IPyTables

March 28, 2015

```
In [1]: from tabipy import Table, TableRow, TableCell, TableHeaderRow
```

This is a package to simply create tables to be displayed in IPython. The tables are rendered as both HTML and LaTeX, so they work both in the browser and if you convert the notebook to LaTeX.

The simplest case is a plain grid:

```
In [2]: Table((4, 1, 8),
              (9, 7, 3),
              (5, 2, 6))
```

Out[2]:

4	1	8
9	7	3
5	2	6

You can add a header row like this:

```
In [3]: Table(TableHeaderRow('a','b','c'),
              (1,  2,  3),
              (2,  4,  6),
             )
```

Out[3]:

a	**b**	**c**
1	2	3
2	4	6

`Table` also accepts `dicts` (or any mapping) with keys as column headers and values as column contents. The order of columns is undefined unless the mapping is an `OrderedDict`.

```
In [4]: Table({'a': (1, 2),
               'b': (2, 4),
               'c': (3, 6)})
```

Out[4]:

a	**b**	**c**
1	2	3
2	4	6

The number of column values (rows) need not be equal:

```
In [5]: Table({'a': (1, 2),
               'b': (2,),
               'c': (3, 6)})
```

Out[5]:

a	**b**	**c**
1	2	3
2		6

You can build a table incrementally using `Table.append_row()`. If you need it, rows also have an `append_cell()` method.

```
In [6]: # Computing values
        t = Table(TableHeaderRow('number', 'square', 'cube'))
        for x in range(1, 11):
            t.append_row((x, x**2, x**3))
        t
```

Out[6]:

number	**square**	**cube**
1	1	1
2	4	8
3	9	27
4	16	64
5	25	125
6	36	216
7	49	343
8	64	512
9	81	729
10	100	1000

You can style cells with the `bg_colour` and `text_colour` parameters. This only works in HTML for the moment; if you convert the notebook to LaTeX, the colours will be ignored.

```
In [7]: # Styling determined by code
        t = Table(TableHeaderRow('divisions', 'result'))
        num = 55
        for x in range(7):
            if num < 1:
                resultcell = TableCell(num, bg_colour='DarkBlue', text_colour='white')
            else:
                resultcell = TableCell(num)
            t.append_row((x, resultcell))
            num /= 3
        t
```

Out[7]:

divisions	**result**
0	55
1	18.333333333333332
2	6.111111111111111
3	2.0370370370370368
4	0.6790123456790123
5	0.22633744855967075
6	0.07544581618655692