## Contents

1 Quick Guide ................................................. 3  
2 API Reference .............................................. 5  
   2.1 API Reference ........................................... 5  
       2.1.1 GraphViz Module ................................. 5  
       2.1.2 Parser Module ................................... 12  
3 Installation .................................................. 15  
4 Contribute ................................................... 17  
5 License ....................................................... 19  
Python Module Index ......................................... 21
PyDotPlus is an improved version of the old pydot project that provides a Python Interface to Graphviz’s Dot language.
CHAPTER 1

Quick Guide

**Warning:** Please Write.
GraphViz Module

Graphviz’s dot language Python interface.
This module provides with a full interface to create handle modify and process graphs in Graphviz’s dot language.

```python
class pydotplus.graphviz.Cluster(graph_name='subG', obj_dict=None, suppress_disconnected=False, simplify=False, **attrs)
```

Class representing a cluster in Graphviz’s dot language.

This class implements the methods to work on a representation of a cluster in Graphviz’s dot language.

```python
def cluster( graph_name='subG', suppress_disconnected=False, attribute=value, ... )
```

- **graph_name**: the cluster’s name (the string ‘cluster’ will be always prepended)
- **suppress_disconnected**: defaults to false, which will remove from the cluster any disconnected nodes.

All the attributes defined in the Graphviz dot language should be supported.
Attributes can be set through the dynamically generated methods:

```python
set_[attribute name], i.e. set_color, set_fontname
```

or using the instance’s attributes:
Cluster.obj_dict['attributes'][attribute name], i.e.
    cluster_instance.obj_dict['attributes']['label'] cluster_instance.obj_dict['attributes']['fontname']

class pydotplus.graphviz.Common
    Common information to several classes.
    Should not be directly used, several classes are derived from this one.

create_attribute_methods (obj_attributes)

get (name)
    Get an attribute value by name.
    Given an attribute ‘name’ it will get its value. There’s always the possibility of using the methods:
        get_'name'()
    which are defined for all the existing attributes.

get_attributes ()

get_parent_graph ()

get_sequence ()

set (name, value)
    Set an attribute value by name.
    Given an attribute ‘name’ it will set its value to ‘value’. There’s always the possibility of using the methods:
        set_'name'(value)
    which are defined for all the existing attributes.

set_parent_graph (parent_graph)

set_sequence (seq)

class pydotplus.graphviz.Dot (*argsl, **argsd)
    A container for handling a dot language file.
    This class implements methods to write and process a dot language file. It is a derived class of the base class ‘Graph’.

create (prog=None, format='ps')
    Creates and returns a Postscript representation of the graph.
    create will write the graph to a temporary dot file and process it with the program given by ‘prog’ (which
    defaults to ‘twopi’), reading the Postscript output and returning it as a string is the operation is successful.
    On failure None is returned.
    There’s also the preferred possibility of using:
        create_‘format’(prog='program')
    which are automatically defined for all the supported formats. [create_ps(), create_gif(), create_dia(), ...]
    If ‘prog’ is a list instead of a string the fist item is expected to be the program name, followed by any
    optional command-line arguments for it:
        ['twopi', '-Tdot', '-s10']

set_graphviz_executables (paths)
    This method allows to manually specify the location of the GraphViz executables.
    The argument to this method should be a dictionary where the keys are as follows:
and the values are the paths to the corresponding executable, including the name of the executable itself.

```python
set_prog (prog)
Sets the default program.
Sets the default program in charge of processing the dot file into a graph.
```

```python
set_shape_files (file_paths)
Add the paths of the required image files.
If the graph needs graphic objects to be used as shapes or otherwise those need to be in the same folder as the graph is going to be rendered from. Alternatively the absolute path to the files can be specified when including the graphics in the graph.
The files in the location pointed to by the path(s) specified as arguments to this method will be copied to the same temporary location where the graph is going to be rendered.
```

```python
write (path, prog=None, format='raw')
Given a filename ‘path’ it will open/create and truncate such file and write on it a representation of the graph defined by the dot object and in the format specified by ‘format’. ‘path’ can also be an open file-like object, such as a StringIO instance.
The format ‘raw’ is used to dump the string representation of the Dot object, without further processing. The output can be processed by any of graphviz tools, defined in ‘prog’, which defaults to ‘dot’ Returns True or False according to the success of the write operation.
There’s also the preferred possibility of using:
```
```python
write_ ‘format’ (path, prog='program')
```which are automatically defined for all the supported formats. [write_ps(), write_gif(), write_dia(), ...]
```

```python
class pydotplus.graphviz.Edge (src='', dst='', obj_dict=None, **attrs)
A graph edge.
This class represents a graph’s edge with all its attributes.
edge(src, dst, attribute=value, ...)
src: source node’s name dst: destination node’s name
All the attributes defined in the Graphviz dot language should be supported.
Attributes can be set through the dynamically generated methods:
```
```python
set_[attribute name], i.e. set_label, set_fontname
or directly by using the instance’s special dictionary:
```
```python
Edge.obj_dict['attributes'][attribute name], i.e.
```
```python
edge_instance.obj_dict['attributes']['label'] edge_instance.obj_dict['attributes']['fontname']
```
```
```python
get_destination ()
Get the edge’s destination node name.
```
```python
get_source ()
Get the edges source node name.
```
```python
parse_node_ref (node_str)
```
```python
to_string ()
Returns a string representation of the edge in dot language.
```
exception pydotplus.graphviz.Error(value)
    General error handling class.

class pydotplus.graphviz.Graph(graph_name='G', obj_dict=None, graph_type='digraph',
    strict=False, suppress_disconnected=False, simplify=False,
    **attrs)
    Class representing a graph in Graphviz’s dot language.

    This class implements the methods to work on a representation of a graph in
    Graphviz’s dot language.

    graph(graph_name='G', graph_type='digraph', strict=False, suppress_disconnected=False,
          attribute=value, ...)
    graph_name: the graph’s name
    graph_type: can be ‘graph’ or ‘digraph’
    suppress_disconnected: defaults to False, which will remove from the graph any
                            disconnected nodes.
    simplify: if True it will avoid displaying equal edges, i.e. only one edge
              between two nodes. removing the duplicated ones.

    All the attributes defined in the Graphviz dot language should be supported.

    Attributes can be set through the dynamically generated methods:
        set_[attribute name], i.e. set_size, set_fontname
    or using the instance’s attributes:
        Graph.obj_dict[’attributes’][attribute name], i.e.
            graph_instance.obj_dict[’attributes’][’label’] graph_instance.obj_dict[’attributes’][’fontname’]

    add_edge(graph_edge)
    Adds an edge object to the graph.
    It takes an edge object as its only argument and returns None.

    add_node(graph_node)
    Adds a node object to the graph.
    It takes a node object as its only argument and returns None.

    add_subgraph(sgraph)
    Adds a subgraph object to the graph.
    It takes a subgraph object as its only argument and returns None.

    del_edge(src_or_list, dst=None, index=None)
    Delete an edge from the graph.
    Given an edge’s (source, destination) node names all matching edges(s) will be deleted if ‘index’ is not
    specified or set to None. If there are several matching edges and ‘index’ is given, only the edge in that
    position will be deleted.
    ‘index’ should be an integer specifying the position of the edge to delete. If index is larger than the number
    of matching edges, no action is taken.
    If edges are deleted it returns True. If no action is taken it returns False.

    del_node(name, index=None)
    Delete a node from the graph.
    Given a node’s name all node(s) with that same name will be deleted if ‘index’ is not specified or set to
    None. If there are several nodes with that same name and ‘index’ is given, only the node in that position
    will be deleted.
‘index’ should be an integer specifying the position of the node to delete. If index is larger than the number of nodes with that name, no action is taken.

If nodes are deleted it returns True. If no action is taken it returns False.

**get_edge** *(src_or_list, dst=None)*
Retrieved an edge from the graph.

Given an edge’s source and destination the corresponding Edge instance(s) will be returned.
If one or more edges exist with that source and destination a list of Edge instances is returned. An empty list is returned otherwise.

**get_edge_defaults** (**attrs**)

**get_edge_list**()
Get the list of Edge instances.
This method returns the list of Edge instances composing the graph.

**get_edges**()

**get_graph_defaults** (**attrs**)

**get_graph_type**()

**get_name**()
Get the graph’s name.

**get_next_sequence_number**()

**get_node** *(name)*
Retrieve a node from the graph.

Given a node’s name the corresponding Node instance will be returned.
If one or more nodes exist with that name a list of Node instances is returned. An empty list is returned otherwise.

**get_node_defaults** (**attrs**)

**get_node_list**()
Get the list of Node instances.
This method returns the list of Node instances composing the graph.

**get_nodes**()
Get the list of Node instances.

**get_simplify**()
Get whether to simplify or not.
Refer to set_simplify for more information.

**get_strict** *(val)*
Get graph’s ‘strict’ mode (True, False).
This option is only valid for top level graphs.

**get_subgraph** *(name)*
Retrieved a subgraph from the graph.
Given a subgraph’s name the corresponding Subgraph instance will be returned.
If one or more subgraphs exist with the same name, a list of Subgraph instances is returned. An empty list is returned otherwise.
get_subgraph_list()
Get the list of Subgraph instances.
This method returns the list of Subgraph instances in the graph.

get_subgraphs()

get_suppress_disconnected(val)
Get if suppress disconnected is set.
Refer to set_suppress_disconnected for more information.

get_top_graph_type()

get_type()
Get the graph’s type, ‘graph’ or ‘digraph’.

set_edge_defaults(**attrs)
set_graph_defaults(**attrs)
set_name(graph_name)
Set the graph’s name.

set_node_defaults(**attrs)
set_parent_graph(parent_graph)

set_simplify(simplify)
Set whether to simplify or not.
If True it will avoid displaying equal edges, i.e. only one edge between two nodes. removing the duplicated ones.

set_strict(val)
Set graph to ‘strict’ mode.
This option is only valid for top level graphs.

set_suppress_disconnected(val)
Suppress disconnected nodes in the output graph.
This option will skip nodes in the graph with no incoming or outgoing edges. This option works also for subgraphs and has effect only in the current graph/subgraph.

set_type(graph_type)
Set the graph’s type, ‘graph’ or ‘digraph’.

to_string()
Returns a string representation of the graph in dot language.
It will return the graph and all its subelements in string from.

exception pydotplus.graphviz.InvocationException(value)
To indicate that a problem occurred while running any of the GraphViz executables.

class pydotplus.graphviz.Node(name='', obj_dict=None, **attrs)
A graph node.
This class represents a graph’s node with all its attributes.
node(name, attribute=value, ...)
name: node’s name
All the attributes defined in the Graphviz dot language should be supported.
add_style (style)

get_name()
    Get the node’s name.

get_port()
    Get the node’s port.

set_name (node_name)
    Set the node’s name.

to_string()
    Returns a string representation of the node in dot language.

class pydotplus.graphviz.Subgraph (graph_name='', obj_dict=None, suppress_disconnected=False, simplify=False, **attrs)
    Class representing a subgraph in Graphviz’s dot language.

    This class implements the methods to work on a representation of a subgraph in Graphviz’s dot language.

subgraph (graph_name='subG', suppress_disconnected=False, attribute=value, ...)

    graph_name: the subgraph’s name

    suppress_disconnected: defaults to false, which will remove from the subgraph any disconnected nodes.

    All the attributes defined in the Graphviz dot language should be supported.

    Attributes can be set through the dynamically generated methods:
        set_[attribute name], i.e. set_size, set_fontname
    or using the instance’s attributes:
        Subgraph.obj_dict['attributes'][attribute name], i.e.
        subgraph_instance.obj_dict['attributes']['label'] subgraph_instance.obj_dict['attributes']['fontname']

pydotplus.graphviz.find_graphviz()
    Locate Graphviz’s executables in the system.

    Tries three methods:
    First: Windows Registry (Windows only) This requires Mark Hammond’s pywin32 is installed.
    Secondly: Search the path It will look for ‘dot’, ‘twopi’ and ‘neato’ in all the directories specified in the PATH environment variable.
    Thirdly: Default install location (Windows only) It will look for ‘dot’, ‘twopi’ and ‘neato’ in the default install location under the “Program Files” directory.

    It will return a dictionary containing the program names as keys and their paths as values.

    If this fails, it returns None.

class pydotplus.graphviz.frozendict (*args, **kw)

    clear
    pop
    popitem
    setdefault
    update

2.1. API Reference
PyDotPlus Documentation, Release 2.0.2

```python
pydotplus.graphviz.get_fobj(fname, mode='w+')
```

Obtain a proper file object.

- **fname** [string, file object, file descriptor] If a string or file descriptor, then we create a file object. If `fname` is a file object, then we do nothing and ignore the specified `mode` parameter.

- **mode** [str] The mode of the file to be opened.

```python
pydotplus.graphviz.graph_from_adjacency_matrix(matrix, node_prefix='', directed=False)
```

Creates a basic graph out of an adjacency matrix.

The matrix has to be a list of rows of values representing an adjacency matrix. The values can be anything: bool, int, float, as long as they can evaluate to True or False.

```python
pydotplus.graphviz.graph_from_dot_data(data)
```

Load graph as defined by data in DOT format.

The data is assumed to be in DOT format. It will be parsed and a Dot class will be returned, representing the graph.

```python
pydotplus.graphviz.graph_from_dot_file(path)
```

Load graph as defined by a DOT file.

The file is assumed to be in DOT format. It will be loaded, parsed and a Dot class will be returned, representing the graph.

```python
pydotplus.graphviz.graph_from_edges(edge_list, node_prefix='', directed=False)
```

Creates a basic graph out of an edge list.

The edge list has to be a list of tuples representing the nodes connected by the edge. The values can be anything: bool, int, float, str.

If the graph is undirected by default, it is only calculated from one of the symmetric halves of the matrix.

```python
pydotplus.graphviz.graph_from_incidence_matrix(matrix, node_prefix='', directed=False)
```

Creates a basic graph out of an incidence matrix.

The matrix has to be a list of rows of values representing an incidence matrix. The values can be anything: bool, int, float, as long as they can evaluate to True or False.

```python
pydotplus.graphviz.is_string_like(obj)
```

Check if `obj` is string.

```python
pydotplus.graphviz.needs_quotes(s)
```

Checks whether a string is a dot language ID.

It will check whether the string is solely composed by the characters allowed in an ID or not. If the string is one of the reserved keywords it will need quotes too but the user will need to add them manually.

```python
pydotplus.graphviz.quote_if_necessary(s)
```

**Parser Module**

Graphviz’s dot language parser.
The dotparser parses graphviz files in dot and dot files and transforms them into a class representation defined by pydotplus.

```python
class pydotplus.parser.DefaultStatement (default_type, attrs)
class pydotplus.parser.P_AttrList (toks)
pydotplus.parser.add_defaults (element, defaults)
pydotplus.parser.add_elements (g, toks, defaults_graph=None, defaults_node=None, defaults_edge=None)
pydotplus.parser.do_node_ports (node)
pydotplus.parser.get_port (node)
pydotplus.parser.graph_definition ()
pydotplus.parser.parse_dot_data (data)
pydotplus.parser.push_attr_list (str, loc, toks)
pydotplus.parser.push_default_stmt (str, loc, toks)
pydotplus.parser.push_edge_stmt (str, loc, toks)
pydotplus.parser.push_graph_stmt (str, loc, toks)
pydotplus.parser.push_node_stmt (s, loc, toks)
pydotplus.parser.push_subgraph_stmt (str, loc, toks)
pydotplus.parser.push_top_graph_stmt (str, loc, toks)
pydotplus.parser.update_parent_graph_hierarchy (g, parent_graph=None, level=0)
```
CHAPTER 3

Installation

For the latest stable version:

```
pip install pydotplus
```

For the development version:

```
pip install https://github.com/carlos-jenkins/pydotplus/archive/master.zip
```
CHAPTER 4

Contribute

https://github.com/carlos-enkins/pydotplus
Copyright (c) 2014 Carlos Jenkins <carlos@jenkins.co.cr>
Copyright (c) 2014 Lance Hepler
Copyright (c) 2004-2011 Ero Carrera <ero@dkbza.org>
Copyright (c) 2004-2007 Michael Krause <michael@krause-software.de>

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
p
pydotplus.graphviz, 5
pydotplus.parser, 12
Index

A
add_defaults() (in module pydotplus.parser), 13
add_edge() (pydotplus.graphviz.Graph method), 8
add_elements() (in module pydotplus.parser), 13
add_node() (pydotplus.graphviz.Graph method), 8
add_style() (pydotplus.graphviz.Node method), 10
add_subgraph() (pydotplus.graphviz.Graph method), 8

C
clear (pydotplus.graphviz.frozendict attribute), 11
Cluster (class in pydotplus.graphviz), 5
Common (class in pydotplus.graphviz), 6
create() (pydotplus.graphviz.Dot method), 6
create_attribute_methods() (pydotplus.graphviz.Common method), 6

D
DefaultStatement (class in pydotplus.parser), 13
del_edge() (pydotplus.graphviz.Graph method), 8
del_node() (pydotplus.graphviz.Graph method), 8
do_node_ports() (in module pydotplus.parser), 13
Dot (class in pydotplus.graphviz), 6

E
Edge (class in pydotplus.graphviz), 7
Error, 7

F
find_graphviz() (in module pydotplus.parser), 11
frozendict (class in pydotplus.graphviz), 11

G
get() (pydotplus.graphviz.Common method), 6
get_attributes() (pydotplus.graphviz.Common method), 6
get_destination() (pydotplus.graphviz.Edge method), 7
get_edge() (pydotplus.graphviz.Graph method), 9
get_edge_defaults() (pydotplus.graphviz.Graph method), 9
get_edge_list() (pydotplus.graphviz.Graph method), 9
get_edges() (pydotplus.graphviz.Graph method), 9
get_fobj() (in module pydotplus.graphviz), 12
get_graph_defaults() (pydotplus.graphviz.Graph method), 9
get_graph_type() (pydotplus.graphviz.Graph method), 9
get_name() (pydotplus.graphviz.Graph method), 9
get_name() (pydotplus.graphviz.Node method), 11
get_next_sequence_number() (pydotplus.graphviz.Graph method), 9
get_node() (pydotplus.graphviz.Graph method), 9
get_node_defaults() (pydotplus.graphviz.Graph method), 9
get_node_list() (pydotplus.graphviz.Graph method), 9
get_nodes() (pydotplus.graphviz.Graph method), 9
get_parent_graph() (pydotplus.graphviz.Common method), 6
get_port() (in module pydotplus.parser), 13
get_port() (pydotplus.graphviz.Node method), 11
get_sequence() (pydotplus.graphviz.Common method), 6
get_simplify() (pydotplus.graphviz.Graph method), 9
get_source() (pydotplus.graphviz.Edge method), 7
get_strict() (pydotplus.graphviz.Graph method), 9
get_subgraph() (pydotplus.graphviz.Graph method), 9
get_subgraph_list() (pydotplus.graphviz.Graph method), 9
get_subgraphs() (pydotplus.graphviz.Graph method), 10
get_suppress_disconnected() (pydotplus.graphviz.Graph method), 10
get_top_graph_type() (pydotplus.graphviz.Graph method), 10
get_type() (pydotplus.graphviz.Graph method), 10
Graph (class in pydotplus.graphviz), 8
graph_definition() (in module pydotplus.parser), 13
graph_from_adjacency_matrix() (in module pydotplus.graphviz), 12
graph_from_dot_data() (in module pydotplus.graphviz), 12
graph_from_dot_file() (in module pydotplus.graphviz), 12
graph_from_edges() (in module pydotplus.graphviz), 12
graph_from_incidence_matrix() (in module pydotplus.graphviz), 12

InvocationException, 10
is_string_like() (in module pydotplus.graphviz), 12
needs_quotes() (in module pydotplus.graphviz), 12
Node (class in pydotplus.graphviz), 10

P
P_AttrList (class in pydotplus.parser), 13
parse_dot_data() (in module pydotplus.parser), 13
parse_node_ref() (pydotplus.graphviz.Edge method), 7
pop (pydotplus.graphviz.frozendict attribute), 11
popitem (pydotplus.graphviz.frozendict attribute), 11
push_attr_list() (in module pydotplus.parser), 13
push_default_stmt() (in module pydotplus.parser), 13
push_edge_stmt() (in module pydotplus.parser), 13
push_graph_stmt() (in module pydotplus.parser), 13
push_node_stmt() (in module pydotplus.parser), 13
push_subgraph_stmt() (in module pydotplus.parser), 13
push_top_graph_stmt() (in module pydotplus.parser), 13
pydotplus.graphviz (module), 5
pydotplus.parser (module), 12

Q
quote_if_necessary() (in module pydotplus.graphviz), 12

S
set() (pydotplus.graphviz.Common method), 6
set_edge_defaults() (pydotplus.graphviz.Graph method), 10
set_graph_defaults() (pydotplus.graphviz.Graph method), 10
set_graphviz_executables() (pydotplus.graphviz.Dot method), 6
set_name() (pydotplus.graphviz.Graph method), 10
set_name() (pydotplus.graphviz.Node method), 11
set_node_defaults() (pydotplus.graphviz.Graph method), 10
set_parent_graph() (pydotplus.graphviz.Common method), 6
set_parent_graph() (pydotplus.graphviz.Graph method), 10
set_prog() (pydotplus.graphviz.Dot method), 7
set_sequence() (pydotplus.graphviz.Common method), 6
set_shape_files() (pydotplus.graphviz.Dot method), 7
set_simplify() (pydotplus.graphviz.Graph method), 10
set_strict() (pydotplus.graphviz.Graph method), 10
set_suppress_disconnected() (pydotplus.graphviz.Graph method), 10
set_type() (pydotplus.graphviz.Graph method), 10
setdefault (pydotplus.graphviz.frozendict attribute), 11
Subgraph (class in pydotplus.graphviz), 11

T
to_string() (pydotplus.graphviz.Edge method), 7
to_string() (pydotplus.graphviz.Graph method), 10
to_string() (pydotplus.graphviz.Node method), 11

U
update (pydotplus.graphviz.frozendict attribute), 11
update_parent_graph_hierarchy() (in module pydotplus.parser), 13

W
write() (pydotplus.graphviz.Dot method), 7