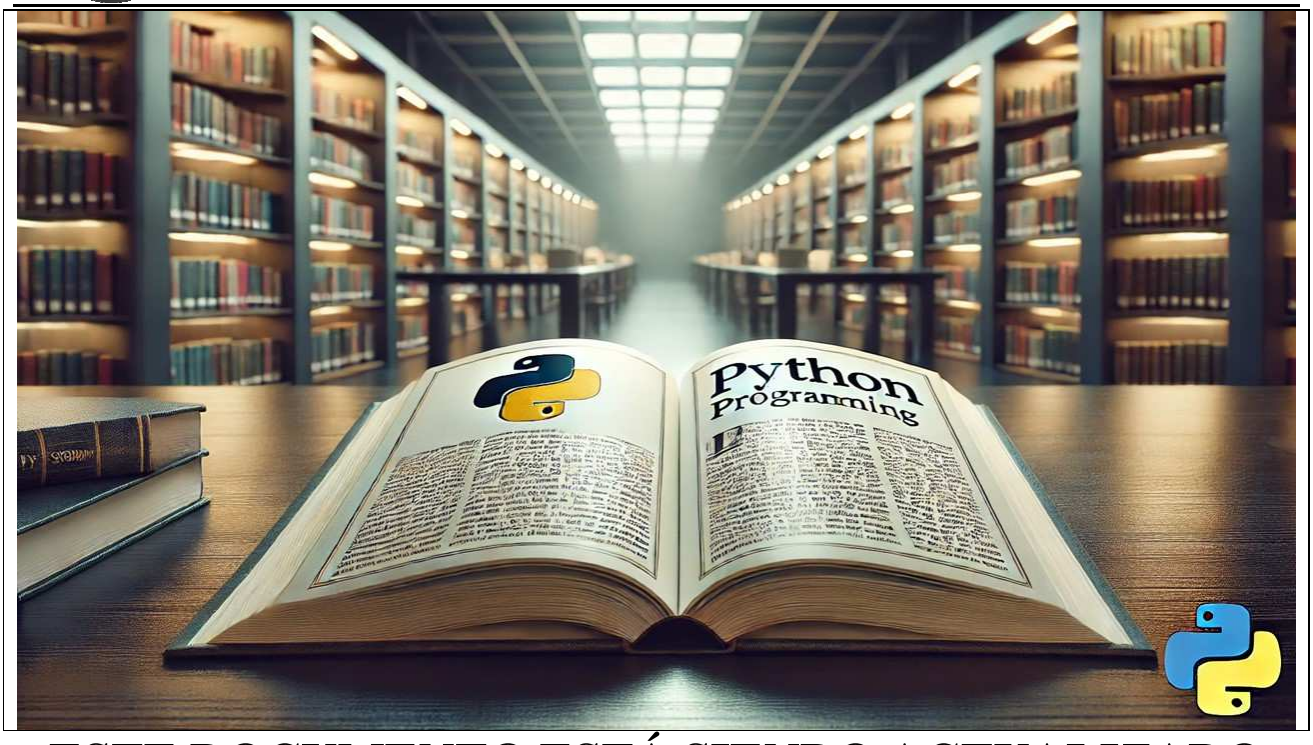


ESTE DOCUMENTO ESTÁ SIENDO ACTUALIZADO

Índice

ReportLab o PyPDF2....2
ReportLab....2
PyPDF2....2
Resumiendo....2
ReportLab....3
Instalar librería....3
Crear Documento PDF....3
Texto en el Documento....3
Preliminares....3
Primer Texto....3
Varios renglones....4
Tamaño de Página, Cálculo de Puntos y Coordenadas....4
Primero lo Primero....4
Listado Tipos de Hojas Estándares Predefinidas....4
Tipo de hoja por defecto....5
Recuerda que....5
Hay que dejar bien claro....5
Relación entre unidad de medida y puntos....6
Hojas definidas por el programador....6
Ejemplos....6
SIEMPRE, al trabajar en tu Documento....7
Mostrar Documento PDF en el Monitor....7
Tipo y Tamaño de Letra....7
Introducción....7
Nombres de las Fuentes Disponibles....8
Detectar Letra: Fuente (Tipo) y Tamaño (Altura).....8
Configurar Fuente (Tipo) y Tamaño de Letra.....8
Justificación y Centrado del Texto.....9

Longitud de una Cadena en Puntos 9
Justificación y Centrado....9
Agregar y/o Numerar las Páginas del Documento. 9
Imprimir Documento PDF Desde tu Programa....10
Alternativa 01....10
Alternativa 02....10
Líneas y Figuras Geométricas....11
Líneas....11
Grosor de línea 11
Rectángulos. 11
Esquinas rectas 11
Esquinas redondeadas 11
Círculos. 12
Elipse. 12
Arco. 13

Ultima Modificación 03/01/2025 03:31:00

ReportLab o PyPDF2

ReportLab: Es ideal para generar documentos PDF desde cero. Permite un control detallado sobre el diseño, incluyendo la creación de gráficos, tablas y texto en distintos formatos y estilos.

PyPDF2: Se utiliza principalmente para manipular archivos PDF ya existentes. Permite tareas como leer, fusionar, dividir y extraer texto de documentos PDF, además de algunas operaciones más avanzadas como agregar metadatos o rotar páginas.

Resumiendo: Usa PyPDF2 para trabajar con PDFs ya creados y ReportLab para crear nuevos documentos PDF.

ReportLab (Ver documentación oficial: https://docs.reportlab.com/) es un paquete de herramientas (librería) de código abierto que podremos usar para manipular documentos PDF desde nuestros programas Python. Esta librería es bastante extensa, con muchas funcionalidades, que nos permitirá incluir en nuestro documento PDF, desde pequeños textos, figuras geométricas o gráficos e ilustraciones. En este pequeño documento encontraremos ejemplos concretos, en donde aplicamos algunas funciones mientras creamos este tipo de documentos. Luego que conozcas las generalidades podrás investigar el resto por tu cuenta.

Instalar librería: **Recuerda** que deberemos instalarla para usarla en nuestros programas.

```
pip install reportlab
```

Crear Documento PDF

Una vez instalada la librería reportlab, ya podremos comenzar, y lo primero que haremos será crear muestro primer documento PDF; al que llamaremos "60_Primero.pdf". **Pero esta vez,** lo dejaremos vacío. Es decir no encontraremos cosas escritas ni dibujadas, ni siquiera una pagina.

```
01. from reportlab.pdfgen import canvas  # Importamos lo necesario
02. # Datos del Archivo
03. Ruta="" # Si creas el documento en la misma carpeta donde esta tu
04.         # programa, deja la Ruta vacía.
05. Documento = "60_Primero.pdf"
06. # creamos una instancia de la clase "canvas.Canvas()"
07. c = canvas.Canvas(Ruta+Documento)
08. # Aca trabajarás
09. c.save() # Guarda (graba) el documento
10.
```

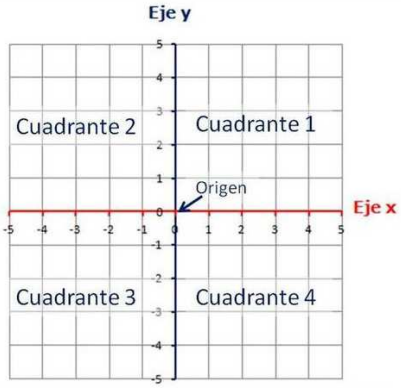
Código Completo: PDF/60_PDF_01_Creando_Documentos_01_Documento_Vacio.txt

Texto en el Documento

Preliminares.

Cuando trabajes con esta librería **(reportlab)**, imagina que tienes una hoja en blanco en la que puedes escribir, dibujar o hacer cualquier cosa que quieras hacer. La escritura, dibujos o lo que sea que quieras hacer, debes realizarlas siempre entre la línea donde le das la identidad (dirección y nombre) a nuestro documento (línea 8) y el método que guarda/crea el documento PDF (línea 10).

Otra cosa que debes saber, hay que especificar el lugar (coordenadas) donde hay que escribir o hacer lo que queremos hacer, encontrando el origen o posición (0,0), igual que con los ejes cartesianos, abajo a la izquierda, y nuestra página es el primer cuadrante: Entonces las posiciones en "X", aumentarán a medida que te corras a la derecha y las posiciones en "Y" aumentarán a medida que subas.



Primer Texto: Usaremos el método "**c.drawString(xi, xi, Texto)**" a la que entregamos como argumentos: Coordenadas del punto inicial donde comenzaremos a escribir y por ultimo el texto.

```
from reportlab.pdfgen import canvas  # Importamos lo necesario
Ruta=""
Documento = "60_Segundo.pdf"
c = canvas.Canvas(Ruta+Documento) # instancia "canvas.Canvas()"
c.drawString(50, 50, "Primer Texto")  # Escribimos el Texto.. :)
c.save()  # Guarda (graba) el documento
```

Acá dispones del código completo:
PDF/60_PDF_01_Creando_Documentos_02_Texto_01_Primeras_Palabras.txt

Ya probaste el segundo ejemplo y no viste lo que escribiste?. Busca otra vez, ábrelo y ahora fíjate en la parte inferior izquierda del documento.

Recuerda que comienza a contar (medir) desde el origen o coordenadas

(0,0) del documento. Como habrás notado, el método (comando o sentencia) que usamos es "**drawString( )**" y que los dos primeros parámetros que le entregamos corresponden a las coordenadas "x,y", continuando con el texto a escribir en el documento PDF.

Varios renglones: Y si quisiéramos escribir varias líneas?. Para esto deberemos especificar las Coordenadas "x" inicial e "y" inicial (en que línea y columna) debe comenzar a escribir cada frase. Veamos un ejemplo:

```
from reportlab.pdfgen import canvas # Importamos lo necesario

Ruta="" # Datos del Archivo
Documento = "60_Ahora_10_Lineas.pdf"

# Coordenadas Iniciales
xi = 50
yi = 800

# Creo en Memoria Documento PDF.
c = canvas.Canvas(Ruta+Documento)

# En ciclo for voy Cambiando coordenadas para cada línea
for i in range(10):
    Texto = "Esta es la linea Numero: " + str(i+1)
    xi = xi + (i*10)  # Incremento las x
    yi = yi - (i*15)  # Resto a las "y" para bajar en el Documento
    c.drawString(xi, yi, Texto)  # Escribo línea en el Documento PDF

# Guarda (graba) Documento en Lugar (ruta) y Nombre Especificado
c.save()
```

Código Completo: PDF/60_PDF_01_Creando_Documentos_02_Texto_02_Diez_lineas.txt

Debes notar en el documento, que parte de la última línea se ha perdido. Esto se debe a que el texto se salió de los márgenes establecidos por la página cargada por defecto.

Ahora seguramente surge la pregunta, y para escribir una línea en la cabecera del documento, cuales serán las coordenadas precisas, o cual es el tamaño de la pagina establecida?

Tamaño de Página, Cálculo de Puntos y Coordenadas

Primero lo Primero: Para comenzar, siempre que crees un documento PDF, necesitarás tener un tipo de Hoja cargado, ya sea que lo definas tu como programador, o lo defina el sistema como hoja por defecto (si no lo hace el programador lo hace el sistema).

A continuación, dispones de una tabla donde encontraras, para cada tipo de hoja (de las más usadas en Argentina) el nombre y los tamaños de cada una expresados en Milímetros, Centímetros, Pulgadas y Puntos.

Dimensiones aproximadas de los formatos estándares, correspondientes a las hojas de la serie A Expresados en: mm, cm, pulgadas y Puntos[1]

Formato	Ancho x Altura (en mm)	Ancho x Altura (en cm)	Ancho x Altura (en pulgadas)	Ancho x Altura (en Puntos)
A0	841 x 1189	84,1 x 118,9	33,1 x 46,8	2383.94 x 3370.40
A1	594 x 841	59,5 x 84,1	23,4 x 33,1	1683.78 x 2383.94
A2	420 x 594	42 x 59,4	16,5 x 23,4	1190.55 x 1683.78
A3	297 x 420	29,7 x 42	11,7 x 16,5	841.89 x 1190.55
A4	**210 x 297**	**21 x 29,7**	**8,3 x 11,7**	595.28 x 841.89
A5	148 x 210	14,8 x 21	5,8 x 8,3	419.53 x 595.28
A6	105 x 148	10,5 x 14,8	4,1 x 5,8	297.64 x 419.53
A7	74 x 105	7,4 x 10,5	2,9 x 4,1	209.76 x 297.64
A8	52 x 74	5,2 x 7,4	2,0 x 2,9	147.40 x 209.76
A9	37 x 52	3,7 x 5,3	1,5 x 2,0	104.88 x 147.40
A10	26 x 37	2,6 x 3,7	1,0 x 1,5	73.70 x 104.88

Importante: Toda Página, tiene sus medidas, tal como las conocemos, pero en reportlab, cuando trabajamos con documentos PDF, las medidas (ancho y alto) estarán expresadas en Puntos (points).

Listado Tipos de Hojas Estándares Predefinidas

El siguiente ejemplo, realiza un listado con todos los tipos de hojas y sus medidas que el sistema pone a tu disposición.

[1] Esto es Valido para Cualquier tipo de hoja. Sin embargo, puedes encontrar diferencias por redondeo en los valores calculados.

```
from reportlab.lib.pagesizes import * #Carga todos los tipos

# Carga (en una lista) los tipos estándar de página importados
Tipos_Pagina = [name for name in globals( ) if name[0] != '_']

# Imprimir la lista de tipos de página estándar con sus tamaños
for Tipo in Tipos_Pagina:
    Tamaños = globals( )[Tipo]   #Consulta Tamaños para cada Pagina
    print(f"{Tipo}: {Tamaños}")
```

Código Completo: PDF/60_PDF_01_Creando_Documentos_02_Texto_03_Listado_Tipo_Paginas_reportlab.txt

Tipo de hoja por defecto

Cuando trabajamos con una instancia del tipo "**Canvas**", si no definimos un tipo de hoja (Tamaño), el sistema define (tal como hizo en los ejemplos anteriores) una hoja estándar (por defecto) que, generalmente es del tipo A4, pero en ocasiones, el Sistema Operativo o el mismo programador, definen otra pagina por defecto. En este caso tendremos que obtener sus datos, para saber si es una pagina estándar (que nombre tiene) y además sus medidas (Ancho y Alto).

Recuerda que**:** Toda Página, tiene sus medidas, tal como las conocemos, pero en reportlab, cuando trabajamos con documentos PDF, las medidas (ancho y alto) estarán expresadas en Puntos (points)..

```
from reportlab.pdfgen import canvas

# Crear documento con fuente y tamaño por defecto
c=canvas.Canvas( Ruta+Documento )

#Tomo (de atributos/variables de la instancia) Tamaño de Hoja por defecto
Formato_Pagina = c._pagesize
Ancho , Alto = Formato_Pagina

print("Tamaño de Pagina: ", Formato_Pagina)

#Tomo (atributos/variables de la instancia) Fuente y tamaño de letra
Fuente = c._fontname
Tamaño = c._fontsize
Texto_01 = f"Fuente por defecto: {Fuente}.ttf."
print(Texto_01)

Texto_02 = f"El tipo de letra es: {Fuente}"
Texto_02 = Texto_02 + f" y el Tamaño (Altura) de letra es: {Tamaño} puntos."
print(Texto_02)
```

Código Completo: PDF/60_PDF_01_Creando_Documentos_02_Texto_03_Detectar_Fuente_y_Tamanio.txt

Todo el sistema esta pensado en Pulgadas, pero la instancia del tipo "**canvas**" también nos permite trabajar con centímetros y/o milímetros (medidas más amigables para nosotros).

Hay que dejar bien claro que, No Importa en que unidades se mida una hoja, ya que tienen siempre el mismo Ancho y el mismo Alto [2].

El tamaño A4 pertenece a la serie A y el número «cuatro» significa que sus dimensiones corresponden a una hoja A0 que se ha cortado cuatro veces por la mitad. Así pues, un tamaño A4 es **16 veces más pequeño que un tamaño A0** [3].	
Tamaño A4 en cm: Una hoja de tamaño A4 mide **21 x 29,7 cm**, es decir, **210 x 297 mm**. Las dimensiones del tamaño A4, como el de los demás formatos de la serie A, está definido por la norma ISO 216 y se utiliza en la mayoría de países. En 1786, el científico alemán Lichtenberg describió la proporción entre la anchura y la longitud de los formatos A. Posteriormente, en 1922, Portsmann y el Instituto alemán de normalización definieron el formato DIN 476, que se convertiría en la norma actual, conocida con el nombre de ISO 216.	**Tamaño A4 en pulgadas:** En Estados Unidos y Canadá, los formatos de la serie A se utilizan muy poco, pero los demás países que han seguido utilizando el sistema británico de unidades de medida, han expresado las dimensiones en pulgadas. Así pues, las medidas del tamaño A4 son del orden de **8,27 x 11,67 pulgadas** (o «inches») y conservan la relación longitud/anchura de √2. [4]

[2] Esto es Valido para Cualquier tipo de hoja. Sin embargo, puedes encontrar diferencias por redondeo en los valores calculados.
[3] https://www.adobe.com/es/creativecloud/design/discover/a4-format.html
[4] https://www.adobe.com/es/creativecloud/design/discover/a4-format.html

Relación entre unidad de medida y puntos [5]

Como se menciono antes, El sistema esta pensado en pulgadas, y una pulgada representa 72 puntos. Partiendo de esta definición, y por medio de un simple calculo (regla de 3 simple) podremos saber cuantos puntos representa un centímetro o un milímetro.

Unidad	Puntos
1 Pulgada	72.0
1 Centímetro	28,35
1 Milímetro	2,834

El Ancho y Alto de todos los tipos de hojas, en un documento PDF, serán medidos en puntos. Por ejemplo, entre otros formatos, una hoja **A4** tiene (**210 Milímetros de ancho x 297 milímetros de alto**), expresado en puntos 595.2 de ancho (width) y 841.8 puntos de alto (height).

El siguiente ejemplo, realiza un listado con todas las unidades de medida y sus equivalencias en Puntos.

```
from reportlab.lib.units import *   #Importa TODAS las unidades disponible

# Crear una copia de globals() para evitar el error de 'RuntimeError'
copiaglobals = dict(globals())

# Mostra las unidades de medida importadas con sus valores
print("Unidades de medida importadas y sus valores:")
for nombre_unidad in copiaglobals:
    if not nombre_unidad.startswith('_'):   # Excluir las variables internas
        valor_unidad = copiaglobals[nombre_unidad]
        print(f"{nombre_unidad}: {valor_unidad}")
```

Código Completo: PDF/60_PDF_01_Creando_Documentos_03_Tamaño_Hoja_00_Lista_Unidades.txt

Hojas definidas por el programador

Al crear un documento PDF, desde nuestro código Python, podremos definir tamaños de hojas alternativas de tres formas distintas, usando un parámetro adicional "**pagesize**", a quien le asignamos el tipo de página a definir, o una tupla (par ordenado en este caso) cuyo primer elemento representa el ancho en puntos y el segundo, el alto, o con un método adicional que también a "reportlab". Resumiendo: cualquiera de los 3 segmentos de código, hacen lo mismo y obtienen igual resultado.

Código A:	C = canvas.Canvas(Ruta+Documento, pagesize=A4)
Código B:	C = canvas.Canvas(Ruta+Documento, pagesize=(Ancho, Alto))
Código C:	C = canvas.Canvas(Ruta+Documento) C.setPageSize((Ancho, Alto))

Cuando el sistema asigna dimensiones por defecto, generalmente son las correspondientes al estándar **A4** y para nosotros, como programadores definir otros estándares, deberemos usar el módulo **reportlab.lib.pagesizes**. Que contiene los datos de cada tipo de hoja disponible. Por ejemplo, entre otras, la hoja tipo **letter**, con 612.0 puntos de ancho (width) y 792.0 puntos de alto (height). El uso de este formato de hoja, es muy frecuente en Estados Unidos. Sugiero que Consultes el "*Listado Tipos de Hojas Estándares Predefinidas*" un poco mas arriba

Ejemplos

Veamos a continuación distintos ejemplos, a los que podrás analizar detenidamente.

Definición hoja Tamaño A4

```
from reportlab.lib.pagesizes import A4
from reportlab.pdfgen import canvas

Ruta="" # Datos del Archivo
Documento = "60_Hoja_A4.pdf"

Ancho , Alto = A4  # A4 retorna tamaño de página

c = canvas.Canvas(Ruta+Documento, pagesize=A4)
```

Desde acá puedes tomar el código completo del ejemplo
PDF/60_PDF_01_Creando_Documentos_03_Tamaño_Hoja_01_A4.txt

Definición hoja Tamaño Letter

```
from reportlab.lib.pagesizes import letter
from reportlab.pdfgen import canvas

Ruta="" # Datos del Archivo
Documento = "60_Hoja_ letter.pdf"

Ancho , Alto = letter # letter retorna el tamaño de página

c = canvas.Canvas(Ruta+Documento, pagesize=letter)
```

Desde acá puedes tomar el código completo del ejemplo
PDF/60_PDF_01_Creando_Documentos_03_Tamaño_Hoja_02_letter.txt

[5] Esto es Valido para Cualquier unidad de medida. Sin embargo, puedes encontrar diferencias por redondeo en los valores calculados.

También es posible definir cualquier tamaño de página en forma personalizada. Para esto usaremos el método "**c.setPageSize((ancho, alto))**" en donde se asigna el tamaño de la hoja por la cantidad de puntos de Ancho y cantidad de puntos en Alto. Las cantidades de puntos pueden coincidir o no con el tamaño de algún tipo de hoja estándar.

Definición de hoja con Puntos Tamaño A3

```
from reportlab.lib.pagesizes import A3
from reportlab.pdfgen import canvas

Ruta="" # Datos del Archivo
Documento = "60_Hoja_Puntos_A3.pdf"

Ancho , Alto = A3  # Retorna Ancho y Alto en puntos

c = canvas.Canvas(Ruta+Documento)
c.setPageSize((Ancho, Alto)) # Defino tamaño en Puntos

Texto = "La Pagina que tiene "
Texto = Texto + str(Ancho) + " PUNTOS de Ancho y "
Texto = Texto + str(Alto) + " de Alto es una Tamaño A3"

print("\n",Texto)  # Lo muestro en el monitor
```

Desde acá puedes tomar el código completo del ejemplo
PDF/60_PDF_01_Creando_Documentos_03_Tamaño_Hoja_03_Puntos.txt

La cantidad de Puntos que usaras al definir el tamaño de página, puede estar o no, relacionados a un tipo de hoja estándar (A4, A3, letter, etc) o a la unidad de medición de tu preferencia, por ejemplo, pulgadas, milímetros o centímetros (entre otras muchas unidades).

Definición de hoja con Puntos Tamaño Personalizado

```
from reportlab.pdfgen import canvas
#from reportlab.lib.units import inch  #72 Puntos por Pulgada
#from reportlab.lib.units import mm  #2,834 por Milímetro
from reportlab.lib.units import cm  #28,3464 por Centímetro

Ruta="" # Datos del Archivo
Documento = "60_Hoja_Personalizada.pdf"

# Defino Hoja de 5x8 pulgadas
#Ancho = 5*inch
#Alto = 8*inch

# Defino Hoja de 150x250 Milímetros
# Ancho = 150*mm
# Alto = 250*mm

# Defino Hoja de 15x25 Centímetros
Ancho = 15*cm
Alto = 25*cm

c = canvas.Canvas(Ruta+Documento)
c.setPageSize((Ancho, Alto)) #Defino tamaño página en Puntos
```

Desde acá puedes tomar el código completo del ejemplo
PDF/60_PDF_01_Creando_Documentos_03_Tamaño_Hoja_04_Pulgadas.txt

SIEMPRE, al trabajar en tu Documento PDF, tendrás que cuidar: NO SALIRSE de los márgenes, ya que perderías parte o toda información.

Mostrar Documento PDF en el Monitor

Es posible que abrir y visualices en el monitor, un documento cualquiera (desde tu programa Python), siempre y cuando exista, conozcas el nombre, ruta de acceso y el Sistema Operativo sepa con que programa visualizarlo. Veamos el segmento de código, en donde creo un documento PDF y luego se muestra automáticamente. Analízalo:

```
import os # Para Mostar PDF por Monitor
from reportlab.lib.pagesizes import A4
from reportlab.pdfgen import canvas

Ruta="" # Datos del Archivo
Documento = "60_Mostrar_X_Monitor_01.pdf"
c = canvas.Canvas(Ruta+Documento, pagesize=A4)

c.drawString(50, Alto - 50, Texto) # Escribo en PDF

c.showPage() # Termina Página
c.save()

# Sistema Operativo Abre y Muestra PDF en Monitor
os.system(Ruta+Documento)
```

Código Completo: PDF/60_PDF_50_Especiales_01_Mostrar_PDF_en_Monitor_01_Automatico.txt

Tipo y Tamaño de Letra

Introducción: El poder manejar el tipo de letra y tamaño es una muy potente herramienta, que permitirá dar un formato más agradable y mejorar notablemente la claridad, ya que entre otras cosas, podremos resaltar textos con colores, grosores, tipos de letras, o Justificar el texto (recordar que las coordenadas dentro del documento si mide en Puntos).

Cuando definimos un Documento, y no definimos un tipo determinado, el documento adoptara por defecto, el tipo de letra definido por el Sistema Operativo, generalmente en Windows es el tipo "**Helvetica**" del tamaño 12, aunque esto puede variar por configuración del usuario, o porque alguna versión del Sistema Operativo especifica tenga de origen otra por defecto. Pero, sin importar la situación, tenemos herramientas para redefinir la fuente de nuestro Documento, o para saber cuan es la definida por defecto y continuar trabajando.).

Nombres de las Fuentes Disponibles

El siguiente ejemplo, realiza un listado con el nombre de todas las fuentes que tienes disponibles para escribir en tu documento, que pertenecen a "reportlab", sin embargo también tendrás disponibles las que pertenecen a Windows pero no serán listadas aquí:

```
from reportlab.pdfgen import canvas

# Crear un lienzo para acceder a las fuentes y sus tamaños
c = canvas.Canvas("temp.pdf")

# Mostrar todos los nombres de fuentes importadas con sus tamaños asociados
print("Nombres de fuentes importadas y sus tamaños asociados:")
for nombre_fuente in c.getAvailableFonts():
    print(f"{nombre_fuente}")

# Cierra Documento PDF o Lienzo
c.save()
```

Código Completo: PDF/60_PDF_01_Creando_Documentos_02_Texto_03_Lista_Nombres_de_Fuentes.txt

Si hubiera alguna fuente que no esté disponible para Windows y tampoco para "reportlab", puedes incluirla temporalmente.
Asegurarse de que la fuente está donde está tu programa o de darle la ruta para que pueda ser encontrada.
IMPORTANTE: La fuente será incluida y la podrás usar solamente durante la ejecución de tu programa, luego desaparece hasta que sea incluida nuevamente.

Código Completo: PDF/60_PDF_01_Creando_Documentos_02_Texto_03_Incluye_Fuente.txt

Detectar Letra: Fuente (Tipo) y Tamaño (Altura).

Si por alguna razón, desconociéramos, o nos resulta más simple que el sistema nos informe Tipo y Tamaño de letra que esta en curso, es muy simple, ya que disponemos de dos variables que pertenecen a la instancia "Cambas" y tienen estos datos. Veamos el ejemplo y analiza (sugiero que obtengas el código completo para realizar el análisis):

```
from reportlab.pdfgen import canvas
from reportlab.lib.pagesizes import A4

c=canvas.Canvas( Ruta+Documento, pagesize=A4)

# Detecto Fuente Registrada y tamaño de letra utilizada
Fuente = c._fontname
Tamaño = c._fontsize

Texto_01 = f"Fuente por defecto: {Fuente}.ttf."
print(Texto_01)
Texto_02 = f" El tipo de letra es: {Fuente}"
Texto_02 = Texto_02 + f" y el Tamaño (Altura) de letra es: {Tamaño} puntos."
print(Texto_02)
```

Código Completo: PDF/60_PDF_01_Creando_Documentos_02_Texto_03_Detectar_Fuente_y_Tamanio.txt

Configurar Fuente (Tipo) y Tamaño de Letra.

Cuando trabajamos con un Documento PDF, siempre y en cualquier momento podremos incorporar, definir y/o cambiar de tipo de letra, o tamaño, según sea necesario.

```
# Fuente a incorporar
Fuente = "Arial"
ArchivoFuente = Fuente + ".ttf"
```

```
TamañoY = 14   # Recuerda: Tamaño hace referencia a la Altura en Puntos

#Cargo Info de la Fuente no Contenida en reportlab
#pdfmetrics.registerFont(TTFont( "Arial", "Arial.ttf"))
pdfmetrics.registerFont(TTFont( Fuente, ArchivoFuente))

#c.setFont("Arial", 14)
c.setFont(Fuente, TamañoY)    #Defino Fuente y TamañoY (en Puntos) de la Letra

Texto_01 = f"Ejemplo de texto {Fuente} tamaño {TamañoY} (puntos de altura)."
c.drawString(30, Alto - 50, Texto_01)

#TamanioX = pdfmetrics.stringWidth(Texto_01, "Arial", 14)
TamañoX = pdfmetrics.stringWidth(Texto_01, Fuente, TamañoY)  # Largo en Puntos
Texto_02 = f"Largo de la cadena: {TamañoX} puntos"
c.drawString(30, Alto - 100, Texto_02)

c.setFont("Helvetica", 14)   #Cambio de Fuente
Texto_03 = "La funcion len() devuelve Largo del String en Caracteres y equivale a la"
c.drawString(30, Alto - 130, Texto_03)
```

Código Completo: PDF/60_PDF_01_Creando_Documentos_02_Texto_04_Configurar_Fuente_y_Tamanio.txt

Justificación y Centrado del Texto.

Para justificar un texto a la derecha o centrarlo en la pagina, primero es necesario calcular la longitud en Puntos de toda la Cadena o Texto; para esto usaremos el método "**stringWidth(Texto,Fuente,Tamaño)**" que pertenece a la librería "pdfmetrics". La que deberemos importar de la siguiente forma "**from reportlab.pdfbase import pdfmetrics**".

Longitud de una Cadena en Puntos: Veamos y analicemos un ejemplo.

Código Completo: PDF/60_PDF_01_Creando_Documentos_02_Texto_04_Tamanio_en_Puntos.txt

Justificación y Centrado: Ya sabiendo el largo en puntos, fuente y tamaño (altura) de la cadena que hay que justificar o centrar, podemos trabajar y obtener los resultados deseados.

Código Completo: PDF/60_PDF_01_Creando_Documentos_02_Texto_05_Justificacion_y_Centrado.txt

Agregar y/o Numerar las Páginas del Documento.

Si bajas y analizas los códigos completos del ejemplo, encontraras que creamos una función que se encarga de escribir el numero de pagina, que la usamos justo antes de la llamada a un nuevo método "**c.showPage()**". Este método le indica a "ReportLab" que ya hemos terminado de trabajar en la hoja actual y queremos pasar a la siguiente. Recuerda que la nueva página se podrá ver cuando hayamos trabajado en la segunda página (no aparecerá en el documento en tanto no se haya dibujado algo) y que es una buena práctica hacerlo antes de invocar al método "**c.save()**". Analiza el Ejemplo:

```
def NroPagina(c):   #Pongo Nro de Página.
    PaginaNumero = c.getPageNumber()   # Cantidad de hojas definidas
    Paginado = f"Página {PaginaNumero}"   #Armo el renglón
    Pos_x = int((Ancho/2.0) - len(Paginado)/2.0)    #Calculo Posición
    c.drawString(Pos_x, 15, Paginado)    #Escribo en el PDF

Texto = "Este es el contenido de la Pagina 01"
c.drawString(50, Alto - 50, Texto)   # Escribo en PDF

NroPagina(c)   # Numero La Página
c.showPage()   # Termina Página

Texto = " Este es el contenido de la Pagina 02"
c.drawString(50, Alto - 50, Texto)   # Escribo en PDF

NroPagina(c)   # Numero La Página
c.showPage()   # Termina Página

c.save()   #Grabo y termino Documento PDF
```

Código Completo: PDF/60_PDF_01_Creando_Documentos_03_Tamaño_Hoja_05_Otra_Hoja.txt

Imprimir Documento PDF Desde tu Programa

Algo muy útil, es imprimir un documento PDF desde tu programa, a continuación dispones dos alternativas, usa la que mejor se adapte a tu aplicación Phyton.

Siempre asegúrate de que el documento PDF exista y que conoces la ruta de acceso. En ambos ejemplos, solo para que sean más amplios, se crea el documento y luego se IMPRIME.

Hay que destacar que existen muchas maneras de imprimir documentos PDF. Aca por simplicidad, optamos por usar los recursos del sistema operativo, quien se encarga de realizar toda la gestión con la impresora; desde Python solo le entregamos el documento y le damos la orden de imprimir. Recomiendo que obtengas los códigos completos y realices los análisis correspondiente.

Alternativa 01: Esta forma es la más simple y genérica, ya que usaremos una librería que generalmente ya esta en tu Sistema, sin embargo, dependiendo de la Configuración de tu equipo (vinculación entre tipo de archivos y programa que lo maneja), podría generar algún problema durante la impresión (problema generalmente molesto). A continuación codificación Python del ejemplo:

```
import os # Para Imprimir PDF
from reportlab.lib.pagesizes import A4
from reportlab.pdfgen import canvas

Ruta="" # Datos del Archivo
Documento = "60_Mostrar_X_Monitor_01.pdf"
c = canvas.Canvas(Ruta+Documento, pagesize=A4)

c.drawString(50, Alto - 50, Texto) # Escribo en PDF

c.showPage( ) # Termina Página
c.save( )

# Sistema Operativo Utiliza Impresora Predeterminada del Sistema
os.system(Ruta+Documento)
```

Código Completo: PDF/60_PDF_50_Especiales_02_Impresora_01_Documento_01_PDF.txt

Alternativa 02: En este segundo caso, usaremos una librería alternativa (aspose.pdf), que seguramente deberás instalar, en la forma acostumbrada, solo escribe en la consola de comandos:

```
pip install aspose.pdf
```

Una vez instalada la biblioteca, ya estas en condiciones de continuar con el programa. Analízalo, y continúa con tu estudio.

```
import aspose.pdf as ap # Para Imprimir PDF
from reportlab.lib.pagesizes import A4
from reportlab.pdfgen import canvas

Ruta="" # Datos del Archivo
Documento = "60_Mostrar_X_Monitor_01.pdf"
c = canvas.Canvas(Ruta+Documento, pagesize=A4)

c.drawString(50, Alto - 50, Texto) # Escribo en PDF

c.showPage( ) # Termina Página
c.save( )

# Se Utilizará Impresora Predeterminada del Sistema
viewer = ap.facades.PdfViewer() # Crear objeto PDFViewer
viewer.bind_pdf( Ruta + Documento ) # Abrir archivo PDF de entrada
viewer.print_document() # Imprimir un documento PDF
viewer.close() # Cerrar archivo PDF
```

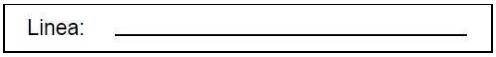
Código Completo: PDF/60_PDF_50_Especiales_02_Impresora_01_Documento_02_PDF_Alternativo.txt

Líneas y Figuras Geométricas

ReportLab permite crear (dibujar) líneas, rectángulos, círculos y otras figuras de una forma sencilla. Veamos todo el tema con ejemplos explicados.

Líneas.

Veamos como realizar (dibujar) una línea. Esto es muy simple, solo necesitaremos invocar (usar) el método "**line(xi, yi, xf, yf)**", a la que entregamos como argumentos las posiciones del punto inicial y final del segmento: **xi, yi, xf, yf**. Solo hay que destacar que si el y inicial es distinto del y final, tendremos una línea inclinada. Veamos el segmento de código, analízalo:

Linea: 

```
# ==== Cálculo de parámetros y dibujo de la línea =======
xi = 50 # Puntos que me desplazo desde Margen Izquierdo
yi = Alto - 50 # Puntos que bajo desde el Margen Superior
xf = xi + 200 # Cuanto me desplazo Desde el x inicial
yf = yi   # yf es igual a yi / Me mantengo la misma Altura
c.line(xi, yi, xf, yf)  # Dibujo Línea
print(" Línea Dibujada. Mira el Documento PDF Creado")
```

Código Completo: PDF/60_PDF_01_Creando_Documentos_04_Dibujando_01_Lineas.txt

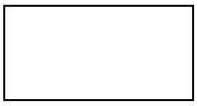
En el código completo, encontraras una línea horizontal y otra inclinada. Siempre puedes dibujar la cantidad de líneas que necesites y en la posición que quieras.

Grosor de línea: Puedes configurar el grosor de la línea, además si tiene las puntas puntiagudas, cuadradas o redondeadas, esta ultima, especial para hacer uniones entre líneas.

Código Completo: PDF/60_PDF_01_Creando_Documentos_04_Dibujando_01_Lineas_con_Grosor.txt

Rectángulos.

Esquinas rectas: Dibujar rectángulos continúa siendo simple. Para ello disponemos del método "**rect(xi, yi, b, h)**". Debes recordar que todo trabajo que realicemos se mide (calculan coordenadas) partiendo del origen, por lo tanto, las coordenadas que le entregaremos como argumento (parámetros), serán las del punto inferior izquierdo del rectángulo ("x" inicial e "y" inicial) y a continuación le damos el tamaño de la Base (Cuanto nos desplazamos desde el "x" inicial) y la Altura (cuanto subiremos desde el "y" inicial). Mira el segmento de código, analízalo detalladamente:

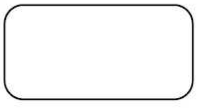
Rectángulo 1: 

```
# Coordenadas del Punto Inferior Izquierdo del Rectángulo
xi = 130   # Puntos que me desde el Margen Izquierdo
yi = Alto - 100   # Puntos que bajo desde el margen Superior

# Tamaño de la Base y Altura del Rectángulo
b = 100   # Tamaño de la Base: Cuanto me desplazo desde la "x" inicial
h = 50   # Tamaño de la Altura: Cuanto subir desde ´"y" inicial

c.rect(xi, yi, b, h)  # Dibujo Rectángulo Esquinas Rectas
print(" Rectángulo Dibujado. Mira el Documento PDF Creado")
```

Código Completo: PDF/60_PDF_01_Creando_Documentos_04_Dibujando_02_Rectangulo_Recto.txt

Esquinas redondeadas: También podremos dibujar rectángulos con las esquinas redondeadas. Para esto usaremos el método:

roundRect(xi, yi, Base, Altura, Radio)

Rectángulo 2: 

Que requiere un quinto argumento (parámetro) que indica el radio según el cual se curvan se curvan las esquinas. Estos argumentos son: "xi, yi" las coordenadas del punto inferior izquierdo del rectángulo, a continuación le entregamos el tamaño de la Base (Cuanto nos desplazamos desde el

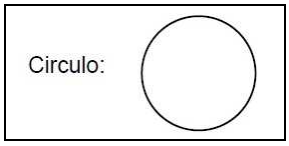
"x" inicial) y la Altura (cuanto subiremos desde el "y" inicial) y finalmente el radio de la curvatura (que tendrá el rectángulo en las esquinas). Veamos el Código Ejemplo:

```
# Coordenadas del Punto Inferior Izquierdo del Rectángulo
xi = 130   # Puntos que me desde el Margen Izquierdo
yi = Alto - 100   # Puntos que bajo desde el margen Superior

# Tamaño de la Base y Altura del Rectángulo
b = 100   # Tamaño de la Base: Cuanto me desplazo desde la "x" inicial
h = 50   # Tamaño de la Altura: Cuanto subir desde ´"y" inicial

c.roundRect(xi, yi, b, h, 10) # Dibujo Rectángulo Esquinas Redondeadas
print(" Rectángulo Dibujado. Mira el Documento PDF Creado")
```

Código Completo: PDF/60_PDF_01_Creando_Documentos_04_Dibujando_02_Rectangulo_Redondeado.txt

Círculos.

En este caso, el método usado es "**c.circle(xi, yi, R)**" y los argumentos que se le entregan son: la posición del centro ("x" Inicial e "y" Inicial), seguido del radio del Circulo. Recuerda que todas las posiciones y tamaños se miden en puntos desde la coordenada 0,0 (abajo a la izquierda de la pantalla). Veamos el Código Ejemplo:

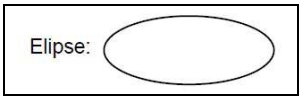


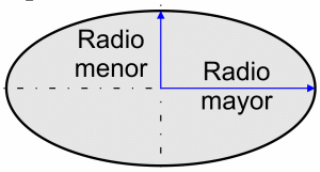

```
# Coordenadas del Centro del Circulo
xi = 140 # Puntos que me desde el Margen Izquierdo
yi = Alto - 100 # Puntos que bajo desde el Margen Superior

R= 30 # Tamaño del Radio

c.circle(xi, yi, R) # Dibujo del Circulo
```

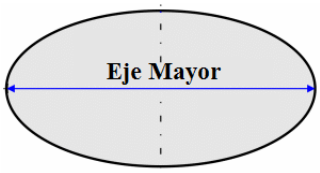
Código Completo: PDF/60_PDF_01_Creando_Documentos_04_Dibujando_03_Circulo.txt

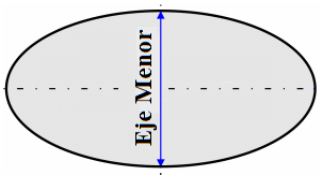
Elipse.

Para dibujar esta figura usaremos el método "c.ellipse(xi, yi, xf, yf)" en donde "x" inicial y "x" final representan las Posiciones del inicio y final del "Eje Mayor" y por otro lado "y" inicial e "y" final representa las posiciones inicial y final del eje Menor. Para mayor claridad veamos las imágenes donde visualizamos cada uno de los ejes por separado.







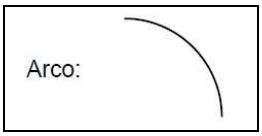


En ocasiones, los ejes mayor y menor, también son llamados Diámetro Mayor y Diámetro Menor, pero en este caso, no tiene importancia. También, al realizar esta figura, puedes intercambiar los tamaños de los ejes, y cambiaras la orientación de la elipse. Ahora analicemos el código completo:

```
# Origen y final del Eje Mayor
xi = 140 # Puntos que me desde el Margen Izquierdo
xf = xi + 150

# Origen y Final del Eje Menor
yi = Alto - 100 # Puntos que bajo desde el Margen Superior
yf = yi - 50

c.ellipse(xi, yi, xf, yf) # Dibujo del Elipse
print(" Elipse Dibujada. Mira el Documento PDF Creado")
```

Código Completo: PDF/60_PDF_01_Creando_Documentos_04_Dibujando_04_Elipse.txt

Arco.



Generalmente esta figura la usaremos para completar figuras o dibujos mayores. El método que usaremos para esta figura es "**c.arc(xi, yi, xf, yf)**", a la que entregamos como argumentos las posiciones del punto inicial y final del arco. A continuación encontraras el segmento de código que lo dibuja, lo ideal es que bajes de la nube el código completo y lo analices:

```
# Coordenadas del Inicio del Arco
xi = 50  #Puntos que desplazo desde Margen Izquierdo
yi = Alto - 50  #Puntos que bajo desde el Margen Superior

# Coordenadas del Final del Arco
xf= 150   #Puntos que desplazo desde Margen Izquierdo
yf= Alto - 150  #Puntos que bajo desde el Margen Superior

c.arc(xi, yi, xf, yf)   #Arco del Circulo
print(" Arco Dibujado. Mira el Documento PDF Creado")
```

Código Completo: PDF/60_PDF_01_Creando_Documentos_04_Dibujando_05_Arco.txt