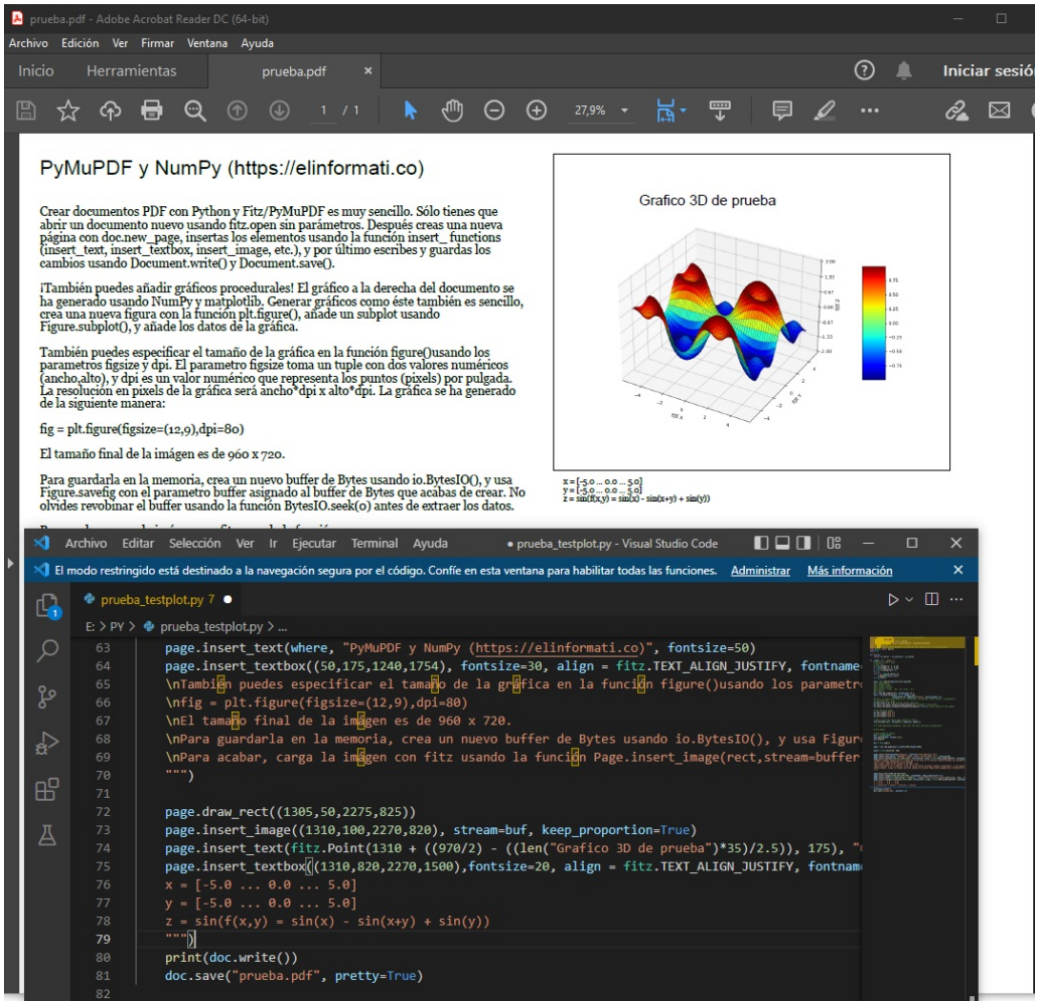
Crear documentos PDF con Python y PyMuPDF

Publicado el El Informatico - 16 de abril de 2022 -



Documento PDF generado con PyMuPDF

En el artículo anterior he mostrado como leer documentos PDF con Python para extraer texto e imágenes, e incluso texto de otras imágenes usando PyMuPDF OCR. Pero PyMuPDF no sólo nos permite extraer contenido de documentos PDF. También nos permite modificar e incluso generar documentos PDF a partir de código en python, de modo que es posible generar documentos de forma procedural para su posterior impresión o distribución.

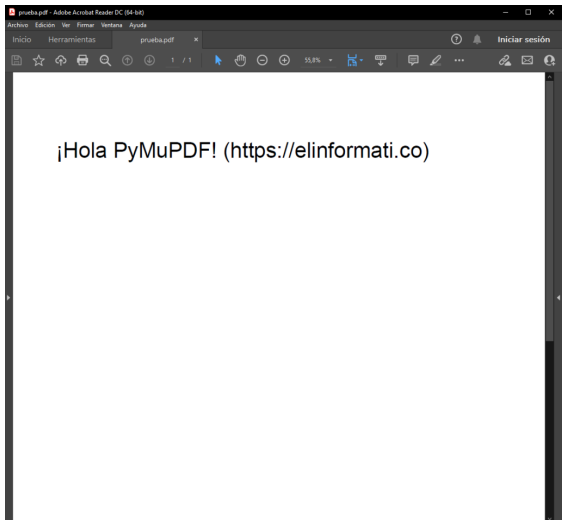
Documento básico en PyMuPDF

Para generar un documento con PyNumPDF, basta con abrir un nuevo documento en la memoria, insertar una nueva página, insertar los elementos, y guardar los cambios realizados.

```
import fitz

if __name__ == "__main__":
    # Nuevo documento
    doc = fitz.open()
    # Nueva página en el documento. Se insertará tras la última página
    pagina = doc.new_page(pno=-1,width=1240,height=1754)
    # Establecemos la posición sobre la que vamos a dibujar
    posicion = fitz.Point(100, 200)
    # Insertamos un texto en la página
    pagina.insert_text(posicion, "¡Hola PyMuPDF!
(https://elinformati.co)", fontsize=50)
    # Guardamos los cambios en el documento
    doc.write()
    # Guardamos el fichero PDF
    doc.save("prueba.pdf", pretty=True)
```

Abre el documento 'prueba.pdf' generado y comprueba que el documento contiene una página con el texto asignado.



Documento básico generado con PyMuPDF

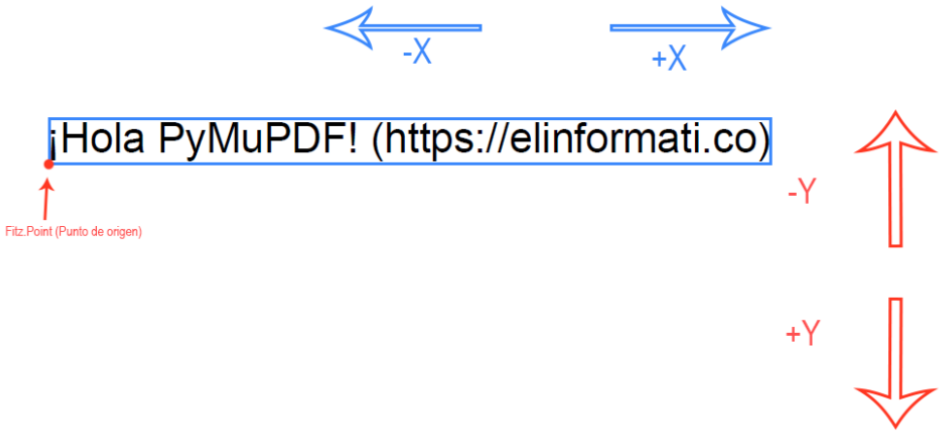
Podemos establecer el tamaño de página que queramos con las propiedades '*width*' y '*height*' de la función *new_page*. Por ejemplo, si queremos añadir una página en formato DIN A4, lo haríamos de la siguiente manera:

```
doc.new_page(pno=-1,width=2480,height=3508)
```

Para dibujar elementos en el documento, dependiendo de lo que queramos dibujar, se nos pedirá unas coordenadas que pueden ser un punto de orígen (desde la esquina inferior izquierda) en los ejex X e Y del documento, o un rectángulo dentro

del cuál se dibujará el elemento.

En éste caso, la posición del texto la asignamos con un Fitz.Point(x,y).

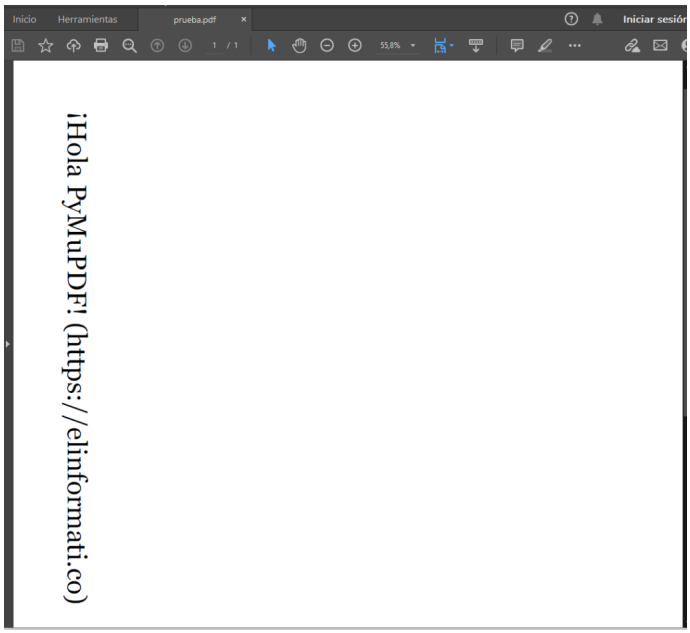


Esquema del documento

La propiedad *fontsize* de la función *insert_text* nos permite establecer un tamaño de fuente para el texto, en pixels.

La función insert_text incluye más parámetros como por ejemplo, *rotate* (ángulo de rotación del texto, sólo en múltiplos de 90), o *fontname* si queremos cambiar la fuente del texto (que tendremos que añadir al documento previamente).

```
if __name__ == "__main__":
    # Nuevo documento
    doc = fitz.open()
    # Nueva página en el documento. Se insertará tras la última página
    pagina = doc.new_page(pno=-1,width=1240,height=1754)
    # Insertamos la fuente Georgia en el documento
    pagina.insert_font(fontname="Georgia",
fontfile="fuentes/georgia.ttf")
    # Establecemos la posición sobre la que vamos a dibujar
    posicion = fitz.Point(100, 150)
    # Insertamos un texto con rotación y fuente en la página
    pagina.insert_text(posicion, "¡Hola PyMuPDF!
(https://elinformati.co)", fontsize=50, rotate=270, fontname="Georgia")
    # Guardamos los cambios en el documento
    doc.write()
    # Guardamos el fichero PDF
    doc.save("prueba.pdf", pretty=True)
```



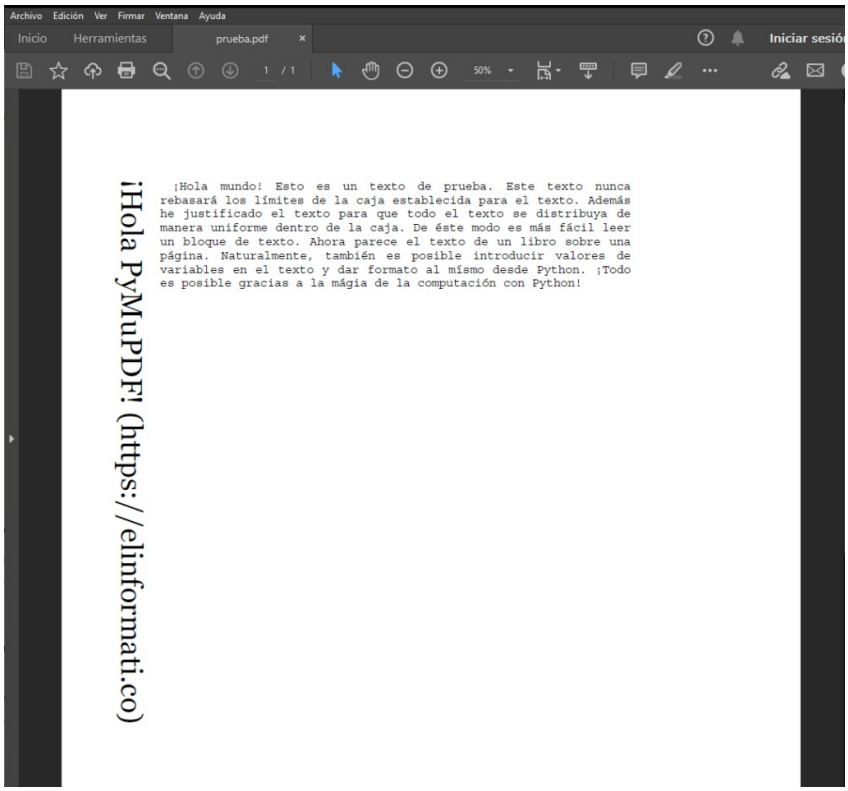
Documento generado con texto rotado

La rotación se realiza siempre desde el punto de orígen del objeto. Es decir, el Fitz.Point que hemos establecido para dibujar el texto.

Bloques de Texto

Podemos insertar bloques de texto en la página con la función insert_textbox. Los textbox nos permiten introducir texto que nunca rebasará los límites de la caja establecida para el texto. También podemos modificar la alineación del texto dentro de la caja.

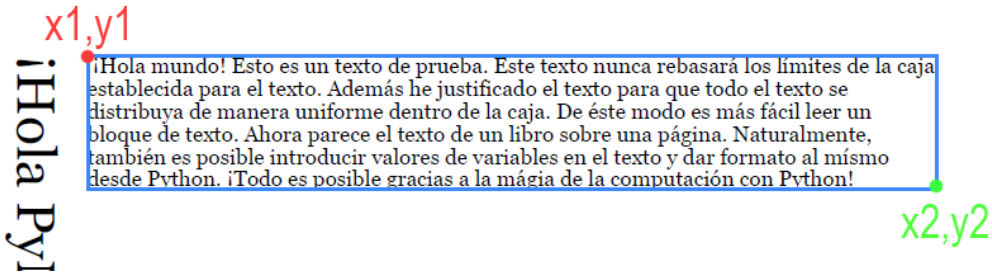
```
if __name__ == "__main__":
    # Nuevo documento
    doc = fitz.open()
    # Nueva página en el documento. Se insertará tras la última página
    pagina = doc.new_page(pno=-1,width=1240,height=1754)
    # Insertamos la fuente Georgia en el documento
    pagina.insert_font(fontname="Georgia",
fontfile="fuentes/georgia.ttf")
    # Establecemos la posición sobre la que vamos a dibujar
    posicion = fitz.Point(100, 150)
    # Insertamos un texto con rotación y fuente en la página
    pagina.insert_text(posicion, "¡Hola PyMuPDF!
(https://elinformati.co)", fontsize=50, rotate=270, fontname="Georgia")
    # Insertamos un textbox con texto justificado
    pagina.insert_textbox((165,150, 955, 650), fontsize=20, align =
fitz.TEXT_ALIGN_JUSTIFY, fontname="Georgia", buffer=""" ¡Hola mundo! Esto
es un texto de prueba. Este texto nunca rebasará los límites de la caja
establecida para el texto. Además he justificado el texto para que todo
el texto se distribuya de manera uniforme dentro de la caja. De éste modo
es más fácil leer un bloque de texto. Ahora parece el texto de un libro
sobre una página. Naturalmente, también es posible introducir valores de
variables en el texto y dar formato al mísmo desde Python. ¡Todo es
posible gracias a la mágia de la computación con Python! """)
    # Guardamos los cambios en el documento
    doc.write()
    # Guardamos el fichero PDF
    doc.save("prueba.pdf", pretty=True)
```



Texto justificado en el documento

El primer parámetro de *insert_textbox* es un tuple con cuatro valores correspondientes con las dos esquinas de la caja dentro de la cual se va a insertar el texto (x1,y1,x2,y2). A ésta caja se le llama '*rect*'. x1,y1 es el punto de orígen de la caja,

y en éste caso se corresponde con la esquina superior izquierda. x2,y2 se corresponden con la esquina inferior derecha, de modo que con esos dos puntos establecemos el diametro de la caja.



Coordenadas del textbox

La propiedad *align* es la que nos permite establecer la alineación del texto. Podemos establecerlo como justificado con fitz.TEXT_ALIGN_JUSTIFY. No obstante, ésto sólo es posible con fuentes símples. Generalmente, todas las fuentes que vienen integradas en los lectores PDF pueden usar ésta propiedad (lista de fuentes). Naturalmente, también es posible centrar o alinear el texto a cualquiera de los dos márgenes de la caja (izquierda o derecha) con TEXT_ALIGN_CENTER, TEXT_ALIGN_LEFT o TEXT_ALIGN_RIGHT (Lista de constantes).

Por último, el parámetro *buffer* es el que contiene el texto.

Color de texto

Podemos modificar el color de cualquier texto con la propiedad *color*.

```
from fitz.utils import getColor

    # Insertamos un texto con rotación y fuente en la página
    pagina.insert_text(posicion, "¡Hola PyMuPDF!
(https://elinformati.co)", color=(0.15,0.55,1),fontsize=50, rotate=270,
fontname="Georgia")
    # Insertamos un textbox con texto justificado
    pagina.insert_textbox((165,150, 955, 650),
color=getColor("mediumorchid"), fontsize=20, align =
fitz.TEXT_ALIGN_RIGHT, fontname="courier", buffer=""" ¡Hola mundo! Esto
es un texto de prueba. Este texto nunca rebasará los límites de la caja
establecida para el texto. Además he justificado el texto para que todo
el texto se distribuya de manera uniforme dentro de la caja. De éste modo
es más fácil leer un bloque de texto. Ahora parece el texto de un libro
sobre una página. Naturalmente, también es posible introducir valores de
variables en el texto y dar formato al mismo desde Python. ¡Todo es
posible gracias a la mágia de la computación con Python! """)
```

Texto con colores

El parametro color toma tuple con 3 o 4 valores (dependiendo de si queremos que el color tenga valor alpha o no). Dichos valores se corresponden con los valores R,G,B,A del color, del 0 al 1 (decimal). No obstante, PyMuPDF incorpora una lista de colores predefinidos, los cuales podemos obtener su valor mediante la función getColor(str), siendo str una cadena de texto con el nombre del color. Podemos usar getColorList() y getColorInfoList() para ver la lista completa (más información).

Dibujo de figuras

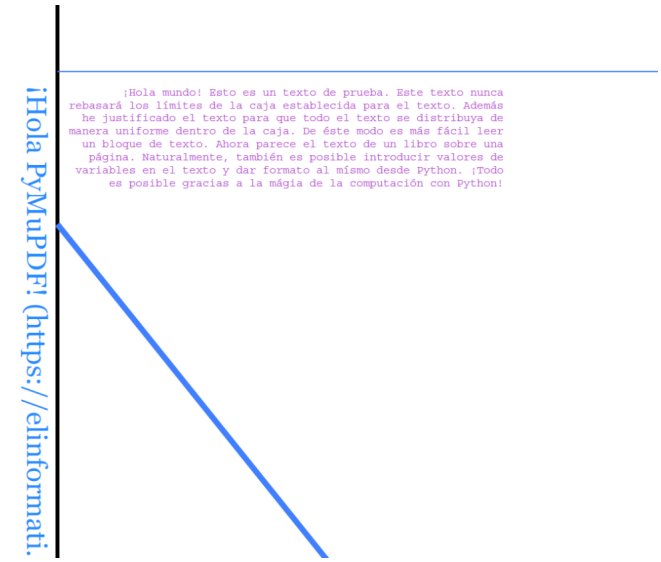
Podemos usar PyMuPDF como un programa de dibujo vectorial y dibujar formas y figuras dentro del documento usando una serie de primitivas.

Dibujar líneas

La primera herramienta que tenemos para dibujar figuras primitivas es draw_line. Como su nombre indica, nos permitirá dibujar líneas sobre la página.

```
    pagina.draw_line(p1=fitz.Point(155,0),p2=fitz.Point(155,1754),color=
(0,0,0),width=5, overlay=True)

pagina.draw_line(p1=fitz.Point(155,125),p2=fitz.Point(1240,125),color=
(0.25,0.5,1),width=2, overlay=True)
    pagina.draw_line(p1=fitz.Point(155,400),p2=fitz.Point(1240,1754),
color=(0.25,0.5,1),width=10, overlay=True)
```



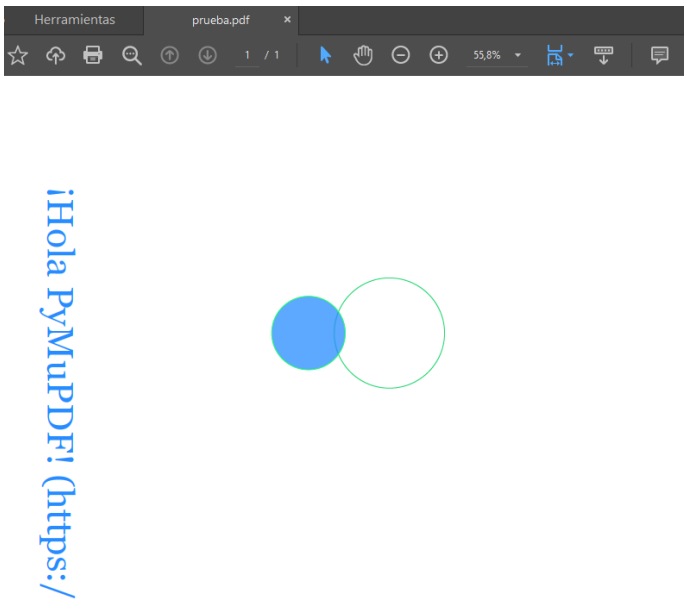
Lineas dibujadas sobre la página

Los parámetros p1 y p2 establecen los dos puntos de la línea. El programa trazará una línea desde p1 a p2. *color* establece el color de la línea. *width* establece el grosor de la línea. Por último, *overlay* especifica si la línea solapa a otras figuras (True) o no (False).

Dibujar círculos

El segundo método es *draw_circle*. Este método nos permite dibujar un círculo, que puede estar o no relleno con un color, de un radio determinado.

```
    pagina.draw_circle(center=fitz.Point(565, 350),radius=75.0,color=
(0.15,0.85,0.45), stroke_opacity=1.0)
    pagina.draw_circle(center=fitz.Point(455, 350),fill=(0.15,0.55,1),
radius=50.0,color=(0.15,1.0,0.45), stroke_opacity=1.0, fill_opacity=0.75)
```



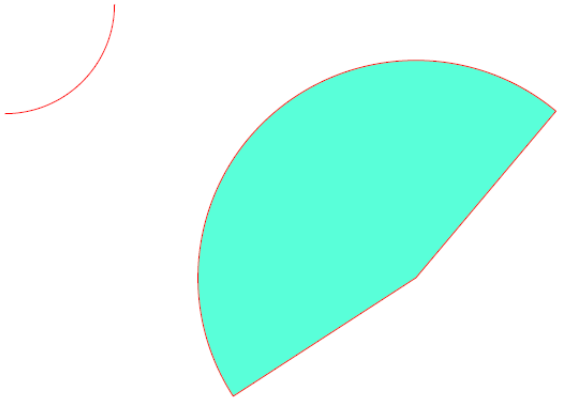
Circulos con draw_circle

La propiedad *center* establece el centro del círculo a dibujar. *radius* establece el valor del radio. *color* establece el color de la línea exterior del círculo. Si queremos que el círculo esté relleno de color, establecemos un color en la propiedad *fill*. En caso contrario, no tendrá relleno. *stroke_opacity* y *fill_opacity* determinan la opacidad de la línea y del relleno respectivamente, con valores de 0 a 1 (decimal).

Arcos

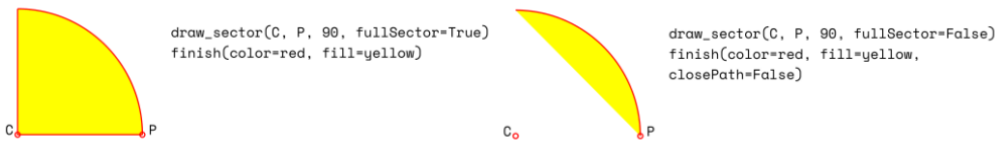
Se pueden dibujar arcos mediante la función *draw_sector*.

```
    pagina.draw_sector(center=fitz.Point(350, 350), beta=90.0,
point=fitz.Point(350,470), fullSector=False, color=(1.0,0,0))
    pagina.draw_sector(center=fitz.Point(800, 650), beta=-163.0,
point=fitz.Point(600,780), fullSector=True, color=(1.0,0,0), fill=
(0.35,1.0,0.85))
```



Arcos

Los parametros *center* y *point* establecen el centro y el primer punto del arco a dibujar. A partir de ahí, dibujará un arco de *beta* grados de ángulo. Dependiendo de si el valor de beta es positivo o negativo, el arco se dibujará en el sentido inverso de las agujas del reloj (positivo) o en el sentido de las agujas (negativo).



draw_sector
(https://pymupdf.readthedocs.io/en/latest/shape.html#Shape.draw_sector)

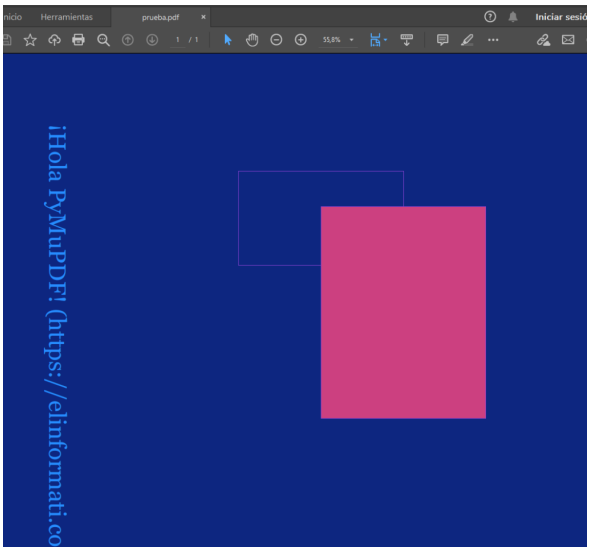
En la documentación para la última versión, el parametro beta es reemplazado por el parametro angle, que funciona de manera similar. Desconozco si ésto es un cambio en la última versión (la mia es la 1.19.0) o si la documentación está desactualizada. Si Python lanza un error diciendo que no exíste el parametro beta, reemplazalo por angle.

El parametro *fullSector*permite dibujar líneas entre el centro y los dos puntos para cerrar la figura cuando se establece a True. En cualquier otro caso, el arco queda sin cerrar.

Dibujar cajas

Podemos dibujar cuadrados y rectángulos usando la función draw_rect:

```
    pagina.draw_rect(rect=(0,0,1240,1754), color=0.0, fill=
(0.05,0.15,0.5), overlay=False)
    pagina.draw_rect(rect=(500,250,850,450), color=(0.5,0.25,0.8),
overlay=True)
    pagina.draw_rect(rect=(675,325,1024,775), color=(0.5,0.25,0.8), fill=
(0.8,0.25,0.5), overlay=True)
```



Podemos usar cuadrados como color de fondo de la página

El parámetro rect toma un rectángulo, similar al usado en insert_textbox, generando una caja mediante el diametro entre los dos puntos (x1,y1,x2,y2). color y fill establecen los colores de la línea exterior y del fondo respectivamente.

Cuando overlay se establece como False, el dibujo evitará solapar con otros elementos de la página. Ésto puede ser útil si, por ejemplo, queremos establecer un color de fondo para la página usando una caja que ocupe toda la página con overlay = False.

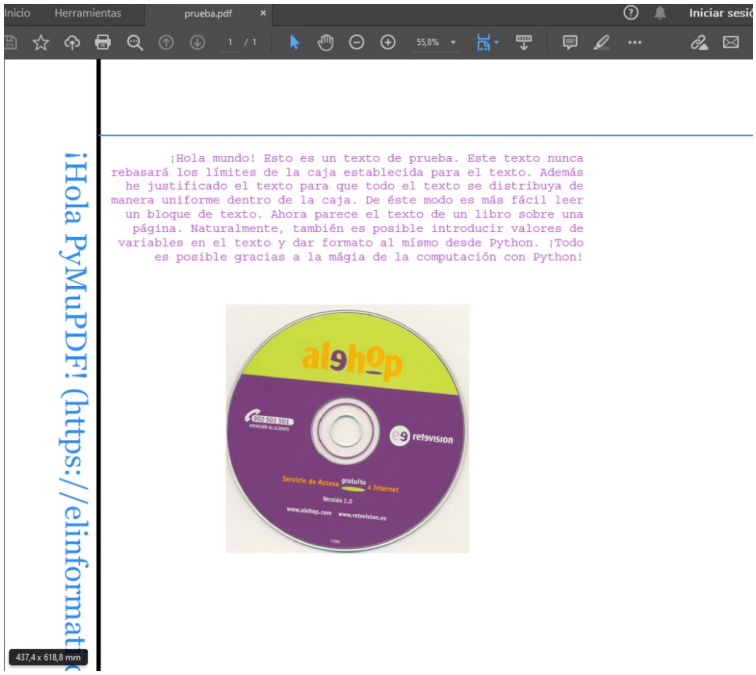
Otras funciones

Para ver un listado de funciones y sus posibles propiedades, visita la wiki de PyMuPDF.

Insertar Imágenes

También podemos insertar imágenes en nuestro documento usando insert_image:

```
pagina.insert_image(rect=(365, 360, 765, 860),filename="imagenes/p8-
83.png", keep_proportion=True, overlay=True)
```



Imágen insertada en el documento

La imágen se insertará desde el rectángulo asignado en el parametro *rect*. Si se establece la propiedad keep_proportion como True, se dibujará manteniendo la relación de aspecto de la imágen. En caso contrario la imágen se estirará para ajustarse al rectángulo asignado.

El parámetro *filename* es el nombre del archivo que queremos insertar. Debe ser un arcchivo de imágen válido. No obstante, también es posible cargar imágenes desde la memoria usando objetos Pixmap (mediante la propiedad *pixmap*) o incluso bytestreams (mediante la propiedad *stream*).

Insertar gráficos de MatPlotLib

Sabiendo que podemos insertar imágenes desde la memoria, podemos generar e insertar gráficos generados con MatPlotLib directamente en nuestro documento PDF:

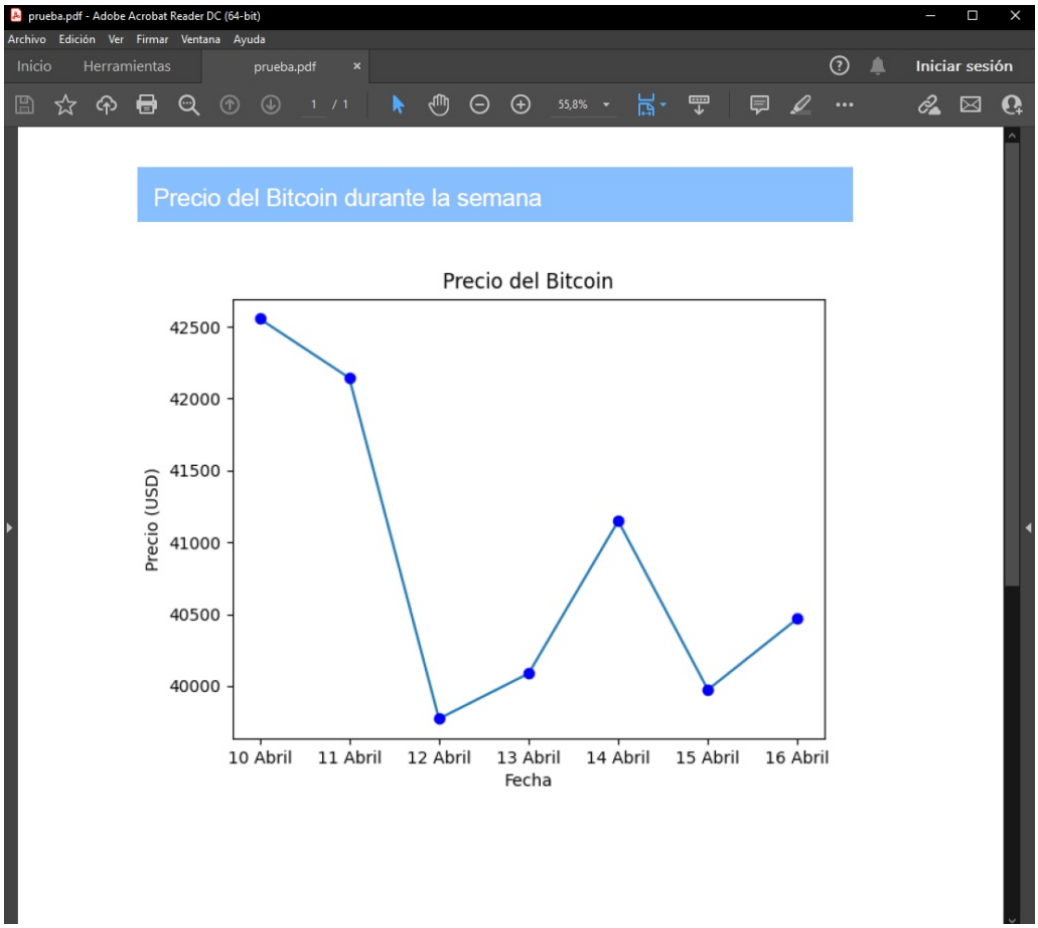
```
import fitz # pip install pymupdf frontend (Do NOT install fitz if using
pymupdf)
from matplotlib import pyplot as plt
import io


if __name__ == "__main__":
    # Datos del eje Y
    datos_y = [42554.72, 42143.33, 39772.773, 40087.55, 41147.74,
39974.57, 40469.68]
    # Datos del eje X
    datos_x = ["10 Abril", "11 Abril", "12 Abril", "13 Abril", "14
Abril", "15 Abril", "16 Abril"]

    # Dibujamos un gráfico de líneas
    plt.plot(datos_x,datos_y)
    # Dibujamos puntos sobre las líneas
    plt.plot(datos_x,datos_y,"ob")
    # Establecemos el título
    plt.title('Precio del Bitcoin')
    # Establecemos los títulos de los ejes
    plt.ylabel('Precio (USD)')
    plt.xlabel('Fecha')

    # Almacenamos la gráfica como un buffer de bytes
    buf = io.BytesIO()
    plt.savefig(buf, format='png')
    buf.seek(0)

    # Creamos un nuevo documento PDF
    doc = fitz.open()
    # Insertamos una nueva página
    pagina = doc.new_page(pno=-1,width=1240,height=1754)
    # Dibujamos un rectángulo
    pagina.draw_rect((150,50,1050,120), fill=(0.15,0.55,1), color=1,
stroke_opacity=0, fill_opacity=0.55, overlay=False)
    # Colocamos un texto sobre el rectángulo
    pagina.insert_text(fitz.Point(170,100), "Precio del Bitcoin durante
la semana", color=1, fontsize=30)
    # Insertamos la gráfica.
    # El tamaño de la gráfica es de 12 * dpi + 9 * dpi
    pagina.insert_image((150,130,150+(12*80),130+(9*80)), stream=buf,
keep_proportion=True)
    # Insertamos los cambios en el documento
    doc.write()
    # Guardamos el archivo PDF
    doc.save("prueba.pdf", pretty=True)
    # Cerramos el documento
    doc.close()
```



Documento PDF con la gráfica insertada

ANTERIOR

Extraer textos e imágenes de documentos PDF y otras imágenes con Python, PyMuPDF y OCRMyPDF

SIGUIENTE

Consejos, trucos y curiosidades de Python

Buscar ...

Entradas Recientes

- Estoy hasta las narices de la web moderna
- Ingenieria inversa básica con Ghidra
- Acerca de la nueva ley transgénero (Y sobre la disfória de género)
- Depresiones causadas por las redes sociales
- ¿Necesito saber matemáticas para aprender informática?
- ¿Es el fín de los discos duros tradicionales?

Categorías

Actualidad | Android | Básicos | Ciberseguridad | Clima | Criptografía | Electronica | Emulación / Virtualización | FOSS | Hacking | Hardware | Informática | Internet | Juegos | Opinion | Otros | Personal | Privacidad | Programación | Tecnología | Time Machine | Tutoriales

RSS

Subscribirse al feed RSS