**Merging PDF files**[**ℑ**](https://pypdf2.readthedocs.io/en/3.0.0/user/merging-pdfs.html#merging-pdf-files)

**Basic Example**[**ℑ**](https://pypdf2.readthedocs.io/en/3.0.0/user/merging-pdfs.html#basic-example)

**from** **PyPDF2** **import** PdfWriter

merger = PdfWriter()

**for** pdf **in** ["file1.pdf", "file2.pdf", "file3.pdf"]:

merger.append(pdf)

merger.write("merged-pdf.pdf")

merger.close()

For more details, see an excellent answer on [StackOverflow](https://stackoverflow.com/questions/3444645/merge-pdf-files) by Paul Rooney.

**Showing more merging options**[**ℑ**](https://pypdf2.readthedocs.io/en/3.0.0/user/merging-pdfs.html#showing-more-merging-options)

**from** **PyPDF2** **import** PdfWriter

merger = PdfWriter()

input1 = open("document1.pdf", "rb")

input2 = open("document2.pdf", "rb")

input3 = open("document3.pdf", "rb")

*# add the first 3 pages of input1 document to output*

merger.append(fileobj=input1, pages=(0, 3))

*# insert the first page of input2 into the output beginning after the second page*

merger.merge(position=2, fileobj=input2, pages=(0, 1))

*# append entire input3 document to the end of the output document*

merger.append(input3)

*# Write to an output PDF document*

output = open("document-output.pdf", "wb")

merger.write(output)

*# Close File Descriptors*

merger.close()

output.close()

**append**[**ℑ**](https://pypdf2.readthedocs.io/en/3.0.0/user/merging-pdfs.html#append)

append has been slighlty extended in PdfWriter.

see [pdfWriter.append](https://pypdf2.readthedocs.io/en/3.0.0/modules/PdfWriter.html" \l "PyPDF2.PdfWriter.append) for more details

**parameters:**

*fileobj*: PdfReader or filename to merge *outline\_item*: string of a outline/bookmark pointing to the beginning of the inserted file. if None, or omitted, no bookmark will be added. *pages*: pages to merge ; you can also provide a list of pages to merge None(default) means that the full document will be merged. *import\_outline*: import/ignore the pertinent outlines from the source (default True) *excluded\_fields*: list of keys to be ignored for the imported objects; if “/Annots” is part of the list, the annotation will be ignored if “/B” is part of the list, the articles will be ignored

examples:

writer.append("source.pdf",(0,10))  # append the first 10 pages of source.pdf

writer.append(reader,"page 1 and 10",[0,9]) #append first and 10th page from reader and create an outline)

During the merging, the relevant named destination will also imported.

If you want to insert pages in the middle of the destination, use merge (which provides (insert) position)

You can now insert the same page multiple times. You can also insert the same page many time at once with a list:

eg: writer.append(reader,[0,1,0,2,0]) will insert the pages (1), (2), with page (0) before, in the middle and after

**add\_page / insert\_page[ℑ](https://pypdf2.readthedocs.io/en/3.0.0/user/merging-pdfs.html" \l "add-page-insert-page" \o "Permalink to this heading)**

It is recommended to use append or merge instead

**reset\_translation[ℑ](https://pypdf2.readthedocs.io/en/3.0.0/user/merging-pdfs.html" \l "reset-translation" \o "Permalink to this heading)**

During the cloning, if an object has been already cloned, it will not be cloned again, a pointer this previously cloned object is returned. because of that, if you add/merge a page that has been already added, the same object will be added the second time. If later you modify any of these two page, both pages can be modified independantly.

To reset, call writer.reset\_translation(reader)

**Advanced cloning**[**ℑ**](https://pypdf2.readthedocs.io/en/3.0.0/user/merging-pdfs.html#advanced-cloning)

In order to prevent side effect between pages/objects objects and all objects linked are linked during merging.

This process will be automatically applied if you use PdfWriter.append/merge/add\_page/insert\_page. If you want to clone an object before attaching it “manually”, use clone function of any PdfObject: eg:

cloned\_object = object.clone(writer)

if you try clone an object already belonging to writer, it will return the same object

cloned\_object == object.clone(writer)  # -> returns True

the same, if you try to clone twice an object it will return the previously cloned object

object.clone(writer) == object.clone(writer)  # -> returns True

Also, note that if you clone an object, you will clone all the objects below including the objects pointed by IndirectObject. because of that if you clone a page that includes some articles (“/B”), not only the first article, but also all the chained articles, and the pages where those articles can be read will be copied. It means that you may copy lots of objects, that will be saved in the output pdf.

In order to prevent, that you can provide the list of defined the fields in the dictionaries to be ignored:

eg: new\_page  = writer.add\_page(reader.pages[0],excluded\_fields=["/B"])