# **Tutorial**

<https://pyfpdf.readthedocs.io/en/latest/Tutorial/index.html>

## **Minimal Example**

Let's start with the classic example:

**from** fpdf **import** FPDF

pdf = FPDF()

pdf.add\_page()

pdf.set\_font('Arial', 'B', 16)

pdf.cell(40, 10, 'Hello World!')

pdf.output('tuto1.pdf', 'F')

After including the library file, we create an FPDF object. The [FPDF](https://pyfpdf.readthedocs.io/en/latest/reference/FPDF/index.html) constructor is used here with the default values: pages are in A4 portrait and the measure unit is millimeter. It could have been specified explicitly with:

pdf = FPDF('P', 'mm', 'A4')

It is possible to use landscape (L), other page formats (such as Letter and Legal) and measure units (pt, cm, in).

There is no page for the moment, so we have to add one with [add\_page](https://pyfpdf.readthedocs.io/en/latest/reference/add_page/index.html). The origin is at the upper-left corner and the current position is by default placed at 1 cm from the borders; the margins can be changed with [set\_margins](https://pyfpdf.readthedocs.io/en/latest/reference/set_margins/index.html).

Before we can print text, it is mandatory to select a font with [set\_font](https://pyfpdf.readthedocs.io/en/latest/reference/set_font/index.html), otherwise the document would be invalid. We choose Arial bold 16:

pdf.set\_font('Arial', 'B', 16)

We could have specified italics with I, underlined with U or a regular font with an empty string (or any combination). Note that the font size is given in points, not millimeters (or another user unit); it is the only exception. The other standard fonts are Times, Courier, Symbol and ZapfDingbats.

We can now print a cell with [cell](https://pyfpdf.readthedocs.io/en/latest/reference/cell/index.html). A cell is a rectangular area, possibly framed, which contains some text. It is output at the current position. We specify its dimensions, its text (centered or aligned), if borders should be drawn, and where the current position moves after it (to the right, below or to the beginning of the next line). To add a frame, we would do this:

pdf.cell(40, 10, 'Hello World!', 1)

To add a new cell next to it with centered text and go to the next line, we would do:

pdf.cell(60, 10, 'Powered by FPDF.', 0, 1, 'C')

****Remark****: the line break can also be done with [ln](https://pyfpdf.readthedocs.io/en/latest/reference/ln/index.html). This method allows to specify in addition the height of the break.

Finally, the document is closed and sent to the browser with [output](https://pyfpdf.readthedocs.io/en/latest/reference/output/index.html). We could have saved it in a file by passing the desired file name.

****Caution****: in case when the PDF is sent to the browser, nothing else must be output, not before nor after (the least space or carriage return matters). If you send some data before, you will get the error message: "Some data has already been output to browser, can't send PDF file". If you send after, your browser may display a blank page.

## **Header, footer, page break and image**

Here is a two page example with header, footer and logo:

**from** fpdf **import** FPDF

**class** **PDF**(FPDF):

**def** **header**(self):

*# Logo*

self.image('logo\_pb.png', 10, 8, 33)

*# Arial bold 15*

self.set\_font('Arial', 'B', 15)

*# Move to the right*

self.cell(80)

*# Title*

self.cell(30, 10, 'Title', 1, 0, 'C')

*# Line break*

self.ln(20)

*# Page footer*

**def** **footer**(self):

*# Position at 1.5 cm from bottom*

self.set\_y(-15)

*# Arial italic 8*

self.set\_font('Arial', 'I', 8)

*# Page number*

self.cell(0, 10, 'Page ' + str(self.page\_no()) + '/{nb}', 0, 0, 'C')

*# Instantiation of inherited class*

pdf = PDF()

pdf.alias\_nb\_pages()

pdf.add\_page()

pdf.set\_font('Times', '', 12)**for** i **in** range(1, 41):

pdf.cell(0, 10, 'Printing line number ' + str(i), 0, 1)

pdf.output('tuto2.pdf', 'F')

[Demo](https://github.com/reingart/pyfpdf/raw/master/tutorial/tuto2.pdf)

This example makes use of the [header](https://pyfpdf.readthedocs.io/en/latest/reference/header/index.html) and [footer](https://pyfpdf.readthedocs.io/en/latest/reference/footer/index.html) methods to process page headers and footers. They are called automatically. They already exist in the FPDF class but do nothing, therefore we have to extend the class and override them.

The logo is printed with the [image](https://pyfpdf.readthedocs.io/en/latest/reference/image/index.html) method by specifying its upper-left corner and its width. The height is calculated automatically to respect the image proportions.

To print the page number, a null value is passed as the cell width. It means that the cell should extend up to the right margin of the page; it is handy to center text. The current page number is returned by the [page\_no](https://pyfpdf.readthedocs.io/en/latest/reference/page_no/index.html) method; as for the total number of pages, it is obtained by means of the special value {nb} which will be substituted on document closure (provided you first called [alias\_nb\_pages](https://pyfpdf.readthedocs.io/en/latest/reference/alias_nb_pages/index.html)). Note the use of the [set\_y](https://pyfpdf.readthedocs.io/en/latest/reference/set_y/index.html) method which allows to set position at an absolute location in the page, starting from the top or the bottom.

Another interesting feature is used here: the automatic page breaking. As soon as a cell would cross a limit in the page (at 2 centimeters from the bottom by default), a break is performed and the font restored. Although the header and footer select their own font (Arial), the body continues with Times. This mechanism of automatic restoration also applies to colors and line width. The limit which triggers page breaks can be set with [set\_auto\_page\_break](https://pyfpdf.readthedocs.io/en/latest/reference/set_auto_page_break/index.html).

## **Line breaks and colors**

Let's continue with an example which prints justified paragraphs. It also illustrates the use of colors.

**from** fpdf **import** FPDF

title = '20000 Leagues Under the Seas'

**class** **PDF**(FPDF):

**def** **header**(self):

*# Arial bold 15*

self.set\_font('Arial', 'B', 15)

*# Calculate width of title and position*

w = self.get\_string\_width(title) + 6

self.set\_x((210 - w) / 2)

*# Colors of frame, background and text*

self.set\_draw\_color(0, 80, 180)

self.set\_fill\_color(230, 230, 0)

self.set\_text\_color(220, 50, 50)

*# Thickness of frame (1 mm)*

self.set\_line\_width(1)

*# Title*

self.cell(w, 9, title, 1, 1, 'C', 1)

*# Line break*

self.ln(10)

**def** **footer**(self):

*# Position at 1.5 cm from bottom*

self.set\_y(-15)

*# Arial italic 8*

self.set\_font('Arial', 'I', 8)

*# Text color in gray*

self.set\_text\_color(128)

*# Page number*

self.cell(0, 10, 'Page ' + str(self.page\_no()), 0, 0, 'C')

**def** **chapter\_title**(self, num, label):

*# Arial 12*

self.set\_font('Arial', '', 12)

*# Background color*

self.set\_fill\_color(200, 220, 255)

*# Title*

self.cell(0, 6, 'Chapter %d : %s' % (num, label), 0, 1, 'L', 1)

*# Line break*

self.ln(4)

**def** **chapter\_body**(self, name):

*# Read text file*

**with** open(name, 'rb') **as** fh:

txt = fh.read().decode('latin-1')

*# Times 12*

self.set\_font('Times', '', 12)

*# Output justified text*

self.multi\_cell(0, 5, txt)

*# Line break*

self.ln()

*# Mention in italics*

self.set\_font('', 'I')

self.cell(0, 5, '(end of excerpt)')

**def** **print\_chapter**(self, num, title, name):

self.add\_page()

self.chapter\_title(num, title)

self.chapter\_body(name)

pdf = PDF()

pdf.set\_title(title)

pdf.set\_author('Jules Verne')

pdf.print\_chapter(1, 'A RUNAWAY REEF', '20k\_c1.txt')

pdf.print\_chapter(2, 'THE PROS AND CONS', '20k\_c2.txt')

pdf.output('tuto3.pdf', 'F')

[Demo](https://github.com/reingart/pyfpdf/raw/master/tutorial/tuto3.pdf)

The [get\_string\_width](https://pyfpdf.readthedocs.io/en/latest/reference/get_string_width/index.html) method allows determining the length of a string in the current font, which is used here to calculate the position and the width of the frame surrounding the title. Then colors are set (via [set\_draw\_color](https://pyfpdf.readthedocs.io/en/latest/reference/set_draw_color/index.html), [set\_fill\_color](https://pyfpdf.readthedocs.io/en/latest/reference/set_fill_color/index.html) and [set\_text\_color](https://pyfpdf.readthedocs.io/en/latest/reference/set_text_color/index.html)) and the thickness of the line is set to 1 mm (against 0.2 by default) with [set\_line\_width](https://pyfpdf.readthedocs.io/en/latest/reference/set_line_width/index.html). Finally, we output the cell (the last parameter to true indicates that the background must be filled).

The method used to print the paragraphs is [multi\_cell](https://pyfpdf.readthedocs.io/en/latest/reference/multi_cell/index.html). Each time a line reaches the right extremity of the cell or a carriage return character is met, a line break is issued and a new cell automatically created under the current one. Text is justified by default.

Two document properties are defined: the title ([set\_title](https://pyfpdf.readthedocs.io/en/latest/reference/set_title/index.html)) and the author ([set\_author](https://pyfpdf.readthedocs.io/en/latest/reference/set_author/index.html)). Properties can be viewed by two means. First is to open the document directly with Acrobat Reader, go to the File menu and choose the Document Properties option. The second, also available from the plug-in, is to right-click and select Document Properties.

## **Installation Notes**

Previously, to import the object you should use the pyfpdf package:

**from** pyfpdf **import** FPDF

After version 1.7, to import it you should use the fpdf package:

**from** fpdf **import** FPDF