Click to verify

Author: David Hernández Sanz License: FPDF This extension allows to draw lines, rectangles, ellipses, polygons and curves with line style. Public methods are: • SetLineStyle($style: array) Sets line style. Parameters are: - style: an array with the following possible keys: . width: width of the line in user units. . cap: type of cap to put on the line (butt, round, square). The difference between "square" and "butt" is that "square" projects a flat end past the end of the line. . join: miter, round, or bevel. . dash: dash pattern. Is 0 (without dash) or array with series of length values, which are the lengths of the on and off dashes. . phase: modifier of the dash pattern which is used to shift the point at which the pattern starts. . color: draw color. Array with components (red, green, blue). • Line($x1: float, $y1: float, $x2: float, $y2: float, $style: array) Draws a line. Overrides the method from FPDF. Parameters are: - x1, y1: start point. - x2, y2: end point. - style: line style (array like for SetLineStyle). • Rect($x: float, $y: float, $w: float, $h: float, $style: string, $border_style: array, $fill_color: array) Draws a rectangle. Overrides the method from FPDF. Parameters are: - x, y: top left corner. - w, h: width and height. - style: style of rectangle (draw and/or fill: D, F, DF, FD). - border_style: border style of rectangle. Array with index: (all => style) for all borders, or (L => style, T => style, R => style, B => style) for each border. Style is an array like for SetLineStyle. - fill_color: fill color. Array with components (red, green, blue). • Curve($x0: float, $y0: float, $x1: float, $y1: float, $x2: float, $y2: float, $x3: float, $y3: float, $style: string, $line_style: array, $fill_color: array) Draws a Bézier curve. Parameters are: - x0, y0: start point. - x1, y1: control point 1. - x2, y2: control point 2. - x3, y3: end point. - style: style of rectangle (draw and/or fill: D, F, DF, FD). - line_style: line style for curve. Array like for SetLineStyle. - fill_color: fill color. Array with components (red, green, blue). • Ellipse($x0: float, $y0: float, $rx: float, $ry: float, $angle: float, $astart: float, $afinish: float, $style: string, $line_style: array, $fill_color: array, $nSeg: integer) Draws an ellipse. Parameters are: - x0, y0: center point. - rx, ry: horizontal and vertical radius (if ry is 0, this is a circle). - angle: orientation angle (anti-clockwise). - astart: start angle. - afinish: finish angle. - style: style of ellipse (draw and/or fill: D, F, DF, FD, C (D + close)). - line_style: line style for ellipse. Array like for SetLineStyle. - fill_color: fill color. Array with components (red, green, blue). - nSeg: ellipse is made up of nSeg Bézier curves. • Circle($x0: float, $y0: float, $r: float, $astart: float, $afinish: float, $style: string, $line_style: array, $fill_color: array, $nSeg: integer) Draws a circle. Parameters are: - x0, y0: center point. - r: radius. - astart: start angle. - afinish: finish angle. - style: style of circle (draw and/or fill: D, F, DF, FD, C (D + close)). - line_style: line style for circle. Array like for SetLineStyle. - fill_color: fill color. Array with components (red, green, blue). - nSeg: circle is made up of nSeg Bézier curves. • Polygon($p: array, $style: string, $line_style: array, $fill_color: array) Draws a polygon. Parameters are: - p: points. Array with values x0, y0, x1, y1,..., x(np-1), y(np-1). - style: style of polygon (draw and/or fill: D, F, DF, FD). - line_style: line style. Array with index (all => style) for all borders, or (0..np-1 => style) for each border. Style is an array like for SetLineStyle. - fill_color: fill color. Array with components (red, green, blue) • RegularPolygon($x0: float, $y0: float, $r: float, $ns: Integer, $angle: float, $circle: boolean, $style: string, $line_style: array, $fill_color: array, $circle_style: array, $circle_line_style: array, $circle_fill_color: array) Draws a regular polygon. Parameters are: - x0, y0: center point. - r: radius of circumscribed circle. - ns: number of sides. - angle: orientation angle (anti-clockwise). - circle: draw circumscribed circle or not. - style: style of polygon (draw and/or fill: D, F, DF, FD). - line_style: line style. Array with index (all => style) for all borders, or (0..ns-1 => style) for each border. Style is an array like for SetLineStyle. - fill_color: fill color. Array with components (red, green, blue). - circle_style: style of circumscribed circle (draw and/or fill: D, F, DF, FD). - circle_line_style: line style for circumscribed circle. Array like for SetLineStyle. - circle_fill_color: fill color for circumscribed circle. Array with components (red, green, blue). • StarPolygon($x0: float, $y0: float, $r: float, $nv: Integer, $ng: Integer, $angle: float, $circle: boolean, $style: string, $line_style: array, $fill_color: array, $circle_style: array, $circle_line_style: array, $circle_fill_color: array) Draws a star polygon. Parameters are: - x0, y0: center point. - r: radius of circumscribed circle. - nv: number of vertices. - ng: number of gaps (ng % nv = 1 => regular polygon). - angle: orientation angle (anti-clockwise). - circle: draw circumscribed circle or not. - style: style of polygon (draw and/or fill) (D, F, DF, FD). - line_style: line style. Array with index (all => style) for all borders, or (0..n-1 => style) for each border. Style is an array like for SetLineStyle. - fill_color: fill color. Array with components (red, green, blue). - circle_style: style of circumscribed circle (draw and/or fill) (D, F, DF, FD) (if draw). - circle_line_style: line style for circumscribed circle. Array like for SetLineStyle (if draw). - circle_fill_color: fill color for circumscribed circle. Array with components (red, green, blue). • RoundedRect($x: float, $y: float, $w: float, $h: float, $r: float, $round_corner: string, $style: string, $border_style: string, $fill_color: array) Draws a rounded rectangle. Parameters are: - x, y: top left corner. - w, h: width and height. - r: radius of the rounded corners. - round_corner: draws rounded corners or not. String with a 0 (not rounded i-corner) or 1 (rounded i-corner) in i-position. Positions are, in order: top left, top right, bottom right and bottom left. - style: style of rectangle (draw and/or fill: D, F, DF, FD). - border_style: border style of rectangle. Array like for SetLineStyle. - fill_color: fill color. Array with components (red, green, blue). View the result here. ZIP | TGZ Line(float x1, float y1, float x2, float y2) Draws a line between two points. x1 Abscissa of first point. y1 Ordinate of first point. x2 Abscissa of second point. y2 Ordinate of second point. SetLineWidth, SetDrawColor The following code snippets show examples of rendering various shapes. Lines¶ Using line() to draw a thin plain orange line: from fpdf import FPDF pdf = FPDF() pdf.add_page() pdf.set_line_width(0.5) pdf.set_draw_color(r=255, g=128, b=0) pdf.line(x1=50, y1=50, x2=150, y2=100) pdf.output("orange_plain_line.pdf") Drawing a dashed light blue line: from fpdf import FPDF pdf = FPDF() pdf.add_page() pdf.set_line_width(0.5) pdf.set_draw_color(r=0, g=128, b=255) pdf.set_dash_pattern(dash=2, gap=3) pdf.line(x1=50, y1=50, x2=150, y2=100) pdf.output("blue_dashed_line.pdf") Circle¶ Using circle() to draw a disc filled in pink with a grey outline: from fpdf import FPDF pdf = FPDF() pdf.add_page() pdf.set_line_width(2) pdf.set_draw_color(240) pdf.set_fill_color(r=230, g=30, b=180) pdf.circle(x=50, y=50, radius=50, style="FD") pdf.output("circle.pdf") This method changed parameters in release 2.8.1 Ellipse¶ Using ellipse(), filled in grey with a pink outline: from fpdf import FPDF pdf = FPDF() pdf.add_page() pdf.set_line_width(2) pdf.set_draw_color(r=230, g=30, b=180) pdf.set_fill_color(240) pdf.ellipse(x=50, y=50, w=100, h=50, style="FD") pdf.output("ellipse.pdf") Rectangle¶ Using rect() to draw nested squares: from fpdf import FPDF pdf = FPDF() pdf.add_page() for i in range(15): pdf.set_fill_color(255 - 15*i) pdf.rect(x=5+5*i, y=5+5*i, w=200-10*i, h=200-10*i, style="FD") pdf.output("squares.pdf") Using rect() to draw rectangles with round corners: from fpdf import FPDF pdf = FPDF() pdf.add_page() pdf.set_draw_color(200) y = 10 pdf.rect(60, y, 33, 28, round_corners=True, style="D") pdf.set_fill_color(0, 255, 0) pdf.rect(100, y, 50, 10, round_corners=("BOTTOM_RIGHT"), style="DF") pdf.set_fill_color(255, 255, 0) pdf.rect(160, y, 10, 10, round_corners=("TOP_LEFT", "BOTTOM_LEFT"), style="F") pdf.output("round_corners_rectangles.pdf") Polygon¶ Using polygon(): from fpdf import FPDF pdf = FPDF() pdf.add_page() pdf.set_line_width(2) pdf.set_fill_color(r=255, g=0, b=0) coords = ((100, 0), (5, 69), (41, 181), (159, 181), (195, 69)) pdf.polygon(coords, style="DF") pdf.output("polygon.pdf") Arc¶ Using arc(): from fpdf import FPDF pdf = FPDF() pdf.add_page() pdf.set_line_width(2) pdf.set_fill_color(r=255, g=0, b=0) pdf.arc(x=75, y=75, a=25, b=25, start_angle=90, end_angle=260, style="FD") pdf.set_fill_color(r=255, g=0, b=255) pdf.arc(x=105, y=75, a=25, b=50, start_angle=180, end_angle=360, style="FD") pdf.set_fill_color(r=0, g=255, b=0) pdf.arc(x=135, y=75, a=25, b=25, start_angle=0, end_angle=130, style="FD") pdf.output("arc.pdf") Solid arc¶ Using solid_arc(): from fpdf import FPDF pdf = FPDF() pdf.add_page() pdf.set_line_width(2) pdf.set_fill_color(r=255, g=0, b=0) pdf.solid_arc(x=75, y=75, a=25, b=25, start_angle=90, end_angle=260, style="FD") pdf.set_fill_color(r=255, g=0, b=255) pdf.solid_arc(x=105, y=75, a=25, b=50, start_angle=180, end_angle=360, style="FD") pdf.set_fill_color(r=0, g=255, b=0) pdf.solid_arc(x=135, y=75, a=25, b=25, start_angle=0, end_angle=130, style="FD") pdf.output("solid_arc.pdf") Bezier Curve¶ New in 2.8.0 Using bezier() to create a cubic Bézier curve: from fpdf import FPDF pdf = FPDF() pdf.add_page() pdf.set_fill_color(r=255, g=0, b=255) pdf.bezier([(20, 80), (40, 20), (60, 80)], style="DF") pdf.output("bezier.pdf") One of the nice properties of Bézier curves is that they can be chained: Note that, for smooth joining cubic Bézier curves, neighbor control points around the joining point must mirror each other (cf. Wikipedia). Source code: test_bezier_chaining() in test_bezier.py Regular Polygon¶ Using regular_polygon(): from fpdf import FPDF pdf = FPDF() pdf.add_page() pdf.set_line_width(0.5) pdf.set_fill_color(r=30, g=200, b=0) pdf.regular_polygon(x=40, y=80, polyWidth=30, rotateDegrees=270, numSides=3, style="FD") pdf.set_fill_color(r=10, g=30, b=255) pdf.regular_polygon(x=80, y=80, polyWidth=30, rotateDegrees=135, numSides=4, style="FD") pdf.set_fill_color(r=165, g=10, b=255) pdf.regular_polygon(x=120, y=80, polyWidth=30, rotateDegrees=198, numSides=5, style="FD") pdf.set_fill_color(r=255, g=125, b=10) pdf.regular_polygon(x=160, y=80, polyWidth=30, rotateDegrees=270, numSides=6, style="FD") pdf.output("regular_polygon.pdf") Regular Star¶ Using star(): from fpdf import FPDF pdf = FPDF() pdf.add_page() pdf.set_line_width(0.5) pdf.set_fill_color(r=255, g=0, b=0) pdf.star(x=40, y=80, r_in=5, r_out=15, rotate_degrees=0, corners=3, style="FD") pdf.set_fill_color(r=0, g=255, b=255) pdf.star(x=80, y=80, r_in=5, r_out=15, rotate_degrees=90, corners=4, style="FD") pdf.set_fill_color(r=255, g=255, b=0) pdf.star(x=120, y=80, r_in=5, r_out=15, rotate_degrees=180, corners=5, style="FD") pdf.set_fill_color(r=255, g=0, b=255) pdf.star(x=160, y=80, r_in=5, r_out=15, rotate_degrees=270, corners=6, style="FD") pdf.output("star.pdf") Path styling¶ line_width specifies the thickness of the line used to stroke a path stroke_join_style defines how the corner joining two path components should be rendered: from fpdf import FPDF from fpdf.enums import StrokeJoinStyle pdf = FPDF() pdf.add_page() pdf.set_line_width(5) pdf.set_fill_color(r=255, g=128, b=0) with pdf.local_context(stroke_join_style=StrokeJoinStyle.ROUND): pdf.regular_polygon(x=50, y=120, polyWidth=100, numSides=8, style="FD") pdf.output("regular_polygon_rounded.pdf") stroke_cap_style defines how the end of a stroke should be rendered. This affects the ends of the segments of dashed strokes, as well. from fpdf import FPDF from fpdf.enums import StrokeCapStyle pdf = FPDF() pdf.add_page() pdf.set_line_width(5) pdf.set_fill_color(r=255, g=128, b=0) with pdf.local_context(stroke_cap_style=StrokeCapStyle.ROUND): pdf.line(x1=50, y1=50, x2=150, y2=100) pdf.output("line_with_round_ends.pdf") There are even more specific path styling settings supported: dash_pattern, stroke_opacity, stroke_miter_limit... All of those settings can be set in a local_context(). February 17, 2025 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): If you want to try out the finished PHP script, you can grab the entire code at the end of the article. Installing FPDF 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. Starting the PHP script 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: