Interference Between Frame Borders, Heading Borders & Paragraph Borders, Top & Bottom

This document with it's snappy title is provided to give a working copy of a bug as reported in Python3docs Bugs #7. This bug is as follows:-

> 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.
>
> *Note*: the 2nd one above has already been reported in bug5.odt.

To produce this document the following steps were taken:

(a) The styles from chapter_01.odt were loaded into this document under LO 24.8.3.2

(b) The Heading "Lambda Functions" was copied from chapter_04.odt at the top of the (next) page. Note that this Heading is set to have a 0.5cm space below the paragraph.

(c) The SideBar *"The print() Function"* was copied from chapter_04.odt onto the next line of the page, and set to be 0.5cm to the right of centre (to more clearly show the interaction between the 1st 2 entities).

(d) The same SideBar is anchored to the paragraph above (the Heading), and is set to fit to the *bottom* of that paragraph. The Frame fails to obey the Heading's 0.5cm bottom border space and, in addition, penetrates into the Heading's coloured background.

(e) A *"Code Box 2"* was copied into the document below the SideBar.

Note:

> The Frame's *Position* is set relative to *"Paragraph Text Area"*. It is unbelievably difficult to get that parameter permanently accepted by the *Property* dialog. It keeps getting switched after <OK> is pressed to *"Entire Paragraph Area"*. I'm well aware of how often I can be a *numpty*, but not as often as that dialog switches that setting. There is another bug there.

Example (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
```