Documentation and examples of sectioner.py 1.0 boxerer.py 1.0 and miarticle.cls

Manuel Gutierrez Algaba
irmina@ctv.es
http://www.ctv.es/USERS/irmina/texpython.htm

January 1999

1 Copyright issues

2 Introduction

This document explains all the details for the user of boxerer.py, sectioner.py and miarticle.cls

2.1 boxerer.py

When I wrote boxerer.py there wasn't any automated utility for doing input-output-function boxes. Of course, you can write them directly in LaTeX. But boxerer.py has two major advantages, it's easier to use and it's faster to 'write'. Besides it can be used as an interface by CASE programs. The kinds of available drawings are good for:

- Expressing the internal structure of large pieces of code, resembling their inner and overall structure.
- Expressing the interfaces of modules or functions ,that is, expressing modules and functions as black boxes.

And this is an utility written in python.

`http://www.python.org`

I imagine that it could be written in TeX but It's 3 times easier to use python. And what's more important TeX programmers have a model , if they want to do the translation.
Another point, the python code could be improved ...

And, it generates tex code. So if you want it in postscript, gif or whatever, use the programs dvi, gs or grab directly from a window!

2.2 sectioner.py and miarticle.cls

When I wrote sectioner.py there wasn't any automated utility for handling relative sectioning in LaTeX/ TeX, or I didn't know anything about it. Of course, you can write traditional sections in LaTeX. But sectioner.py has several major advantages:

- It's easier to use because you don't need to know if you are in a section, subsection, . . . , you just go up or go down a level.
- It's more abstract. You don't need to take any decision about the class of the document (article, report, letter, miarticle, . . .), this is done when you filter the tex source through sectioner.py. This , somehow, follows the true spirit of TeX:

  To pay attention to logical design.

- It allows a greater structuration and modularity of LaTeXcode, see details below 3.2.
- You are still compatible with LaTeX, because you can flatten your sources of sectioner-LaTeXinto pure LaTeX(using sectioner.py), and nobody would never know you used it.

And this is an utility written in python.

`http://www.python.org`

I imagine that it could be written in TeX but It's 8 times easier to use python. And what's more important TeX programmers have a model , if they want to do the translation.
Another point, the python code could be improved . . .

3 How to use it

3.1 Using boxerer.py

Always use a "from boxerer import * " at the beginning of your python code.

3.1.1 Using the class 'included'

This piece of python code illustrates the full capabilities of included, basically, you can put lot of structured stuff in it, this is very good, when you program has got very rich in terms of modularity and information. This code

```
r0 = ("men\\_prin.py",None,"Here it's the main menu is defined")
r1 = ("Graphic representation", [("tkinter.py",None,
                                  "Main calling module"),r0],
                                  "Graphic interfases with the user, mainly")
r2 = ("Structures of the languages(grammar)", None)
r3 = ("Configuration", [("Database of words",None),
        ("Labels",None)])
i = included(("Lritaunas Peki",[r1,r2,r3],
```

```
              " This programm aims to teach \
vocabulary and grammar of different languages."))
f = open("result.tex","w")
i.genera(f)
f.close()
```

generates this:

Lritaunas Peki
This programm aims to teach vocabulary and grammar of different languages.

Graphic representation
Graphic interfases with the user, mainly

tkinter.py
Main calling module

men_prin.py
Here it's the main menu is defined

Structures of the languages(grammar)

Configuration

Database of words

Labels

As you notice, what it generates is LaTeXcode, it can be included in a figure, or not, as you please.

3.1.2 Using the class 'inputoutputfunction'

This piece of python code illustrates the full capabilities of inputoutputfunction, basically, you say what are inputs, what are outputs, and what are functions and that's all. The generated code can be inserted directly or it can be put in a figure. If it's included directly ,then it's a good idea to insert a `\newline` after and before the `\input` command, otherwise, it can be messed with the text below it. This code

```
example2=inputoutputfunction('taxman',
"This is your best friend, who helps you when you earn too much")
example2.do_inputs(["Your income","Your properties","The Law"])
example2.do_outputs(["Your taxes","Your fines",
                     "Historical information"])
example2.do_functions(["Compute your taxes","Check your lies",
  "Send you to prison","Bribery"])
example2.generate_latex_code("result2.tex",200)
```

generates fig 1

When you don't want to put its outputs(for example) then you got a figure as fig 2

Besides, if you put

```
example2=inputoutputfunction('taxman',
"This is your best friend, who helps you when you earn too much",
'es')
```

Figure 1: Input-output-function figure

taxman
This is your best friend, who helps you when you earn too much

Inputs

* Your income
* Your properties
* The Law

Functions

* Compute your taxes
* Check your lies
* Send you to prison
* Bribery

Outputs

* Your taxes
* Your fines
* Historical information

Figure 2: Input-output-function figure, without outputs

doctor
This is your best friend, who helps you when you are tired of life

Inputs

* Your income
* Your properties
* Your confidence

Functions

* Kill you
* Heal you

Outputs
None

,then it generates the labels in Spanish.

3.2 Using sectioner.py

This program is a small revolution in the LaTeX–world, I think. Let's take a look at a sectioner-LaTeX code:

```
\documentclass[10pt]{miarticle}
\begin{document}
\n
Lritaunas Peki Documentation
Lritaunas Peki is a programm for learning languages.

\+
Structure of the Code

\+
Graphic user interfaces
\+
Ideas
Basically, you must think about it as a nested layers of metawidgets.
So we can reuse the code.
\n
Python mega widgets
This is a nice library of megawidgets, that supports scrollbars,
multiple entries, dialogs...
\+
My meta widgets.
Trying to isolate GUI from tk and from Pmw(python megawidgets) I
wrote a higher set of widgets.

\+
Specialized metawidgets
This widgets are those really used in the GUI, they're composites
of my meta widgets, and they make calls to the structures of
the core code~\ref{lookupindictionaries}


\-2
\n
What you really see in your screen
Blah, blah

\-2


/minput{strucincode.tex}

\n
And yeah more blah
Blah , and blah

\end{document}
```

Well, as you notice, `\sections` have been replaced by `\n,\+ or \-`. The line below a `\n,\+ or \-` is the title of the (sub)section or whatever. There's no binding with the style of the document, if you replace miarticle by report, then the sectioning of report style will be used. The only thing you musn't forget is that the title of it CAN BE ONLY IN THE NEXT LINE TO THE `\n,\+ or \-`.

The `\+2 or \-3` things say you go two levels up or down. And the `\n` stays at the same level.

And the last thing is `/minput{strucincode.tex}` , this includes more code in sectioner–LaTeXstyle. As a matter of fact, all this story about sectioner is that it's better for modularity, look at this:

```
\n
Structures used

\+
Dictionaries
\+
Functions
\+
Lookups
\label{lookupindictionaries}
This is probably one of the most used flavors.. blah, blah

\-3
```

As you may notice , `\-3` makes that all that module is encapsulated, in terms of sectioning. Depending where you include it , you got sections, subsubsections, or paragraphs, or whatever. And you can use labels, because, sectioner is just a filter, when it finishes its work, you got pure LaTeX, code. Of course you can nest to the level you want (if your style allows it).

3.2.1 How to call to sectioner.py

```
python sectioner.py method inputfile outputfile
```

method can be: manolo (for miarticle.cls), article or report. inputfile would be highnest.tex in this case, and outputfile whatever you want. As a example lets take a look at the output (I wrote o.tex as outputfile):

```
\documentclass[10pt]{miarticle}
\begin{document}
\part{Lritaunas Peki Documentation}
Lritaunas Peki is a programm for learning languages.

\section{Structure of the Code}

\subsection{Graphic user interfaces}
\subsubsection{Ideas}
Basically, you must think about it as a nested layers of metawidgets.
So we can reuse the code.
\subsubsection{Python mega widgets}
This is a nice library of megawidgets, that supports scrollbars,
multiple entries, dialogs...
\sssection{My meta widgets.}
Trying to isolate GUI from tk and from Pmw(python megawidgets) I
wrote a higher set of widgets.

\ssssection{Specialized metawidgets}
This widgets are those really used in the GUI, they're composites
of my meta widgets, and they make calls to the structures of
the core code~\ref{lookupindictionaries}


\subsubsection{What you really see in your screen}
Blah, blah



\section{Structures used}

\subsection{Dictionaries}
\subsubsection{Functions}
\sssection{Lookups}
\label{lookupindictionaries}
This is probably one of the most used flavors.. blah, blah


\section{And yeah more blah}
Blah , and blah

\end{document}
```

3.2.2 About miarticle.cls

It's a good style when you require more than 7 levels of nesting. It has a small bug, it says some warning when compiling. Press Return twice and forget it.

3.3 Sources of help

Well, the best you can do is to take a look at the end of boxerer.py and sectioner.py where you can see how boxerer generates the draws and how sectioner filter. Secondly, you should take a look at the source of this document, that is: less textoolspro.tex

The figures are included using a simple `\input` command.

4 Caveats and bugs

Boxerer has no bugs. Sectioner has no bugs, miarticle has that nasty warning about that number (forget it). Even so, there'll be some bugs.

5 Bye bye

I hope this documentation helps you to use these utilities. It's not difficult and greatly profitable. And if you want to get similar drawings or improve some of them just take a look at the code, once you get accostumed to it , you'll find it quite logical.