The PdfFileWriter Class

class PyPDF2.**PdfFileWriter**

This class supports writing PDF files out, given pages produced by another class (typically `PdfFileReader`).

addAttachment(*fname*, *fdata*)

Embed a file inside the PDF.

Parameters:
- **fname** (*str*) – The filename to display.
- **fdata** (*str*) – The data in the file.

Reference:
https://www.adobe.com/content/dam/Adobe/en/devnet/acrobat/pdfs/PDF32000_2008.pdf
Section 7.11.3

addBlankPage(*width=None*, *height=None*)

Appends a blank page to this PDF file and returns it. If no page size is specified, use the size of the last page.

Parameters:
- **width** (*float*) – The width of the new page expressed in default user space units.
- **height** (*float*) – The height of the new page expressed in default user space units.

Returns: the newly appended page

Return type: `PageObject`

Raises PageSizeNotDefinedError: if width and height are not defined and previous page does not exist.

addBookmark(*title*, *pagenum*, *parent=None*, *color=None*, *bold=False*, *italic=False*, *fit='/Fit'*, **args*)

Add a bookmark to this PDF file.

Parameters:
- **title** (*str*) – Title to use for this bookmark.
- **pagenum** (*int*) – Page number this bookmark will point to.
- **parent** – A reference to a parent bookmark to create nested bookmarks.
- **color** (*tuple*) – Color of the bookmark as a red, green, blue tuple from 0.0 to 1.0
- **bold** (*bool*) – Bookmark is bold
- **italic** (*bool*) – Bookmark is italic
- **fit** (*str*) – The fit of the destination page. See `addLink()` for details.

addJS(*javascript*)

Add Javascript which will launch upon opening this PDF.

Parameters: **javascript** (*str*) – Your Javascript.

```
>>> output.addJS("this.print({bUI:true,bSilent:false,bShrinkToFit:true});")
# Example: This will launch the print window when the PDF is opened.
```

addLink(*pagenum*, *pagedest*, *rect*, *border=None*, *fit='/Fit'*, **args*)

Add an internal link from a rectangular area to the specified page.

Parameters:
- **pagenum** (*int*) – index of the page on which to place the link.

- pagedest (*int*) – index of the page to which the link should go.
- rect – `RectangleObject` or array of four integers specifying the clickable rectangular area `[xLL, yLL, xUR, yUR]`, or string in the form `"[ xLL yLL xUR yUR ]"`.
- border – if provided, an array describing border-drawing properties. See the PDF spec for details. No border will be drawn if this argument is omitted.
- fit (*str*) – Page fit or 'zoom' option (see below). Additional arguments may need to be supplied. Passing `None` will be read as a null value for that coordinate.

Valid zoom arguments (see Table 8.2 of the PDF 1.7 reference for details):

`/Fit`	No additional arguments
`/XYZ`	[left] [top] [zoomFactor]
`/FitH`	[top]
`/FitV`	[left]
`/FitR`	[left] [bottom] [right] [top]
`/FitB`	No additional arguments
`/FitBH`	[top]
`/FitBV`	[left]

`addMetadata`(*infos*)

Add custom metadata to the output.

Parameters: infos (*dict*) – a Python dictionary where each key is a field and each value is your new metadata.

`addPage`(*page*)

Adds a page to this PDF file. The page is usually acquired from a `PdfFileReader` instance.

Parameters: page (*PageObject*) – The page to add to the document. Should be an instance of `PageObject`

`appendPagesFromReader`(*reader*, *after_page_append=None*)

Copy pages from reader to writer. Includes an optional callback parameter which is invoked after pages are appended to the writer.

Parameters: reader – a PdfFileReader object from which to copy page annotations to this writer object. The writer's annots

will then be updated :callback after_page_append (function): Callback function that is invoked after

each page is appended to the writer. Callback signature:

param writer_pageref (PDF page reference):

Reference to the page appended to the writer.

`cloneDocumentFromReader`(*reader*, *after_page_append=None*)

Create a copy (clone) of a document from a PDF file reader

Parameters: reader – PDF file reader instance from which the clone should be created.

Callback after_page_append (function):

Callback function that is invoked after each page is appended to the writer. Signature includes a reference to the appended page (delegates to appendPagesFromReader). Callback signature:

param writer_pageref (PDF page reference):
Reference to the page just appended to the document.

cloneReaderDocumentRoot(*reader*)

Copy the reader document root to the writer.

Parameters: reader – PdfFileReader from the document root should be copied.

:callback after_page_append

encrypt(*user_pwd*, *owner_pwd=None*, *use_128bit=True*)

Encrypt this PDF file with the PDF Standard encryption handler.

Parameters:
- user_pwd (*str*) – The "user password", which allows for opening and reading the PDF file with the restrictions provided.
- owner_pwd (*str*) – The "owner password", which allows for opening the PDF files without any restrictions. By default, the owner password is the same as the user password.
- use_128bit (*bool*) – flag as to whether to use 128bit encryption. When false, 40bit encryption will be used. By default, this flag is on.

getNumPages()

Returns: the number of pages.
Return type: int

getPage(*pageNumber*)

Retrieves a page by number from this PDF file.

Parameters: pageNumber (*int*) – The page number to retrieve (pages begin at zero)
Returns: the page at the index given by *pageNumber*
Return type: PageObject

getPageLayout()

Get the page layout. See setPageLayout() for a description of valid layouts.

Returns: Page layout currently being used.
Return type: str, None if not specified

getPageMode()

Get the page mode. See setPageMode() for a description of valid modes.

Returns: Page mode currently being used.
Return type: str, None if not specified

insertBlankPage(*width=None*, *height=None*, *index=0*)

Inserts a blank page to this PDF file and returns it. If no page size is specified, use the size of the last page.

Parameters:
- width (*float*) – The width of the new page expressed in default user space units.
- height (*float*) – The height of the new page expressed in default user space units.
- index (*int*) – Position to add the page.

Returns: the newly appended page
Return type: PageObject

Raises PageSizeNotDefinedError:

if width and height are not defined and previous page does not exist.

insertPage(*page*, *index=0*)

Insert a page in this PDF file. The page is usually acquired from a `PdfFileReader` instance.

Parameters:
- **page** (*PageObject*) – The page to add to the document. This argument should be an instance of `PageObject`.
- **index** (*int*) – Position at which the page will be inserted.

pageLayout

Read and write property accessing the `getPageLayout()` and `setPageLayout()` methods.

pageMode

Read and write property accessing the `getPageMode()` and `setPageMode()` methods.

removeImages(*ignoreByteStringObject=False*)

Removes images from this output.

Parameters: **ignoreByteStringObject** (*bool*) – optional parameter to ignore ByteString Objects.

removeLinks()

Removes links and annotations from this output.

removeText(*ignoreByteStringObject=False*)

Removes images from this output.

Parameters: **ignoreByteStringObject** (*bool*) – optional parameter to ignore ByteString Objects.

setPageLayout(*layout*)

Set the page layout

Parameters: **layout** (*str*) – The page layout to be used

Valid layouts are:

`/NoLayout`	Layout explicitly not specified
`/SinglePage`	Show one page at a time
`/OneColumn`	Show one column at a time
`/TwoColumnLeft`	Show pages in two columns, odd-numbered pages on the left
`/TwoColumnRight`	Show pages in two columns, odd-numbered pages on the right
`/TwoPageLeft`	Show two pages at a time, odd-numbered pages on the left
`/TwoPageRight`	Show two pages at a time, odd-numbered pages on the right

setPageMode(*mode*)

Set the page mode.

Parameters: **mode** (*str*) – The page mode to use.

Valid modes are:

`/UseNone`	Do not show outlines or thumbnails panels
`/UseOutlines`	Show outlines (aka bookmarks) panel
`/UseThumbs`	Show page thumbnails panel
`/FullScreen`	Fullscreen view

/UseOC	Show Optional Content Group (OCG) panel
/UseAttachments	Show attachments panel

updatePageFormFieldValues(*page*, *fields*)

Update the form field values for a given page from a fields dictionary. Copy field texts and values from fields to page.

Parameters:
- **page** – Page reference from PDF writer where the annotations and field data will be updated.
- **fields** – a Python dictionary of field names (/T) and text values (/V)

write(*stream*)

Writes the collection of pages added to this object out as a PDF file.

Parameters: **stream** – An object to write the file to. The object must support the write method and the tell method, similar to a file object.