

Image Frames Interfere with Adjoining Borders

This document is provided to give a 2nd working copy of the same bug as reported in [Python3docs Bugs #3](#), with extra information supplied.

Comment

The report (below) was made at [BugZilla#164297](#) on Dec 12 2024. The bug was first documented at [GitHub Python3Docs Bug#02](#) and then documented separately as [Bug#03](#).

- a) Version 24.8.3.2 is affected by this bug
- b) Grandmother version 3.3.0.4 is affected by this bug
- c) All other versions between 3.3 & 24.8 are affected by this bug.
- d) Michael Stahl declared that *"this is working as designed"*[™] (good grief)

I now understand that the reason that LibreOffice has been riddled with bugs for the last 7+ years is *by design* rather than through ignorance, laziness or inability. There now follows extra comment + demonstration of this bug:-

original bug report:

The bottom border of an image-in-frame-with-caption will coalesce with the top inline-border of a paragraph. In other words, the below-paragraph spacing of an image will be added to the internal top-spacing of the next paragraph, rather than being placed between the two entity's borders.

Extra comment:

The original report is almost accurate (if wordy), though it is probably more accurate to say (as in the H1 Heading at top) that the Image+Caption Frame *"Interferes"* with adjoining borders. It also helps if a person can see this in action. These are the features to pay attention to:-

1. I believe this to be a frame bug.
2. The bug occurs with *anything* that is framed and is positioned next to something with a border.
3. The point below is that 3 of the paragraphs have a blue inline-border & thus the issue becomes measurable.
4. There are 5 demonstration pages below; the 1st two pages are individual examples of the two ingredients, whilst the final 3 pages are snapshots in a very short film.

How to produce this document

1. This document created from scratch under 24.8.3.2.
`menu:Help | About LibreOffice` shows: [Build](#)
2. All Styles within [Python3docs chapter_01.odt](#) then imported into this doc.
3. The 1st 3 pages from [bug02+.odt](#) copied into this doc & edited.
4. The image+caption “*Figure 3.1: IDLE’s Python Shell*” within [chapter_01.odt](#) then pasted into a new page (p#5) of this doc (image is anchored to a paragraph, but set to place itself at page centre-top).

There are 2 Paragraphs on that page. Before insertion the cursor was within the Pilcrow (‘¶’ - *EOP marker*) of the 1st paragraph & the image+caption inserted via a `Ctrl+V`. Notice how the image nevertheless anchored itself to the 2nd paragraph. Yet another bug.

Paragraph pilcrows are displayed via either `Ctrl+F10` or by placing a tick into `menu:View | Formatting Marks`.

5. 4 paragraphs from p8 of [chapter_01.odt](#) were copied in similar fashion into another new, blank page (p#5+1).
(last 3 paragraphs are Code Box 2 (CB2) paragraphs, which have an inline text-border)
6. Yet another new page (p#7) was created. 3 x plain-text (style: ‘Body Text / Text Body’) paragraphs were created on that page. The single word of text “para1”, “para2” + “para3” were placed into the relative paragraphs. That thus names the ladder of paragraphs for the image+caption to climb & descend.

Last, a 2nd copy of the same image was pasted into the pilcrow for para3. The outside frame of the image was then changed to place the image at paragraph centre-top (rather than page centre-top). Finally, a 2nd copy of the same 4 paragraphs from p#5+1 was pasted into the same pilcrow (which now was below the image).

The end result of all this is the words “para1”, “para2” at the top of p#7, followed by the image, then “para3”, then a plain-text paragraph, finally a CB2 3-line paragraph (which has an embedded blue border).

7. Next, 2 new pages (p#8 + p#9) were created & all the contents of p#7 were pasted into each page. For each of those pages, the image was selected on the page and then the down-arrow cursor (↓) was pressed.
8. For p#8 the cursor was pressed until the words “Let’s look ... the details:” appeared above the image
9. For p#9 the cursor continued to be pressed until the CB2 top-border appeared above the image.

Demonstrations:

Refer to the final pages (p#5 to p#9) for a practical demonstration of this bug.

Things to note:-

i. to recap:

- a) Each image+caption in all example pages is identical, with the sole exception that the p#5 image (*Figure 3.1*) is placed relative to the “Page text area” whereas all others are placed relative to the “Paragraph text area”. Moving the image+caption — as described in section#7 in the previous page — will actually auto-change any “Page text” placing within Setup to “Paragraph text” instead.
- b) Further, each image+caption frame is set with 0 cm borders; Wrap is OFF; spacing is 0cm except for bottom = 0.5 cm.
- c) All text paragraphs are identical in borders + spacing to the image+caption frame, with the sole extra wrinkle that *Code Box 2* (CB2) paragraphs have the check-box selected for “do not add space between paragraphs of the same style” (which means that only the bottom para has a bottom spacing).
- d) The combo of the above means that text elements will ‘flow’ around the frame of the image+caption. In addition, it also means that, once selected, the image+caption frame can be moved up & down the ladder of the surrounding text paragraphs by means of the keyboard up- (↑) or down-arrow (↓) cursor.

ii. in p#7 (*Figure 3.2*):

- a) The bottom spacing below ALL paragraphs + image is 0.5 cm
- b) The 3 x CB2 paragraphs are surrounded by an embedded blue border & the spacing from text above to the top of the border is 0.5 cm

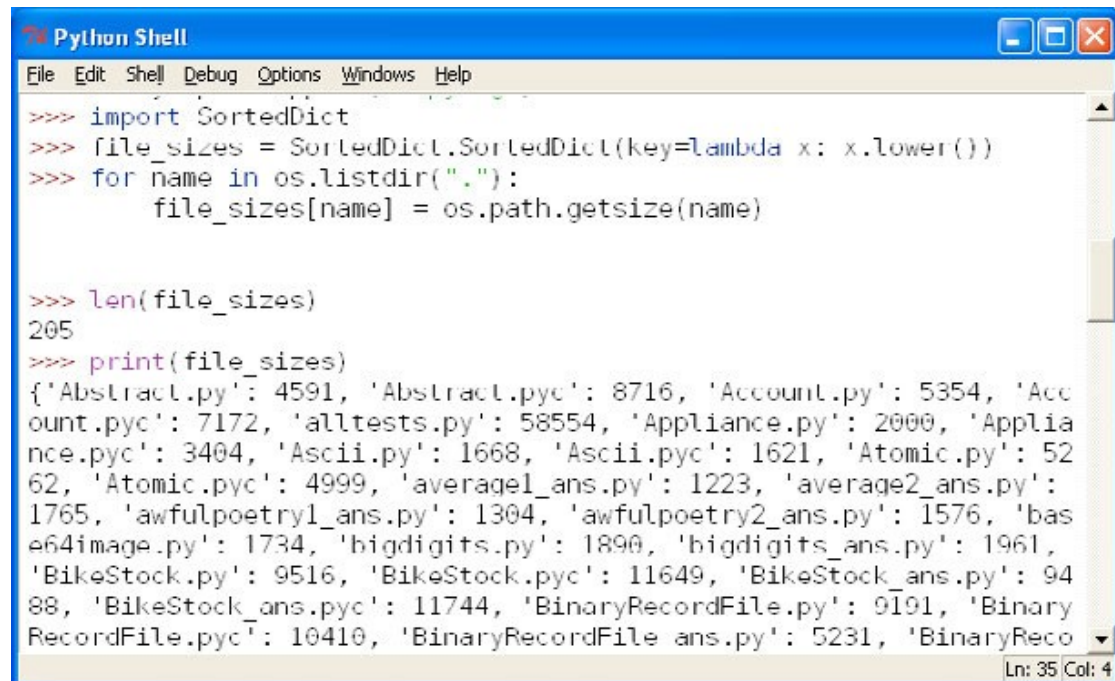
iii. in p#8 (*Figure 3.3*):

- a) This is the clearest demonstration of the bug.
- b) The text paragraph “Let’s look ... the details:” is above the image+caption; notice how the text bottom spacing is too small.
- c) The CB2 paragraphs are below the image+caption. Notice how the frame bottom spacing is too large.

- d) Where has the *CB2* top-border gone?
- iv. in p#9 (*Figure 3.4*):
 - a) This is probably the most blatant demonstration of the bug.
 - b) The *CB2* top-border now reappears above the image. I believe this to be a display error, in that the *CB2* border '*belongs*' to the *CB2* text & should not be separated from it. The current display looks daft.
 - c) This daft bug has been in place for more than 7 years, and is a Grandparent gift from Apache (if I recall the heredity of LibreOffice correctly). Time to fix it, methinks.
- v. All frames show this bug.

Examples

(see next pages):



```
Python Shell
File Edit Shell Debug Options Windows Help
>>> import SortedDict
>>> file_sizes = SortedDict.SortedDict(key=lambda x: x.lower())
>>> for name in os.listdir("."):
>>>     file_sizes[name] = os.path.getsize(name)

>>> len(file_sizes)
205
>>> print(file_sizes)
{'Abstract.py': 4591, 'Abstract.pyc': 8716, 'Account.py': 5354, 'Account.pyc': 7172, 'alltests.py': 58554, 'Appliance.py': 2000, 'Appliance.pyc': 3404, 'Ascii.py': 1668, 'Ascii.pyc': 1621, 'Atomic.py': 5262, 'Atomic.pyc': 4999, 'averagel_ans.py': 1223, 'average2_ans.py': 1765, 'awfulpoetry1_ans.py': 1304, 'awfulpoetry2_ans.py': 1576, 'base64image.py': 1734, 'bigdigits.py': 1890, 'bigdigits_ans.py': 1961, 'BikeStock.py': 9516, 'BikeStock.pyc': 11649, 'BikeStock_ans.py': 9488, 'BikeStock_ans.pyc': 11744, 'BinaryRecordFile.py': 9191, 'BinaryRecordFile.pyc': 10410, 'BinaryRecordFile_ans.py': 5231, 'BinaryReco
```

Figure 3.1: IDLE's Python Shell

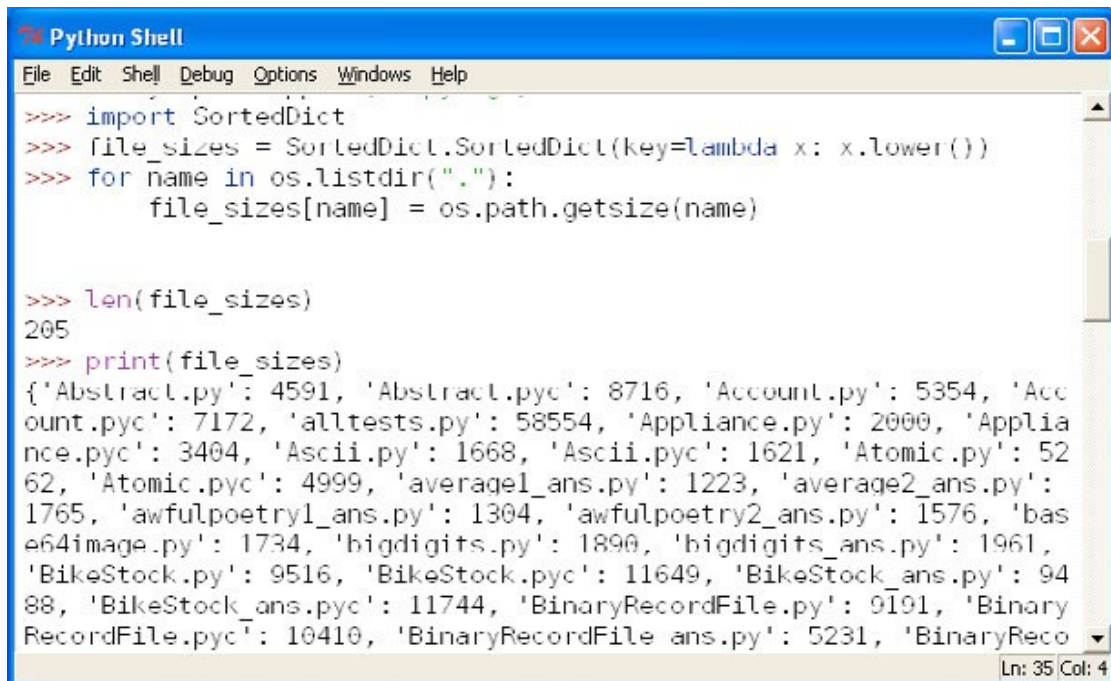
some words

Let's look at a few tiny examples, and then discuss some of the details:

```
x = "blue"  
y = "green"  
z = x
```

para1

para2



```
Python Shell
File Edit Shell Debug Options Windows Help

>>> import SortedDict
>>> file_sizes = SortedDict.SortedDict(key=lambda x: x.lower())
>>> for name in os.listdir("."):
>>>     file_sizes[name] = os.path.getsize(name)

>>> len(file_sizes)
205
>>> print(file_sizes)
{'Abstract.py': 4591, 'Abstract.pyc': 8716, 'Account.py': 5354, 'Account.pyc': 7172, 'alltests.py': 58554, 'Appliance.py': 2000, 'Appliance.pyc': 3404, 'Ascii.py': 1668, 'Ascii.pyc': 1621, 'Atomic.py': 5262, 'Atomic.pyc': 4999, 'average1_ans.py': 1223, 'average2_ans.py': 1765, 'awfulpoetry1_ans.py': 1304, 'awfulpoetry2_ans.py': 1576, 'base64image.py': 1734, 'bigdigits.py': 1890, 'bigdigits_ans.py': 1961, 'BikeStock.py': 9516, 'BikeStock.pyc': 11649, 'BikeStock_ans.py': 9488, 'BikeStock_ans.pyc': 11744, 'BinaryRecordFile.py': 9191, 'BinaryRecordFile.pyc': 10410, 'BinaryRecordFile_ans.py': 5231, 'BinaryReco
```

Figure 3.2: IDLE's Python Shell

para3

Let's look at a few tiny examples, and then discuss some of the details:

```
x = "blue"
y = "green"
z = x
```

para1

para2

para3

Let's look at a few tiny examples, and then discuss some of the details:

A screenshot of the IDLE Python Shell window. The window has a blue title bar with the text "Python Shell" and standard window controls. Below the title bar is a menu bar with "File", "Edit", "Shell", "Debug", "Options", "Windows", and "Help". The main area is a text editor with a light beige background, showing a Python script. The script imports SortedDict, creates a SortedDict of file sizes, and prints the result. The output shows a dictionary of file names and their sizes. The status bar at the bottom right indicates "Ln: 35 Col: 4".

```
>>> import SortedDict
>>> file_sizes = SortedDict.SortedDict(key=lambda x: x.lower())
>>> for name in os.listdir("."):
>>>     file_sizes[name] = os.path.getsize(name)

>>> len(file_sizes)
205
>>> print(file_sizes)
{'Abstract.py': 4591, 'Abstract.pyc': 8716, 'Account.py': 5354, 'Account.pyc': 7172, 'alltests.py': 58554, 'Appliance.py': 2000, 'Appliance.pyc': 3404, 'Ascii.py': 1668, 'Ascii.pyc': 1621, 'Atomic.py': 5262, 'Atomic.pyc': 4999, 'average1_ans.py': 1223, 'average2_ans.py': 1765, 'awfulpoetry1_ans.py': 1304, 'awfulpoetry2_ans.py': 1576, 'base64image.py': 1734, 'bigdigits.py': 1890, 'bigdigits_ans.py': 1961, 'BikeStock.py': 9516, 'BikeStock.pyc': 11649, 'BikeStock_ans.py': 9488, 'BikeStock_ans.pyc': 11744, 'BinaryRecordFile.py': 9191, 'BinaryRecordFile.pyc': 10410, 'BinaryRecordFile_ans.py': 5231, 'BinaryReco
```

Figure 3.3: IDLE's Python Shell

```
x = "blue"
y = "green"
z = x
```


para1

para2

para3

Let's look at a few tiny examples, and then discuss some of the details:



```
Python Shell
File Edit Shell Debug Options Windows Help

>>> import SortedDict
>>> file_sizes = SortedDict.SortedDict(key=lambda x: x.lower())
>>> for name in os.listdir("."):
>>>     file_sizes[name] = os.path.getsize(name)

>>> len(file_sizes)
205
>>> print(file_sizes)
{'Abstract.py': 4591, 'Abstract.pyc': 8716, 'Account.py': 5354, 'Account.pyc': 7172, 'alltests.py': 58554, 'Appliance.py': 2000, 'Appliance.pyc': 3404, 'Ascii.py': 1668, 'Ascii.pyc': 1621, 'Atomic.py': 5262, 'Atomic.pyc': 4999, 'average1_ans.py': 1223, 'average2_ans.py': 1765, 'awfulpoetry1_ans.py': 1304, 'awfulpoetry2_ans.py': 1576, 'base64image.py': 1734, 'bigdigits.py': 1890, 'bigdigits_ans.py': 1961, 'BikeStock.py': 9516, 'BikeStock.pyc': 11649, 'BikeStock_ans.py': 9488, 'BikeStock_ans.pyc': 11744, 'BinaryRecordFile.py': 9191, 'BinaryRecordFile.pyc': 10410, 'BinaryRecordFile_ans.py': 5231, 'BinaryReco
```

Figure 3.4: IDLE's Python Shell

```
x = "blue"
y = "green"
z = x
```