# Skalieren von PDF-Inhalten für den Druck

Ich muss Dokumente drucken, und mir ist aufgefallen, dass der Weißraum um den Inhalt oft zu groß ist, was Papier verschwendet und den Text kleiner erscheinen lässt, als er sein sollte. Dieses Skript hilft dabei, den PDF-Inhalt automatisch so zu skalieren, dass er besser auf die Seite passt, indem es den Inhaltsbereich erkennt und ihn vergrößert, um die Seite zu füllen, während ein kleiner Rand eingehalten wird.

```
import subprocess
import sys
import os
from PIL import Image
from pdf2image import convert_from_path

MARGIN_PERCENT = 0.005
DPI = 72

def convert_pixels_to_points(pixels, dpi):
    """Konvertiert Pixel in Punkte."""
    return pixels * 72 / dpi

def get_image_dimensions(image):
    """Ermittelt die Bildabmessungen in Pixel und Punkten."""
    width, height = image.size
    dpi = image.info.get('dpi', (DPI, DPI))
    width_points = convert_pixels_to_points(width, dpi[0])
    height_points = convert_pixels_to_points(height, dpi[1])
    return width, height, width_points, height_points, dpi

def analyze_whitespace(image, width, height):
    """Analysiert den Weißraum, um den Begrenzungsrahmen des Inhalts zu finden."""
    left_margin_px = width
    right_margin_px = 0
    top_margin_px = height
    bottom_margin_px = 0
    found_content = False

    for x in range(width):
        for y in range(height):
            pixel = image.getpixel((x, y))
            if isinstance(pixel, tuple):
```

```
                if any(c < 250 for c in pixel):
                    if not found_content:
                        left_margin_px = x
                        top_margin_px = y
                        found_content = True
                    right_margin_px = max(right_margin_px, x)
                    bottom_margin_px = max(bottom_margin_px, y)
            elif pixel < 250:
                if not found_content:
                    left_margin_px = x
                    top_margin_px = y
                    found_content = True
                right_margin_px = max(right_margin_px, x)
                bottom_margin_px = max(bottom_margin_px, y)

    if not found_content:
        return None, None, None, None

    right_margin_px = width - right_margin_px
    bottom_margin_px = height - bottom_margin_px
    return left_margin_px, right_margin_px, top_margin_px, bottom_margin_px

def calculate_scale_factor(input_pdf):
    """
    Ermittelt die Abmessungen der ersten Seite eines PDFs, analysiert den Weißraum
    und berechnet den Skalierungsfaktor basierend auf dem PDF-Inhalt und den Zielabmessungen im A4-Format mit
    Gibt den Skalierungsfaktor zurück oder None, falls ein Fehler auftritt.
    """
    print(f"Berechne Skalierungsfaktor für: {input_pdf}")
    try:
        images = convert_from_path(input_pdf, first_page=1, last_page=1)
        if not images:
            print("  Konnte PDF nicht in ein Bild konvertieren.")
            return None

        image = images[0]
        width, height, width_points, height_points, dpi = get_image_dimensions(image)

        margins = analyze_whitespace(image, width, height)
        if margins[0] is None:
```

```
        print("  Konnte den Begrenzungsrahmen des Inhalts nicht bestimmen.")
        left_margin_points = 0
        right_margin_points = 0
        top_margin_points = 0
        bottom_margin_points = 0
    else:
        left_margin_px, right_margin_px, top_margin_px, bottom_margin_px = margins
        content_width_px = right_margin_px - left_margin_px
        content_height_px = bottom_margin_px - top_margin_px

        left_margin_points = convert_pixels_to_points(left_margin_px, dpi[0])
        right_margin_points = convert_pixels_to_points(right_margin_px, dpi[0])
        top_margin_points = convert_pixels_to_points(top_margin_px, dpi[1])
        bottom_margin_points = convert_pixels_to_points(bottom_margin_px, dpi[1])

        print(f"  Inhaltsrahmen: links={left_margin_px}, oben={top_margin_px}, rechts={right_margin_px}, u
        print(f"  Inhaltsabmessungen (Pixel): Breite={content_width_px}, Höhe={content_height_px}")
        print(f"  Ränder (Punkte): links={left_margin_points}, rechts={right_margin_points}, oben={top_mar

    print(f"  Erkannte Abmessungen: Breite={width_points}, Höhe={height_points}")

    width_margin_points = min(left_margin_points, right_margin_points)
    height_margin_points = min(top_margin_points, bottom_margin_points)

    content_width = width_points - width_margin_points * 2
    content_height = height_points - height_margin_points * 2

    target_width = width_points * (1 - 2 * MARGIN_PERCENT)
    target_height = height_points * (1- 2 * MARGIN_PERCENT)

    width_scale = target_width / content_width
    height_scale = target_height / content_height

    print(f"  Inhaltsabmessungen (Punkte): Breite={content_width}, Höhe={content_height}")

    if content_width <= 0 or content_height <= 0:
        print("Fehler: Konnte die Inhaltsabmessungen nicht bestimmen.")
        return None

    print(f"  Zielabmessungen: Breite={target_width}, Höhe={target_height}")
```

```
        print(f"  Berechnete Breitenskala: {width_scale}, Höhenskala: {height_scale}")

        scale_factor = min(width_scale, height_scale)
        print(f"  Endgültiger Skalierungsfaktor: {scale_factor}")

        return scale_factor

    except Exception as e:
        print(f"Fehler beim Ermitteln der PDF-Abmessungen oder beim Berechnen des Skalierungsfaktors: {e}")
        return None


def scale_pdf(input_pdf, output_pdf, scale_factor):
    """Skaliert ein PDF mit pdfjam."""
    print(f"Skaliere {input_pdf} zu {output_pdf} mit Skalierungsfaktor: {scale_factor}")
    try:
        subprocess.run(
            [
                "pdfjam",
                "--scale",
                str(scale_factor),
                input_pdf,
                "--outfile",
                output_pdf,
            ],
            check=True,
        )
        print(f"Erfolgreich skaliert {input_pdf} zu {output_pdf}")
    except subprocess.CalledProcessError as e:
        print(f"Fehler beim Skalieren des PDFs: {e}")
    except FileNotFoundError:
        print("Fehler: pdfjam-Befehl nicht gefunden. Bitte stellen Sie sicher, dass es installiert ist und sic


if __name__ == "__main__":
    if len(sys.argv) != 3:
        print("Verwendung: python scale-pdf.py <input_pdf> <output_pdf>")
        sys.exit(1)

    input_pdf = sys.argv[1]
```

```
output_pdf = sys.argv[2]
print(f"Eingabe-PDF: {input_pdf}, Ausgabe-PDF: {output_pdf}")

if not os.path.exists(input_pdf):
    print(f"Fehler: Eingabe-PDF-Datei nicht gefunden: {input_pdf}")
    sys.exit(1)

scale_factor = calculate_scale_factor(input_pdf)
if scale_factor is None:
    sys.exit(1)

scale_pdf(input_pdf, output_pdf, scale_factor)
```