Convert Excel to PDF in Python: The Ultimate 2024 Guide

Alexander Stock · Follow
7 min read · Oct 8, 2023

Microsoft Excel is now a vital tool for data management and analysis. However, there are cases where sharing the original Excel file isn't ideal due to reasons like formatting preservation, cross-device compatibility, or preventing unauthorized changes. That's when converting Excel files to PDF format becomes necessary.

In this blog post, I will guide you through the process of converting Excel files to PDF using Python, demonstrating it with eight specific examples.

- Convert a Worksheet to PDF with Default Page Settings
- Convert a Worksheet to PDF with Custom Page Margins
- Cut Unnecessary Blank Areas while Converting a Worksheet to PDF
- Specify Page Size while Converting a Worksheet to PDF
- Preserve Gridlines while Converting a Worksheet to PDF
- Specify Page Orientation while Converting a Worksheet to PDF
- Convert a Cell Range to PDF
- Convert a Workbook to PDF

Install Dependency

This solution requires Spire.XLS for Python to be installed as the dependency, which is a Python library for reading, creating and manipulating Excel documents in a Python program. You can install Spire.XLS for Python by executing the following pip commands.

```
pip install Spire.XLS
```

Convert a Worksheet to PDF with Default Page Settings

In Microsoft Excel, page settings refer to the various parameters that determine how your worksheet will be printed or displayed when converted to a physical page or a digital format like PDF. These settings allow you to control aspects such as page orientation, paper size, margins, scaling, and more.

If you have already set the desired page settings for your worksheet, you can convert it to PDF using the **Worksheet.SaveToPdf** method provided by Spire.XLS.

```
from spire.xls import *
from spire.xls.common import *

# Create a Workbook object
workbook = Workbook()

# Load an Excel document
```

```
workbook.LoadFromFile("C:\\Users\\Administrator\\Desktop\\Input.xlsx")

# Get a specific worksheet (index starts at zero)
sheet = workbook.Worksheets[0]

# Convert the worksheet to PDF file
sheet.SaveToPdf("output/WorksheetToPdf.pdf")

# Dispose resources
workbook.Dispose()
```

Convert a Worksheet to PDF with Custom Page Margins

By adjusting the margin settings, you can control the spacing between the content and the edges of the PDF page. This is useful when you want to place the content at the center of the page, or ensure that important data or elements are not too close to the page borders in the converted PDF.

Spire.XLS offers the **PageSetup** class to work with page settings including margins. This class provides properties such as **TopMargin**, **BottomMargin**, **LeftMargin**, and **RightMargin**, which are responsible for configuring the margins on each side of the page.

```
from spire.xls import *
from spire.xls.common import *

# Create a Workbook object
workbook = Workbook()

# Load an Excel document
workbook.LoadFromFile("C:\\Users\\Administrator\\Desktop\\Input.xlsx")

# Fit worksheet on one page
workbook.ConverterSetting.SheetFitToPage = True

# Get a specific worksheet (index starts at zero)
sheet = workbook.Worksheets[1]

# Get the PageSetup object
pageSetup = sheet.PageSetup

# Set page margins
pageSetup.TopMargin = 0.1
pageSetup.BottomMargin = 0.1
pageSetup.LeftMargin = 0.1
pageSetup.RightMargin = 0.1
```

```
# Convert the worksheet to PDF file
sheet.SaveToPdf("output/CostomizePageMargins.pdf")

# Dispose resources
workbook.Dispose()
```

Cut Unnecessary Blank Areas while Converting a Worksheet to PDF

When converting worksheets to PDFs, you can eliminate unnecessary blank areas. This ensures that the resulting PDF file accurately represents the content of the worksheet without any excessive empty space.

To cut unnecessary blank areas, you need first to set the page margins to small values or zero. This minimizes the empty space surrounding the content. Secondly, you're required to set the **Workbook.ConverterSetting.SheetFitToPage** property to true, which makes sure that the PDF page size is adjusted to fit the content area of the worksheet.

```
from spire.xls import *
from spire.xls.common import *

# Create a Workbook object
workbook = Workbook()

# Load an Excel document
workbook.LoadFromFile("C:\\Users\\Administrator\\Desktop\\Input.xlsx")

# Fit worksheet on one page
workbook.ConverterSetting.SheetFitToPage = True

# Get a specific worksheet (index starts at zero)
sheet = workbook.Worksheets[1]

# Get the PageSetup object
pageSetup = sheet.PageSetup

# Set page margins
pageSetup.TopMargin = 0.1
pageSetup.BottomMargin = 0.1
pageSetup.LeftMargin = 0.1
pageSetup.RightMargin = 0.1

# The PDF page size is adjusted to fit the content area
workbook.ConverterSetting.SheetFitToPage = True
```

```
# Convert the worksheet to PDF file
sheet.SaveToPdf("output/RemoveBlankArea.pdf")

# Dispose resources
workbook.Dispose()
```

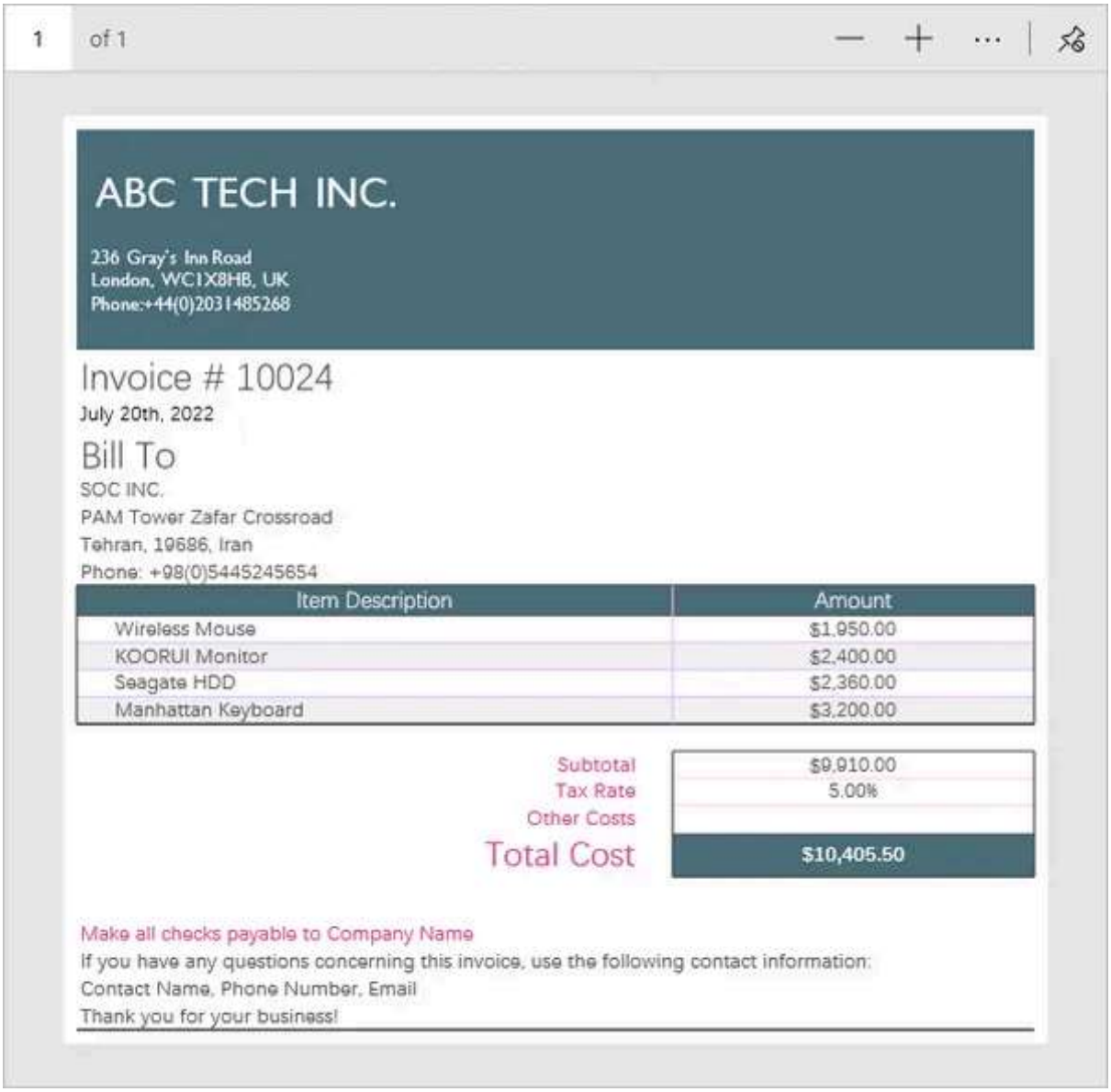


Figure 1. Convert Worksheet to PDF without Unnecessary Blank Areas

Specify Page Size while Converting a Worksheet to PDF

Converting a worksheet to PDF with a standard page size like A4 ensures compatibility, consistency, and convenience when sharing, viewing, and printing the document. It promotes better document organization and aligns with widely accepted international standards.

Within the **PageSetup** class, you can utilize the **PaperSize** property to define the page size when converting a document to PDF. This property allows you to specify

standard paper sizes such as A3, A4, B4, B5, or even a custom paper size according to your specific requirements.

```
from spire.xls import *
from spire.xls.common import *

# Create a Workbook object
workbook = Workbook()

# Load an Excel document
workbook.LoadFromFile("C:\\Users\\Administrator\\Desktop\\Input.xlsx")

# Fit worksheet on one page
workbook.ConverterSetting.SheetFitToPage = True

# Fit worksheet on one page while retaining paper size
workbook.ConverterSetting.SheetFitToPageRetainPaperSize = True

# Get a specific worksheet (index starts at zero)
sheet = workbook.Worksheets[1]

# Get the PageSetup object
pageSetup = sheet.PageSetup

# Set the page size to A4
pageSetup.PaperSize = PaperSizeType.PaperA4

# Convert the worksheet to PDF file
sheet.SaveToPdf("output/StandardPageSize.pdf")

# Dispose resources
workbook.Dispose()
```

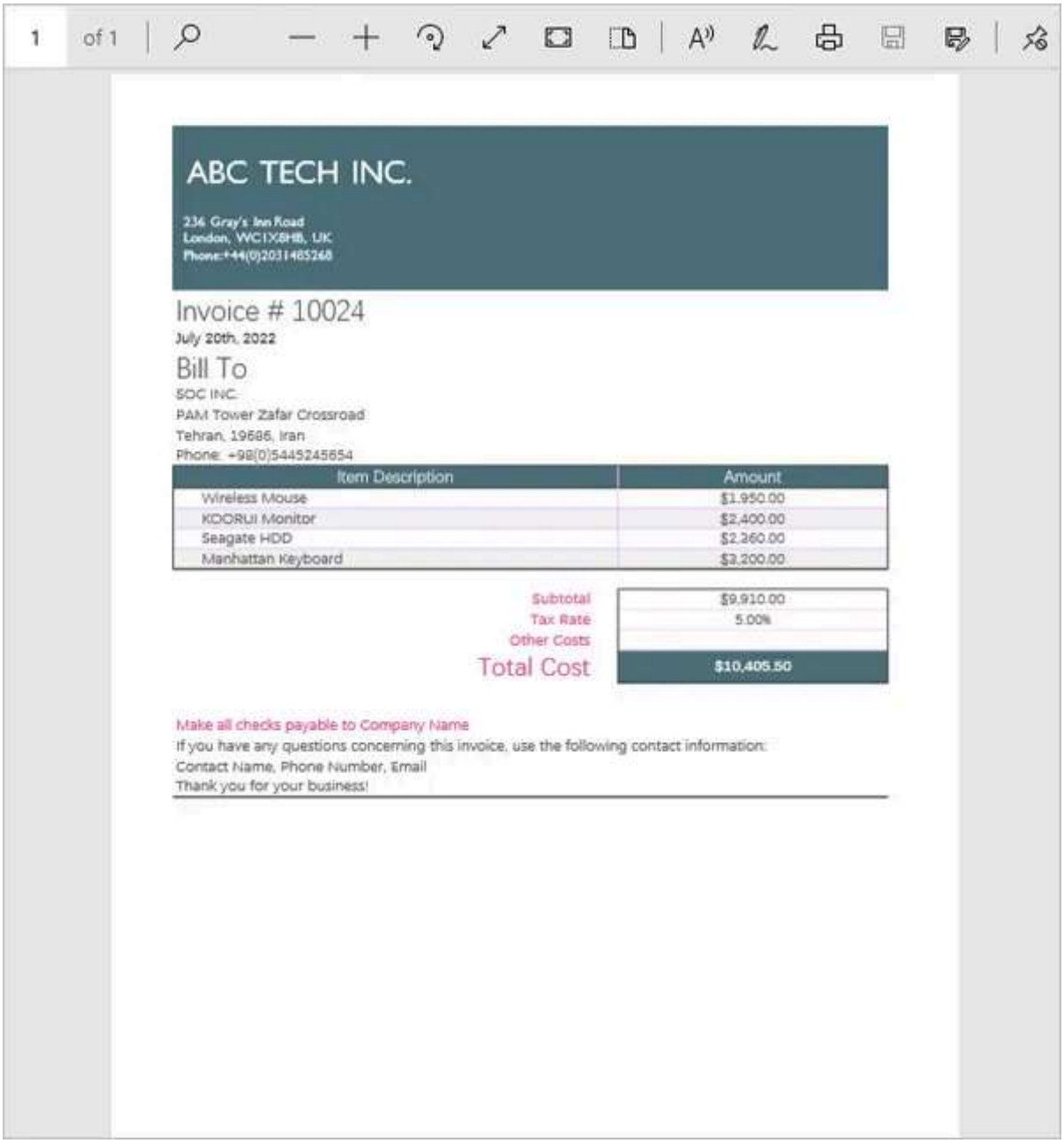


Figure 2. Convert Worksheet to PDF with a Standard Page Size

Preserve Gridlines while Converting a Worksheet to PDF

By preserving the gridlines during the conversion process, the resulting PDF file will maintain the visual representation of the grid structure from the original worksheet. This can be helpful for maintaining readability, data alignment, and overall visual consistency.

To enable the gridlines when converting worksheets to PDFs, set the **PageSetup.IsPrintGridlines** to True. The default value is False.

```
from spire.xls import *
from spire.xls.common import *
```

```
# Create a Workbook object
workbook = Workbook()

# Load an Excel document
workbook.LoadFromFile("C:\\Users\\Administrator\\Desktop\\Input.xlsx")

# Get a specific worksheet (index starts at zero)
sheet = workbook.Worksheets[1]

# Get the PageSetup object
pageSetup = sheet.PageSetup

# Preserve gridlines
pageSetup.IsPrintGridlines = True

# Convert the worksheet to PDF file
sheet.SaveToPdf("output/PreserveGridlines.pdf")

# Dispose resources
workbook.Dispose()
```

Specify Page Orientation while Converting a Worksheet to PDF

When converting a worksheet to PDF, you may want to specify the page orientation to ensure that the content is presented in the desired layout. To do so, you set the **PageSetup.Orientation** to either **Lanscape** or **Portrait**.

```
from spire.xls import *
from spire.xls.common import *

# Create a Workbook object
workbook = Workbook()

# Load an Excel document
workbook.LoadFromFile("C:\\Users\\Administrator\\Desktop\\Input.xlsx")

# Get a specific worksheet (index starts at zero)
sheet = workbook.Worksheets[1]

# Get the PageSetup object
pageSetup = sheet.PageSetup

# Set page orientation to Lanscape (or Portrait)
pageSetup.Orientation = PageOrientationType.Landscape

# Convert the worksheet to PDF file
sheet.SaveToPdf("output/PageOrientation.pdf")
```

```
# Dispose resources
workbook.Dispose()
```

Convert a Cell Range to PDF

By converting a specific cell range to PDF, you can create a focused and concise PDF document containing only the selected data. This is particularly useful when you need to share or distribute specific portions of a worksheet without including unnecessary information.

To specify a range of cells for conversion, you can assign a value, such as “A7:B11,” to the **PageSetup.PrintArea** property. This allows you to define the specific cell range that you want to include when converting the worksheet to PDF.

```
from spire.xls import *
from spire.xls.common import *

# Create a Workbook object
workbook = Workbook()

# Load an Excel document
workbook.LoadFromFile("C:\\Users\\Administrator\\Desktop\\Input.xlsx")

# Fit worksheet on one page
workbook.ConverterSetting.SheetFitToPage = True

# Get a specific worksheet (index starts at zero)
sheet = workbook.Worksheets[1]

# Get the PageSetup object
pageSetup = sheet.PageSetup

# Set print area
pageSetup.PrintArea = "A7:B11"

# Convert the range to PDF file
sheet.SaveToPdf("output/CellRangeToPdf.pdf")

# Dispose resources
workbook.Dispose()
```

Convert a Workbook to PDF

You can convert your workbook into a single PDF document with each sheet appearing on a separate page. This allows for easy navigation and viewing of each

sheet's content within the PDF file.

Using Spire.XLS, you have the capability to convert a workbook to a different file format, such as PDF, by employing the **Workbook.SaveToFile** method. Prior to the conversion process, you may need to customize the page settings for each sheet within the workbook and configure other specific options using the properties available under the **ConverterSetting** object.

```
from spire.xls import *
from spire.xls.common import *

# Create a Workbook object
workbook = Workbook()

# Load an Excel document
workbook.LoadFromFile("C:\\Users\\Administrator\\Desktop\\Input.xlsx")

# Fit worksheet on one page
workbook.ConverterSetting.SheetFitToPage = True

# Fit worksheet on one page while retaining paper size
workbook.ConverterSetting.SheetFitToPageRetainPaperSize = True

# Iterate through the worksheets in the workbook
for sheet in workbook.Worksheets:

    # Get the PageSetup object
    pageSetup = sheet.PageSetup

    # Set the page size to A4
    pageSetup.PaperSize = PaperSizeType.PaperA4

# Convert the workbook to PDF
workbook.SaveToFile("output/WorkbookToPdf.pdf", FileFormat.PDF)

# Dispose resources
workbook.Dispose()
```

Get a Free Trial License

The code snippets mentioned above produce PDF documents that include a red evaluation message at the center of each page. If you wish to remove the watermark, you can get a 30-day trial license here.

Conclusion

In this blog post, I provide a comprehensive summary of different scenarios for converting Excel files to PDF. I cover topics such as adjusting page size, setting page orientation, managing page margins, specifying the conversion area, and more. I hope that this post proves to be valuable and informative for your needs.

Related Topics

- Convert Word to PDF in Python
- Extract Data from Excel in Python
- Add Borders to Excel in Python
- Create Pivot Tables in Excel in Python

Python Excel Pdf API Convert

Follow

Written by Alexander Stock

180 Followers · 4 Following

Experienced software development consultant and blogger with more than 10 years in the field, focusing on office document tools and knowledge sharing.

Responses (1)

Luis Xbox

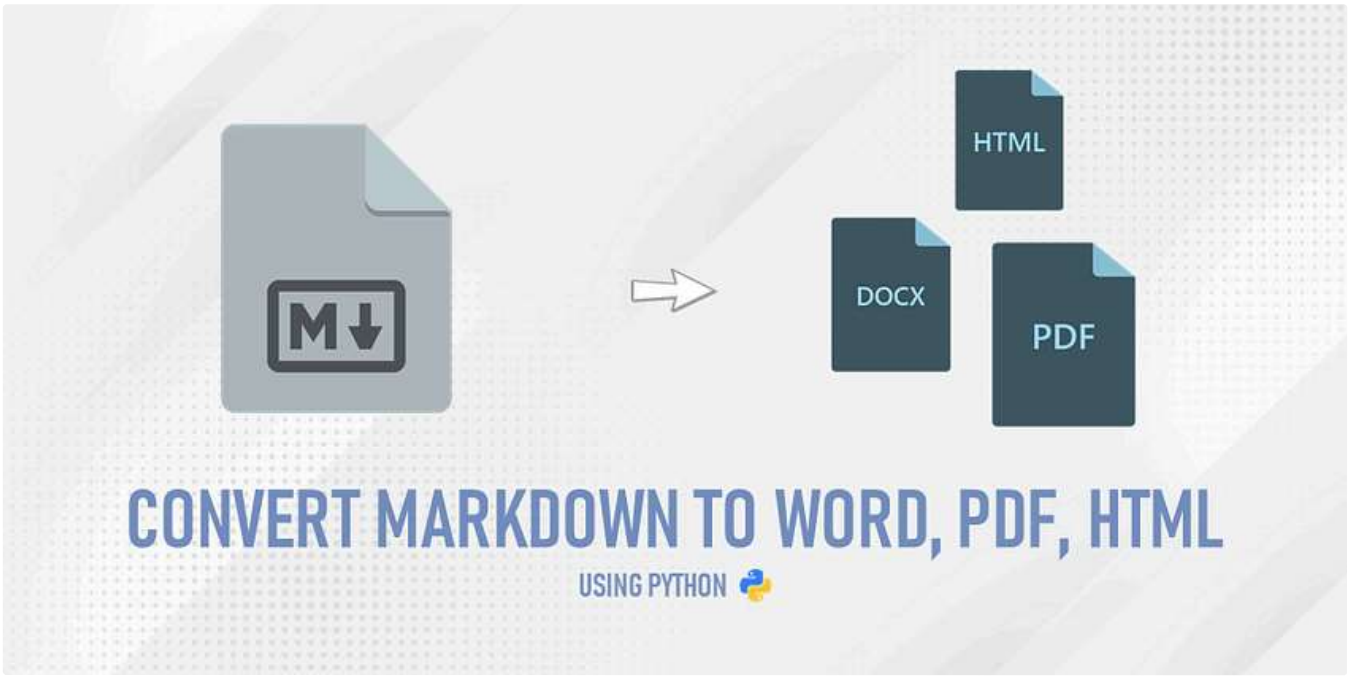
What are your thoughts?

Abdullah said
Dec 21, 2024

this is paid feature, 2500$ which is crazy

3 Reply

More from Alexander Stock

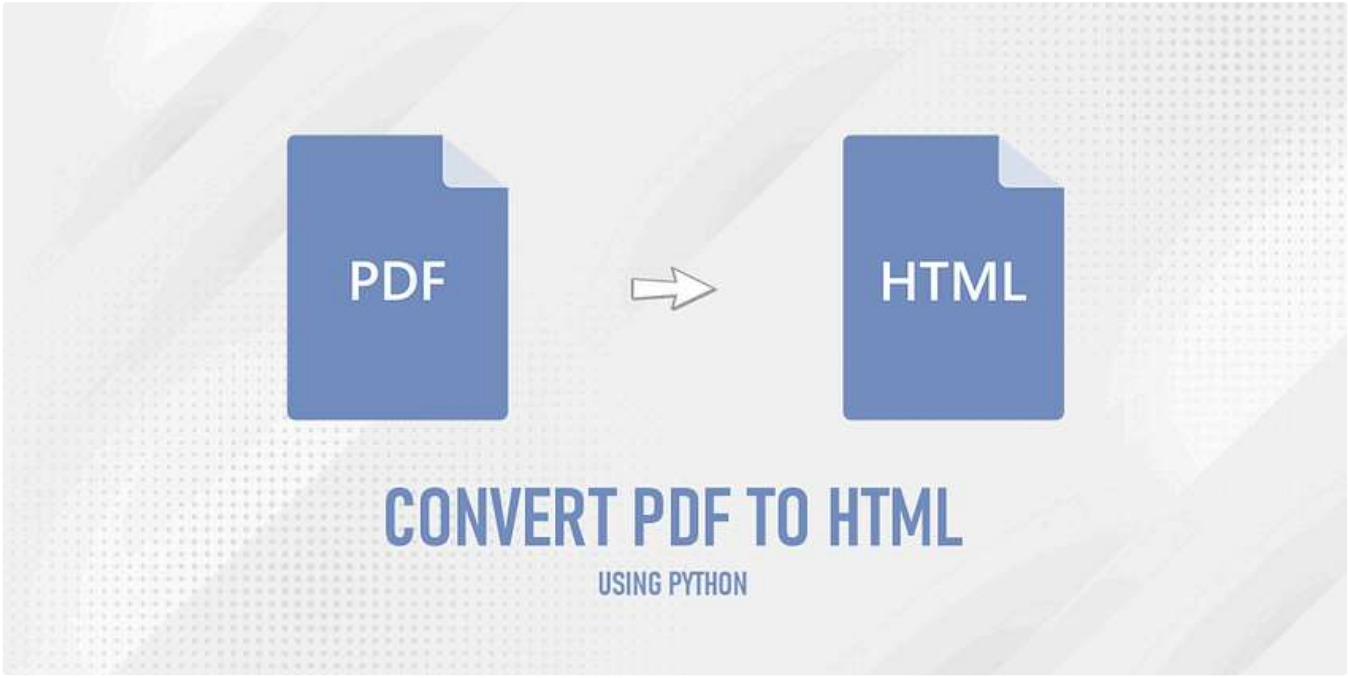


Alexander Stock

How to Convert Markdown to Word, PDF, and HTML with Python

Markdown has become a popular markup language due to its simplicity and versatility, allowing users to format text easily without the need...

Sep 30, 2024 11 1



Alexander Stock

Convert PDF to HTML with Python (Developer Guide)

By transforming PDF documents into HTML files, it becomes easier to integrate documents into web-based environment, incorporate interactive...

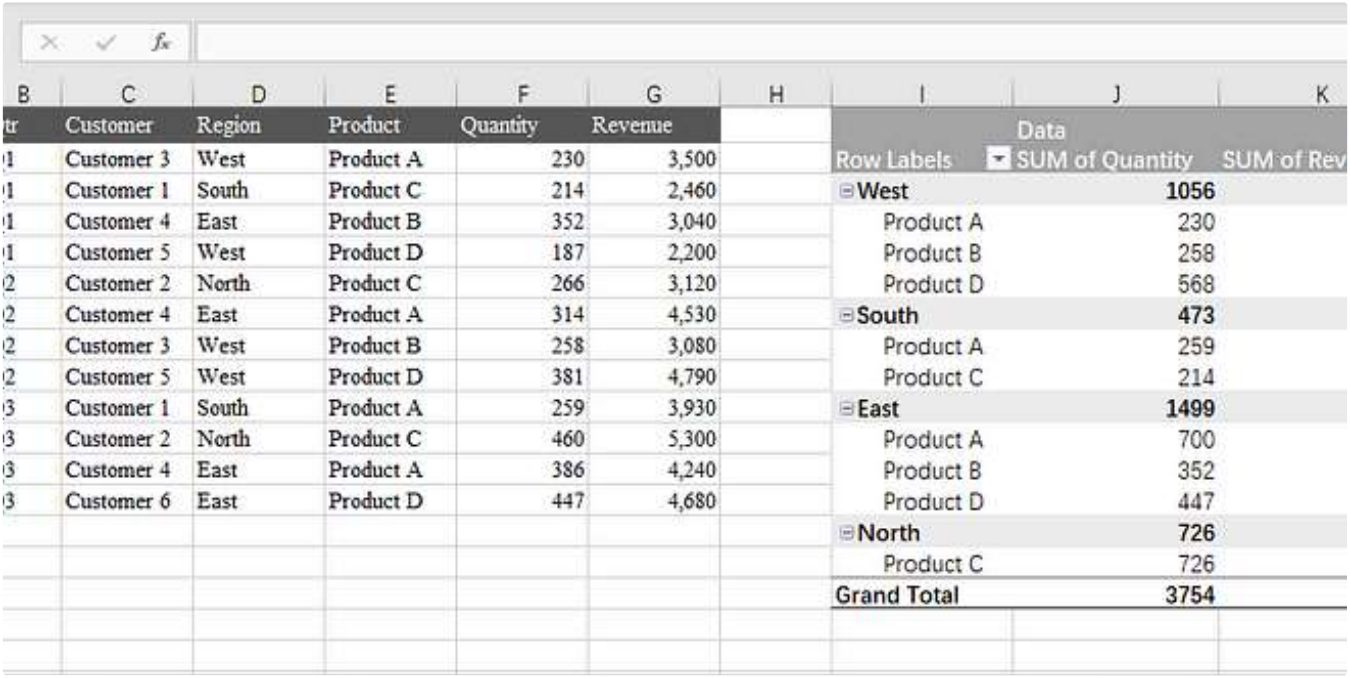
Apr 29, 2024 117 3

Alexander Stock

5 Ways to Convert PDF to Word in Python: A Comparison Guide

This guide provides 5 solutions for converting PDF to Word in Python s, highlighting the pros and cons of each solution.

Jul 8, 2024 11 2



tr	Customer	Region	Product	Quantity	Revenue
1	Customer 3	West	Product A	230	3,500
1	Customer 1	South	Product C	214	2,460
1	Customer 4	East	Product B	352	3,040
1	Customer 5	West	Product D	187	2,200
2	Customer 2	North	Product C	266	3,120
2	Customer 4	East	Product A	314	4,530
2	Customer 3	West	Product B	258	3,080
2	Customer 5	West	Product D	381	4,790
3	Customer 1	South	Product A	259	3,930
3	Customer 2	North	Product C	460	5,300
3	Customer 4	East	Product A	386	4,240
3	Customer 6	East	Product D	447	4,680

Row Labels	SUM of Quantity	SUM of Rev
West	1056	
Product A	230	
Product B	258	
Product D	568	
South	473	
Product A	259	
Product C	214	
East	1499	
Product A	700	
Product B	352	
Product D	447	
North	726	
Product C	726	
Grand Total	3754	

Alexander Stock

Create Pivot Tables in Excel Using Python

Pivot tables are a powerful feature in Microsoft Excel that allow users to analyze and summarize large amounts of data quickly and...

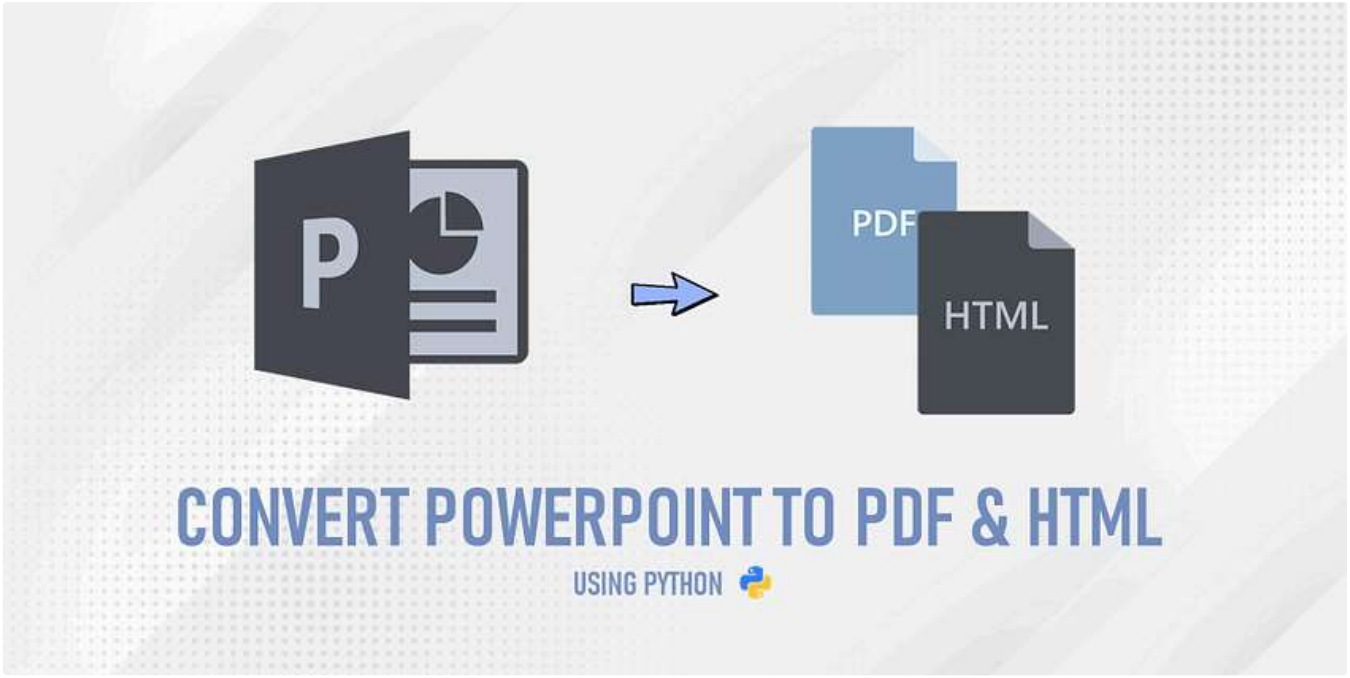
Jan 29, 2024 10 2

See all from Alexander Stock

Open in app ↗

Medium Search

Recommended from Medium



Alexander Stock

Convert PowerPoint Files to PDF and HTML from Simple Python Code

Converting PowerPoint files to PDF and HTML formats offers enhanced accessibility and versatility for presentations. PDFs preserve the...

Oct 18, 2024 52

In Stackademic by Kevin Meneses González

How to Extract Text from PDFs Using Python: A Practical Guide

Imagine this: you've just received a multi-page PDF filled with vital information, and you need to extract key details for a report...

Dec 3, 2024 108

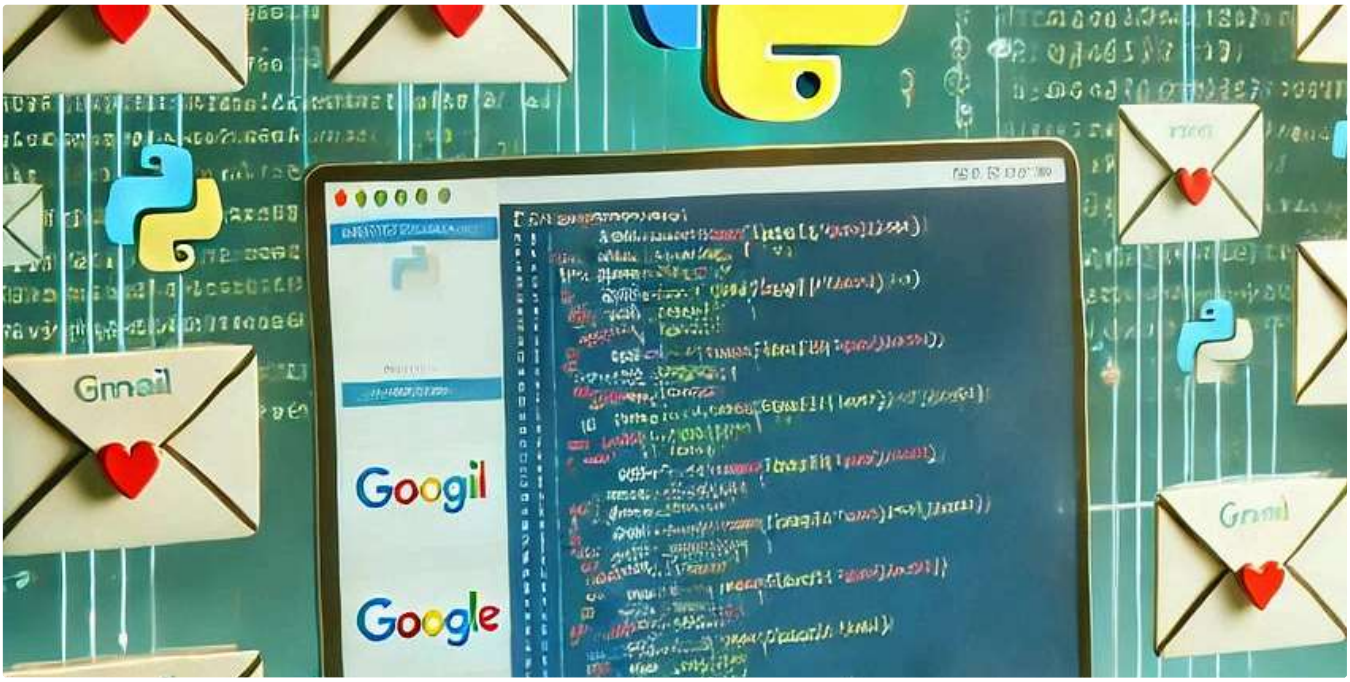
Lists

Coding & Development
11 stories · 1029 saves

Predictive Modeling w/ Python
20 stories · 1852 saves

Practical Guides to Machine Learning
10 stories · 2221 saves

ChatGPT
21 stories · 985 saves

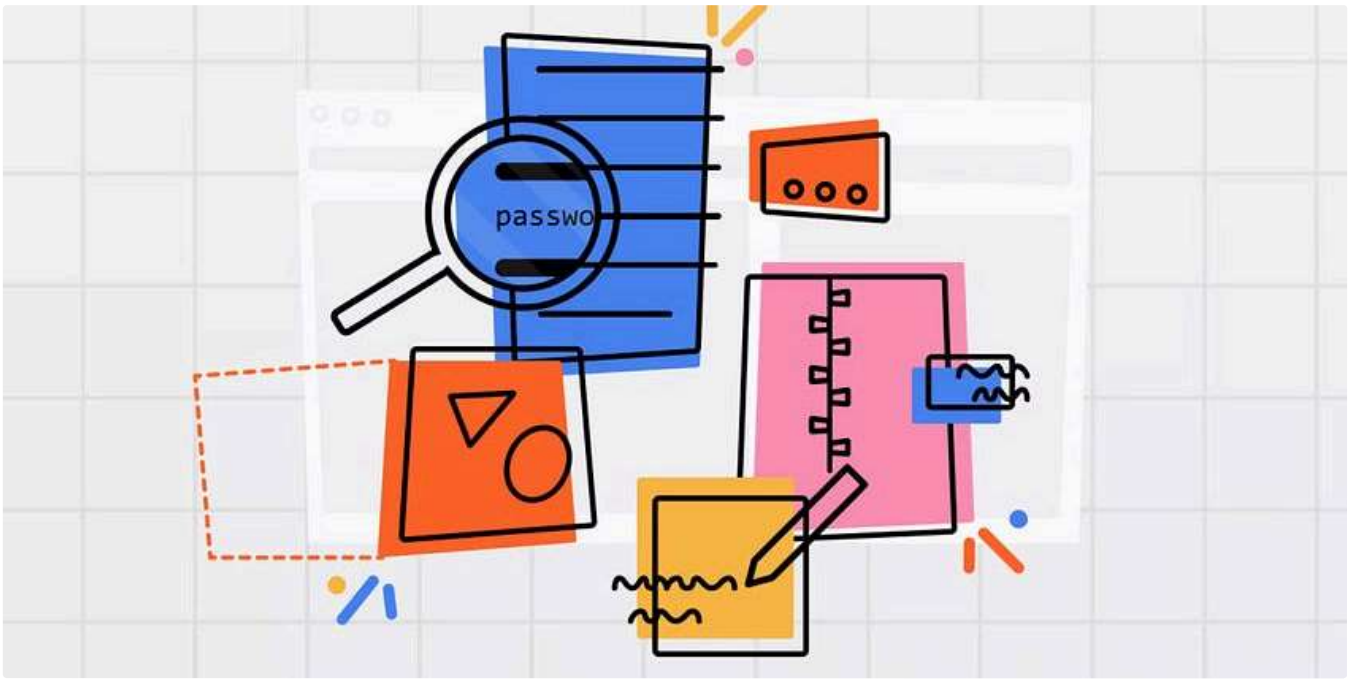


Shantanu Patra

Fetching and Analyzing Mail from Gmail using Python

Email communication is a key part of our daily digital life, and accessing email programmatically can be useful for automating tasks such...

Oct 1, 2024 23 1



Avinash Maheshwari

Unlocking Document Processing with Python: Advanced File Partitioning and Text Extraction

Processing and extracting information from diverse document formats is essential for numerous applications. Python's unstructured library...

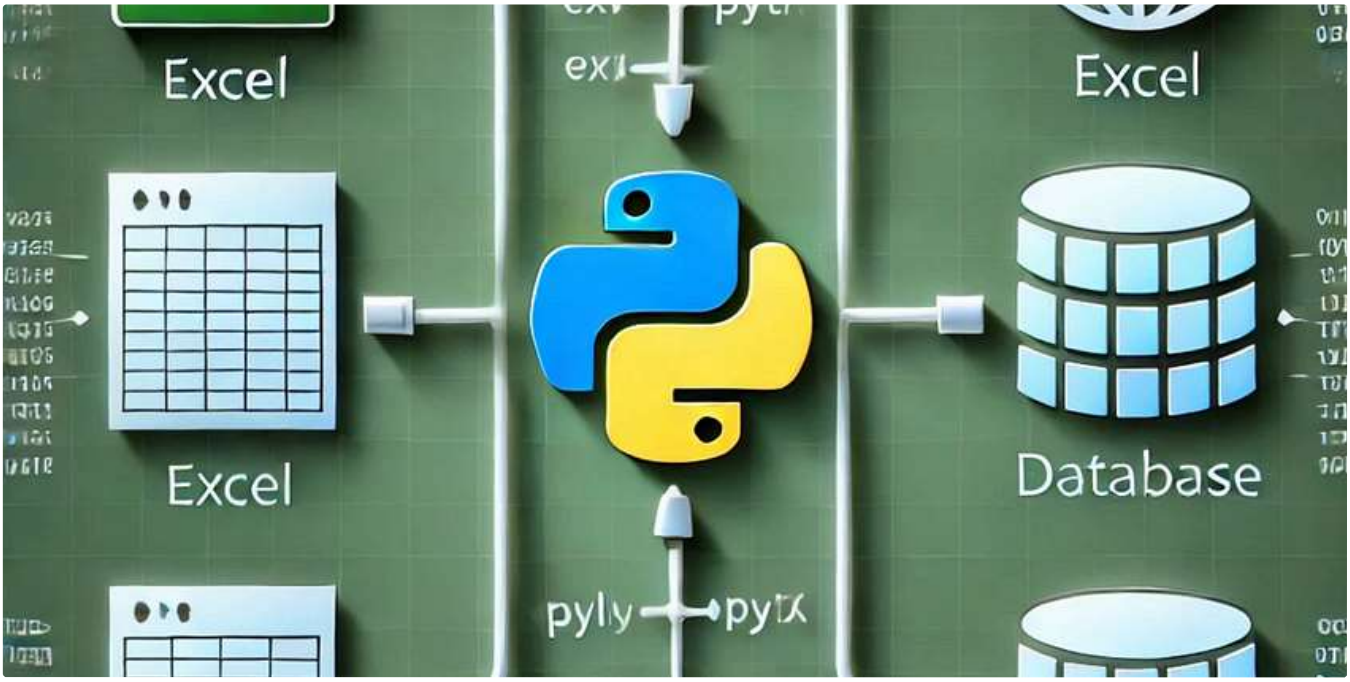
Dec 1, 2024 140 2

Juanrosario

Real-time Email Processing using Python and IMAP

1. Introduction

Oct 11, 2024 1



Balakrishna

Python Database Automation: Effortlessly Generate CREATE TABLE Statements from Excel

Automation is changing the way we work, and Python is at the heart of this transformation. One area where automation can significantly...

Oct 8, 2024

See more recommendations