Frame Borders Wrongly Interfere with Adjacent Borders, Top & Bottom

This document is provided to give a 2nd working copy of the same bug as reported in Python3docs Bugs #07, with extra information supplied.

Comment

The report (below) was made at BugZilla#164304 on Dec 12 2024. The bug was first documented at GitHub Python3Docs Bug#07 and then documented separately as Bug#07+.

a) Version 24.8.3.2 (Build) is affected by this bug
b) The problem begins to show at Grandfather version 3.3.0.4
c) The above means that ALL LibreOffice versions are affected by this bug.

original bug report:

> In Writer, the top border of a Frame will interfere with the bottom border of a Heading. In addition, the bottom border of the same Frame will coalesce with the top inline-border of a paragraph. In other words, the top & bottom borders of the Frame are interfering with both other entity's borders, and not respecting their personal space.

Extra comment:

> The original report is accurate but has attracted zero response other than one blind, ignorant, inaccurate copy/pasted comment.

How to produce this document

1. This document created in Writer under 24.8.3.2.
   `menu:Help | About LibreOffice` shows: Build
2. The entire document bug06+.odt was copied as this doc & then edited.
3. Content on the final page#4 was deleted and replaced with content from the final page of bug07.odt.
4. The Heading 3 at top of page#4 had it's font-size increased to that of a standard Heading 1 (32 pt), to allow the interference from the SideBar to be as obvious as possible.
5. The SideBar (a Frame that contains text) had it's border changed to a dotted +0A2746 line; that was to try to provide a contrast to the Code Box 2 (CB2) paragraph below.
6. The SideBar is offset to the right by 0.5 cm, to allow Frame interference with the Heading's background ("Area") colour to be as visually obvious as possible. Both the Heading + the SideBar's Lower Spacing (and all text paragraphs) are also equal to 0.5 cm.

Demonstration:

Refer to the final page#4 for a practical demonstration of this bug.

Things to note:–

i. to recap:

   a) This bug concerns Frames; in this context that is *“SideBar ‘The print() Function’”;* it is a Frame that contains text.

   b) At issue is the way that no frame since the very first 3.3.0.4 version of LibreOffice Writer up to and including the latest version will behave decently with any other entity that it has a border with.

ii. This frame is set to anchor to a paragraph (same paragraph as the Heading is anchored to).

   a) The frame’s position is set to be at the bottom of that paragraph (it fails to do that).

   b) The Heading paragraph has a 0.5 cm bottom spacing, but the frame fails to honour that.

   c) The problems above are absent if the SideBar is set to anchor to a Character.

   d) The Frame has a 0.5 cm bottom spacing, but that is not honoured (the spacing is placed immediately above the CB2 text, rather than between the Frame & the CB2’s blue-border). In addition, the CB2’s top blue border is missing entirely, presumably behind the Frame.

      This section has already been reported separately as bug05+.odt.

Example

(see next page):

Lambda Functions

The print() Function

The `print()` function accepts any number of positional arguments, and has three keyword arguments, *sep*, *end*, and *file*. All the keyword arguments have defaults. The *sep* parameter's default is a space; if two or more positional arguments are given, each is printed with the *sep* in between, but if there is just one positional argument this parameter does nothing. The *end* parameter's default is `\n`, which is why a newline is printed at the end of calls to `print()`. The file parameter's default is *sys.stdout*, the standard output stream, which is usually the console.

Any of the keyword arguments can be given the values we want instead of using the defaults. For example, *file* can be set to a file object that is open for writing or appending, and both *sep* and *end* can be set to other strings, including the empty string.

If we need to print several items on the same line, one common pattern is to print the items using `print()` calls where *end* is set to a suitable separator, and then at the end to call `print()` with no arguments, since this just prints a newline. For an example, see the `print_digits()` function (180 <).

```
lambda parameters: expression
```