Continue

The script includes several classes and functions to accomplish this task. The `PDFExtractor` class is used to download and process PDF files. * The `download_file` function downloads a file from a specified URL. * The `break_pdf` method extracts individual pages from a PDF file and saves them as separate files. * There are two text extraction methods (`extract_text_algo_1` and `extract_text_algo_2`) that use different approaches to extract text from the PDF. The first method uses Python's built-in functionality, while the second method uses the PyPDF2 library with a custom device for conversion. * The `extract_text` function compares the extracted text using both methods and returns the longer one. * The `extarct_table` function reads the table content from the PDF file using the tabula library. Additionally, there is a method called `tiff_header_for_CCITT` that generates a TIFF header for a given image size. import PyPDF2 with open("sample.pdf", "rb") as pdf_file: read_pdf = PyPDF2.PdfFileReader(pdf_file) number_of_pages = read_pdf.getNumPages() page = read_pdf.pages[0] page_content = page.extractText() print(page_content) Working with LTTextBoxes from y-axis bottom up is feasible, allowing subtraction from the page's mediabox: x0, y0_orig, x1, y1_orig = some_lobj.bbox; y0 = page.mediabox[3] - y1_orig; y1 = page.mediabox[3] - y0_orig. Moreover, LTTextBoxes possess a .get_text() method, which retrieves their textual content as a string. This is in addition to the bbox property. Each LTTextBox comprises LTChars (characters explicitly drawn by the PDF) and LTAnnos (spaces added by PDFMiner based on character distance). By combining these properties, you can display the coordinates of each text block. It's worth noting that LTFigures, unlike other Stack Overflow answers, do not require recursion since they directly contain LTChar objects rather than grouping them into LTTextBoxes.