

## My Project

Generated by Doxygen 1.8.12



# Contents

<b>1</b>	<b>Namespace Index</b>	<b>1</b>
1.1	Packages . . . . .	1
<b>2</b>	<b>Class Index</b>	<b>3</b>
2.1	Class List . . . . .	3
<b>3</b>	<b>File Index</b>	<b>5</b>
3.1	File List . . . . .	5
<b>4</b>	<b>Namespace Documentation</b>	<b>7</b>
4.1	Axes Namespace Reference . . . . .	7
4.1.1	Function Documentation . . . . .	7
4.1.1.1	get_axes() . . . . .	7
4.1.2	Variable Documentation . . . . .	8
4.1.2.1	x_coordinates . . . . .	8
4.1.2.2	y_coordinates . . . . .	8
4.2	Graph Namespace Reference . . . . .	8
4.2.1	Function Documentation . . . . .	8
4.2.1.1	coord() . . . . .	8
4.2.2	Variable Documentation . . . . .	8
4.2.2.1	Graph . . . . .	8
4.3	InputParser Namespace Reference . . . . .	9
4.3.1	Function Documentation . . . . .	9
4.3.1.1	checktype() . . . . .	9
4.3.1.2	clean_data() . . . . .	9
4.4	Scale Namespace Reference . . . . .	9
4.4.1	Function Documentation . . . . .	10
4.4.1.1	get_scale() . . . . .	10

<b>5</b>	<b>Class Documentation</b>	<b>11</b>
5.1	Graph.Graph Class Reference	11
5.1.1	Detailed Description	12
5.1.2	Constructor & Destructor Documentation	12
5.1.2.1	__init__()	12
5.1.3	Member Function Documentation	12
5.1.3.1	plot_function()	12
5.1.3.2	plot_point()	12
5.1.3.3	plot_points()	13
5.1.3.4	plot_points_with_line()	13
5.1.4	Member Data Documentation	14
5.1.4.1	data	14
5.1.4.2	dx	14
5.1.4.3	dy	14
5.1.4.4	graph	14
5.1.4.5	graph_height	14
5.1.4.6	graph_width	14
5.1.4.7	markings	14
5.1.4.8	master	14
5.1.4.9	scale	14
5.1.4.10	scale_x	14
5.1.4.11	scale_y	15
5.1.4.12	x_offset	15
5.1.4.13	y_offset	15
<b>6</b>	<b>File Documentation</b>	<b>17</b>
6.1	C:/Users/sarth/Desktop/All code/Axes.py File Reference	17
6.2	C:/Users/sarth/Desktop/All code/Graph.py File Reference	17
6.3	C:/Users/sarth/Desktop/All code/InputParser.py File Reference	18
6.4	C:/Users/sarth/Desktop/All code/Scale.py File Reference	18
	<b>Index</b>	<b>19</b>

# Chapter 1

## Namespace Index

### 1.1 Packages

Here are the packages with brief descriptions (if available):

<a href="#">Axes</a>	7
<a href="#">Graph</a>	8
<a href="#">InputParser</a>	9
<a href="#">Scale</a>	9



## Chapter 2

# Class Index

### 2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

#### [Graph.Graph](#)

XyGraph constructing class

Author: Hatim Rehman

This class represents a graph object that can plot points with data contained within a list of tuples

[ (x1, y1), (x2, y2), ... , (xn, yn) ] . . . . . [11](#)





## Chapter 3

# File Index

### 3.1 File List

Here is a list of all files with brief descriptions:

C:/Users/sarth/Desktop/All code/ <a href="#">Axes.py</a> . . . . .	17
C:/Users/sarth/Desktop/All code/ <a href="#">Graph.py</a> . . . . .	17
C:/Users/sarth/Desktop/All code/ <a href="#">InputParser.py</a> . . . . .	18
C:/Users/sarth/Desktop/All code/ <a href="#">Scale.py</a> . . . . .	18



## Chapter 4

# Namespace Documentation

### 4.1 Axes Namespace Reference

#### Functions

- def `get_axes` (window, markings, scale\_x, scale\_y)  
*On a Canvas object, constructs X and Y axes appropriately placed.*

#### Variables

- dictionary `x_coordinates` = { }
- dictionary `y_coordinates` = { }

#### 4.1.1 Function Documentation

##### 4.1.1.1 `get_axes()`

```
def Axes.get_axes (
    window,
    markings,
    scale_x,
    scale_y )
```

On a Canvas object, constructs X and Y axes appropriately placed.

Returns a new canvas with this done. Vertical line: Start from the middle with respect to x, draw a line from top to bottom with respect to y. Horizontal line: Start from the middle with respect to y, draw a line from left to right with respect to x. Then calls `_get_labels()` to add the labels to the two lines.

#### Parameters

<i>window</i>	The window to draw on.
<i>markings</i>	The number of labels to put on each axis.
<i>scale_x</i>	The scale value to use on the x axis.
<i>scale_y</i>	The scale value to use on the y axis.

## 4.1.2 Variable Documentation

### 4.1.2.1 x\_coordinates

dictionary Axes.x\_coordinates = { }

### 4.1.2.2 y\_coordinates

dictionary Axes.y\_coordinates = { }

## 4.2 Graph Namespace Reference

### Classes

- class [Graph](#)  
*xyGraph constructing class*  
*Author: Hatim Rehman*  
*This class represents a graph object that can plot points with data contained within a list of tuples*  
*[ (x1, y1), (x2, y2), ... , (xn, yn) ]*

### Functions

- def [coord](#) (x, y)

### Variables

- [Graph](#) = [Graph](#)( 6 , [ (-4,4), (-2,1), (-1,1) , (1,1), (2,2), (4,5), (5,3) ])

## 4.2.1 Function Documentation

### 4.2.1.1 coord()

```
def Graph.coord (
    x,
    y )
```

## 4.2.2 Variable Documentation

### 4.2.2.1 Graph

[Graph.Graph](#) = [Graph](#)( 6 , [ (-4,4), (-2,1), (-1,1) , (1,1), (2,2), (4,5), (5,3) ])

## 4.3 InputParser Namespace Reference

### Functions

- def `clean_data` (data)  
*This function breaks apart the user input and assigns the coordinate pairs to a dictionary (data structure).*
- def `checktype` (x, y)  
*This function checks if the data input type is correct(i.e.*

### 4.3.1 Function Documentation

#### 4.3.1.1 `checktype()`

```
def InputParser.checktype (  
    x,  
    y )
```

This function checks if the data input type is correct(i.e.

integers or floats). If data type is not correct function outputs an exception message.

Author: Sarthak Desai

#### Parameters

<code>x</code>	is a list of x values from all the co-ordinate pairs.
<code>y</code>	is a list of y values from all the co-ordinate pairs.

#### 4.3.1.2 `clean_data()`

```
def InputParser.clean_data (  
    data )
```

This function breaks apart the user input and assigns the coordinate pairs to a dictionary (data structure).

If there are inconsistencies in the user input (for example a missing y value)the function prints an exception message.

Author: Sarthak Desai

#### Parameters

<code>data</code>	is the list of co-ordinate pairs inputted by the user.
-------------------	--

## 4.4 Scale Namespace Reference

## Functions

- def `get_scale` (data\_set, round\_to=0)

*get\_scale*

*Author: Louis Bursey*

*Function to get the appropriate max and mins for the scale of the graph*

*Use round\_to to create padding for the graph, ie if the graph goes in increments of 5, set round\_to to 5 to keep the graph neat.*

### 4.4.1 Function Documentation

#### 4.4.1.1 `get_scale()`

```
def Scale.get_scale (
    data_set,
    round_to = 0 )
```

`get_scale`

Author: Louis Bursey

Function to get the appropriate max and mins for the scale of the graph

Use round\_to to create padding for the graph, ie if the graph goes in increments of 5, set round\_to to 5 to keep the graph neat.

Returns a tuple in the format [x,y]

#### Parameters

<i>data_set,the</i>	set of data being graphed, in a (x,y) dictionary
<i>round_to,an</i>	optional parameter to make the scale a multiple of round_to

## Chapter 5

# Class Documentation

### 5.1 Graph.Graph Class Reference

xyGraph constructing class

Author: Hatim Rehman

This class represents a graph object that can plot points with data contained within a list of tuples

[ (x1, y1), (x2, y2), ... , (xn, yn) ]

#### Public Member Functions

- `def __init__ (self, n, data=None)`  
*The constructor method Takes in 3 parameters.*
- `def plot_point (self, coord, kwargs)`  
*Plot points method Takes in a coordinate to plot, with keyword args that may be used to style the coordinate.*
- `def plot_points (self, data=None, kwargs)`  
*Plot points method takes in multiple coordinates and calls the `plot_point()` method from its own API on each method.*
- `def plot_points_with_line (self, data, kwargs)`  
*Plot points with line method takes in multiple coordinates and calls the `plot_point()` method from its own API on each method.*
- `def plot_function (self, func, x_interval=None, kwargs)`  
*Plots a python function onto the graph Outputs a graph with the func parameter plotted.*

#### Public Attributes

- `data`
- `markings`
- `scale`
- `scale_x`
- `scale_y`
- `master`
- `graph`
- `graph_height`
- `graph_width`
- `x_offset`
- `y_offset`
- `dx`
- `dy`

### 5.1.1 Detailed Description

xyGraph constructing class

Author: Hatim Rehman

This class represents a graph object that can plot points with data contained within a list of tuples

[ (x1, y1), (x2, y2), ... , (xn, yn) ]

### 5.1.2 Constructor & Destructor Documentation

#### 5.1.2.1 \_\_init\_\_()

```
def Graph.Graph.__init__ (
    self,
    n,
    data = None )
```

The constructor method Takes in 3 parameters.

Outputs a graph with plotted data that is entered as a parameter.

##### Parameters

<i>self</i>	The object pointer.
<i>n</i>	The number of markings to appear on the x and y axes
<i>data</i>	The data to plot

### 5.1.3 Member Function Documentation

#### 5.1.3.1 plot\_function()

```
def Graph.Graph.plot_function (
    self,
    func,
    x_interval = None,
    kwargs )
```

Plots a python function onto the graph Outputs a graph with the func parameter plotted.

##### Parameters

<i>self</i>	The object pointer.
<i>func</i>	A python function that takes in a double value and returns a double value
<i>x_interval</i>	A list of double values that the function should pass through [ x1, ... , xn ]
<i>kwargs</i>	Keyword arguments for Tkinter's create_circle() method

#### 5.1.3.2 plot\_point()

```
def Graph.Graph.plot_point (
```



```

    self,
    coord,
    kwargs )

```

Plot points method Takes in a coordinate to plot, with keyword args that may be used to style the coordinate.

Outputs a graph with the coord parameter plotted.

#### Parameters

<i>self</i>	The object pointer.
<i>coord</i>	A dictionary with the format { 'x': <i>value</i> , 'y': <i>value</i> }
<i>kwargs</i>	Keyword arguments for Tkinter's create_circle() method

#### 5.1.3.3 plot\_points()

```

def Graph.Graph.plot_points (
    self,
    data = None,
    kwargs )

```

Plot points method takes in multiple coordinates and calls the [plot\\_point\(\)](#) method from its own API on each method.

Outputs a graph with the data parameter plotted.

#### Parameters

<i>self</i>	The object pointer.
<i>data</i>	A list of tuples [ (x1, y1), (x2, y2), ... , (xn, yn) ]
<i>kwargs</i>	Keyword arguments for Tkinter's create_circle() method

#### 5.1.3.4 plot\_points\_with\_line()

```

def Graph.Graph.plot_points_with_line (
    self,
    data,
    kwargs )

```

Plot points with line method takes in multiple coordinates and calls the [plot\\_point\(\)](#) method from its own API on each method.

Then calls its private function generating method that creates a polynomial that passes through each point using the Lagrange polynomial interpolation theorem. Outputs a graph with the data parameter plotted.

#### Parameters

<i>self</i>	The object pointer.
<i>data</i>	A list of tuples [ (x1, y1), (x2, y2), ... , (xn, yn) ]
<i>kwargs</i>	Keyword arguments for Tkinter's create_circle() method

## 5.1.4 Member Data Documentation

### 5.1.4.1 data

`Graph.Graph.data`

### 5.1.4.2 dx

`Graph.Graph.dx`

### 5.1.4.3 dy

`Graph.Graph.dy`

### 5.1.4.4 graph

`Graph.Graph.graph`

### 5.1.4.5 graph\_height

`Graph.Graph.graph_height`

### 5.1.4.6 graph\_width

`Graph.Graph.graph_width`

### 5.1.4.7 markings

`Graph.Graph.markings`

### 5.1.4.8 master

`Graph.Graph.master`

### 5.1.4.9 scale

`Graph.Graph.scale`

### 5.1.4.10 scale\_x

`Graph.Graph.scale_x`

#### 5.1.4.11 `scale_y`

`Graph.Graph.scale_y`

#### 5.1.4.12 `x_offset`

`Graph.Graph.x_offset`

#### 5.1.4.13 `y_offset`

`Graph.Graph.y_offset`

The documentation for this class was generated from the following file:

- `C:/Users/sarth/Desktop/All code/Graph.py`



## Chapter 6

# File Documentation

### 6.1 C:/Users/sarth/Desktop/All code/Axes.py File Reference

#### Namespaces

- [Axes](#)

#### Functions

- def [Axes.get\\_axes](#) (window, markings, scale\_x, scale\_y)  
*On a Canvas object, constructs X and Y axes appropriately placed.*

#### Variables

- dictionary [Axes.x\\_coordinates](#) = { }
- dictionary [Axes.y\\_coordinates](#) = { }

### 6.2 C:/Users/sarth/Desktop/All code/Graph.py File Reference

#### Classes

- class [Graph.Graph](#)  
*xyGraph constructing class*  
*Author: Hatim Rehman*  
*This class represents a graph object that can plot points with data contained within a list of tuples*  
*[ (x1, y1), (x2, y2), ... , (xn, yn) ]*

#### Namespaces

- [Graph](#)

## Functions

- def [Graph.coord](#) (x, y)

## Variables

- [Graph.Graph](#) = Graph( 6 , [ (-4,4), (-2,1), (-1,1) , (1,1), (2,2), (4,5), (5,3) ])

## 6.3 C:/Users/sarth/Desktop/All code/InputParser.py File Reference

### Namespaces

- [InputParser](#)

### Functions

- def [InputParser.clean\\_data](#) (data)  
*This function breaks apart the user input and assigns the coordinate pairs to a dictionary (data structure).*
- def [InputParser.checktype](#) (x, y)  
*This function checks if the data input type is correct(i.e.*

## 6.4 C:/Users/sarth/Desktop/All code/Scale.py File Reference

### Namespaces

- [Scale](#)

### Functions

- def [Scale.get\\_scale](#) (data\_set, round\_to=0)  
*get\_scale*  
*Author: Louis Bursey*  
*Function to get the appropriate max and mins for the scale of the graph*  
*Use round\_to to create padding for the graph, ie if the graph goes in increments of 5, set round\_to to 5 to keep the graph neat.*

# Index

- `__init__`
    - `Graph::Graph`, 12
- Axes, 7
  - `get_axes`, 7
  - `x_coordinates`, 8
  - `y_coordinates`, 8
- `C:/Users/sarth/Desktop/All code/Axes.py`, 17
- `C:/Users/sarth/Desktop/All code/Graph.py`, 17
- `C:/Users/sarth/Desktop/All code/InputParser.py`, 18
- `C:/Users/sarth/Desktop/All code/Scale.py`, 18
- `checktype`
  - `InputParser`, 9
- `clean_data`
  - `InputParser`, 9
- `coord`
  - `Graph`, 8
- `data`
  - `Graph::Graph`, 14
- `dx`
  - `Graph::Graph`, 14
- `dy`
  - `Graph::Graph`, 14
- `get_axes`
  - Axes, 7
- `get_scale`
  - Scale, 10
- `Graph`, 8
  - `coord`, 8
  - `Graph`, 8
- `graph`
  - `Graph::Graph`, 14
- `Graph.Graph`, 11
- `Graph::Graph`
  - `__init__`, 12
  - `data`, 14
  - `dx`, 14
  - `dy`, 14
  - `graph`, 14
  - `graph_height`, 14
  - `graph_width`, 14
  - `markings`, 14
  - `master`, 14
  - `plot_function`, 12
  - `plot_point`, 12
  - `plot_points`, 13
  - `plot_points_with_line`, 13
- `scale`, 14
- `scale_x`, 14
- `scale_y`, 14
- `x_offset`, 15
- `y_offset`, 15
- `graph_height`
  - `Graph::Graph`, 14
- `graph_width`
  - `Graph::Graph`, 14
- `InputParser`, 9
  - `checktype`, 9
  - `clean_data`, 9
- `markings`
  - `Graph::Graph`, 14
- `master`
  - `Graph::Graph`, 14
- `plot_function`
  - `Graph::Graph`, 12
- `plot_point`
  - `Graph::Graph`, 12
- `plot_points`
  - `Graph::Graph`, 13
- `plot_points_with_line`
  - `Graph::Graph`, 13
- Scale, 9
  - `get_scale`, 10
- `scale`
  - `Graph::Graph`, 14
- `scale_x`
  - `Graph::Graph`, 14
- `scale_y`
  - `Graph::Graph`, 14
- `x_coordinates`
  - Axes, 8
- `x_offset`
  - `Graph::Graph`, 15
- `y_coordinates`
  - Axes, 8
- `y_offset`
  - `Graph::Graph`, 15