ipy_table Reference

Table Creation

To create a table call make_table on an array (a list of equal sized lists) or a `numpy.ndarray`.

`make_table()` creates a table in interactive mode. Subsequent calls to modify styles (e.g. `apply_theme()`, `set_cell_style()`, etc.) will re-render the table with the new style modifications.

In [1]:
```
from ipy_table import *
example_table = [[i for i in range(j,j+4)] for j in range(0,30,10)]
make_table(example_table)
```

Out[1]:

0	1	2	3
10	11	12	13
20	21	22	23

Built-in Styles

ipy_table implements three pre-defined table styles (basic, basic_left, and basic_both) which provide bold gray headers and alternating colored rows for three different header configurations.

In [2]:
```
make_table(example_table)
apply_theme('basic')
```

Out[2]:

0	**1**	**2**	**3**
10	11	12	13
20	21	22	23

In [3]:
```
make_table(example_table)
apply_theme('basic_left')
```

Out[3]:

0	1	2	3
10	11	12	13
20	21	22	23

In [4]:
```
import copy
example_table2 = copy.deepcopy(example_table)  # Copy the example table
example_table2[0][0] = ''   # Clear the contents of the upper left corner cell
make_table(example_table2)
apply_theme('basic_both')
```

Out[4]:

	1	**2**	**3**
10	11	12	13
20	21	22	23

set_cell_style()

Sets the style of a single cell. For a list of the available style options, see **Syle Options** below.

In [5]:
```
make_table(example_table)
set_cell_style(1, 2, color='red')
```

Out[5]:

0	1	2	3
10	11	12	13
20	21	22	23

set_row_style()

Sets the style for a row of cells. For a list of the available style options, see **Syle Options** below.

In [6]:
```
make_table(example_table)
set_row_style(0, color='lightGreen')
```

Out[6]:

0	1	2	3
10	11	12	13
20	21	22	23

set_column_style()

Sets the style for a column of cells. For a list of the available style options, see **Syle Options** below.

In [7]:
```
make_table(example_table)
set_column_style(1, color='lightBlue')
```

Out[7]:

0	1	2	3
10	11	12	13
20	21	22	23

set_global_style()

Sets the style for all cells. For a list of the available style options, see **Syle Options** below.

In [8]:
```
make_table(example_table)
set_global_style(color='Pink')
```

Out[8]:

0	1	2	3
10	11	12	13
20	21	22	23

Style options

bold

In [9]:
```
make_table(example_table)
set_row_style(1, bold=True)
```

Out[9]:

0	1	2	3
10	**11**	**12**	**13**
20	21	22	23

italic

In [10]:
```
make_table(example_table)
set_row_style(1, italic=True)
```

Out[10]:

0	1	2	3
10	*11*	*12*	*13*
20	21	22	23

color

Sets background cell color by name. The color name can be any any standard web/X11 color name. For a list see http://en.wikipedia.org/wiki/Web_colors

In [11]:
```
make_table(example_table)
set_row_style(1, color='Orange')
```

Out[11]:

0	1	2	3
10	11	12	13
20	21	22	23

thick_border

Accepts a comma delimited list of cell edges, which may be any of: left, top, right, bottom. You can also speify 'all' to include all edges.

In [12]:
```
make_table(example_table)
set_cell_style(0,0, thick_border='left,top')
set_cell_style(2,3, thick_border='right,bottom')
```

Out[12]:

0	1	2	3
10	11	12	13
20	21	22	23

In [13]:
```
make_table(example_table)
set_row_style(1, thick_border='all')
```

Out[13]:

0	1	2	3
10	11	12	13
20	21	22	23

no_border

Accepts a comma delimited list of cell edges, which may be any of: left, top, right, bottom. You can also speify 'all' to include all edges.

In [14]:
```
make_table(example_table)
set_cell_style(0,0, no_border='left,top')
set_cell_style(2,3, no_border='right,bottom')
```

Out[14]:

0	1	2	3
10	11	12	13
20	21	22	23

In [15]:
```
make_table(example_table)
set_row_style(1, no_border='all')
```

Out[15]:

0	1	2	3
10	11	12	13
20	21	22	23

row_span

In [16]:
```
make_table(example_table)
set_cell_style(0, 0, row_span=3)
```

Out[16]:

	1	2	3
0	11	12	13
	21	22	23

column_span

In [17]:
```
make_table(example_table)
set_cell_style(1,1, column_span=3)
```

Out[17]:

0	1	2	3
10	11		
20	21	22	23

width

Sets the cell width in pixels.

In [18]:
```
make_table(example_table)
set_cell_style(0,0, width=100)
```

Out[18]:

0	1	2	3
10	11	12	13
20	21	22	23

align

Sets the cell alignment. Accpets any of: left, right, center.

In [19]:
```
make_table(example_table)
set_cell_style(0, 0, width='100')
set_cell_style(0, 0, align='right')
set_cell_style(1, 0, align='center')
```

Out[19]:

0	1	2	3
10	11	12	13
20	21	22	23

wrap

Turns text wrapping on or off. By default wraping is off.

In [20]:
```
example_table2 = copy.deepcopy(example_table)
example_table2[0][0] = 'This cell has wrap set'
example_table2[0][1] = 'This cell does not have wrap set'
make_table(example_table2)
set_cell_style(0, 0, width=50,wrap=True)
set_cell_style(0, 1, width=50)
```

Out[20]:

This cell has wrap set	This cell does not have wrap set	2	3
10	11	12	13
20	21	22	23

float_format

Sets the display format for floating point values.

The float format string is a standard Python "%" format string (and should contain one and only one %f reference). See http://docs.python.org/2/library/stdtypes.html#string-formatting-operations

The float format only affects cells that contain `float` or `numpy.float64` data types, so you can use set_global_style to set a global floating point format and only those cells containing floating point data will be affected.

The default float format is '%0.4f'.

In [21]:
```
from ipy_table import *
example_table2 = [[i + float(i)/100.0 + i/10000.0 for i in range(j,j+4)] for j in ra
make_table(example_table2)
set_cell_style(0, 0, float_format='%0.1f')
set_cell_style(1, 0, float_format='%0.6f')
set_cell_style(2, 0, float_format='$%0.2f')
```

Out[21]:

0.0	1.0101	2.0202	3.0303
10.101000	11.1111	12.1212	13.1313
$20.20	21.2121	22.2222	23.2323

Class interface

In [22]:
```
t = IpyTable(example_table)
t.set_cell_style(1, 1, color='DarkCyan')
```

```
In [23]: t.render()
```

Out[23]:

0	1	2	3
10	11	12	13
20	21	22	23

Debug mode

Setting `debug=True` will casue ipy_table to display the html source text whenever it renders a table.

```
In [24]: make_table(example_table, debug=True)
```

```
<table border="1" cellpadding="3" cellspacing="0"  style="border:1px solid
black;border-collapse:collapse;"><tr><td>0</td><td>1</td><td>2</td><td>3</td>
</tr><tr><td>10</td><td>11</td><td>12</td><td>13</td></tr><tr><td>20</td>
<td>21</td><td>22</td><td>23</td></tr>
```

Out[24]:

0	1	2	3
10	11	12	13
20	21	22	23

HTML Text

The HTML text representation of the current table can be obtained by calling `get_table_html()`

```
In [25]: get_table_html()
```

```
Out[25]: '<table border="1" cellpadding="3" cellspacing="0"  style="border:1px solid
black;border-collapse:collapse;"><tr><td>0</td><td>1</td><td>2</td><td>3</td>
</tr><tr><td>10</td><td>11</td><td>12</td><td>13</td></tr><tr><td>20</td>
<td>21</td><td>22</td><td>23</td></tr>'
```

Tabulate

Use `tabulate(list, n)` to display a list (not an array) of data in a table with n columns.

In [26]:
```
tabulate(range(20), 6)
```

Out[26]:

0	1	2	3	4	5
6	7	8	9	10	11
12	13	14	15	16	17
18	19				

`tabulate()` creates a table object just like `make_table()`, so the same style operations can be applied.

In [27]:
```
set_cell_style(1, 2, color='yellow')
```

Out[27]:

0	1	2	3	4	5
6	7	8	9	10	11
12	13	14	15	16	17
18	19				

Version

In [28]:
```
import ipy_table as ipt
ipt.__version__
```

Out[28]:
```
1.08
```