WORKFLOW AUTOMATION

PYPDF2 Library: How Can You Work With PDF Files in Python?

by **Dhanashree** 20 MIN READ

Published: Aug 16, 2022 • Updated: Mar 31, 2024

Table of Contents

Looking to extract data from documents? Try Nanonets OCR software. Extract data from receipts, bills, PDF documents, and more on autopilot. No Credit Card. Forever Free plans.

Start your free trial

The most popular file type is Portable Document Format, also known as PDF. It can be an ebook, digitally signed agreements, password-protected documents, or scanned documents like passports.

PDF is the most extensively used digital format, and the International Standards Organization (ISO) maintains it as an open standard.

> ***PDF is the most widely used document format, with over 73 million new PDF files saved every day on Gmail & Drive.***

This shows the enormous amount of data stored within these file types, which are generally difficult to edit or modify. Here in this blog, we will see how you can use the Python library, PyPDF2 to work with PDF files and perform the following tasks:

- Extract text from PDF file using PyPDF2
- Encrypt a PDF file using PyPDF2
- Rotate, merge and split PDF files
- Adding a watermark to a PDF file

So, let's read on.

PyPDF2 isn't the only Python library you can use for PDF OCR using python. Here are some common Python PDF libraries:

- **PDFQuery:** PDFQuery is a PDF scraping library, and it is a fast and user-friendly python wrapper for PyQuery, PDFMiner, and XML.
- **Tabula.py:** It is a Python wrapper around tabula-java used to read tables in PDF. Tabula.py enables you to read tables and can be converted into Pandas DataFrame.
- **Slate:** It is used to extract text from PDF files, depending on the PDFMiner package. Slate is a lightweight annotation tool that supports annotation in Python.
- **PDFMiner**: It is an open-source PDF library used to extract text from PDF. You can use PDFMiner to perform analysis on data. However, it only supports Python3.
- **pdflib:** PDFlib is a library for creating PDFs in python. This development library contains several levels for creating, personalizing, and importing PDFs.
- **Xpdf:** It is a Python wrapper for pdf.
- **PyPDF2:** It is one of the best-known python libraries that enable you to perform tasks on PDFs, including merging PDF files, extracting document information, splitting PDF pages, and much more.

Here in this article, we will discuss the PyPDF2 library, known as one of the best libraries to manipulate PDF in Python and is available on every platform.

Dealing with PDFs often involves repetitive tasks, like extracting text or managing documents. Imagine streamlining these tedious processes with the power of workflow automation. Nanonets introduces an innovative platform at **Nanonets' Workflow Automation** that enables you to swiftly automate manual tasks, integrate seamlessly with numerous apps, and even harness AI to revolutionize how you handle PDF workflows. Say goodbye to the manual grind and hello to efficiency; let Nanonets transform your document management into a smooth, automated experience.

Learn More

What is the Best PDF Library for Python?

The best library for working with PDFs in Python is PyPDF2. It’s lightweight, fast, and well-documented. The library is available on the Python Package Index (PyPI).

If you need to create a PDF file from scratch, you’ll want to use PyPDF2 because it has robust support for creating new documents. If you need to parse an existing record, then PyPDF2 is perfect because it has better support for detecting different types of fonts and other features.

Introduction to PyPDF2 Library

PyPDF2 is a Python library that allows the manipulation of PDF documents. It can be used to create new PDF documents, modify existing ones and extract content from documents. PyPDF2 is a pure Python library that requires no non-standard modules.

The low-level API (based on Pygments) allows writing programs that generate or efficiently manipulate documents. The high-level API (based on ReportLab) enables the creation of complex documents such as forms, books, or magazines with just a few lines of code.

PyPDF2 supports:

- Converting PDF files into images (png or jpeg) or text files;
- Converting PDF to text, image to text
- Creating new PDF documents from scratch;
- Editing existing PDFs by adding, removing, replacing, or modifying pages;
- Modifying existing PDFs by rotating pages, adding watermarks, changing fonts, etc.;
- Signing documents with digital signatures (certificates must be present);

PyPDF2 has been designed with performance in mind. It uses native C code to handle the most time-consuming tasks (such as parsing) but never sacrifices the simplicity of its interface. The library is also thread-safe, and its memory footprint is not much larger than the one required by Python (around 1MB).

How do I read a PDF in PyPDF2?

Although PyPDF2 doesn't have a method specifically for reading remote files, you can use **Python's urllib.request module** to read the remote file in bytes before passing it to the **PdfFileReader()** function with the file in the format of the byte. The remaining steps resemble reading a local PDF file.

What is the difference between PyPDF2 and PyPDF4?

PyPDF2 is a library used to create, manipulate and decode portable documents. It supports PDF 1.4, 1.5, and 1.6, as well as all the security features in PDF 1.7, including digital signatures and permissions.

PyPDF2 has no dependencies other than the Python standard library. It is pure Python code, but it does use C extensions for some algorithms to improve performance.

PyPDF4 is an advanced tool for working with PDF documents on the macOS, Windows, and Linux platforms. It includes:

- A GUI for creating PDF files from scratch with text, images, and shapes; A browser for viewing PDF files
- A command-line tool for performing all other operations on existing files or creating new ones from scratch using Python scripts or shell commands such as sed(1) or awk(1).

PyPDF4 is based on the PyPDF2 library and shares its license but has additional features like:

- Support for page label annotations
- Support for encrypted documents (including their decryption at runtime)
- Support for document encryption/decryption
- Support for bookmarks with links between pages
- Password support protected documents (including their decryption at runtime).

Want to scrape data from PDF ? Check out Nanonets' to extract data, scan QR codes, and read barcodes from documents easily.

Try Nanonets for Free

What is the Use of PyPDF2?

The PyPDF2 library can be used in many different ways:

- PyPDF2 is used as a command line tool to create or modify PDFs. This is useful if you have shell access to a server but not a graphical desktop environment.
- It runs the library inside Python scripts by importing it as a module and calling its functions. This is useful if you want to automate tasks directly or build complex applications using PyPDF2.
- It can read, parse and write PDFs. It can be used as a command line tool or as a library.
- It's written in pure Python with no external dependencies (except for Python itself).
- It supports Unicode strings so that it can handle non-English characters.

PyPDF2 Use Cases

Here are some of the use cases where PyPDF2 can be used for:

Converting PDF to Word or Other Formats

For instance, if you want to convert a PDF to Word or another format, you'll have to download a separate program for each conversion. And if you're trying to do this on multiple documents at a time, the process can be slow and cumbersome.

PyPDF2 is a command-line tool that offers an alternative way of working with PDFs. You can use it as part of your regular workflow or as a Python program.

Merging Multiple PDFs Together

Users may use PyPDF2 to modify the contents of a PDF document. For example, add or remove pages from a PDF or extract text. It's also possible to add images and other objects to existing PDFs.

Modifying the Contents of a PDF Document

Merging multiple PDFs together can be done using PyPDF2 as well. This can be done by specifying the input paths for each PDF file and then combining them together into one document.

Splitting a Large Document into Smaller Ones

If you need to split a large document into smaller ones, PyPDF2 is the library you should use. The library supports splitting documents by page, every n pages, and by range of pages.

It also supports splitting documents based on their metadata, which is useful if you want to split documents according to their author or title.

PyPDF2 Installation

There are various methods for installing PyPDF2. The most popular choice is to employ pip.

Python 3.6 and up is needed to run PyPDF2.

A package installer called pip is typically included with Python. It allows you to set up PyPDF2:

```
pip install PyPDF2
```

You can just install PyPDF2 for your current user if you're not a superuser (a system administrator or root):

```
pip install --user PyPDF2
```

You'll need to install a few additional requirements if you want to use PyPDF2 to encrypt or decrypt AES PDFs. RC4 encryption is supported by using the standard installation.

```
pip install PyPDF2[crypto]
```

Extracting Document Details with PyPDF2

PyPDF2 is a Python library for working with PDF documents. It can be used to parse PDFs, modify them, and create new PDFs. PyPDF2 can be used to extract some text and metadata from a PDF. This can be helpful if you're automating some processes on your existing PDF files.

The current categories of data that can be extracted are as follows:

- Author
- Creator
- Producer
- Subject
- Title
- amount of pages

To utilize this example, you must locate a PDF. Any PDF that is readily available on your computer may be used.

Here's a code example for this:

```
# get_doc_info.py
from PyPDF2 import PdfFileReader
def get_info(path):
with open(path, 'rb') as f:
pdf = PdfFileReader(f)
info = pdf.getDocumentInfo()
number_of_pages = pdf.getNumPages()
print(info)
author = info.author
creator = info.creator
producer = info.producer
```

```
subject = info.subject
title = info.title
if __name__ == '__main__':
path = 'reportlab-sample.pdf'
get_info(path)
```

PyPDF.pdf.DocumentInformation will return comprising useful attributes, including author, creator, producer, subject, and title.

By printing **the DocumentInformation** object, you will get the required output like this:

The PyPDF2 package's PdfFileReader is imported here. A class called PdfFileReader offers several ways to deal with PDF files.

In this instance, you **call.getDocumentInfo()**, which will provide you a DocumentInformation object. Most of this information is what you're most interested in.

Additionally, you may obtain the document's page count by calling the reader **object.getNumPages()** method.

You can use the information variable's various instance attributes to extract the remaining document metadata that you require. You take a printout of the information and give it back for future use.

The**.extractText()** function in PyPDF2 can be used on its page objects (not shown in this example), although it is not very effective. Some PDFs will yield text, while others will return an empty string.

Check out the Nanonets instead if you want to extract text from a PDF. Since it was created expressly for extracting text from PDFs, Nanonets is significantly more capable.

Extracting Text from PDF with PyPDF2

Extracting text from PDF using PyPDF2 is hard as it has limited support for text extraction. The return of the code will not be in a proper format. You may get a series of line break characters due to PyPDF2's limited support.

Let’s see how you can extract text from a PDF:

```
# extracting_text.py
from PyPDF2 import PdfFileReader
def text_extractor(path):
with open(path, 'rb') as f:
pdf = PdfFileReader(f)
# get the first page
page = pdf.getPage(1)
print(page)
print('Page type: {}'.format(str(type(page))))
text = page.extractText()
print(text)
if __name__ == '__main__':
path = 'reportlab-sample.pdf'
text_extractor(path)
```

Output:

Additional Uses for the PyPDF2 Module

Many operations can be carried out on PDF files using the PyPDF2 module, including:

- Reading the PDF file's text, as we have just done above
- Changing the angle at which a PDF file page is rotated
- by combining two or more PDF files on a specific page.
- Attaching multiple PDF files sequentially.
- To obtain information such as the creator, author, creation date, etc., locate all the metadata for every PDF file.
- Even better, we can use the text from a text file to produce a new PDF document.

Automate PDF processing like document upload, extracting text, splitting PDF, rearranging PDF pages, edit PDF metadata, and more with workflow automation.

Try Nanonets for Free

Other PyPDF2 Tutorials

How to Rotate Pages of a PDF File?

The Python module PyPDF2 is a library used to manipulate PDF files. It's straightforward to use and is available for many different platforms.

Here we'll see how we can rotate the pages of a pdf file. Save the PDF in another file and run the following code:

```
import PyPDF2
pdf_in = open('original.pdf', 'rb')
pdf_reader = PyPDF2.PdfFileReader(pdf_in)
pdf_writer = PyPDF2.PdfFileWriter()
for pagenum in range(pdf_reader.numPages):
page = pdf_reader.getPage(pagenum)
page.rotateClockwise(180)
pdf_writer.addPage(page)
pdf_out = open('rotated.pdf', 'wb')
pdf_writer.write(pdf_out)
pdf_out.close()
pdf_in.close()
```

Instead of using codes, use Nanonets to rotate PDFs with no-code workflow automation. Over 10,000+ users use Nanonets to automate PDF processing.

Try Nanonets for free, or set up a call with us.

How to Merge PDF Files?

After scanning multiple pages of a document or storing numerous pages as separate documents on your computer, merging PDF files is frequently necessary.

Numerous programs, including Adobe and online applications, can help do this task swiftly. However, most of them are either for sale or may not offer enough security measures.

Open your preferred editor, then make a new file called "pdfMerger.py." Make sure the Python program is located in the same directory as the PDF files that will be attached.

You can combine two or more PDF files by using the following block of code:

```
from PyPDF2 import PdfFileMerger, PdfFileReader
merger = PdfFileMerger()
merger.append(PdfFileReader(open(filename1, 'rb')))
merger.append(PdfFileReader(open(filename2, 'rb')))
merger.write("merged.pdf")
```

The code above appears pretty straightforward, but what if you want to combine more than two files? For each file, you want to add, line 3 would need to be repeated, which would make your application rather long. In this circumstance, a for loop can be used.

Another method to combine multiple PDF files is shown in the following code.

```
import PyPDF2
def merge_pdfs(_pdfs):
    mergeFile = PyPDF2.PdfFileMerger()
    for _pdf in _pdfs:
        mergeFile.append(PyPDF2.PdfFileReader(_pdf, 'rb'))
    mergeFile.write("New_Merged_File.pdf")
if __name__ == '__main__':
    _pdfs = ['file1.pdf', 'file2.pdf', 'file3.pdf']
    merge_pdfs(_pdfs)
```

An additional way of merging PDF files is using no-code workflows on Nanonets. You can merge PDF files and create custom OCR models on Nanonets. Set up a 10 minute call with our team to learn more!

How to Split Pages from a PDF File?

For various reasons, you may often want to extract a specific page from a large PDF file or combine several PDF files into one. This can be accomplished with certain PDF editor software. Still, you may find that the split and merge features are typically not included in the free version or that processing so many pages or files makes them too laborious. In this article, I'll share a straightforward Python script that you can use to split or combine several PDF files.

Using PdfFileReader to read the original file will allow you to access a specific page by its page number when you wish to extract a particular page from the PDF file and create it as a separate PDF file (page number starts from 0). The add page function of the PdfFileWriter allows you to add a PDF page to a brand-new PDF object and save it.

Here is an example of code that separates the file1.pdf's first page into a separate PDF file called first page.pdf.

```
from PyPDF2 import PdfFileWriter, PdfFileReader
input_pdf = PdfFileReader("file1.pdf")
output = PdfFileWriter()
output.addPage(input_pdf.getPage(0))
with open("first_page.pdf", "wb") as output_stream:
output.write(output_stream)
```

How to Merge Pages of a PDF File?

You can use PdfFileMerger to combine multiple PDF files into a single document. Even though you may also use PdfFileWriter to accomplish this, merging pages without editing them first makes using PdfFileMerger more straightforward.

The sample code that uses the PdfFileMerger's append method to add multiple PDF files and write them into a single file called merged is shown below.

```
from PyPDF2 import PdfFileReader, PdfFileMerger
pdf_file1 = PdfFileReader("file1.pdf")
pdf_file2 = PdfFileReader("file2.pdf")
output = PdfFileMerger()
output.append(pdf_file1)
output.append(pdf_file2)
with open("merged.pdf", "wb") as output_stream:
output.write(output_stream)
```

If you want to add certain pages from your original file to the new PDF file, you can use the pages argument of the append function to give a tuple containing the beginning and ending page numbers.

If you wish to specify where your pages go, you must use the merge function because the append function will continually add new pages at the end. It enables you to select the page's location on which you wish to insert new pages.

If you have a lot of files to process, you can automate splitting, merging and rotating PDF pages with a simple no-code workflow process on Nanonets.

Try Nanonets for free, or set up a call with us.

Encrypting the PDF File

A PDF file can be encrypted using a password or a digital certificate. The encryption method is chosen by the user when the file is created. A password-protected PDF file can be opened, edited, and printed by anyone who knows the password. It cannot be opened or edited by someone who does not know the password. A digitally signed document is also protected from unauthorized editing. Still, it also includes an electronic signature that can be verified by anyone who has access to the original document or its digital signature.

```
for page in range(pdf.getNumPages()):
pdfwrite.addPage(pdf.getPage(page))
pdfwrite.encrypt(user_pwd=password, owner_pwd=None,
use_128bit=True)
with open(outputpdf, 'wb') as fh:
pdfwrite.write(fh)
```

You can password protect a PDF file using the above code just like this:

How to Add a Watermark to a PDF File?

A watermark is a text or graphic overlay on your document's front. It can help you protect your work from unauthorized use or misuse and show which records have been modified or printed. You can add text and graphics to make custom watermarks for your documents.

Here’s a code snippet about how to add a watermark to a PDF File:

```
import PyPDF2
pdf_file = "doc.pdf"
watermark = "watermark.pdf"
merged_file = "merged.pdf"
input_file = open(pdf_file,'rb')
input_pdf = PyPDF2.PdfFileReader(input_file)
watermark_file = open(watermark,'rb')
watermark_pdf = PyPDF2.PdfFileReader(watermark_file)
pdf_page = input_pdf.getPage(0)
watermark_page = watermark_pdf.getPage(0)
pdf_page.mergePage(watermark_page)
output = PyPDF2.PdfFileWriter()
output.addPage(pdf_page)
merged_file = open(merged_file,'wb')
output.write(merged_file)
merged_file.close()
watermark_file.close()
input_file.close()
```

Output:

Here is how the first page of original (left) and watermarked (right) PDF file looks like:

Three arguments must be carefully considered while using the encrypt function.

- User password user pwd is used to limit file opening and reading;
- User password is one step below the owner pwd, str. The file can be opened without any limitations when it is given. Default owner pwd and user pwd are the same if not supplied;
- Use the 128bit Boolean option to specify whether or not to utilize 128 bits for a password. False indicates a 40-bit password should be used; True is the default;

Working with PDF files using Nanonets

Nanonets extract text from invoice PDF

Nanonets has an OCR API that can be used to extract text from PDF documents, including invoices, receipts, customer orders, claim forms, and more. It can also identify handwritten documents and characters from 200+ languages. Furthermore, you can automate all aspects of data extraction by using automated workflows. Nanonets GUI allows you to extract data from unstructured PDFs on the go with pre-trained OCR templates. You can also create your custom model in 15 minutes.

Nanonets is an online OCR software; therefore, you can use all the features from your browser without downloading anything.

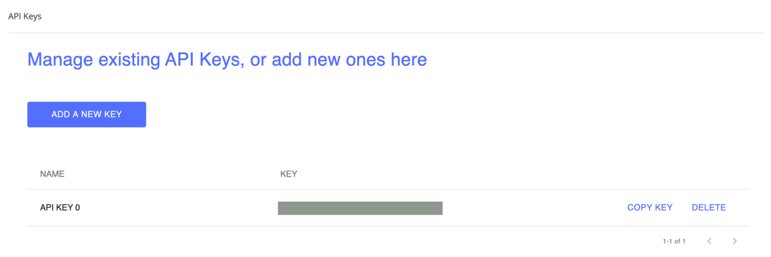
You can start using Nanonets by using the GUI interface: **https://app.nanonets.com/**

Or, you can access Nanonets OCR API with the following steps.

Step 1: Install dependencies using GitHub library

```
git clone https://github.com/NanoNets/nanonets-ocr-sample-python.git
cd nanonets-ocr-sample-python
sudo pip install requests tqdm
```

Step 2: Get your free Nanonets API Key
Get your free API Key from https://app.nanonets.com/#/keys



Step 3: Set the API key as an Environment Variable

```
export NANONETS_API_KEY=YOUR_API_KEY_GOES_HERE
```

Step 4: Create a New Model on the interface

```
python ./code/create-model.py
```

Note: This generates a MODEL_ID that you need for the next step

Step 5: Add Model Id as Environment Variable

```
export NANONETS_MODEL_ID=YOUR_MODEL_ID
```

Note: you will get YOUR_MODEL_ID from the previous step

Step 6: Upload the Training Data
The training data is found in `images` (image files) and `annotations` (annotations for the image files)

```
python ./code/upload-training.py
```

Step 7: Train the Model
Once the Images have been uploaded, begin training the Model

```
python ./code/train-model.py
```

Step 8: Get Model State
The model takes ~2 hours to train. You'll be notified via email once the model is trained. You can check model state with the following

```
python ./code/model-state.py
```

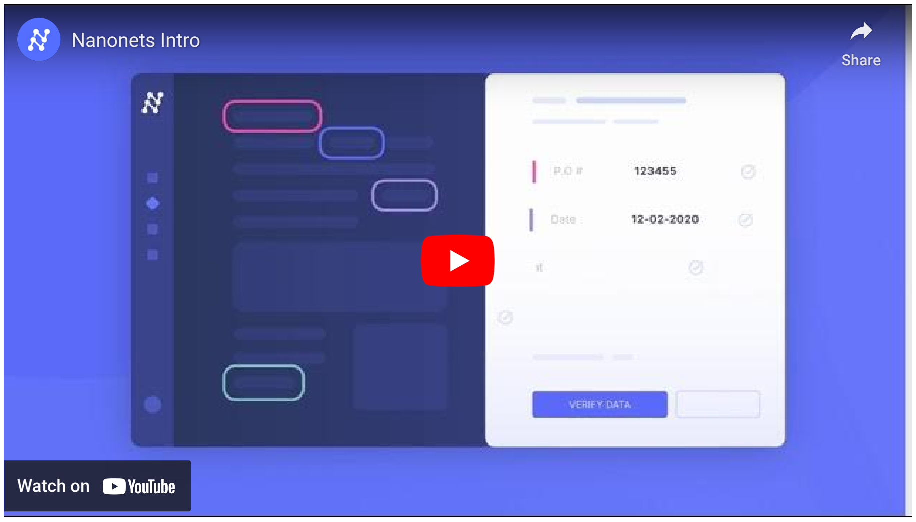
Step 9: Make Prediction
Once the model is trained. You can make predictions using the model

```
python ./code/prediction.py ./images/151.jpg
```

Nanonets - Best AI PDF OCR engine

Nanonets is an AI-based PDF OCR software that extracts text and tables from PDFs, handwritten scanned documents, emails, or images with 95% accuracy. Nanonets GUI is a no-code platform that allows you to automate data extraction using rule-based workflows.

The upshot? The time spent on manual processing nosedives, freeing up employees to focus on more strategic tasks that drive growth.



Capterra Rating: 4.9

G2 Rating: 4.9

Start Free Trial

Get Product Demo

Conclusion

PyPDF2 is one of the easiest ways to convert PDF files, and it's completely open source. If you're in a hurry, the excellent online documentation will have you up and running in minutes. If you have questions or need more help, the friendly PyPDF2 community will gladly offer their assistance. As well as being simple to use, PyPDF2 is exceptionally lightweight—it has no other dependencies besides Python (which means it'll work on almost every platform imaginable).

Moreover, PyPDF2 is distributed under a BSD-style license, so you can bundle it with your software if you like. In short, this is an excellent tool for manipulating PDFs, and we recommend that Python developers check it out.

FAQs

Can Python Read a PDF?

Python has no native support for reading PDF files, so this isn't something you will be able to do with a single line of code. But plenty of third-party libraries allow Python to read PDFs and convert them into other formats, such as HTML or plain text.

Another question arises here if Python reads a PDF, then:

Can Python read Excel files too?

Yes, Python can read Excel files. Pandas make it simple to import an Excel file into Python. You must use read_excel to achieve this objective.

Is PyPDF2 Open Source?

PyPDF2 is open-source software licensed under the LGPL.

Also, PyPDF2 is available for download in source code form. It can be installed using pip or downloading the zip file and extracting it to your chosen directory.

The PyPDF2 library includes several command-line tools that can be used to convert PDF files into other formats. These tools are installed with the Python module when it is installed.

Is PyPDF2 Safe?

PyPDF2 aims to provide a pure Python interface to **libpdf** (the C++ PDF Reference Library) rather than having a separate C extension module linked to Python.

The primary goal of PyPDF2 is to make it easier for developers to create PDF applications without having to worry about installing a complicated development environment or dealing with multiple versions of external libraries.

Can Excel extract data from PDF?

Yes, Excel can extract data from PDF.

Excel is a great tool for manipulating data and is easy to use. It's also very powerful and can be used to handle many different types of data.

In addition, Excel is a big advantage because you can use it on any platform (Windows, Mac, Linux), and you don't need any special software.

The process of extracting data from PDF is not straightforward, but we will show you how to do it step by step.

What is OCR Python?

OCR Python is a fully-featured OCR library written in pure Python. It wraps the Tesseract open source OCR engine and provides a simple API for developers to use. OCR, Optical Character Recognition, converts scanned text images into searchable, digital text.

OCR Python uses Tesseract's high-quality output as its base, and it can be used with any other OCR engine that uses the Leptonica or Harp libraries (such as GOCR).

If you want to digitize documents using OCR, then this library will help you quickly and easily.

Why is Text Extraction from PDF Hard?

Text extraction from PDF is hard. There are many reasons for this:

The PDF format was designed to be read by humans, not machines. The world's most popular document format has many neat features that make it easy for people to read, but it's a pain for computers to deal with.

PDFs can contain any content (text, charts, images, etc.), and they can be laid out in any way you want. This means there is no standard way to extract text from a PDF file — every file has its unique layout.

The text in a given PDF might not be located where you expect it to be! Some PDFs have tables of contents or indexes containing all the document's text; others have footnotes or endnotes; others have headers and footers that repeat at regular intervals; others use frames or layers instead of pages (this is rare).

OCR vs. Text Extraction

Text can be extracted from photographs using optical character recognition (OCR). OCR software is what accomplishes this. The most well-known Open Source OCR program is the tesseract OCR engine.

PyPDF2 isn't an OCR program.

Nanonets online OCR & OCR API have many interesting use cases that could optimize your business performance, save costs and boost growth. Find out how Nanonets' use cases can apply to your product.

Get Started

Read more:

- How to edit PDF metadata?
- How to rearrange PDF pages?
- How to use Nanonets PDF scraper?
- Free PDF OCR tool
- Top OCR Software for Mac

- How to use Google Drive OCR

Related content

WORKFLOW AUTOMATION

8 Ways to Use ChatGPT for Finance

WORKFLOW AUTOMATION

Order entry automation simplified

WORKFLOW AUTOMATION

What is the Role of AI in Lending and Loan Management?

WORKFLOW AUTOMATION

Calendly Meeting Analytics