Interference Between Table Borders & Paragraph Borders

This document is provided to give a working copy of a bug as reported in Python3docs Bugs #4. This bug is as follows:-

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

To produce this document the following steps were taken:

(a) The styles from bug3.odt were loaded into this document under LO 24.8.3.2

(b) The table-in-frame *"Frame 3.3: Dictionary Methods Table"* was copied from chapter_03.odt into this document.

(c) A *"Code Box 2"* was copied into the document below the image

Note:

> Almost certainly this is an issue with Frames rather than Tables, and if so will be a copy of Python3docs Bugs #3.

Example (next page):

Table 3.1: Dictionary Methods

Syntax	*Description*
`d.clear()`	Removes all items from `dict d`
`d.copy()`	Returns a shallow copy of `dict d`
`d.fromkeys(s, v)`	Returns a `dict` whose keys are the items in sequence `s` and whose values are `None` or `v` if `v` is given
`d.get(k)`	Returns `key k`'s associated value, or `None` if `k` isn't in `dict d`
`d.get(k, v)`	Returns `key k`'s associated value, or `v` if `k` isn't in `dict d`
`d.items()`	Returns a viewError: Reference source not found of all the `(key, value)` pairs in `dict d`
`d.keys()`	Returns a viewError: Reference source not found of all the keys in `dict d`
`d.pop(k)`	Returns `key k`'s associated value and removes the item whose `key` is `k`, or returns `v` if `k` isn't in `dict d`
`d.popitem()`	Returns and removes an arbitrary `(key, value)` pair from `dict d`, or raises a *KeyError* exception if `d` is `empty`
`d.setdefault(k, v)`	The same as the `dict.get()` method, except that if the `key` is not in `dict d`, a new item is inserted with the `key k`, and with a *value* of `None` or of `v` if `v` is given
`d.update(a)`	Adds every `(key, value)` pair from `a` that isn't in `dict d` to `d`, and for every *key* that is in both `d` and `a`, replaces the corresponding value in `d` with the one in `a`—`a` can be a *dictionary*, an *iterable* of `(key, value)` pairs, or *keyword* arguments
`d.values()`	Returns a viewError: Reference source not found of all the *values* in `dict d`

Shallow and deep copying

146 ➤

```
d = {}.fromkeys("ABCD", 3) # d == {'A': 3, 'B': 3, 'C': 3, 'D': 3}
s = set("ACX")             # s == {'A', 'C', 'X'}
matches = d.keys() & s     # matches == {'A', 'C'}
```