Documentation and examples of gentabtex.py 1.2

Manuel Gutierrez Algaba
carmonense@andaluciajunta.es
http://algaba.sdf-eu.org/gentab.cgi

April 2004- January 2006

Contents

1 Copyright issues **1**

2 Requirements **1**

3 Introduction **1**

4 Examples and usage **2**
4.1 First example, no real advantage!: python and spanish !!! 2
4.2 Second example, two advantages 3
4.3 Third example. Using crosses 4
4.4 Filling row by row 6

5 End **7**

1 Copyright issues

2 Requirements

This package requires python to work. `www.python.org`

3 Introduction

High layer tables rendering engine written in python. If :

- Your tables are huge and sparse and you need to put very few values in it, or you must put all except some.
- You experience some difficulty in balancing the load, that is, the ammount of data of each column and test the width of each column until you get a

desirable result. You don't want narrow columns with full of data while some others are too empty and hold very few data.

- You find a bit messy the notation of LaTeX when dealing with tables

Then this package might be for you. First of all you won't ever need to write tables in LaTeX again. You write a python programm that writes LaTeX code. You place a layer between your tables and LaTeX.

4 Examples and usage

In my opinion the best way to learn is to see examples and experience yourself. All examples taken from the end of the very same gentabtex.py.

4.1 First example, no real advantage!: python and spanish !!!

Let's start with a rather simple example. Let's see a broken table, a table with so huge paragraphs that they get out the paper.

```
\begin{tabular}{lllll}
colores & buen gusto & coche & gallumbos & fruta\\
rojo & es dificil conseguir algo que no sobresalte y
quede bien & implica conduccion agresiva &  & tomate\\
azul & los pantalones, camisas, color discreto
 donde los haya & es un color que como tal rara vez se da, es mas
 corriente en los nortes &  & alga\\
verde & si no es fuerte o amerillento & no esta de moda &  & melon\\
amarillo & hortero en la mayoria de las ocasiones & es el
color que mejor se ve &  & melon\\
\end{tabular}
```

Ok, this is not the best way of writing a table, and you can use paragraph of fixed size and many other things. But anyway, it's rendered thus:

colores	buen gusto	coche
rojo	es dificil conseguir algo que no sobresalte y quede bien	implica conduccion agresiva
azul	los pantalones, camisas, color discreto donde los haya	es un color que como tal rara vez s
verde	si no es fuerte o amerillento	no esta de moda
amarillo	hortero en la mayoria de las ocasiones	es el color que mejor se ve

That table can be generated using gentabtex (see the end of the code of gentabtex.py)

```
    g = gentabtex().en("rojo").en("azul").en("verde").en("amarillo"). \
cab(["colores", "buen gusto", "coche", "gallumbos", "fruta"]).otraen(1). \
en(["es dificil conseguir algo que no sobresalte y quede bien",
    "los pantalones, camisas, color discreto donde los haya",
    "si no es fuerte o amerillento",
    "hortero en la mayoria de las ocasiones"]).otraen(2).\
    en(["implica conduccion agresiva",
        "es un color que como tal rara vez se da, es mas corriente en los nortes",
        "no esta de moda",
```

```
        "es el color que mejor se ve"]).\
equis("ultimacolumna", ["tomate", "alga", "melon", "melon"]).\
defaspectocolumna("")

    g.sync()
    print g
```

You can identify each element of the python programm with the latter table, you don't really need to know python or spanish (yes, the instructions are writen spanish).

Basically, that code it says that the heading (cab) is composed of "colores", "buen gusto", ... Moreover, the pairs `otraen(1)` and `en(...)` (anotherin(number) and in(...)) say what to put in each *column*, lastly, you place with `equis` something in the last column (`ultimacolumna`). Besides, you have to sync the table so it's internally rendered and then you can get it using `print g`.

Till now, no gain. Just some strange python code and even more strange in Spanish !

4.2 Second example, two advantages

Now, let's see other example taken from gentabtex.py.
Let's see the four lines:

```
g. lamatrizdispersa. calculapesos()
g.repartepesos(8, 0.3)
g. daaspectautoma()
print g
```

Once matrix have been synced, we can calculate weights (`calculapesos`) on the sparse matrix (`lamatrizdispersa`) and we require it to give us an "automatic aspect" (`daaspectautoma`, literally "give automatic aspect"), with only that we get:

colores	buen gusto	coche	gallumfruta
rojo	es dificil conseguir algo que no sobresalte y quede bien	implica conduccion agresiva	tomate
azul	los pantalones, camisas, color discreto donde los haya	es un color que como tal rara vez se da, es mas corriente en los nortes	alga
verde	si no es fuerte o amerillento	no esta de moda	melon
amarillo	ortero en la mayoria de las ocasiones	es el color que mejor se ve	melon

Which is a *huge* improvement, because:

- You have only specified the total width of the table `repartepesos(8, 0.3)`, *that's 8 cm width* with a minimum column width of 0.3 cm.

- If you have to resize the width, because you use package `multicolumn`, for example, you don't have to worry about your tables, just specify the new width, and they' ll get the best
- If you modify your table, and add a lot of info into one column, your table will balance itself, so no column will hang full of data while the others are almost empty

Still some words would require a bigger column width, perhaps using 0.5 instead of 0.3, or splitting "gallumbos" in "gallum- bos" would help.

4.3 Third example. Using crosses

Some tables are like this:

0	1	2	3	4	5	6
Recorrer carpetas/Ejecutar archivo	x	x	x	x		
Listar carpeta/Leer datos	x	x	x	x	x	
Atributos de lectura	x	x	x	x	x	
Atributos extendidos de lectura	x	x	x	x	x	
Crear archivos/Escribir datos	x	x				x
Crear carpetas/Anexar datos	x	x				x
Atributos de escritura	x	x				x
Atributos extendidos de escritura	x	x			x	
Eliminar subcarpetas y archivos	x					
Eliminar	x	x				
Permisos de lectura	x	x	x	x	x	x
Cambiar permisos	x					
Tomar posesin	x	x				
Sincronizar	x	x	x	x	x	x

1 control total
2 modificar
3 leer y ejecutar
4 listar el contenido de la carpeta
5 lectura
6 escribir

That's full of crosses, and while it's no problem to do it manually, it's a bit dull and error prone when the number of crosses increases.

Look at the code:

```
\begin{tabular}{lllllll}
0 & 1 & 2 & 3 & 4 & 5 & 6\\
Recorrer carpetas/Ejecutar archivo & x & x & x & x & x &  & \\
Listar carpeta/Leer datos & x & x & x & x & x & x & \\
Atributos de lectura & x & x & x & x & x & x & \\
Atributos extendidos de lectura & x & x & x & x & x & x & \\
Crear archivos/Escribir datos & x & x & x &  &  &  & x\\
Crear carpetas/Anexar datos & x & x & x &  &  &  & x\\
Atributos de escritura & x & x & x &  &  &  & x\\
Atributos extendidos de escritura & x & x & x &  &  & x & \\
Eliminar subcarpetas y archivos & x & x &  &  &  &  & \\
Eliminar & x & x & x &  &  &  & \\
```

```
Permisos de lectura & x & x & x & x & x & x\\
Cambiar permisos & x &  &  &  &  & \\
Tomar posesin & x & x &  &  &  & \\
Sincronizar & x & x & x & x & x & x\\
\end{tabular}
```

Just imagine dozen columns !

But if you can choose the columns by their properties, like gentabtex.py does, then:

```
    g2 = gentabtex().en(["Recorrer carpetas/Ejecutar archivo",
      "Listar carpeta/Leer datos",
      "Atributos de lectura", "Atributos extendidos de lectura",
      "Crear archivos/Escribir datos", "Crear carpetas/Anexar datos",
      "Atributos de escritura", "Atributos extendidos de escritura",
      "Eliminar subcarpetas y archivos", "Eliminar",
      "Permisos de lectura", "Cambiar permisos",
      "Tomar posesin", "Sincronizar"]).\
cab([0,1,2,3,4,5,6]). \
equis(1, "toda").equis(2, "todamenos", [('c', 'subcarpeta'),
                                        ('c', 'Cambiar permisos')]).\
 equis("todafila", [('c', 'Permisos de lectura'),
                    ('c', 'Sincronizar')]).\
 equis(3, "desdehasta", (('c',"Recorrer"),('c', "extendidos"))).\
  equis(4, "desdehasta", (('c',"Recorrer"),('c', "extendidos"))).\
 equis(5, "desdehasta", (('c',"Listar"),('c', "extendidos"))).\
 equis(5, "en", [('c',"extendidos de escritura")]).\
 equis(6, "desdehasta", (('c',"Crear archivos"),('c', "Atributos de escritura"))).\
 coletilla("""\\\\
1 control total

2 modificar

3 leer y ejecutar

4 listar el contenido de la carpeta

5 lectura

6 escribir

""").\
 sync()
```

Let's see, the commands:

- The first `en`, `en(["Recorrer carpetas/Ejecut"...])` populates the first column
- The `cab([0,1,2...])` populates the heading

- Now, let's see the first `equis` (equis is the spanish word for cross), `equis(1,"toda")` says that all (toda) the second column will be marked

- The second equis

```
.equis(2, "todamenos", [('c', 'subcarpeta'),
 ('c', 'Cambiar permisos')]).\
```

  says that the *third* column will be marked except (menos) those rows that contain either "subcarpeta" or "Cambiar permisos"

- The equis

```
equis("todafila", [('c', 'Permisos de lectura'),
('c', 'Sincronizar')]).\
```

  says that all the row (fila) which contains "Permisos de lectura" or "Sincronizar" will be marked.

- The equis

```
equis(3, "desdehasta",
(('c',"Recorrer"),('c', "extendidos"))).\
```

  says that in the *fourth* row it will be marked those cells, starting from top, from (desde) any row containing "Recorrer" to (hasta) any row containing "extendidos" and no more will be marked, with this instruction. That's the program starts in row 1, and compares if column 0 has "Recorrer" if not it continues, when it reaches to such column 0 storing "Recorrer" it marks in that row and in the 4th column, and it continues marking row by row at 4th column till it reaches to a row which column 0 is "extendidos" then stops marking.

- Finally the

```
 equis(5, "en", [('c',
"extendidos de escritura")]).\
```

  places a mark (cross) into all the cells whose column is the 5th and whose row has at column0 a text containing "extendidos de escritura".

4.4 Filling row by row

Finally, we'll see how to write a table row by row, with the advantage of self-balancing of columns:

Protocolo	Velocidad nominal	Frecuencia base	Capacidad exigible al cable	Categora de cable requerida
100Base-T4	100 Mbps	12.5 MHz	12.5 MHz	Cat-3
802.12 (VG)	100 Mbps	15 MHz	15 MHz	Cat-3
100Base-TX	100 Mbps	31.25 MHz	80 MHz	Cat-5
FDDI (*)	100 Mbps	31.25 MHz	80 MHz	Cat-5
ATM (**)	155 Mbps	77.5 MHz	100 MHz	Cat-5

which is achieved with:

```
   tabcabtec = gentabtex().cab(["Protocolo", "Velocidad nominal",
     "Frecuencia base", "Capacidad exigible al cable",
"Categora de cable requerida"]).\
   fil([ "10Base-T", "10 Mbps", "10 MHz", "10 MHz", "Cat-3"]).\
   fil(["100Base-T4", "100 Mbps", "12.5 MHz", "12.5 MHz", "Cat-3"]).\
   fil(["802.12 (VG)", "100 Mbps", "15 MHz", "15 MHz", "Cat-3"]).\
   fil(["100Base-TX", "100 Mbps", "31.25 MHz", "80 MHz", "Cat-5"]).\
   fil(["FDDI  (*)", "100 Mbps", "31.25 MHz", "80 MHz", "Cat-5"]).\
   fil(["ATM  (**)", "155 Mbps", "77.5 MHz", "100 MHz", "Cat-5"]).\
   sync().calculapesos().repartepesos(6, 0.5). daaspectautoma()
   print tabcabtec
```

I think it's self explainatory.

5 End

End.