Handling Text Frames

dehyphenateText(...)
```
dehyphenateText(["name"]) -> bool
```

Does dehyphenation on text frame "name". If "name" is not given the currently selected item is used.

May raise WrongFrameTypeError if the target frame is not a text frame

deleteText(...)
```
deleteText(["name"])
```

Deletes any text in the text frame "name". If there is some text selected, only the selected text will be deleted. If "name" is not given the currently selected item is used.

getAllText(...)
```
getAllText(["name"]) -> string
```

Returns the text of the text frame "name" and of all text frames which are linked with this frame. If this textframe has some text selected, the selected text is returned. If "name" is not given the currently selected item is used.

getColumnGap(...)
```
getColumnGap(["name"]) -> float
```

Returns the column gap size of the text frame "name" expressed in points. If "name" is not given the currently selected item is used.

getColumns(...)
```
getColumns(["name"]) -> integer
```

Gets the number of columns of the text frame "name". If "name" is not given the currently selected item is used.

getFont(...)
```
getFont(["name"]) -> string
```

Returns the font name for the text frame "name". If this text frame has some text selected the value assigned to the first character of the selection is returned. If "name" is not given the currently selected item is used.

getFontSize(...)
```
getFontSize(["name"]) -> float
```

Returns the font size in points for the text frame "name". If this text frame has some text selected the value assigned to the first character of the selection is returned. If "name" is not given the currently selected item is used.

getLineSpacing(...)
```
getLineSpacing(["name"]) -> float
```

Returns the line spacing ("leading") of the text frame "name" expressed in points. If "name" is not given the currently selected item is used.

getText(...)
```
getText(["name"]) -> string
```

Returns the text of the text frame "name". If this text frame has some text selected, the selected text is returned. All text in the frame, not just currently visible text, is returned. If "name" is not given the currently selected item is used.

getTextColor(...)
```
getLineColor(["name"]) -> string
```

Returns the name of the line color of the object "name". If "name" is not given the currently selected item is used.

getTextDistances(...)
```
getTextDistances(["name"]) -> tuple
```

Returns the text distances of the text frame "name" expressed in points. The distances are returned as a tuple like (left, right, top, bottom). If "name" is not given the currently selected item is used.

getTextLength(...)
```
getTextLength(["name"]) -> integer
```

Returns the length of the text in the text frame "name". If "name" is not given the currently selected item is used.

getTextLines(...)
```
getTextLines(["name"]) -> integer
```

Returns the number of lines of the text in the text frame "name". If "name" is not given the currently selected item is used.

getTextShade(...)

```
getLineShade(["name"]) -> integer
```

Returns the shading value of the line color of the object "name". If "name" is not given the currently selected item is used.

hyphenateText(...)

```
hyphenateText(["name"]) -> bool
```

Does hyphenation on text frame "name". If "name" is not given the currently selected item is used.

May raise WrongFrameTypeError if the target frame is not a text frame

insertText(...)

```
insertText("text", pos, ["name"])
```

Inserts the text "text" at the position "pos" into the text frame "name". Text must be UTF encoded (see setText() as reference) The first character has an index of 0. Inserting text at position -1 appends it to the frame. If "name" is not given the currently selected Item is used.

May throw IndexError for an insertion out of bounds.

isPDFBookmark(...)

```
isPDFBookmark(["name"]) -> bool
```

Returns true if the text frame "name" is a PDF bookmark. If "name" is not given the currently selected item is used.

May raise WrongFrameTypeError if the target frame is not a text frame

selectText(...)

```
selectText(start, count, ["name"])
```

Selects "count" characters of text in the text frame "name" starting from the character "start". Character counting starts at 0. If "count" is zero, any text selection will be cleared. If "name" is not given the currently selected item is used.

May throw IndexError if the selection is outside the bounds of the text.

setColumns(...)

```
setColumns(nr, ["name"])
```

Sets the number of columns of the text frame "name" to the integer "nr". If "name" is not given the currently selected item is used.

May throw ValueError if number of columns is not at least one.

setColumnGap(...)

```
setColumnGap(size, ["name"])
```

Sets the column gap of the text frame "name" to the value "size". If "name" is not given the currently selected item is used.

May throw ValueError if the column gap is out of bounds (must be positive).

setFont(...)

```
setFont("font", ["name"])
```

Sets the font of the text frame "name" to "font". If there is some text selected only the selected text is changed. If "name" is not given the currently selected item is used.

May throw ValueError if the font cannot be found.

setFontSize(...)

```
setFontSize(size, ["name"])
```

Sets the font size of the text frame "name" to "size". "size" is treated as a value in points. If there is some text selected only the selected text is changed. "size" must be in the range 1 to 512. If "name" is not given the currently selected item is used.

May throw ValueError for a font size that's out of bounds.

setLineSpacing(...)

```
setLineSpacing(size, ["name"])
```

Sets the line spacing ("leading") of the text frame "name" to "size". "size" is a value in points. If "name" is not given the currently selected item is used.

May throw ValueError if the line spacing is out of bounds.

setPDFBookmark(...)

```
setPDFBookmark("toggle", ["name"])
```

Sets wether (toggle = 1) the text frame "name" is a bookmark nor not. If "name" is not given the currently selected item is used.

May raise WrongFrameTypeError if the target frame is not a text frame

setText(...)

```
setText("text", ["name"])
```

Sets the text of the text frame "name" to the text of the string "text". Text must be UTF8 encoded - use e.g. unicode(text, 'iso-8859-2'). See the FAQ for more details. If "name" is not given the currently selected item is used.

setTextAlignment(...)

```
setTextAlignment(align, ["name"])
```

Sets the text alignment of the text frame "name" to the specified alignment. If "name" is not given the currently selected item is used. "align" should be one of the ALIGN_ constants defined in this module - see dir(scribus).

May throw ValueError for an invalid alignment constant.

setTextDistances(...)

```
setTextDistances(left, right, top, bottom, ["name"])
```

Sets the text distances of the text frame "name" to the values "left" "right", "top" and "bottom". If "name" is not given the currently selected item is used.

May throw ValueError if any of the distances are out of bounds (must be positive).

setTextScalingH(...)

```
setTextScalingH(scale, ["name"]))
```

Sets the horizontal character scaling of the object "name" to "scale" in percent. If "name" is not given the currently selected item is used.

setTextScalingV(...)

```
setTextScalingV(scale, ["name"]))
```

Sets the vertical character scaling of the object "name" to "scale" in percent. If "name" is not given the currently selected item is used.

setTextColor(...)

```
setTextColor("color", ["name"])
```

Sets the text color of the text frame "name" to the color "color". If there is some text selected only the selected text is changed. If "name" is not given the currently selected item is used.

setTextShade(...)

```
setTextShade(shade, ["name"])
```

Sets the shading of the text color of the object "name" to "shade". If there is some text selected only the selected text is changed. "shade" must be an integer value in the range from 0 (lightest) to 100 (full color intensity). If "name" is not given the currently selected item is used.

setTextStroke(...)

```
setTextStroke("color", ["name"])
```

Set "color" of the text stroke. If "name" is not given the currently selected item is used.

textOverflows(...)

```
textOverflows(["name", nolinks]) -> integer
```

Returns 1 if there are overflowing characters in text frame "name", 0 if not. If is nolinks set to non zero value it takes only one frame - it doesn't use text frame linking. Without this parameter it search all linking chain.

May raise WrongFrameTypeError if the target frame is not an text frame