tecnología Intel.

### **¿Qué es OpenCV?**

OpenCV (Open Source Computer Vision) es una librería de programación de código abierto dirigida principalmente a la visión por computador en tiempo

	<b>INFORME FINAL DE TRABAJO DE GRADO</b>	Código	FDE 089
		Versión	03
		Fecha	2015-01-22

real, desarrollada por la división rusa de Intel en el centro de Nizhni Nóvgorod. Actualmente también cuenta con el apoyo de Willow Garage y la compañera de visión Itseez. El uso es gratuito bajo la licencia open source BSD. La librería OpenCV es multiplataforma. Está optimizada para ser usada en procesadores Intel, porque si la librería detecta que las librerías de Intel IPP (Integrated Performance Primitives) se encuentran en el sistema, hará uso automáticamente para acelerar el rendimiento de la aplicación. También cuenta con apoyo de SIMD, optimizaciones OpenMP, optimizaciones para Intel TBB (Threading Building Blocks) y a partir de la versión 2.4.8 apoya instrucciones vectoriales NEON para sistemas ARM (OpenCV.org, s.f.).

OpenCV permite desarrollar en C, C ++ o Python y es compatible con el IDE QT Creator y sus correspondientes librerías QT.

Áreas de Aplicación:

- Odometria visual
- Sistema de reconocimiento facial
- Reconocimiento de gestos
- La interacción hombre-ordenador (HCI)
- Robótica móvil
- Comprensión del movimiento
- Identificación del objeto
- Segmentación y reconocimiento
- Visión estereoscópica: percepción de profundidad desde 2 cámaras
- Estructura de movimiento (SFM)
- Rastreo de movimiento
- Realidad aumentada

	INFORME FINAL DE TRABAJO DE GRADO	Código	FDE 089
		Versión	03
		Fecha	2015-01-22

### 3. METODOLOGÍA

---

1. Transformación gamma de una imagen: recibe una imagen en escala de grises como parámetro, imagen, y retorna el cálculo de la transformación gamma dado por la ecuación:

$I_{\text{gamma}} = I_m \cdot \gamma$   
 Imagen\_gamma=ajustar\_gamma(I<sub>m</sub>,001)

2. Umbralización de una imagen: esta función recibe como parámetros una imagen escala de grises y un umbral (margen), y aplica los diferentes tipos de umbralización. Si el margen está compuesto por 2 valores, se aplica la umbralización binaria (los valores por debajo del límite inferior se hacen 0 y los valores por encima del límite superior se hacen 255).

```

import cv2
from UmbralizacionImagen import umbralizar_imagen as ui
imagen=cv2.imread('salto.jpg')
imagenGris = cv2.cvtColor(imagen, cv2.COLOR_BGR2GRAY)
ui(imagenGris)

```

3. Conversión de imagen en formato HSL a RGB: esta función recibe como parámetro una imagen en formato HSL y la transforma en formato RGB, mostrando tanto la imagen original, como la transformada y cada uno de sus canales.

```

import cv2
from hsl2rgb import convertir_hsl_2_rgb as ch
imagenHSL = cv2.imread('lenaHsv.png') #imagen en formato HSL
ch(imagenHSL)

```

4. Conversión de imagen en formato RGB a HSL: esta función recibe como parámetro una imagen en formato RGB y la transforma en formato

 Institución Universitaria	INFORME FINAL DE TRABAJO DE GRADO	Código	FDE 089
		Versión	03
		Fecha	2015-01-22

HSL, mostrando tanto la imagen original, como la transformada y cada uno de sus canales.

```
import cv2
from rgb2hsl import convertir_rgb_2_hsl as ch
imagenRGB = cv2.imread('tigre.jpg') #imagen en formato HSL
ch(imagenRGB)
```

5. Conversión de imagen en formato RGB a CMY: esta función recibe como parámetro una imagen en formato RGB y la transforma en formato CMY. Para esto, pre-procesa la imagen, convirtiéndola en formato de punto flotante. Muestra tanto la imagen original, así como la imagen resultante y cada uno de sus canales.

```
Import cv2
from rgb2cmy import convertir_rgb_2_cmy as ch
imagenRGB = cv2.imread('tigre.jpg') #imagen en formato RGB
ch(imagenRGB)
```

6. Filtro Gaussiano: esta función toma como parámetro una imagen en escala de gris, la cual contiene ruido, una ventana y aplica el filtro gaussiano en dicha imagen en la región indicada.

```
import cv2
import matplotlib.pyplot as plt
from ruido_gaussiano import agregar_ruido_gaussiano as rg
from filtro_Gaussiano import aplicar_filtro_gaussiano as fg
sigma_ruido=35
imagen_original= cv2.imread('tigre.jpg',0)
imagen_ruidosa = rg(imagen_original,sigma_ruido)
imagen_filtrada=fg(imagen_ruidosa,(5,5))
```

7. Filtro Convolución: esta función toma como parámetro una imagen en escala de gris, la cual contiene ruido, una ventana y aplica el filtro convolución en dicha imagen en la región indicada.

	INFORME FINAL DE TRABAJO DE GRADO	Código	FDE 089
		Versión	03
		Fecha	2015-01-22

```

import cv2
import matplotlib.pyplot as plt
import numpy as np
from filtro_Convolucion import aplicar_filtro_convolucion as fc
imagen_original= cv2.imread('tigre.jpg',0)
imagen_ruidosa = imagen_original + 3 * imagen_original.std() *
np.random.random(imagen_original.shape)
imagen_filtrada=fc(imagen_ruidosa,(5,5))

```

8. Filtro mediana: esta función toma como parámetro una imagen en escala de gris, la cual contiene ruido y aplica el filtro mediana. Es ideal para imágenes que presentan ruido tipo sal y pimienta.

```

import cv2
import matplotlib.pyplot as plt
import numpy as np
from ruido_sal_pimienta import agregar_ruido_sal_pimienta as rsp
#agregar ruido sal pimienta
from filtro_Mediana import aplicar_filtro_mediana as fm
imagen_original= cv2.imread('tigre.jpg')
imagen_ruidosa = rsp(imagen_original)
imagen_filtrada=fm(imagen_ruidosa,5)

```

9. Filtro Promedio: esta función toma como parámetro una imagen en escala de gris y una ventana, la cual contiene ruido y aplica el filtro promedio basado en dicha ventana.

```

import cv2
import matplotlib.pyplot as plt
import numpy as np
from filtro_Promedio import aplicar_filtro_promedio as fp
imagen_original= cv2.imread('tigre.jpg',0)
imagen_ruidosa = imagen_original + 3 * imagen_original.std() *
np.random.random(imagen_original.shape)
ventana=(5,5)
imagen_filtrada=fp(imagen_ruidosa,ventana)

```

	INFORME FINAL DE TRABAJO DE GRADO	Código	FDE 089
		Versión	03
		Fecha	2015-01-22

10. Filtro Laplaciano: esta función recibe como parámetro una imagen en escala de grises y aplica el filtro laplaciano. Este tipo de filtro es muy utilizado para detectar bordes y contornos.

```
import cv2
import matplotlib.pyplot as plt
import numpy as np
from Filtro_Laplaciano import aplicar_filtro_laplaciano as fl
imagen_original = cv2.imread('edificios.jpg',0)
imagen_filtrada=fl(imagen_original)
```

11. Filtro Sobel: esta función recibe como parámetro una imagen en escala de grises y aplica el filtro sobel para tratar de calcular su gradiente, tanto en el eje X como en el Y

```
import cv2
import matplotlib.pyplot as plt
import numpy as np
from filtro_Sobel import aplicar_filtro_sobel as fl
imagen_original = cv2.imread('zebra.jpg',0)
imagen_filtrada1, imagen_filtrada2=fl(imagen_original)
```

12. Erosión: Requiere como parámetro de entrada una imagen binaria, esta función es útil para eliminar pequeños ruidos blancos, separar dos objetos conectados, etc.

```
import cv2
import numpy as np
img = cv2.imread('A.png',0)
kernel = np.ones((7,7),np.uint8)
erosion = cv2.erode(img,kernel,iterations = 1)
```

13. Dilatación: La función de dilatación es lo opuesto a la erosión. Recibe como entrada una imagen binaria, en este proceso un elemento de píxel es '1' si al menos un píxel de la imagen de los que caen dentro de la ventana del kernel es '1'. De tal forma que la dilatación aumenta el tamaño de los objetos de primer plano, es decir, la región blanca.

	INFORME FINAL DE TRABAJO DE GRADO	Código	FDE 089
		Versión	03
		Fecha	2015-01-22

```

import cv2
import numpy as np
img = cv2.imread('A.png',0)
kernel = np.ones((7,7),np.uint8)
dilatacion = cv2.dilate(img,kernel,iterations = 1)

```

14.Apertura: La función de apertura consiste en la erosión de la imagen seguida de una dilatación, es útil para eliminar el ruido.

```

import cv2
import numpy as np
img = cv2.imread('A2.png',0)
kernel = np.ones((7,7),np.uint8)
apertura = cv2.morphologyEx(img, cv2.MORPH_OPEN, kernel)

```

15.Cierre: La función Cierre es el opuesto de Apertura, es decir, dilatación seguida de erosión. Es útil para cerrar pequeños agujeros dentro de los objetos de primer plano, o pequeños puntos negros en el objeto.

```

import cv2
import numpy as np
img = cv2.imread('A3.png',0)
kernel = np.ones((7,7),np.uint8)
cierre = cv2.morphologyEx(img, cv2.MORPH_CLOSE, kernel)

```

16.Gradient: Esta función es la diferencia entre la dilatación y la erosión de una imagen. El resultado se verá como el contorno del objeto.

```

import cv2
import numpy as np
img = cv2.imread('B.png',0)
kernel = np.ones((7,7),np.uint8)
gradiente = cv2.morphologyEx(img, cv2.MORPH_GRADIENT, kernel)

```

17.Histograma de una Imagen a escala de grises: La función histograma recibe como parámetro una imagen a escala de grises y muestra una gráfica en donde se puede apreciar la frecuencia con las que aparecen los distintos niveles de intensidad, normalmente el nivel de intensidad



	INFORME FINAL DE TRABAJO DE GRADO	Código	FDE 089
		Versión	03
		Fecha	2015-01-22

está en el rango de 0 a 255, en donde el valor 0 representa el color negro y 255 el color blanco, utilizando el histograma de una imagen podemos modificar sus características, por ejemplo, el brillo y contraste.

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
imagen_original = cv2.imread('lena.png',0)
plt.hist(imagen_original.ravel(),256,[0,256]);
plt.show()
imagen_ecualizada = cv2.equalizeHist(imagen_original)
plt.hist(imagen_ecualizada.ravel(),256,[0,256]);
res = np.hstack((imagen_original,imagen_ecualizada)) #Agrupa las imágenes una al lado de la otra
```

18. Histograma de una Imagen a color: Esta función necesita como entrada una imagen a color, lo que obtenemos como resultado es una gráfica que muestra la frecuencia con la que aparecen los distintos niveles de cada uno de los canales, azul, verde y rojo.

Se recorren los tres canales de la imagen y se calcula el histograma para cada uno, se asigna un color diferente a cada uno, de tal forma que es posible identificarlo en la gráfica.

```
import cv2
from matplotlib import pyplot as plt
img = cv2.imread('tigre.jpg')
cv2.imshow('tigre.jpg', img)
color = ('b','g','r')
for i, c in enumerate(color):
    hist = cv2.calcHist([img], [i], None, [256], [0, 256])
    plt.plot(hist, color = c)
    plt.xlim([0,256])
```

	INFORME FINAL DE TRABAJO DE GRADO	Código	FDE 089
		Versión	03
		Fecha	2015-01-22

#### 4. CONCLUSIONES, RECOMENDACIONES Y TRABAJO FUTURO

Este apartado es el corazón del manuscrito, donde la mayoría de los lectores irán después de leer el resumen. Se recomienda tener en cuenta las siguientes consideraciones:

- Python es un lenguaje que se hizo muy popular en corto tiempo, principalmente debido a su simplicidad y legibilidad de código. Permite al programador expresar sus ideas en menos líneas de código sin reducir la legibilidad. Además de ser completamente open source, se integra perfectamente con librerías específicas para la implementación de operaciones relacionadas con la visión artificial, de forma legible y amigable. Pese a que las operaciones que incluyen modificaciones en los valores de cada pixel pueden ralentizarse debido a que es un lenguaje más lento que los de bajo nivel, existen opciones para sortear dicho obstáculo como la instalación de Cython, un híbrido entre el lenguaje C y Python.
- OpenCV no cuenta con una función nativa para la transformación de imágenes en formato RGB a CMY y viceversa, por lo tanto, es necesario aplicar una función matemática basada en manejo de vectores y matrices.
- Las aplicaciones de reconocimiento facial en tiempo real, así como el análisis de video son potenciales frentes de acción para futuros trabajos.

 Institución Universitaria	INFORME FINAL DE TRABAJO DE GRADO	Código	FDE 089
		Versión	03
		Fecha	2015-01-22

## REFERENCIAS

---

Cologne, P. (s.f.). *python.org*. Obtenido de

<https://www.python.org/dev/peps/pep-0008/>


*Open Source Initiative*. (s.f.). Obtenido de <https://opensource.org/>

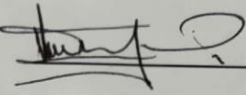
*OpenCV.org*. (s.f.). Obtenido de <https://opencv.org/about.html>

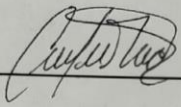
*PythonSoftwareFoundation*. (s.f.). Obtenido de

<https://docs.python.org/3/license.html?highlight=history>

 Institución Universitaria	<b>INFORME FINAL DE TRABAJO DE GRADO</b>	Código	FDE 089
		Versión	03
		Fecha	2015-01-22

 Institución Universitaria	<b>INFORME FINAL DE TRABAJO DE GRADO</b>	Código	FDE 089
		Versión	03
		Fecha	2015-01-22

FIRMA ESTUDIANTES  \_\_\_\_\_  
Mayra Alvarez \_\_\_\_\_

FIRMA ASESOR  \_\_\_\_\_

FECHA ENTREGA: \_\_\_\_\_

FIRMA COMITÉ TRABAJO DE GRADO DE LA FACULTAD \_\_\_\_\_

RECHAZADO \_\_\_\_\_ ACEPTADO \_\_\_\_\_ ACEPTADO CON MODIFICACIONES \_\_\_\_\_

ACTA NO. \_\_\_\_\_

FECHA ENTREGA: \_\_\_\_\_

 Institución Universitaria	INFORME FINAL DE TRABAJO DE GRADO	Código	FDE 089
		Versión	03
		Fecha	2015-01-22

FIRMA CONSEJO DE FACULTAD \_\_\_\_\_

ACTA NO. \_\_\_\_\_

FECHA ENTREGA: \_\_\_\_\_