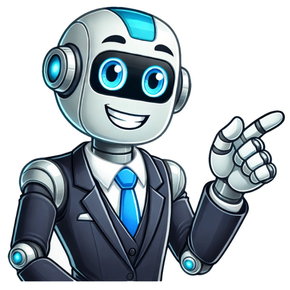
Click to prove you're human



Many hours have I searched for a fast and easy, but mostly accurate, way to get the number of pages in a PDF document. Since I work for a graphic printing and reproduction company that works a lot with PDFs, the number of pages in a document must be precisely known before they are processed. PDF documents come from many different clients, so they aren't generated with the same application and/or don't use the same compression method. Here are some of the answers I found insufficient or simply NOT working: Using Imagick (a PHP extension) Imagick requires a lot of installation, apache needs to restart, and when I finally had it working, it took amazingly long to process (2-3 minutes per document) and it always returned 1 page in every document (haven't seen a working copy of Imagick so far), so I threw it away. That was with both the getNumberImages() and identifyImage() methods. Using FPDI (a PHP library) FPDI is easy to use and install (just extract files and call a PHP script), BUT many of the compression techniques are not supported by FPDI. It then returns an error: FPDF error: This document (test_1.pdf) probably uses a compression technique which is not supported by the free parser shipped with FPDI. Opening a stream and search with a regular expression: This opens the PDF file in a stream and searches for some kind of string, containing the pagecount or something similar. $f = "test1.pdf"; $stream = fopen($f, "r"); $content = fread ($stream, filesize($f)); if(!$stream || !$content) return 0; $count = 0; // Regular Expressions found by Googling (all linked to SO answers): $regex = "/\/Count\s+(\d+)/"; $regex2 = "/\/Page\W*(\d+)/"; $regex3 = "/\/N\s+(\d+)/"; if(preg_match_all($regex, $content, $matches)) $count = max($matches); return $count; /\/Count\s+(\d+)/ (looks for /Count) doesn't work because only a few documents have the parameter /Count inside, so most of the time it doesn't return anything. Source. /\/Page\W*(\d+)/ (looks for /Page) doesn't get the number of pages, mostly contains some other data. Source. /\/N\s+(\d+)/ (looks for /N) doesn't work either, as the documents can contain multiple values of /N ; most, if not all, not containing the pagecount. Source. See the answer below April 25, 2017 at 10:49 PM by Dr. Drang I deal with lots of PDFs at work, and sometimes I want to report on the total number of pages I've reviewed on a given project. I've used various ad hoc methods to collect and sum up these page counts but have never come up with an automated technique. When there's deadline pressure to get a report out, there's no time—or it seems like there's no time—to put together a decent automated solution. Today I decided to make time. There are several ways to get the number of pages in a PDF. You can open it in Preview and see the page count in the title bar. You can do a Get Info on the file. If you have PDFtk installed, you can run it from the command line using the dump_data operation. That'll get you a crapload of info on the file (over 2,000 lines for the file I'm using as an example), but you can limit it to just the number of pages by filtering the output.1 $ pdftk File\ 3.pdf dump_data | grep NumberOfPages NumberOfPages: 420 I decided to use one of the command-line tools Apple provides to access the metadata used by Spotlight. These tools all begin with "md," and the one that can output the page count is mdls (metadata list). Like pdftk, mdls will, by default, spit out a lot of information on a file. $ mdls File\ 3.pdf _kMDItemOwnerUserID = 501 kMDItemContentCreationDate = 2017-04-26 01:29:28 +0000 kMDItemContentModificationDate = 2017-04-26 01:29:42 +0000 kMDItemContentType = "com.adobe.pdf" kMDItemContentTypeTree = ("com.adobe.pdf", "public.item", "com.adobe.pdf", "public.data", "public.composite-content", "public.content") kMDItemDateAdded = 2017-04-26 01:29:28 +0000 kMDItemDisplayName = "File 3.pdf" kMDItemEncodingApplications = ("Pixel Translations (PIXPDF 58.5.1.1422)") kMDItemFSContentChangeDate = 2017-04-26 01:29:42 +0000 kMDItemFSCreationDate = 2017-04-26 01:29:28 +0000 kMDItemFSCreatorCode = "" kMDItemFSFinderFlags = 0 kMDItemFSHasCustomIcon = (null) kMDItemFSInvisible = 0 kMDItemFSIsExtensionHidden = 0 kMDItemFSIsStationery = (null) kMDItemFSLabel = 0 kMDItemFSName = "File 3.pdf" kMDItemFSNodeCount = (null) kMDItemFSOwnerGroupID = 20 kMDItemFSOwnerUserID = 501 kMDItemFSSize = 41753844 kMDItemFSTypeCode = "" kMDItemKind = "Portable Document Format (PDF)" kMDItemLastUsedDate = 2017-04-26 01:42:08 +0000 kMDItemLogicalSize = 41753844 kMDItemNumberOfPages = 420 kMDItemPageHeight = 807.36 kMDItemPageWidth = 606 kMDItemPhysicalSize = 41754624 kMDItemSecurityMethod = "None" kMDItemUseCount = 1 kMDItemUsedDates = ("2017-04-25 05:00:00 +0000") kMDItemVersion = "1.4" The only line we want is the one that starts with kMDItemNumberOfPages. We can limit the output to just that line by using the -name option and get rid of the label with the -raw option: $ mdls -name kMDItemNumberOfPages -raw File\ 3.pdf 420 The thing about -raw is that it doesn't put a newline after the output, which is sometimes good and sometimes bad. We'll deal with that in a bit. Now we're ready to build a shell script, called pdfpages, that does the following: Prints a usage string if you don't give it any arguments. Prints just the number of pages in the file if you give it one argument. Prints the number of pages and the name for each file and the grand total of pages if you give it more than one argument. To demonstrate: $ pdfpages Usage: pdfpages $ pdfpages File\ 3.pdf 420 $ pdfpages *.pdf 330 File 1.pdf 532 File 2.pdf 420 File 3.pdf 1282 Total Here's the source code for pdfpages: bash: 1: #!/bin/bash 2: 3: if [$# == 0]; then 4: echo "Usage: pdfpages " 5: elif [$# == 1]; then 6: echo `mdls -name kMDItemNumberOfPages -raw "$1"` 7: else 8: sum=0 9: for f in "$@"; do 10: count=`mdls -name kMDItemNumberOfPages -raw "$f"` 11: echo -e "$count\t$f" 12: ((sum += count)) 13: done 14: echo -e "$sum\tTotal" 15: fi Recall that $# provides the number of arguments to the script, so Lines 3 and 5 test for the no argument and one argument conditions, respectively. Line 4 is the usage message printed if there are no arguments. Line 6 runs when there's one argument and is basically what we showed above. Putting the mdls command in backticks runs it and feeds the output to echo, which adds the trailing newline and prevents the next command prompt from appearing on the same line as the output.2 Lines 8–14 handle the case of more than one argument. We start by initializing the running sum in Line 8. Then we loop through all the arguments with the for on Line 9. For each file, Line 10 runs the mdls command and puts the output in the variable count. Line 11 prints the page count and file name for the current file, and Line 12 increments the running sum.3 When the loop is finished, Line 14 prints out the total. Throughout the script, please note the use of double quotes around the $@, $1, and $f variables. This keeps the script from shitting the bed when file paths include spaces. This is about as complicated a shell script as I would ever want to write. If I find myself wanting to add features, I'll probably rewrite it as a Python script using the subprocess module to run the mdls command. After writing pdfpages, I wondered how it would have worked on a older project in which I gave up trying to count all the PDF pages I was sent because there were just too many spread over too many files. Getting the number of PDF files (just over 1,000) in a nested folder structure was easy using standard tools: find . -iname *.pdf | wc -l Now I could get the number of pages in those files with find . -iname *.pdf -print0 | xargs -0 pdfpages It chugged away for about half a minute and told me I'd been given nearly 16,000 pages to review. Maybe it was better I didn't know. The PDF or Portable Document Format is one of the most popular file formats that is used for documents to share it through email or other ways. One of the main reasons why this format is preferred is that users can set restrictions by using it. A user who creates a PDF file can prevent others from editing it or copying content from it with a password. A user can even protect the file with a password so no one can open it without his permission. It is even possible to restrict printing so one can only view the file and not print it. A common problem is when you have a restricted file where printing is disabled. You may have to print the file as it has important content. If you have to print such a file and were wondering how to do it, we will tell you in this guide on how to print protected PDF files. Part 1: How to Enable Print Option in PDF without Knowing Password There are different ways by which you can enable the print option in a PDF file without the password. Let's look at how to enable print option in PDF using two different ways through utilities. 1.1 Using Online Tool to Enable Print Option in PDF It is possible to enable print in PDF by using a utility that is available online. One of these utilities is iLovePDF. This free online utility helps you carry out various operations on PDF files. You can convert PDF files to other formats, merge files, split them, and unlock PDF files. Let us see how to enable print option in PDF by unlocking the file using iLovePDF. The following steps explain how to print protected PDF using this utility: Step 1: The first thing to do is to visit the official website of iLovePDF. Step 2: Scroll down the home page and search for the option Unlock PDF. You can use this option to unlock the PDF file thus allowing you to enable printing. Step 3: You will see a button asking you to Select PDF Files. You can click the button to upload PDF files from your computer or you can directly drop your PDF files. You can upload PDF files from Google Drive or Dropbox. Step 4: After uploading, click the red button Unlock PDF to start the process.Select the file and click OK. Step 5: After finishing, the unlocked PDF file will be downloaded automatically or you can download it manually. As you can see, the method is simple. However, it has some disadvantages: This is an online utility. If you do not have Internet connection, you cannot use it. The file is saved on this utility's website. The site claims not to store files but there is an element of risk involved, particularly if you are trying to unlock confidential documents. There is always the possibility of hacking when your data is on a website. 1.2 Using Third-Party Software to Enable Print Option in PDF A unique and innovative solution to deal with problematic PDF files is Passper for PDF. This software allows you to easily and effectively unlock PDF files. You can remove all restrictions placed on the file, including print restrictions. Removing restrictions on PDF files is easy and super fast, it takes hardly 3 seconds to get it done. What we can expect from Passper for PDF: All restrictions on PDF files including edit, copy, print and comment can be removed by using Passper for PDF. It is an easy-to-use program. It only takes 3 steps to complete the removal process. It will only take about 3 seconds to remove restrictions on PDF files. Additionally, Passper for PDF can be used to recover password with 4 attack modes when you forgot it. The tool is available in trial version. You can free download it to test whether your PDF files are supportable or not. Free Download Buy Now We explain how to enable print option in PDF by using Passper for PDF. The following are the steps involved: Step 1 Once open the software, it gives you two options to: Recover passwords and Remove Restrictions. To enable print in PDF files please choose Remove Restrictions. Step 2 You will be asked to choose a PDF file that is protected and for which you want to remove the restrictions. Select the file from your system and click OK. The name of the file selected is displayed. If you made a mistake while selecting, you can use the back button to redo this step. Step 3 Click the Remove button to begin the process. The software is so fast that it can accomplish this job within just 1 to 2 seconds. If you know the password that used to restricted PDF files, then you can use Adobe Acrobat. But you should make sure that the Adobe Acrobat you use is pro version. Now, you can follow the steps below to remove the print restriction on your PDF file: Step 1: Open the restricted PDF file on Adobe Acrobat Pro version. Step 2: Click on the padlock located in the left panel. Click Permission Details to bring up the Document Properties window. Step 3: Click on the Change Settings button. It will require you to enter a permissions password. Step 4: Now, you can change the permission settings for your PDF file. The above three methods explained how to enable print option in PDF. Now that you have seen the three options, it is time to do a comparison to understand the differences between the three methods and which of these three methods are more advantageous. iLovePDF Passper for PDF Adobe Acrobat Ease to use simple simple medium Internet Needed Safe or Not not safe very safe safe Need Password to Remove Restrictions Crack Password From the above, we can see that Passper for PDF has multiple advantages. Apart from removing print restrictions, the software has an option to recover lost passwords. The multiple features make it a preferred tool to use. I am a student. I created one dissertation and saved it in PDF format with a password. However, I lost the password to open it. To get rid of the protection I tried many free solutions found online, but failed. Then I downloaded Passper for PDF to test it. I have to say that it is really easy to use. I paid for the Passper for PDF 1-year plan and it never disappointed me. If you are a student like me, I recommend it! By Jason M. Contreras Share — copy and redistribute the material in any medium or format for any purpose, even commercially. Adapt — remix, transform, and build upon the material for any purpose, even commercially. The licensor cannot revoke these freedoms as long as you follow the license terms. Attribution — You must give appropriate credit , provide a link to the license, and indicate if changes were made . You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use. ShareAlike — If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original. No additional restrictions — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits. You do not have to comply with the license for elements of the material in the public domain or where your use is permitted by an applicable exception or limitation . No warranties are given. The license may not give you all of the permissions necessary for your intended use. For example, other rights such as publicity, privacy, or moral rights may limit how you use the material. Share — copy and redistribute the material in any medium or format for any purpose, even commercially. Adapt — remix, transform, and build upon the material for any purpose, even commercially. The licensor cannot revoke these freedoms as long as you follow the license terms. Attribution — You must give appropriate credit , provide a link to the license, and indicate if changes were made . You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use. ShareAlike — If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original. No additional restrictions — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits. You do not have to comply with the license for elements of the material in the public domain or where your use is permitted by an applicable exception or limitation . No warranties are given. The license may not give you all of the permissions necessary for your intended use. For example, other rights such as publicity, privacy, or moral rights may limit how you use the material. Working with PDFs in Python is easy with the PdfReader class. One useful method is getNumPages. It helps you count the number of pages in a PDF file.This article explains how to use getNumPages. It also includes examples and tips for beginners. Let's dive in!What is PdfReader.getNumPages?The getNumPages method is part of the PdfReader class. It returns the total number of pages in a PDF file. This is useful for tasks like splitting or analyzing PDFs.To use it, you need the PyPDF2 library. If you don't have it, check out our guide on how to install Python PdfReader.How to Use PdfReader.getNumPagesFirst, install the PyPDF2 library. Then, import the PdfReader class. Open a PDF file and call getNumPages to count its pages.Here's an example: # Import PdfReader from PyPDF2 from PyPDF2 import PdfReader # Open the PDF file reader = PdfReader("example.pdf") # Get the number of pages num_pages = reader.getNumPages() # Print the result print(f"The PDF has {num_pages} pages.") The PDF has 10 pages. In this example, the code reads a PDF file named example.pdf. It then prints the total number of pages.Common Errors and FixesSometimes, you might encounter errors. For example, you could see "No module named PdfReader". This means the PyPDF2 library is not installed.To fix this, install the library using pip. For more details, read our guide on fixing the "No module named PdfReader" error.Practical Use CasesThe getNumPages method is helpful in many scenarios. For example, you can use it to split a PDF into smaller files. Or, you can check if a PDF meets a page limit.If you need to extract specific pages, check out our guide on using PdfReader.getPage.ConclusionThe PdfReader.getNumPages method is a simple yet powerful tool. It helps you count pages in a PDF file quickly. With this knowledge, you can handle PDFs more efficiently in Python.Remember to install the PyPDF2 library and handle errors properly. Happy coding! Many hours have I searched for a fast and easy, but mostly accurate, way to get the number of pages in a PDF document. Since I work for a graphic printing and reproduction company that works a lot with PDFs, the number of pages in a document must be precisely known before they are processed. PDF documents come from many different clients, so they aren't generated with the same application and/or don't use the same compression method. Here are some of the answers I found insufficient or simply NOT working: Using Imagick (a PHP extension) Imagick requires a lot of installation, apache needs to restart, and when I finally had it working, it took amazingly long to process (2-3 minutes per document) and it always returned 1 page in every document (haven't seen a working copy of Imagick so far), so I threw it away. That was with both the getNumberImages() and identifyImage() methods. Using FPDI (a PHP library) FPDI is easy to use and install (just extract files and call a PHP script), BUT many of the compression techniques are not supported by FPDI. It then returns an error: FPDF error: This document (test_1.pdf) probably uses a compression technique which is not supported by the free parser shipped with FPDI. Opening a stream and search with a regular expression: This opens the PDF file in a stream and searches for some kind of string, containing the pagecount or something similar. $f = "test1.pdf"; $stream = fopen($f, "r"); $content = fread ($stream, filesize($f)); if(!$stream || !$content) return 0; $count = 0; // Regular Expressions found by Googling (all linked to SO answers): $regex = "/\/Count\s+(\d+)/"; $regex2 = "/\/Page\W*(\d+)/"; $regex3 = "/\/N\s+(\d+)/"; if(preg_match_all($regex, $content, $matches)) $count = max($matches); return $count; /\/Count\s+(\d+)/ (looks for /Count) doesn't work because only a few documents have the parameter /Count inside, so most of the time it doesn't return anything. Source. /\/Page\W*(\d+)/ (looks for /Page) doesn't get the number of pages, mostly contains some other data. Source. /\/N\s+(\d+)/ (looks for /N) doesn't work either, as the documents can contain multiple values of /N ; most, if not all, not containing the pagecount. Source. See the answer below This post shows how to count Pages in multiple PDF files in Windows 11/10. While there are so many best free PDF readers available to use, counting the total number of pages present in different PDF documents is not possible with most such tools as either only one PDF can be opened at a time or there is no such option present. So, if you ever need to check how many pages are there in your PDF collection, then the PDF page counter options covered below will be helpful. Whether you have scanned PDF files or password-protected PDFs, the page count for all the PDF documents can be done with ease.How to count pages in multiple PDF files together in Windows 11/10To count pages in multiple PDF files together on a Windows 11/10 computer, use the following options:Use a web browserUse PDFMate Free PDF MergerUse PDF Count.Let's have a look at all these options.1] Use a web browserThis is a very simple option to use and doesn't require any extra tool to install on your PC. All the popular browsers (Edge, Chrome, Firefox, Opera, etc.) support PDF files and when you open a PDF file in a browser, the total number of pages is visible on the PDF toolbar. Depending upon the browser you use, the page count is visible in the middle section or left section of the toolbar.So, what you need to do is open PDFs in different tabs of your web browser and the page count for each PDF will be visible to you. You can also check the page count for a PDF using Document Properties. And then, you can make a total of all the pages and note it somewhere for later use.In a similar way, you can also use Adobe Acrobat Reader DC (free) to count PDF pages.2] PDFMate Free PDF MergerPDFMate Free PDF Merger tool, as the name indicates, helps to combine or merge PDF files to create a single PDF document. But this freeware can also be used to encrypt PDF, convert PNG, TIF, and JPG to PDF, extract pages from PDF, and count pages of multiple PDF files at once.You can get the setup file of PDFMate Free PDF Merger software from pdfmate.com. Launch the software after installation and then you can add a folder containing your PDF documents or use Add Files button to add the selected files.After adding the PDF files, you can check the Total Page column which will show the page count for each PDF in the list. The tool works well, but it fails to load a password-protected PDF file.Related: How to see the word count in Word and PowerPoint3] PDF CountPDF Count (or TTFA PDF Page Counter) is also a free software and one of the best options to count pages in multiple PDF files together. With this option, you don't have to open PDF files. Just add them to its interface using Add Files button and the total page count for all PDF documents will be visible on the top left part. Also, you can add as many PDF files as you want. If a PDF is password protected, then you can also provide the password to enter that PDF in the list.Apart from this, it also provides a page count for each individual PDF separately. You will see the document name and page count for that PDF document. The option to export the total page count as TXT and Excel files (XLSX or XLS) is also there. To use this tool, download it from softpedia.com.That's all!Also read: How to sign a PDF using free software or online toolsHow do I combine PDF files in Windows 11?There are multiple ways to combine PDF files into one PDF for free on a Windows 11/10 PC. You can use some free online PDF merger tools or free PDF merger software like PDFMate Free PDF Merger and PDF24 Creator. A free Microsoft Store app called PDF Merger & Splitter can also be used to create a single PDF file from multiple PDFs.Can you print multiple PDFs at once in Windows 11?Yes, you can print multiple PDFs at once in Windows 11. In fact, you can do the same on Windows 10 and other versions of Windows as well. All you need to do is to choose your desired PDF files and do the usual for printing. It prints one PDF and puts others in queue.Read next: How to extract Tables from PDF documents. Many hours have I searched for a fast and easy, but mostly accurate, way to get the number of pages in a PDF document. Since I work for a graphic printing and reproduction company that works a lot with PDFs, the number of pages in a document must be precisely known before they are processed. PDF documents come from many different clients, so they aren't generated with the same application and/or don't use the same compression method. Here are some of the answers I found insufficient or simply NOT working: Using Imagick (a PHP extension) Imagick requires a lot of installation, apache needs to restart, and when I finally had it working, it took amazingly long to process (2-3 minutes per document) and it always returned 1 page in every document (haven't seen a working copy of Imagick so far), so I threw it away. That was with both the getNumberImages() and identifyImage() methods. Using FPDI (a PHP library) FPDI is easy to use and install (just extract files and call a PHP script), BUT many of the compression techniques are not supported by FPDI. It then returns an error: FPDF error: This document (test_1.pdf) probably uses a compression technique which is not supported by the free parser shipped with FPDI. Opening a stream and search with a regular expression: This opens the PDF file in a stream and searches for some kind of string, containing the pagecount or something similar. $f = "test1.pdf"; $stream = fopen($f, "r"); $content = fread ($stream, filesize($f)); if(!$stream || !$content) return 0; $count = 0; // Regular Expressions found by Googling (all linked to SO answers): $regex = "/\/Count\s+(\d+)/"; $regex2 = "/\/Page\W*(\d+)/"; $regex3 = "/\/N\s+(\d+)/"; if(preg_match_all($regex, $content, $matches)) $count = max($matches); return $count; /\/Count\s+(\d+)/ (looks for /Count) doesn't work because only a few documents have the parameter /Count inside, so most of the time it doesn't return anything. Source. /\/Page\W*(\d+)/ (looks for /Page) doesn't get the number of pages, mostly contains some other data. Source. /\/N\s+(\d+)/ (looks for /N) doesn't work either, as the documents can contain multiple values of /N ; most, if not all, not containing the pagecount. Source. See the answer below Many hours have I searched for a fast and easy, but mostly accurate, way to get the number of pages in a PDF document. Since I work for a graphic printing and reproduction company that works a lot with PDFs, the number of pages in a document must be precisely known before they are processed. PDF documents come from many different clients, so they aren't generated with the same application and/or don't use the same compression method. Here are some of the answers I found insufficient or simply NOT working: Using Imagick (a PHP extension) Imagick requires a lot of installation, apache needs to restart, and when I finally had it working, it took amazingly long to process (2-3 minutes per document) and it always returned 1 page in every document (haven't seen a working copy of Imagick so far), so I threw it away. That was with both the getNumberImages() and identifyImage() methods. Using FPDI (a PHP library) FPDI is easy to use and install (just extract files and call a PHP script), BUT many of the compression techniques are not supported by FPDI. It then returns an error: FPDF error: This document (test_1.pdf) probably uses a compression technique which is not supported by the free parser shipped with FPDI. Opening a stream and search with a regular expression: This opens the PDF file in a stream and searches for some kind of string, containing the pagecount or something similar. $f = "test1.pdf"; $stream = fopen($f, "r"); $content = fread ($stream, filesize($f)); if(!$stream || !$content) return 0; $count = 0; // Regular Expressions found by Googling (all linked to SO answers): $regex = "/\/Count\s+(\d+)/"; $regex2 = "/\/Page\W*(\d+)/"; $regex3 = "/\/N\s+(\d+)/"; if(preg_match_all($regex, $content, $matches)) $count = max($matches); return $count; /\/Count\s+(\d+)/ (looks for /Count) doesn't work because only a few documents have the parameter /Count inside, so most of the time it doesn't return anything. Source. /\/Page\W*(\d+)/ (looks for /Page) doesn't get the number of pages, mostly contains some other data. Source. /\/N\s+(\d+)/ (looks for /N) doesn't work either, as the documents can contain multiple values of /N ; most, if not all, not containing the pagecount. Source. See the answer below Photo by CURVD® on UnsplashWhen working with PDFs, you might need to extract specific information about the pages, such as the page count, size, rotation, orientation, or other attributes. This article will walk you through the process of extracting page information from PDFs using Python. It covers the following topics:Python Module to Extract Page Information from PDFsWe will use the Spire.PDF for Python module to extract page information from PDF documents. This module provides a comprehensive set of functions that allow us to create, read, edit, and convert PDF documents within Python applications.We need to install Spire.PDF for Python from PyPI by running the following command in the terminal:pip install Spire.PdfOnce the module is installed, we can use it to easily extract various page information from PDF documents. Let's get started!Extract Page Count from PDF with PythonThe number of pages in a PDF is a fundamental piece of information, especially for document processing tasks such as pagination, splitting, or rearranging.Getting the number of pages in a PDF can be easily done through two steps:Open the PDF document through the PdfDocument class.Get the number of pages in the document through the PdfDocument.Pages.Count property.The following code example shows how to get the total number of pages of a PDF document using Python and Spire.PDF for Python:from spire.pdf.common import *from spire.pdf import *# Open the PDF documentpdf = PdfDocument("Sample.pdf")# Get the total number of pagespage_count = pdf.Pages.Count# Print the page countprint(f"The PDF has {page_count} pages.")pdf.Close()Extract Page Size from PDF with PythonPage size refers to the dimensions (width and height) of a PDF page, typically measured in points, where 1 point equals 1/72 inch. Extracting the page size is crucial for ensuring consistent formatting, printing, and display across various documents.To retrieve the width and height of a PDF page, you can utilize the PdfPageBase.Size.Width and PdfPageBase.Size.Height properties. Below is a simple example demonstrating how to accomplish this:from spire.pdf.common import *from spire.pdf import *# Open the PDF documentpdf = PdfDocument("Sample.pdf")# Get the first page by its index (zero-based)page = pdf.Pages[0]# Get the width and height of the first pagewidth = page.Size.Widthheight = page.Size.Height# Print the dimensions of the first pageprint(f print(f"The first page dimensions are {width} points x {height} points.")pdf.Close()Key Points:The extracted page dimensions are in points by default.You can use the PdfUnitConvertor class provided by Spire.PDF for Python to convert between points and other units of measurement, such as inches, pixels, centimeters, and millimeters. Here is how you can achieve the conversion:# Create a PdfUnitConvertor objectconverter = PdfUnitConvertor()# Convert points to inchesinch_value = converter.ConvertUnits(point_value, PdfGraphicsUnit.Point, PdfGraphicsUnit.Inch)# Convert points to pixelspixel_value = converter.ConvertUnits(point_value, PdfGraphicsUnit.Point, PdfGraphicsUnit.Pixel)# Convert points to centimeterscentimeter_value = converter.ConvertUnits(point_value, PdfGraphicsUnit.Point, PdfGraphicsUnit.Centimeter)# Convert points to millimetersmillimeter_value = converter.ConvertUnits(point_value, PdfGraphicsUnit.Point, PdfGraphicsUnit.Millimeter)Extract Page Rotation from PDF with PythonSome pages in a PDF may be rotated to a specific angle, such as 0°, 90°, 180°, or 270°. Knowing this information is crucial when working with documents that require specific orientation adjustments for proper viewing or printing.To retrieve the rotation angle of a PDF page, you can utilize the PdfPageBase.Rotation property. Here's a simple example demonstrating how you can extract the rotation angle of a PDF page using Python and Spire.PDF for Python:from spire.pdf.common import *from spire.pdf import *# Open the PDF documentpdf = PdfDocument("Sample.pdf")# Get the first page by its index (zero-based)page = pdf.Pages[0]# Get the rotation information of the first pagerotation_info = page.Rotation# Determine the rotation angleif rotation_info == PdfPageRotateAngle.RotateAngle0: rotation_angle = "0 degrees (Normal)"elif rotation_info == PdfPageRotateAngle.RotateAngle90: rotation_angle = "90 degrees (Rotated clockwise)"elif rotation_info == PdfPageRotateAngle.RotateAngle180: rotation_angle = "180 degrees (Upside down)"elif rotation_info == PdfPageRotateAngle.RotateAngle270: rotation_angle = "270 degrees (Rotated counterclockwise)"else: rotation_angle = "Unknown rotation"# Print the rotation angle of the first pageprint(f"The rotation angle of the first page is {rotation_angle}.")pdf.Close()Extract Page Orientation from PDF with PythonThe orientation of a PDF page can be either portrait (default) or landscape. While Spire.PDF for Python doesn't provide a direct method to detect page orientation, you can determine it by comparing the page's width and height. If the width is greater than the height, the page is in landscape mode; otherwise, it's in portrait mode.Here is a simple example demonstrating how to check the orientation of a PDF page using Spire.PDF for Python:from spire.pdf.common import *from spire.pdf import *# Open the PDF documentpdf = PdfDocument("Sample.pdf")# Get the first page by its index (zero-based)page = pdf.Pages[0]# Get the width and height of the first pagewidth = page.Size.Widthheight = page.Size.Height# Check if the page is in portrait or landscape modeif width > height: print("The first page is in landscape mode.")else: print("The first page is in portrait mode.")pdf.Close()Extract Page Labels from PDF with PythonPDF page labels are used to provide custom names or labels to pages, which may differ from the actual page numbers (e.g., roman numerals for the introduction, followed by Arabic numerals for the main content). Extracting these labels is useful when processing documents with non-standard numbering.To retrieve the label of a PDF page, you can utilize the PdfPageBase.PageLabel property. Here is a simple example demonstrating how to accomplish this:from spire.pdf.common import *from spire.pdf import *# Open the PDF documentpdf = PdfDocument("Sample.pdf")# Get the first page by its index (zero-based)page = pdf.Pages[0]# Get the label of the first pagelabel = page.PageLabel# Print the label of the first pageprint(f"The label of the first page is: {label}")pdf.Close()Extract Page Boxes Information from PDF with PythonA PDF page contains various boxes that define its boundaries for different purposes. Here's a brief overview:Media Box: Defines the overall dimensions of the page.Crop Box: Specifies the visible area of the page when viewed or printed.Bleed Box: Defines the region to which content should be clipped when printed, ensuring no unprinted margins.Trim Box: Represents the final dimensions of the page after trimming.Art Box: Specifies the extent of the page's meaningful content.The following code example describes how to extract the media box, crop box, bleed box, trim box, and art box information of a PDF page using Python and Spire.PDF for Python:from spire.pdf.common import *from spire.pdf import *# Open the PDF documentpdf = PdfDocument("Sample.pdf")# Get the first page by its index (zero-based)page = pdf.Pages[0]# Get the media, crop, bleed, trim, and art boxes of the first pagemedia_box = page.MediaBoxcrop_box = page.CropBoxbleed_box = page.BleedBoxtrim_box = page.TrimBoxart_box = page.ArtBox# Print the dimensions and coordinates of each boxprint(f"Media Box: Width = {media_box.Width}, Height = {media_box.Height}, X = {media_box.X}, Y = {media_box.Y}")print(f"Crop Box: Width = {crop_box.Width}, Height = {crop_box.Height}, X = {crop_box.X}, Y = {crop_box.Y}")print(f"Bleed Box: Width = {bleed_box.Width}, Height = {bleed_box.Height}, X = {bleed_box.X}, Y = {bleed_box.Y}")print(f"Trim Box: Width = {trim_box.Width}, Height = {trim_box.Height}, X = {trim_box.X}, Y = {trim_box.Y}")print(f"Art Box: Width = {art_box.Width}, Height = {art_box.Height}, X = {art_box.X}, Y = {art_box.Y}")pdf.Close()ConclusionThis blog demonstrated how to extract various page information from PDF documents using Python, including page count, page size, page orientation, page rotation angle, page label, and page boxes. We hope you find it helpful.Related Topics The PDF was developed by Adobe back in the early 90s and it has become increasingly popular since the advent of the Internet and Social Media. PDF files typically contain both text and images and it is these images that can often increase the file size, in some cases dramatically so. Many users, especially those at work, require files that are not prohibitive in size so that when they share or receive these files their mailboxes don't get blown. They also want to ensure they don't use up all their allocated storage on their device. That is why compressing files, specifically bigger files like PDF are so popular. You can use the Zamzar PDF compression tool to reduce the size of your PDF file without impacting the quality of your file thereby still allowing you to share or print these files. Many hours have I searched for a fast and easy, but mostly accurate, way to get the number of pages in a PDF document. Since I work for a graphic printing and reproduction company that works a lot with PDFs, the number of pages in a document must be precisely known before they are processed. PDF documents come from many different clients, so they aren't generated with the same application and/or don't use the same compression method. Here are some of the answers I found insufficient or simply NOT working: Using Imagick (a PHP extension) Imagick requires a lot of installation, apache needs to restart, and when I finally had it working, it took amazingly long to process (2-3 minutes per document) and it always returned 1 page in every document (haven't seen a working copy of Imagick so far), so I threw it away. That was with both the getNumberImages() and identifyImage() methods. Using FPDI (a PHP library) FPDI is easy to use and install (just extract files and call a PHP script), BUT many of the compression techniques are not supported by FPDI. It then returns an error: FPDF error: This document (test_1.pdf) probably uses a compression technique which is not supported by the free parser shipped with FPDI. Opening a stream and search with a regular expression: This opens the PDF file in a stream and searches for some kind of string, containing the pagecount or something similar. $f = "test1.pdf"; $stream = fopen($f, "r"); $content = fread ($stream, filesize($f)); if(!$stream || !$content) return 0; $count = 0; // Regular Expressions found by Googling (all linked to SO answers): $regex = "/\/Count\s+(\d+)/"; $regex2 = "/\/Page\W*(\d+)/"; $regex3 = "/\/N\s+(\d+)/"; if(preg_match_all($regex, $content, $matches)) $count = max($matches); return $count; /\/Count\s+(\d+)/ (looks for /Count) doesn't work because only a few documents have the parameter /Count inside, so most of the time it doesn't return anything. Source. /\/Page\W*(\d+)/ (looks for /Page) doesn't get the number of pages, mostly contains some other data. Source. /\/N\s+(\d+)/ (looks for /N) doesn't work either, as the documents can contain multiple values of /N ; most, if not all, not containing the pagecount. Source. See the answer below Convenience No software to download. Just select your file, pick a format to convert to and away you go. Experience We have been successfully converting files since 2006, with millions of happy customers. Support Got a file you can't convert? Just email us and we'll ask our dedicated engineers to take a look for you. Speed We aim to complete all our conversions in under 10 minutes. Formats We support 1200+ file formats. When dealing with a PDF file in our regular lifestyle, we may need to know how many pages there are. If the page number is out of bounds, you may receive an error while accessing any page or anything from the PDF. We can count the number of pages in a PDF file or Get number of pages in Pdf Python to avoid these types of problems, Count Number Of Pages In Pdf Python, Python Pdf Page Count, Python includes a variety of built-in functions. To count the pages of a PDF file, we can use the Python inbuilt library 'PyPDF2' Pypdf2 Get Number Of Pages, Pypdf2 Number Of Pages, Python Count Pdf Pages, Python Get Pdf Page Count, Python Get Number Of Pages In Pdf, Python Pdf Number Of Pages, Python Count Pages In Pdf, Count Pdf Pages Python, Pypdf2 Count Pages, Fitz Get Number Of Pages, Pymupdf Get Number Of Pages, Pdf Page Count Python, Pypdf2 Get Page Number, Python Read Pdf, Read Pdf Python, Fitz Page Count processes are mentioned below. Before we work with the module PyPDF2 we should first install it. pip install PyPDF2 Output: Collecting PyPDF2 Downloading PyPDF2-1.26.0.tar.gz (77 kB) |[redacted]| 77 kB 2.8 MB/s Building wheels for collected packages: PyPDF2 Building wheel for PyPDF2 (setup.py) ... done Created wheel for PyPDF2: filename=PyPDF2-1.26.0-py3-none-any.whl size=61102 sha256=818687400c96b3b bcd3f38aecf0870e60e7c5241f0c6aa10b8fc45ffb26de2f1 Stored in directory: /root/.cache/pip/wheels/80/1a/24/648467ade3a77ed20f35cfd2badd32134e96dd25c a811e64b3 Successfully built PyPDF2 Installing collected packages: PyPDF2 Successfully installed PyPDF2-1.26.0 Count of Number of Pages in a PDF File in Python Below are the ways to count the number of pages in the given pdf file: Using PyPDF2 module Using pymupdf module Method #1: Using PyPDF2 module Approach: Import PyPDF2 module using the import keyword Open the PDF file in read-binary mode(converts file into binary format) using the open() function and store it in a variable. Pass the above PDF file to the PdfFileReader() function to read the pdf file and store it in another variable. Apply numPages attribute on the above read pdf to count the total number of pages in the given PDF file and store it in another variable. Print the count of the total number of pages in the given PDF file. The Exit of the Program. Below is the implementation: # Import PyPDF2 module using the import keyword import PyPDF2 # Open the PDF file in read-binary mode(converts file into binary format) using the open() # function and store it in a variable. gvn_file = open('btechgeeks.pdf', 'rb') # Pass the above PDF file to the PdfFileReader() function to read the pdf file and # store it in another variable. pdf_read = PyPDF2.PdfFileReader(gvn_file) # Apply numPages attribute on the above read pdf to count the total number of pages in # the given PDF file and store it in another variable. pages_count = pdf_read.numPages # Print the count of total number of pages in the given PDF file print("The total number of pages in the given PDF file = ", pages_count) Output: The total number of pages in the given PDF file = 8 Method #2: Using pymupdf module First, install the pymupdf module before going to the code given below: pip install pymupdf Approach: Import fitz function using the import keyword Open the PDF file using the open() function and store it in a variable. Apply pageCount on the above pdf file to get the count of total number of pages in a given PDF file and print the result. The Exit of the Program. Below is the implementation: # Import fitz function using the import keyword import fitz # Open the PDF file using the open() function and store it in a variable. gvn_pdffile = fitz.open('btechgeeks.pdf') # Apply pageCount on the above pdf file to get the count of total number of # pages in a given PDF file and print the result. print("The total number of pages in the given PDF file: ") gvn_pdffile.pageCount Output: The total number of pages in the given PDF file: 8 Recommended Reading On: How to Find the Page Number of a Text from a PDF File in Python? Trusted by over 10,000+ studentsWe are your complete resource for starting, growing, and monetizing an online business - from start to finish.Start a BlogI have over a decade of experience building and managing high-traffic authority websites that rank in the search engines and convert like crazy.I've developed a simple, proven process to scale the traffic, stickiness, and conversions of a website. Now I'm looking to teach those same methods to you.Digital marketing can be overwhelming, which is why we've done our best to simplify things down do their base levels.We've designed all of our resources with simplicity in mind. This is our guiding light, as we fully understand that learning digital marketing can be overwhelming at times. We've created our lessons to be simple to read, understand, and implement.The experts behind our lessons aren't just teaching - they're experienced professionals who have done this before. Rest assured you'll be learning from the best in the business.We don't just provide simple information - we back it up with helpful assets and offer actionable recommendations that you can take. Save yourself hours of searching with our reviews, curated collections, and insightful recommendations.Our team has spent years collecting real-world resources and examples that you can reference while building your online business. You'll see our suggestions implemented in the real world, from people we've never even met. Many hours have I searched for a fast and easy, but mostly accurate, way to get the number of pages in a PDF document. Since I work for a graphic printing and reproduction company that works a lot with PDFs, the number of pages in a document must be precisely known before they are processed. PDF documents come from many different clients, so they aren't generated with the same application and/or don't use the same compression method. Here are some of the answers I found insufficient or simply NOT working: Using Imagick (a PHP extension) Imagick requires a lot of installation, apache needs to restart, and when I finally had it working, it took amazingly long to process (2-3 minutes per document) and it always returned 1 page in every document (haven't seen a working copy of Imagick so far), so I threw it away. That was with both the getNumberImages() and identifyImage() methods. Using FPDI (a PHP library) FPDI is easy to use and install (just extract files and call a PHP script), BUT many of the compression techniques are not supported by FPDI. It then returns an error: FPDF error: This document (test_1.pdf) probably uses a compression technique which is not supported by the free parser shipped with FPDI. Opening a stream and search with a regular expression: This opens the PDF file in a stream and searches for some kind of string, containing the pagecount or something similar. $f = "test1.pdf"; $stream = fopen($f, "r"); $content = fread ($stream, filesize($f)); if(!$stream || !$content) return 0; $count = 0; // Regular Expressions found by Googling (all linked to SO answers): $regex = "/\/Count\s+(\d+)/"; $regex2 = "/\/Page\W*(\d+)/"; $regex3 = "/\/N\s+(\d+)/"; if(preg_match_all($regex, $content, $matches)) $count = max($matches); return $count; /\/Count\s+(\d+)/ (looks for /Count) doesn't work because only a few documents have the parameter /Count inside, so most of the time it doesn't return anything. Source. /\/Page\W*(\d+)/ (looks for /Page) doesn't get the number of pages, mostly contains some other data. Source. /\/N\s+(\d+)/ (looks for /N) doesn't work either, as the documents can contain multiple values of /N ; most, if not all, not containing the pagecount. Source. See the answer below In this article, we will see how can we count the total number of pages in a PDF file in Python, For this article there is no such prerequisite, we will use PyPDF2 library for this purpose. PyPDF2 is a free and open-source pure-Python PyPDF library capable of performing many tasks like splitting, merging, cropping, and transforming the pages of PDF files. It can also add custom data, viewing options, and passwords to PDF files. PyPDF2 can retrieve text and metadata from PDFs as well. Refer to this "Working with PDF files in Python" to explore about PyPDF2 Installing required libraryExecute the below command to install the PyPDF2 library in the command prompt or terminal. pip install PyPDF2Step to Count the number of pages in a PDF fileStep 1: Import PyPDF2 library into the Python program import PyPDF2Step 2: Open the PDF file in read binary format using file handling file = open('your pdf file path', 'rb')Step 3: Read the pdf using the PdfReader() function of the PyPDF2 library pdfReader = PyPDF2.PdfReader(file)Note: These above three steps are similar for all methods that we are going to see using an example. Methods to count PDF pagesWe are going to learn three methods to count the number of pages in a PDF file which are as follows: By using the len(pdfReader.pages) property.By using the getNumPages() method.By using the pages property and len() function.Method 1: Using len(pdfReader.pages) propertylen(pdfReader.pages) is a property of PdfReader Class that returns the total number of pages in the PDF file. totalPages1 = len(pdfReader.pages)For Example: Python3 # importing PyPDF2 library import PyPDF2 # opened file as reading (r) in binary (b) mode file = open('/home/hardik/GFG_Temp/dbmsFile.pdf', 'rb') # store data in pdfReader pdfReader = PyPDF2.PdfReader(file) # count number of pages totalPages = len(pdfReader.pages) # print number of pages print(f"Total Pages: {totalPages}") Output: Total Pages: 10In the above example, we imported the PyPDF2 module and opened the file using file handling in read binary format after that with the help of PdfReader() function of PyPDF2 module we read the pdf file which we opened previously, then with the help of the numPages property of the module we counted the total pages of PDF file and stored the total number of pages in a variable "totalPages" for further usage and at last, we print the variable holding the total page count of PDF file. Method 2: Using getNumPages() methodgetNumPages() is a method of PdfReader class that returns an integer specifying a total number of pages and it takes no argument this method is deprecated since version 1.28.0 but we can still use another method that comes in its replacement is next method discussed. totalPages2 = pdfReader.getNumPages() Python3 # importing PyPDF2 library import PyPDF2 # opened file as reading (r) in binary (b) mode file = open('/home/hardik/GFG_Temp/dbmsFile.pdf', 'rb') # store data in pdfReader pdfReader = PyPDF2.PdfReader(file) # count number of pages totalPages = pdfReader.getNumPages() # print number of pages print(f"Total Pages: {totalPages}") Output: Total Pages: 10In the above example, we imported the PyPDF2 module and opened the file using file handling in reading binary format after that with the help of the PdfReader() function of PyPDF2 module we read the pdf file that we opened previously, then with the help of getNumPages() method of the module we counted the total pages of PDF file and stored the total number of pages in a variable "totalpages" for further usage and at last, we print the variable holding the total page count of PDF file. Method 3: Using pages property and len() functionpages is a read-only property that emulates a list of Page objects and using len() function which is Python's inbuilt function to count the length of a sequence is used combinedly to determine the total pages of the PDF. totalPages3 = len(pdfReader.pages) Python3 # importing PyPDF2 library import PyPDF2 # opened file as reading (r) in binary (b) mode file = open('/home/hardik/GFG_Temp/dbmsFile.pdf', 'rb') # store data in pdfReader pdfReader = PyPDF2.PdfReader(file) # count number of pages totalPages = len(pdfReader.pages) # print number of pages print(f"Total Pages: {totalPages}") Output: Total Pages: 10In the above example we imported the PyPDF2 module and opened the file using file handling in read binary format then with the help of PdfReader() function of PyPDF2 module we read the pdf file which we opened previously, then with the help of the pages property of the module we get the list of all the pages of PDF file and with the help of len() function we counted the total pages returned by pages property and stored the total number of pages in a variable "totalpages" for further usage and at last, we print the variable holding the total page count of PDF file. Many hours have I searched for a fast and easy, but mostly accurate, way to get the number of pages in a PDF document. Since I work for a graphic printing and reproduction company that works a lot with PDFs, the number of pages in a document must be precisely known before they are processed. PDF documents come from many different clients, so they aren't generated with the same application and/or don't use the same compression method. Here are some of the answers I found insufficient or simply NOT working: Using Imagick (a PHP extension) Imagick requires a lot of installation, apache needs to restart, and when I finally had it working, it took amazingly long to process (2-3 minutes per document) and it always returned 1 page in every document (haven't seen a working copy of Imagick so far), so I threw it away. That was with both the getNumberImages() and identifyImage() methods. Using FPDI (a PHP library) FPDI is easy to use and install (just extract files and call a PHP script), BUT many of the compression techniques are not supported by FPDI. It then returns an error: FPDF error: This document (test_1.pdf) probably uses a compression technique which is not supported by the free parser shipped with FPDI. Opening a stream and search with a regular expression: This opens the PDF file in a stream and searches for some kind of string, containing the pagecount or something similar. $f = "test1.pdf"; $stream = fopen($f, "r"); $content = fread ($stream, filesize($f)); if(!$stream || !$content) return 0; $count = 0; // Regular Expressions found by Googling (all linked to SO answers): $regex = "/\/Count\s+(\d+)/"; $regex2 = "/\/Page\W*(\d+)/"; $regex3 = "/\/N\s+(\d+)/"; if(preg_match_all($regex, $content, $matches)) $count = max($matches); return $count; /\/Count\s+(\d+)/ (looks for /Count) doesn't work because only a few documents have the parameter /Count inside, so most of the time it doesn't return anything. Source. /\/Page\W*(\d+)/ (looks for /Page) doesn't get the number of pages, mostly contains some other data. Source. /\/N\s+(\d+)/ (looks for /N) doesn't work either, as the documents can contain multiple values of /N ; most, if not all, not containing the pagecount. Source. See the answer below