

Primeros pasos con Python:

Manipulando imágenes

Rodolfo Ferro

[@FERRORODOLFO](https://twitter.com/FERRORODOLFO)

Abril 29, 2017

Universidad de Guanajuato
CIMAT A.C.



Tabla de contenido

1. Sobre Python
2. Instalación
3. Primeros pasos
4. Manipulando imágenes
5. Creando imágenes

Sobre Python

*Python es un lenguaje interpretado, sencillo,
versátil y poderoso. La belleza del lenguaje
radica en su sintaxis.*



Un poco de historia (I)

- Python fue creado a inicios de los 90's por Guido van Rossum en el *Stichting Mathematisch Centrum* (CWI), en los Países Bajos, como un sucesor del lenguaje ABC.
- Guido sigue siendo el principal autor, aunque incluye muchas contribuciones por parte de otros.

Un poco de historia (II)

- En mayo del 2000, Guido y el equipo core de desarrollo de Python se mudan a *BeOpen.com* para formar el equipo BeOpen PythonLabs.
- En octubre del mismo año, los PythonLabs se mudan a Digital Creations (hoy Zope Corporation) y en 2001 se crea la *Python Software Foundation* (PSF), una organización sin fines de lucro creada específicamente para poseer todo lo relacionado con propiedad intelectual sobre Python.

Python en el mundo del SL

- Todos los releases de Python son Open Source.
- De acuerdo a la *Free Software Foundation*:

The screenshot shows a web browser displaying the Free Software Foundation (FSF) website at fsf.org. The page is titled "Python". The FSF logo is visible in the top left, and a "JOIN FSF NOW" button is in the top right. The main navigation menu includes links for about, campaigns, licensing, membership, resources, community, donate, and shop. Below the menu, a sub-navigation bar includes "Meet the free software gang >" and a search bar with a "Search" button. The main content area features a red background with a large black cartoon snake logo. Text on the page describes Python as a multi-purpose programming language and its popularity among developers. A sidebar on the right contains a "Sign up" form for newsletter subscribers, with fields for email address and a "Subscribe me" button. At the bottom, there's a "News" section with a link to a SecureDrop and Alexandre Oliva award article from March 25, 2017.

Meet the free software gang >

Python

by sdubois Contributions — Published on Jun 24, 2010 12:53 PM

Python is a multi-purpose programming language that packs a punch. A popular choice for developers, Python makes writing complex programs less difficult through its clear syntax. Python can be used for writing near anything, from a small desktop application to a powerful web application.

While there are many programming languages for one to choose from, Python is the favorite of many programmers in the free software community.

- Find more programming languages in the Free Software Directory



SecureDrop and Alexandre Oliva are 2016 Free Software Awards winners

Mar 25, 2017

Recordando un poco...

Un **Software Libre** es aquel que respeta las cuatro libertades que la *FSF* establece:

- La libertad de usar el programa con cualquier propósito.
- La libertad de estudiar cómo funciona el programa y modificarlo, adaptándolo a tus necesidades.
- La libertad de distribuir copias del programa, con lo cual puedes ayudar a tu próximo.
- La libertad de mejorar el programa y hacer públicas esas mejoras a los demás, de modo que toda la comunidad se beneficie.

Nota: El Software Libre no es necesariamente gratuito.

Instalación

Instalando Python

- Ir al sitio oficial: <https://www.python.org/downloads/>
- Descargar la versión 3.6.x.

The screenshot shows the Python Software Foundation website's main page. At the top, there is a navigation bar with links for Python, PSF, Docs, PyPI, Jobs, and Community. Below the navigation bar, the Python logo is displayed next to the word "python". A search bar with a magnifying glass icon and a "GO" button is present. To the right of the search bar are links for "Socialize" and "Sign In". The main content area features a large banner with the text "Download the latest version for Mac OS X" and two download buttons: "Download Python 3.6.1" and "Download Python 2.7.13". Below the banner, there is a link to "Wondering which version to use? Here's more about the difference between Python 2 and 3.". Further down, there are links for "Looking for Python with a different OS? Python for Windows, Linux/UNIX, Mac OS X, Other" and "Want to help test development versions of Python? Pre-releases". To the right of the text, there is a graphic of two brown boxes hanging from yellow and white striped parachutes against a blue sky with white clouds. At the bottom of the page, there is a section titled "Looking for a specific release?" with a link to "Python releases by version number:". This section includes a table with three rows of data:

Release version	Release date	Click for more
Python 3.6.1	2017-03-21	Download Release Notes
Python 3.4.6	2017-01-17	Download Release Notes

Instalando Python

En Windows:

- Preferentemente instalarlo en el disco local c://.
- Dar check a la opción "Add Python to PATH".

En Linux/Unix:

- No hace falta algo adicional, todo es hermoso.
- Es más, por default ya viene instalada la versión 2.7 de Python.

Instalando paquetes

Al instalar Python, se instala **pip** (*Python Package Index*), que es un gestor de paquetes de Python. Usando **pip** instalaremos:

- **NumPy**, que incluye herramientas de métodos numéricos.
- **Matplotlib**, que incluye herramientas de visualización.

Nota: Python se instala con su librería estándar. Lo recomendable es instalar una distribución (como *Anaconda*) para una mejor gestión de paquetes, además de instalarse por default los más usados.

Paquetes de manipulación de imágenes

Existen diversos paquetes para manipular imágenes:

- **PIL** (*Python Imaging Library*)
- **OpenCV** (*OpenSource Computer Vision*)
- **scikit-image** (*Scientific Kit - Image*)
- Etc.

Cada uno trata de manera distinta las imágenes y se enfocan en cosas diferentes. Como es un taller de primeros pasos, no usaremos alguno de estos paquetes, sino que trabajaremos desde cero, sólo con **numpy** y **matplotlib**.

Instalando numpy y matplotlib

Abriremos una consola y ejecutaremos los siguientes comandos.

Para instalar numpy:

```
$ sudo pip install numpy
```

Para instalar matplotlib:

```
$ sudo pip install matplotlib
```

Nota: En Windows se omite el comando `sudo`.

Primeros pasos

El prompt de Python



Lo básico (I)

La belleza de Python radica en la sintaxis, los bloques de código deben tener la misma indentación, pues las llaves {} se utilizan para una estructura de datos (diccionarios).

Un ejemplo:

```
while True:  
    # El bloque de código dentro del ciclo debe  
    # estar en el mismo nivel de indentación  
  
    print("Infinite loop.")
```

Lo básico (II)

- No hace falta terminar cada línea con punto y coma.
- Los scripts de Python pueden ejecutarse desde consola, usando un IDE o con Jupyter notebooks.
- Todos los scripts de Python deben ser de extensión .py (excepto por los Jupyter notebooks).
- Para fines prácticos, nosotros correremos scripts desde consola y escribiremos código desde un bloc de notas, por lo que es recomendable instalar uno tipo Atom o Sublime Text.

Variables y operaciones (I)

- Las variables no se declaran, sólo se definen y Python sabe el tipo de variable que se está utilizando.
- Python cuenta con 5 tipos de datos estándar: números, cadenas de texto, listas, tuplas y diccionarios.
- Puedes realizar asignación múltiple.
- Las operaciones básicas con números son: +, -, *, /, //, **.
- En las cadenas de texto también puede usarse + (concatenación) y * (repeticIÓN).

Variables y operaciones (II)

Los tipos de variables que existen en Python son:

- **Enteros**, como 3.
- **Longs**, como 51924361L.
- **Flotantes**, como 3.14e4.
- **Complejos**, como 4.53-7j.
- **Strings**, como "Hola mundo".
- **Listas**, como [1., 3, 5.3, "perrito"].
- **Tuplas**, como (1., 3, 5.3, "perrito").
- **Diccionarios**, como {"Nombre":"Rodolfo", "Apellido":"Ferro"}

Condicionales

La sintaxis correspondiente es:

```
if/elif/else
    if condition:
        # Block content
    elif condition:
        # Block content
    else:
        # Block content
```

Ciclos

La sintaxis correspondiente es:

```
for
    for idx in iterable:
        # Block content
```

```
while
    while condition:
        # Block content
```

Funciones

La sintaxis correspondiente es:

Funciones

```
def function_name(parameters):  
    # Block content  
    return values
```

No hay limitación en los parámetros y valores de retorno, en el sentido de puede pasársele cualquier variable o función como argumento y cualesquiera variables pueden ser retornadas en una función.

Para concluir la sección...

Para poder concluir la sección y comenzar a tirar nuestras primeras líneas de código en Python, tengamos claro lo siguiente:

- ¿Qué es slicing?
- ¿Por qué conviene utilizar arreglos de NumPy en vez de listas de Python?
- Computacionalmente, ¿qué es una imagen?
- ¿Cuántas dimensiones tiene una imagen?
- ¿Hay alguna otra pregunta?



Repositorio del taller:

<https://github.com/RodolfoFerro/FLISoL17>

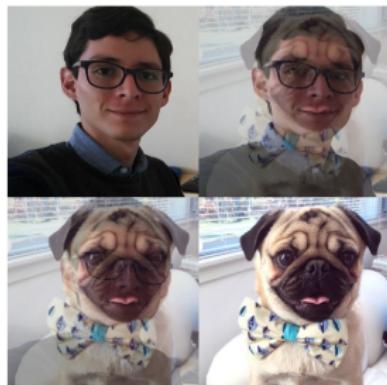
En este punto, descargar solamente la carpeta de imágenes.

Manipulando imágenes

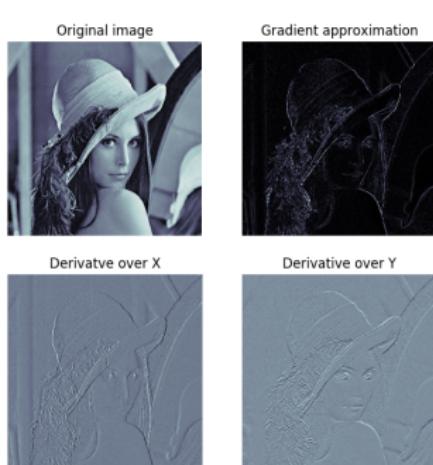
Objetivos

Queremos ser capaces de recrear las siguientes imágenes:

When sabes programar y te programas una transición de opacidad $(1 - \alpha)A + \alpha B$ para dos imágenes A y B , con $\alpha \in [0, 1]$ y te conviertes en un meme de pug:



This plot was created with the magic of Python by @FernRaddatz.



Para ello...

Lo que necesitamos aprender es lo siguiente:

1. Desplegar una imagen
2. Operar una imagen
3. Realizar una transición de opacidad
4. Acceder a pixeles y subsecciones
5. Cortando imágenes
6. Programar derivadas

Importando NumPy y Matplotlib

Se pueden importar librerías simplemente haciendo `import librería`, sin embargo, se pueden importar módulos específicos (ej. `from matplotlib import pyplot`) o importar módulos/librerías con un alias.

Por simplicidad nosotros importaremos **NumPy** y **Matplotlib** como sigue:

Importamos usando un alias:

```
import numpy as np  
import matplotlib.pyplot as plt
```

1. Desplegando una imagen

Matplotlib tiene una función llamada `imread`, cuyo argumento es una dirección y sirve para leer una imagen y guardarla como un arreglo de NumPy.

Podemos imprimir la matriz usando la función `print` y mostrarla en una ventana haciendo lo que sigue:

Después de importar...

```
img = plt.imread("../imgs/Lenna.png")
plt.imshow(img)
plt.show()
```

Script: `show_img.py`

2. Operando una imagen

Dado que las imágenes son matrices, es muy fácil operar imágenes utilizando NumPy.

Basta hacer la operación directa en la imagen o utilizar alguna función de NumPy para operar:

Después de importar y cargar una imagen inicial...

```
opr = img ** 0.5  
opr = np.cos(img)  
plt.imshow(opr)  
plt.show()
```

Script: operating_imgs.py

3. Transición de opacidad

Una transición de opacidad para dos imágenes A y B está definida como sigue:

$$op_img = (1 - \alpha) A + \alpha B,$$

tomando valores de α en el intervalo $[0,1]$.

Por ejemplo el primer estado intermedio:

Habiendo cargado una imagen inicial y una final...

```
op_img = 0.7*im_i + 0.3*im_f  
plt.imshow(op_img)  
plt.show()
```

Script: `opacity.py`

4. Accediendo a pixeles y subsecciones

Dado que las imágenes son matrices, accedemos a un pixel como si accediéramos a un elemento cualquiera de la matriz que representa a la imagen:

$$\text{pixel}_{\{i,j\}} = \text{img}[i, j]$$

De la misma manera, utilizando slicing podemos acceder a regiones de la imagen:

$$\text{region}_{\{x \times y\}} = \text{img}[y_i : y_j, x_i : x_j]$$

5. Cortando una imagen

Con lo anterior, es muy fácil cortar y obtener una nueva imagen:

$$\text{region}_{\{X \times Y\}} = \text{img}[y_i : y_j, x_i : x_j],$$

simplemente hace falta guardar el resultado.

Habiendo importado librerías...

```
img = plt.imread("Lenna.png")
cropped_img = img[220:300, 210:370]
plt.imshow(cropped_img)
plt.show()
img = plt.imsave("Lenna.png", cropped_img)
```

Script: crop_img.py

6. Derivando una imagen

La derivada de una imagen no es más que un barrido calculando las diferencias entre el pixel en el que te encuentras y el anterior:

$$diff_x = img[:, 1:j] - img[:, 0:j-1]$$

$$diff_y = img[1:i, :] - img[0:i-1, :]$$

Con estas derivadas podemos construir una aproximación del gradiente de una imagen:

$$grad \sim (diff_x^2 + diff_y^2)^{0.5}$$

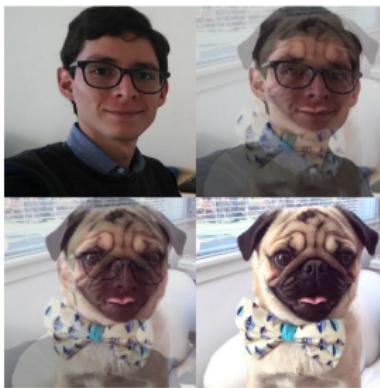
Script: `derivatives.py`

Reto para llevar

Reto 1:

Dar formato a la transición de opacidad para que quede como sigue:

When sabes programar y te programas una transición de opacidad $(1 - \alpha)A + \alpha B$ para dos imágenes A y B , con $\alpha \in [0, 1]$ y te conviertes en un meme de pug:



This plot was created with the magic of Python by @FernRodolfo.

Reto para llevar

Reto 2:

Nuestras funciones de derivadas de imagen funcionan sólo con imágenes que son cuadradas, ¿por qué?

¿Cómo **corregir** esto para que sirva con la imagen `leaf.png`?
(HINT: Sobel operator.)



Para concluir la sección...

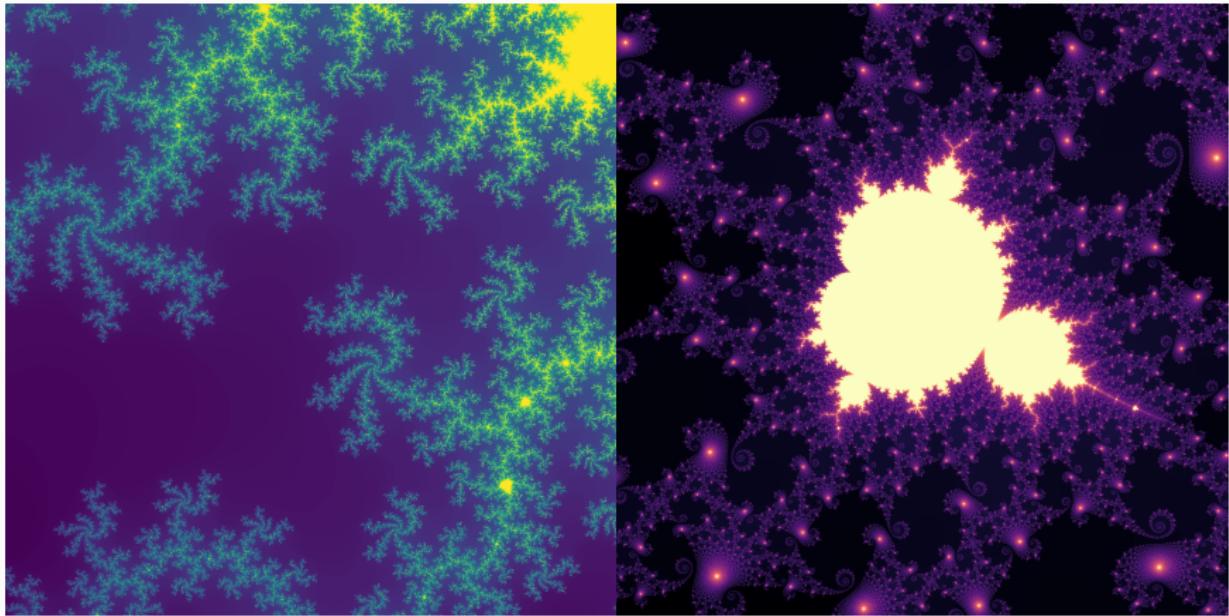
Para poder concluir la sección y continuar con la siguiente, respondamos:

- ¿Resulta sencillo manipular imágenes (arreglos de NumPy, en general)?
- ¿Por qué las derivadas de las imágenes ayudan a encontrar bordes?
- ¿Hay alguna otra pregunta?

Creando imágenes

Objetivos

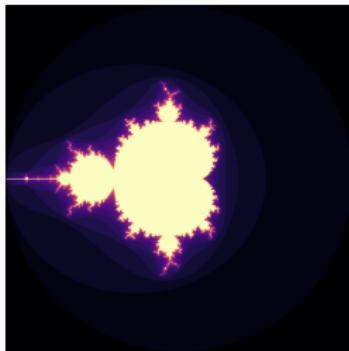
Queremos ser capaces de recrear las siguientes imágenes:



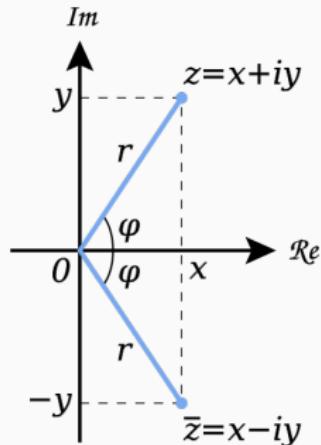
¿Por qué un fractal?

Como una imagen es una matriz, es muy fácil que seamos capaces de crear una imagen con contenido basura, por lo que resulta ser más interesante construir una imagen a partir de un ente geométrico.

Primero necesitamos entender cómo se construye un fractal, en particular, el Conjunto de Mandelbrot.



Construyendo el conjunto de Mandelbrot (I)



Recordando que un complejo es un número de la forma $x + iy$ y que tiene una representación en el plano; el conjunto de Mandelbrot se construye al iterar un número complejo z en la función $f_c(z) = z^2 + c$, para un complejo c .

Construyendo el conjunto de Mandelbrot (II)

Por ejemplo, comenzando en $z = 0 + i0 = 0$, para el complejo $c = 0.25 + i0 = 0.25$, tendríamos:

$$f_{0.25}(0) = 0^2 + 0.25 = 0.25$$

$$f_{0.25}(0.25) = 0.25^2 + 0.25 = 0.625 + 0.25 = 0.65$$

$$f_{0.25}(0.65) = 0.65^2 + 0.25 = 0.4225 + 0.25 = 0.4235$$

$$\vdots \quad = \quad \vdots$$

De aquí podemos notar que la sucesión no crece arbitrariamente. Sin embargo, ¿qué sucede con otros números, como $1(1 + i0)$?

Construyendo el conjunto de Mandelbrot (III)

- La idea es crear una función que itere un complejo dado hasta un máximo de iteraciones (en caso de no crecer arbitrariamente).
- Teniendo dicha función, habrá que usarla dentro de otra para que vaya recorriendo cada complejo en una región y vaya creando el fractal.

Nota: El color del pixel en ese punto está asociado con el nivel de iteración, por lo que será necesario tener una condición de paro necesaria en caso de crecer arbitrariamente. Esta condición de paro está relacionada con un acotamiento del conjunto.

Código (I)

Función que itera:

```
def mandel(x, y, max_iters):
    c = complex(x, y)
    z = 0j
    for i in range(max_iters):
        z = z*z + c
        if z.real * z.real + z.imag * z.imag >= 4:
            return 255 * i // max_iters

    return 255
```

Script: `mandel.py`

Código (II)

Función que calcula el fractal...

```
def create_fractal(min_x, max_x, min_y, max_y,
                    image, iters):

    height = image.shape[0]
    width = image.shape[1]

    pixel_size_x = (max_x - min_x) / width
    pixel_size_y = (max_y - min_y) / height
```

Script: mandel.py

Código (III)

...continuación:

```
for x in range(width):
    real = min_x + x * pixel_size_x
    for y in range(height):
        imag = min_y + y * pixel_size_y
        image[y, x] = color

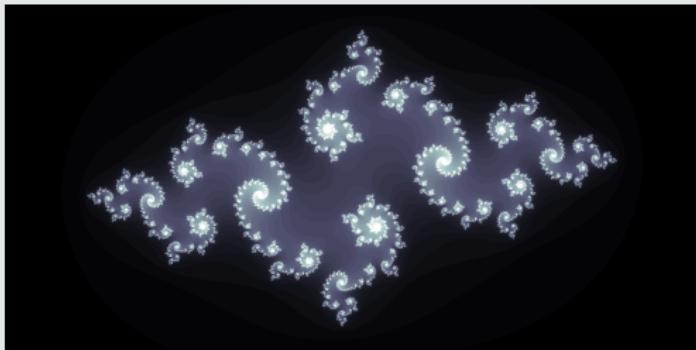
return image
```

Script: mandel.py

Reto para llevar

Reto 3:

Construir un conjunto de Julia a partir del código `mandel.py`.
(Pueden revisar a [@JuliaSetBot](#) en Twitter para ver ejemplos de conjuntos de Julia generados con sus respectivos complejos.)



Para concluir la sección...

Para poder concluir la sección y continuar con la siguiente, respondamos:

- ¿Qué sucede si iteramos otra función, en vez de $f_c(z) = z^2 + c$?
- ¿Cómo se puede mejorar el código para más iteraciones?
- ¿Hay alguna otra pregunta?

¿Preguntas?

Referencias (I)

- [1] Python Software Foundation.
History of the software.
History and Licence, disponible en
<https://docs.python.org/3/license.html>, 2017.
- [2] Hipertextual.
¿Qué es Software Libre?
Diferencias entre Software Libre y Open Source, 2014.
- [3] Hipertextual.
¿Qué es Open Source?
Diferencias entre Software Libre y Open Source, 2014.

Referencias (II)

[4] The Hitchhiker's Guide to Python.

Python Imaging Library.

Image Manipulation, 2016.

[5] The Hitchhiker's Guide to Python.

OpenSource Computer Vision.

Image Manipulation, 2016.

[6] Python Variable Types.

Tutorials Point.

Python Basic Tutorial.

Referencias (III)

[7] Image tutorial.

Importing image data into Numpy arrays.

matplotlib, disponible en

http://matplotlib.org/users/image_tutorial.html.

[8] Image tutorial.

Plotting numpy arrays as images.

matplotlib, disponible en

http://matplotlib.org/users/image_tutorial.html.

Referencias (IV)

- [9] Complex number.

Complex number.

Wikipedia, the free encyclopedia, disponible en

https://en.wikipedia.org/wiki/Complex_number.

- [10] The Mandelbrot Set.

The Mandelbrot Set - Numberphile.

Numberphile, disponible en

<https://www.youtube.com/watch?v=NGMRB40922I>.

- [11] IBM developerWorks.

How To Quickly Compute The Mandelbrot Set In Python.

JeanFrancoisPuget, 2015.



@FERRORODOLFO

Repositorio del taller:

<https://github.com/RodolfoFerro/FLISoL17>

¡Gracias por su atención!