

Selecting a Caption Selects the Image

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

Comment

The combo of report#1 + report#2 (below) were made at [BugZilla#164295](#) on Dec 12 2024. The bug was first documented at [GitHub Python3Docs Bug#2](#). After a fortnight there were 3 significant responses to the BugZilla report:

- a) The latest 25.2.0.0.beta1+ version is still affected by this bug
Version 6.1.0.0.alpha0+ is *unaffected* by the bug
- b) It was introduced at version 6.5 on Fri Dec 6 14:04:55 2019 +0100
- c) Michael Stahl declared that *"this is working as designed"* (good grief)

Bugs #1 + #2 do NOT appear when using LibreOffice 6.1.0.3.
Bugs #3 + #4 DO appear using 6.1.0.3.

I now understand that the reason that LibreOffice has been riddled with bugs for the last 5 years is *by design* rather than through ignorance, laziness or inability. There now follow the original + updated bug reports:-

original bug report:

Selecting the whole of an image caption text (eg for rename) also selects the image. If the caption is then deleted the image itself will also be deleted.

update bug report 2:

If text exists immediately below the image-plus-caption, then selecting the top paragraph of text + it's Pilcrow ('¶' - EOL marker) will also select the image + caption above it. Deleting the text + paragraph will then delete both the paragraph + image + caption.

update bug report 3:

Placing the cursor behind the first Pilcrow ('¶') below the image+caption, then entering a command to create a new page, creates that page above the image rather than below. Doing the same within any other Pilcrow correctly creates a new page below the paragraph.

update bug report 4 (dupe of [bug03.odt](#)):

*Placing Code Box 2 paragraphs below the image+caption, as in the final page (p9) of this bug-report, illustrates the way that the image captures & integrates the 1st paragraph of text within itself + mangles it's border. The top-line of a Code Box 2 integral border is placed above the image, and the bottom spacing of the image frame (here 0.5 cm) is added **above** the paragraph text rather than between the two borders.*

How to produce this document

1. This document created from scratch under 24.8.3.2.
2. All Styles within [Python3docs chapter_01.odt](#) then imported into this doc.
3. The 1st page from [bug02.odt](#) copied into this doc & later edited.
4. The image+caption “Figure 2.1: IDLE’s Python Shell” within [chapter_01.odt](#) pasted into a new page (p#6 of this doc (*image is anchored to a paragraph*)).
5. 4 paragraphs from p8 of [chapter_01.odt](#) copied into another new page (p7). (*last 3 paragraphs are Code Box 2 paragraphs, which have an inline border*)
6. Inside yet another new page (p#8), a 2nd copy of the same image is pasted into the pilcrow at top of page, then a 2nd copy of the same 4 paragraphs pasted into the same pilcrow (which now is below the image).
7. Create a final new page (p#9), paste all the contents of p#8 into it, then remove (what used to be) the top paragraph of p6. That will leave the image+caption plus, below it, the 3 lines that are within a *Code Box 2* paragraph style.

Demonstrations:

Refer to the final pages for a practical demonstration of this set of bugs.

LibreOffice 6.1.0.3 does NOT contain these bugs. A set of old AppImages that include that version are available from libreoffice.soluzioniopen.com. Refer to [AppImage Installation \(Python3docs on Github\)](#) for brief help on installing an AppImage.

To easily demonstrate this problem (1) (see page#6, #8 or #9):

- i. Click within the Image caption
- ii. Press the <HOME> key
(cursor should show at front of ‘Figure’)
- iii. Hold down <SHIFT> key and then press <END> key
- iv. Under LO v7 & later image + caption are now both selected
(s/b just the caption)

To easily demonstrate this problem (2) (see page#8 or #9):

- i. Do the same as (i) above, but this time start with the 1st letter of the 1st line on the paragraph that is below the Image.
- ii. Select to the end of the 1st line
(no problem)
- iii. Select to the end of the whole paragraph
(Under LO v7 & later the whole of the image above + caption + paragraph below has been selected)
(that’s greedy; deleting the paragraph will now delete everything)

Note: This only happens on pages below that contain both an image + text below the image.

To easily demonstrate this problem (3):

- i. Place your cursor to the left of the 1st Pilcrow (‘¶’) below any image+caption.
- ii. Enter the command to insert a new page
(this is either `menu:Insert | Page Break` or `kbd:Ctrl+Enter`). Note that the Page Break has been inserted *before* your current page & not *after* it.
- iii. Trying the same command on a page that has more than one Pilcrow, after the 1st Pilcrow, will work as expected (which is Page Break *after* the current page).

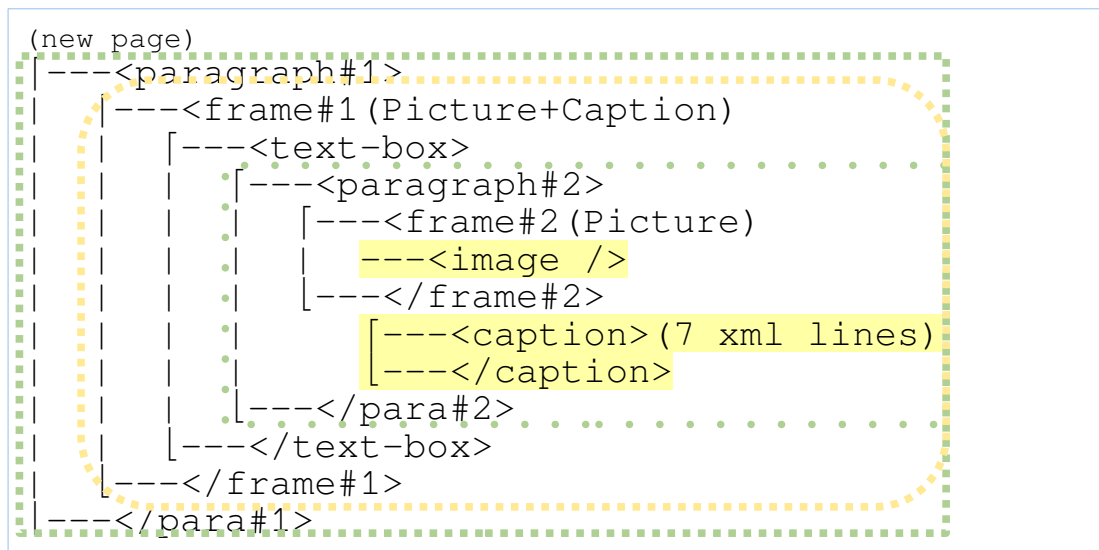
To easily demonstrate this problem (4:)

- i. Make a visual inspection of the final page (p#9) of the examples below.
- ii. Note that the image+caption-frame bottom spacing (which here is set at 0.5 cm) is placed above the 1st line of text below it.
- iii. The text top-border correct location should be between the bottom of the frame & the top of the text border. However, the text has been captured by the image+caption, and the border has been placed above the frame. I believe that this may be a frame error, rather than an image error.

Encapsulations:

1st Page

There now follows a schematic layout of the encapsulations of the 1st Example page (page#6). It contains an Image + Caption only; the schema is revealed by `content.xml` within the ODT. Note that [7-Zip](#) can show + extract files contained within this ODT + other archive files.



There is just an **image** + a **caption** on page#6. *paragraph#1* (p1) contains everything, starting with *frame#1* (fr1) which contains everything else. *frame#1* contains a *text-box* (tb), and the *text-box* contains yet another *paragraph* (p2) and that contains everything else.

The schema layout above became so complicated that I was forced to stop highlighting with colours & designs (it was making it muddier, not clearer).

Both the **image** and **caption** are contained within *paragraph#2* (p2). The **image** is contained within *frame#2* (fr2) and that frame is used to split the **caption** to be either above or (as in this case) below the **image**.

This is how the encapsulations each end up:

image:	p1°fr1°tb°p2°fr2°	image°	fr2°p2°tb°fr1°p1°	[5 encapsulations]
caption:	p1°fr1°tb°p2°	caption°	p2°tb°fr1°p1°	[4 encapsulations]

Last Page + Comment

The 1st line of the (added) Text paragraphs is inserted as ‘bald’ text¹ into the very end of the image+caption (see also [bug02-info.txt](#)). Thus, ‘x = "blue"’ is placed into the content *after* `</frame#1>` and *before* `</para#1>`. That line of text thus becomes an integral component of the entire image+caption. A commenter has stated that this is “by design”. I believe it to be a bug.

raal has commented within the [Bugzilla Report](#) that the bug began at LO v6.5, identified the commit in 2019 that caused it, and asked Michael Stahl to take a look. Stahl is the one that commented “this is working as designed”.

I’ve never seen a clearer set of bugs. As one obvious comment, what on earth is the 1st paragraph of text below a frame doing being incorporated into the frame above it? Why are they not maintained as independent entities?

Examples (next pages):

¹ Other text is entered into the xml together with embedded Style links, but this text is unadorned by such links.



```
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)

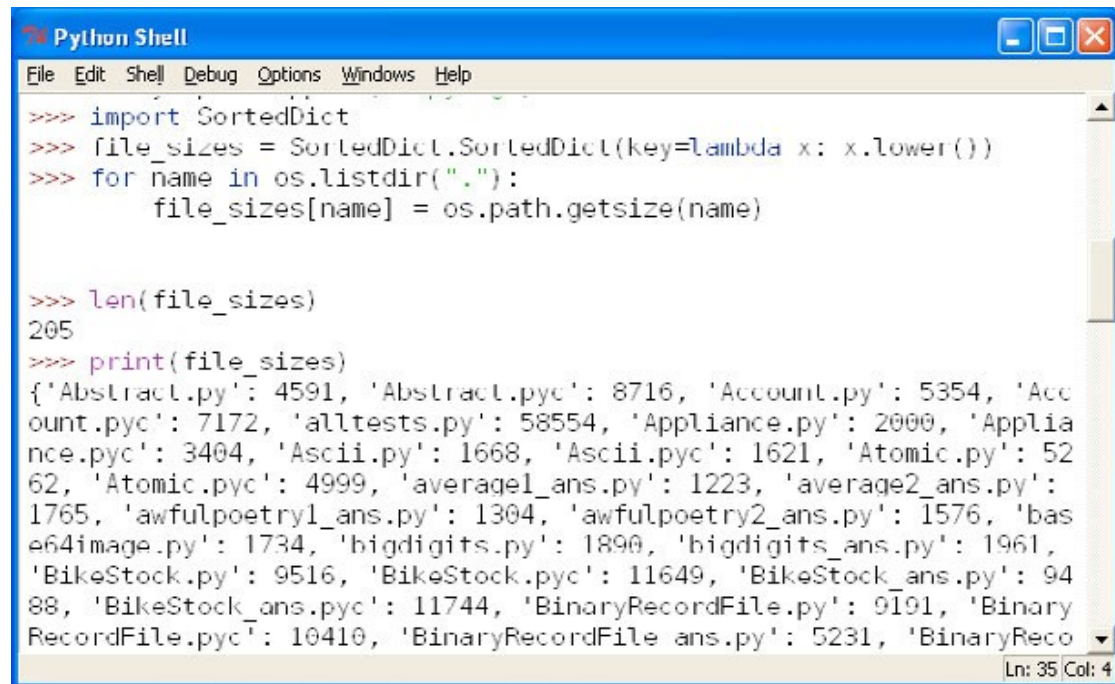
>>> len(file_sizes)
205
>>> print(file_sizes)
{'Abstract.py': 4591, 'Abstract.pyc': 8716, 'Account.py': 5354, 'Account.pyc': 7172, 'alltests.py': 58554, 'Appliance.py': 2000, 'Appliance.pyc': 3404, 'Ascii.py': 1668, 'Ascii.pyc': 1621, 'Atomic.py': 5262, 'Atomic.pyc': 4999, 'averagel_ans.py': 1223, 'average2_ans.py': 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
```

Ln: 35 Col: 4

Figure 2.2: IDLE's Python Shell

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

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



```
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 2.3: IDLE's Python Shell

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

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


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 text area contains the following Python code:

```
>>> import SortedDict
>>> file_sizes = 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
```

The code is color-coded: keywords are in blue, strings in red, and comments in green. The output shows the length of the dictionary is 205 and then prints the dictionary contents. The window status bar at the bottom right shows "Ln: 35 Col: 4".

Figure 2.4: IDLE's Python Shell

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