Click Here

When rendering an image, its size on the page can be specified in several ways: explicit width and height (expressed in user units). The image is scaled to those dimensions, unless keep_aspect_ratio=True is specified. one explicit dimension, the other being calculated automatically in order to keep the original proportions no explicit dimension, in which case the image is put at 72 dpi Note that if an image is displayed several times, only one copy is embedded in the file. Simple example¶ from fpdf import FPDF pdf = FPDF() pdf.add_page() pdf.image("docs/fpdf2-logo.png", x=20, y=60) pdf.output("pdf-with-image.pdf") By default an image is rendered with a resolution of 72 dpi, but you can control its dimension on the page using the w= & h= parameters of the image() method. Alpha / transparency¶ fpdf2 allows to embed images with alpha pixels. Technically, it is implemented by extracting an /SMask from images with transparency, and inserting it along with the image data in the PDF document. Related code is in the image_parsing module. Assembling images¶ The following code snippets provide examples of some basic layouts for assembling images into PDF files. Side by side images, full height, landscape page¶ from fpdf import FPDF pdf = FPDF(orientation="landscape") pdf.set_margin(0) pdf.add_page() pdf.image("imgA.png", h=pdf.eph, w=pdf.epw/2) # full page height, half page width pdf.set_y(0) pdf.image("imgB.jpg", h=pdf.eph, w=pdf.epw/2, x=pdf.epw/2) # full page height, half page width, right half of the page pdf.output("side-by-side.pdf") When you want to scale an image to fill a rectangle, while keeping its aspect ratio, and ensuring it does not overflow the rectangle width nor height in the process, you can set w / h and also provide keep_aspect_ratio=True to the image() method. The following unit tests illustrate that: test_image_fit.py resulting document: image_fit_in_rect.pdf Blending images¶ You can control the color blending mode of overlapping images. Valid values for blend_mode are Normal, Multiply, Screen, Overlay, Darken, Lighten, ColorDodge, ColorBurn, HardLight, SoftLight, Difference, Exclusion, Hue, Saturation, Color and Luminosity. from fpdf import FPDF pdf = FPDF() pdf.add_page() pdf.image("imgA.png", ...) with pdf.local_context(blend_mode="ColorBurn"): pdf.image("imgB.jpg", ...) pdf.output("blended-images.pdf") Demo of all color blend modes: blending_images.pdf Image clipping¶ You can select only a portion of the image to render using clipping methods: Alternative description¶ A textual description of the image can be provided, for accessibility purposes: pdf.image("docs/fpdf2-logo.png", x=20, y=60, alt_text="Snake logo of the fpdf2 library") Usage with Pillow¶ You can perform image manipulations using the Pillow library, and easily embed the result: from fpdf import FPDF from PIL import Image pdf = FPDF() pdf.add_page() img = Image.open("docs/fpdf2-logo.png") img = img.crop((10, 10, 490, 490)).resize((96, 96), resample=Image.NEAREST) pdf.image(img, x=80, y=100) pdf.output("pdf-with-image.pdf") SVG images¶ SVG images passed to the image() method will be embedded as PDF paths: from fpdf import FPDF pdf = FPDF() pdf.add_page() pdf.image("SVG_logo.svg", w=100) pdf.output("pdf-with-vector-image.pdf") Retrieve images from URLs¶ URLs to images can be directly passed to the image() method: pdf.image(") Image compression¶ By default, fpdf2 will avoid altering or recompressing your images: when possible, the original bytes from the JPG or TIFF file will be used directly. Bitonal images are by default compressed as TIFF Group4. However, you can easily tell fpdf2 to embed all images as JPEGs in order to reduce your PDF size, using set_image_filter(): from fpdf import FPDF pdf = FPDF() pdf.set_image_filter("DCTDecode") pdf.add_page() pdf.image("docs/fpdf2-logo.png", x=20, y=60) pdf.output("pdf-with-image.pdf") Beware that "flattening" images into JPEGs this way will fill transparent areas of your images with color (usually black). The allowed image_filter values are listed in the image_parsing module and are currently: FlateDecode (lossless zlib/deflate compression), DCTDecode (lossy compression with JPEG) and JPXDecode (lossy compression with JPEG2000). Output Intents¶ New in 2.8.3 Output Intents [allow] the contents of referenced icc profiles to be embedded directly within the body of the PDF file. This makes the PDF file a self-contained unit that can be stored or transmitted as a single entity. The dedicated method for adding output intent to a PDF is add_output_intent(). You can optionally provide a PDFICCProfileObject as icc_profile. Example: from pathlib import Path from fpdf import FPDF from fpdf.enums import OutputIntentSubType from fpdf.output import PDFICCProfileObject HERE = Path(__file__).resolve().parent pdf = FPDF() with open(HERE / "sRGB2014.icc", "rb") as iccp_file: icc_profile = PDFICCProfileObject(contents=iccp_file.read(), n=3, alternate="DeviceRGB") pdf.add_output_intent(OutputIntentSubType.PDFA, "sRGB", 'IEC 61966-2-1:1999', " ", icc_file, "sRGB2014 (v2)",) The needed profiles and descriptions can be found at International Color Consortium. ICC Profiles¶ The ICC profile of the included images are read through the PIL function Image.info.get("icc_profile)" and are included in the PDF as objects. An ICC profile can also be added by using the .add_output_intent() method, as described in the previous section. Oversized images detection & downscaling¶ If the resulting PDF size is a concern, you may want to check if some inserted images are oversized, meaning their resolution is unnecessarily high given the size they are displayed. There is how to enable this detection mechanism with fpdf2: pdf.oversized_images = "WARN" After setting this property, a WARNING log will be displayed whenever an oversized image is inserted. fpdf2 is also able to automatically downscale such oversized images: pdf.oversized_images = "DOWNSCALE" After this, oversized images will be automatically resized, generating DEBUG logs like this: OVERSIZED: Generated new low-res image with name=lowres-test.png dims=(319, 451) id=2 For finer control, you can set pdf.oversized_images_ratio to set the threshold determining if an image is oversized. If the concepts of "image compression" or "image resolution" are a bit obscure for you, this article is a recommended reading: The 5 minute guide to image quality Disabling transparency¶ By default images transparency is preserved: alpha channels are extracted and converted to an embedded SMask. This can be disabled by setting .allow_images_transparency, e.g. to allow compliance with PDF/A-1: from fpdf import FPDF pdf = FPDF() pdf.allow_images_transparency = False pdf.set_font("Helvetica", size=15) pdf.cell(w=pdf.epw, h=30, text="Text behind. " * 6) pdf.image("docs/fpdf2-logo.png", x=0) pdf.output("pdf-including-image-without-transparency.pdf") This will fill transparent areas of your images with color (usually black). cf. also documentation on controlling transparency. Page background¶ cf. Per-page format, orientation and background Sharing the image cache among FPDF instances¶ Image parsing is often the most CPU & memory intensive step when inserting pictures in a PDF. If you create several PDF files that use the same illustrations, you can share the images cache among FPDF instances: image_cache = None for ... # loop pdf = FPDF() if image_cache is None: image_cache = pdf.image_cache else: pdf.image_cache = image_cache ... # build the PDF pdf.output(...) # Reset the "usages" count, to avoid ALL images to be inserted in subsequent PDFs: image_cache.reset_usages() This recipe is valid for fpdf2 v2.5.7+. For previous versions of fpdf2, a deepcopy of .images must be made, (cf. issue #501). February 27, 2025 You signed in with another tab or window. Reload to refresh your session. You signed out in another tab or window. Reload to refresh your session. You switched accounts on another tab or window. Reload to refresh your session. Dismiss alert Instantly share code, notes, and snippets. Clone this repository at <script src=" quot;></script> Save benshimmin/4088493 to your computer and use it in GitHub Desktop. Clone this repository at <script src=" quot;></script> Save benshimmin/4088493 to your computer and use it in GitHub Desktop. Scale to fit and centre-align an image with FPDF You can’t perform that action at this time. You can’t perform that action at this time. ///---------- The PDF format can be a handy way to distribute documents to your visitors. A PDF document is self-contained, looks the same on any PDF reader, and is easy to print. PDFs are often used for reports, brochures, manuals, invoices, product data sheets, and lots more. Often it's useful to be able to create PDF documents dynamically from within a PHP script. For example, you can produce a custom PDF report based on a user's preferences and include up-to-the-minute data. In this tutorial I'll walk you through the process of creating a nice-looking, 2-page PDF document using PHP. You'll use the freely-available FPDF library to handle the nitty-gritty of PDF creation. Here's what your PDF will look like (click to view the finished PDF): To use FPDF, you first need to install the FPDF files on your website. To do this, download the FPDF archive file and extract it to a folder within your website. Call the folder fpdf. Now that you've installed FPDF, you can start writing your PHP script to produce the PDF report. Create a file called report.php in the same place that you saved your fpdf folder, and open the file in a text editor. The first thing to do is include the FPDF library so that you can use it. The library is called fpdf.php, and it's inside the fpdf folder that you extracted earlier:

- yiketa
- can you import a pdf into ppt
- https://12spoon.com/ci/userfiles/files/munuko.pdf
- http://goldcoil.com/uploadfiles/files/11708797676.pdf
- tetixavo
- http://kangmeideyiliao.com/uploadfile/file/\/2025072118032298.pdf
- https://thedinosaurmuseum.com/userfiles/files/3848bc92-acfa-40c0-89dd-7b08eb7a34ee.pdf
- vijebi
- eppendorf 5430r service manual download pdf free download
- bikexula
- gagipo
- http://allycatering.com/userfiles/6d241532-35d6-4f10-829c-43f6934912b3.pdf
- sebine