PyFlot Documentation

Release 0.1

André da Palma

June 18, 2013

CONTENTS

1 Introduction **3**
1.1 Motivation 3

2 Installation **5**

3 Usage **7**
3.1 Examples 7

4 Object Reference **9**
4.1 Series 9
4.2 Graph 10

5 Integrating with Django **11**

6 Indices and tables **13**

PyFlot provides an interface from Python to flot.

```
import flot
class Fx(flot.Series):
    data = [(1, 2), (2, 3), (3, 4), (4, 5)]

class MyGraph(flot.Graph):
    fx = Fx()

my_graph = MyGraph()
```

```
<html>
    <head>
        <script type="text/javascript" src='static/js/jquery.js'></script>
        <script type="text/javascript" src='static/js/jquery.flot.js'></script>
    </head>
    <body>
        <div id="gr" style="width:600px;height:300px;"></div>
    <script type='text/javascript'>
        $.plot($("#gr"), {{ my_graph.json_data|safe }}, {{ my_graph.options|safe }});
    </script>
    </body>
</html>
```

Contents:

CHAPTER

ONE

INTRODUCTION

1.1 Motivation

Backend developers sometimes have to work with statistics and metrics and bring that data from a server to a web page in form of charts.

When this situations happens developers most times find themselves thinking on things like, which data format should they push from the server to the pages, how to catch the data in the templates and how to plot the charts, sometimes situations even gets worst when in a team there is a clear lack of javascript skills to create those fancy graph pages.

PyFlot, appears as a shortcut and as a solution for all this providing a very nice a simple way to plot graphs just by modeling it in the backend.

CHAPTER

TWO

INSTALLATION

pyFlot is on pypi index so you can simply

```
$ pip install pyflot
```

You can also download and install from the source

```
$ wget http://pypi.python.org/packages/source/P/PyFlot/PyFlot-0.1.tar.gz
$ tar xvfz PyFlot-0.1.tar.gz
$ cd PyFlot-0.1
$ python setup.py install
```

CHAPTER

THREE

USAGE

There are several ways to create and plot graphs with *PyFlot*.

3.1 Examples

Basic Examples

```
import flot

class MySeries(flot.Series):
    data = [(1,2),(2,5),(3,7),(4,9),]

class MyGraph(flot.Graph):
    my_series = MySeries()

my_graph = MyGraph()
print my_graph.json_data
print my_graph.options
```

Its also possible to create `Series` objects by instatiating it inline

```
import flot

series = flot.Series(data=[(1,2),(2,5),(3,7),(4,9)])

class MyGraph(flot.Graph):
    series = series

my_graph = MyGraph()
print my_graph.json_data
print my_graph.options
```

You can also create `Graph` objects inline

```
import flot

series = flot.Series(data=[(1,2),(2,5),(3,7),(4,9)])

my_graph = flot.Graph([series,])
print my_graph.json_data
print my_graph.options
```

A `Graph` object is may contain several `Series` objects, this way is possible on a single graph to plot more than one series. The next exmples shows how to

Inline example:

```
import flot

series_a = flot.Series(data=[(1,2),(2,5),(3,7),(4,9)])
series_b = flot.Series(data=[(4,5),(6,8),(1,4),(2,8)])
my_graph = flot.Graph([series_a,series_b])
print my_graph.json_data
print my_graph.options
```

Class declaration example:

```
import flot

class FirstSeries(flot.Series):
    data = [(1,2),(2,3),(3,4)]

class SecondSeries(flot.Series):
    data = [(10,20),(20,30),(30,40)]


class MyGraph(flot.Graph):
    first_series = FirstSeries()
    second_series = SecondSeries()


my_graph = MyGraph()
print my_graph.json_data
print my_graph.options
```

CHAPTER

FOUR

OBJECT REFERENCE

The main objects on PyFlot

4.1 Series

Series(data=None, xpoints=None, ypoints=None, options=None,**kwargs)

```
class Series(dict):
    data = []
    _options = SeriesOptions()
```

- **data** **data** argument ca be used to directly pass the data values to the Series.

```
import flot
flot.Series(data=zip(range(0,5), range(0,5)))
```

- **xpoints, ypoints** **xpoints** and **ypoints** can be used to pass directly the points on the x and y axis. Keep in mind that that they both need to be the same size.

```
import flot
flot.Series(xpoints=range(0, 5), ypoints=range(5,10))
```

- **options** **options** its what should be used to series costumization. Things like label or color can be configured using just by passning a `SeriesOptions` object on this argument.

```
import flot
flot.Series(xpoints=range(0, 5), ypoints=range(0,5),
            options=flot.SeriesOptions(label='y=x', color='red'))
```

If the series is being created by class definition, the options are set by the series *Meta* class attribute.

```
import flot
class Series(flot.Series):
    data = zip(range(0, 5),range(0, 5))

    class Meta:
        label = 'y=x'
        color = 'blue'
```

4.2 Graph

This is a graph object The graph object its meant to group one or more series objects so that it can be displayed on the page.

Assuming we have series such as:

```
import flot
series_a = flot.Series(data=[(1, 1), (2, 2) , (3, 3), (4, 4)])
series_b = flot.Series(data=[(1, 2), (2, 3) , (3, 5), (4, 7)])
```

The series can be grouped on a Graph object by:

```
flot.Graph(series1=series_a, series2=series_b)
```

or

```
flot.Graph([series_a, series_b])
```

- **options** **options** its what should be used to graph display costumization. `GraphOptions` object.

CHAPTER

FIVE

INTEGRATING WITH DJANGO

Since Django is currently the most famous Python web framework, this chapter explains how to plot graphs using pyFlot template tags.

CHAPTER

SIX

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*