

Import Statements

```
from graphs pac e_p yth on.a pi.client import GraphSpace..., border _wi dth=2, ...)
```

Border width

```
from graphs pac e_p yth on.g ra phs.cl ass es.g sgraph import GSGraph
```

```
from graphs pac e_p yth on.g ra phs.cl ass es.g sgroup import GSGroup
```

```
from graphs pac e_p yth on.g ra phs.cl ass es.g sl ayout import GSLayout
```

Create a Graph

```
G = GSGraph()
```

Create empty graph

```
G.add_ nod e('a')
```

Add node to graph

```
..., label='A', ...)
```

Label to display

```
..., popup= 'html text', ...)
```

Popup text

```
G.add_ edg e('a', 'b')
```

Add undirected edge to graph

```
..., direct ed= True, ...)
```

Directed edge

```
..., popup= 'html text', ...)
```

Popup text

```
G.set_ nam e('My Sample Graph')
```

Add name/title to graph

```
G.set_ tag s([ 'sa mple'])
```

Add tags to graph

Add Node Style

```
G.add_ nod e_s tyl e('a', shape= 'el li pse')
```

Node shape

```
..., color= 'blue', ...)
```

Node color

```
..., color= '#A 152B8', ...)
```

HTML color

```
..., bubble = 'b lue', ...)
```

Bubble style

```
..., height=40, width=20, ...)
```

Custom size

```
..., style= 'da shed', ...)
```

Border style

```
..., border _co lor = 'red', ...)
```

Border color

Add Node Style (cont)

Arguments should be combined into one add_ no de_ style command.

Add Edge Style

```
G.add_ edg e_s tyl e('a', 'b', edge_s tyl e= 'd ot ted')
```

Edge style

```
..., direct ed= True, ...)
```

Directed edge

```
..., color= '#F F5733', ...)
```

HTML color

```
..., arrow_ sha pe= 'tee', ...)
```

Arrow shape

```
..., width=4, ...)
```

Edge width

```
..., color= 'ma genta', ...)
```

Edge color

```
..., arrow_ fil l= 'hol low', ...)
```

Arrow fill

```
..., width=4, ...)
```

Edge width

Arguments should be combined into one add_ ed ge_ style command.

Post Graph to GraphSpace

```
gs = GraphS pac e(' uname', 'pssw d')
```

Connect to GraphSpace

```
graph = gs.pos t_ g raph(G)
```

Post (new) graph to GraphSpace

```
graph = gs.upd ate _gr aph(G)
```

Update (existing) graph to GraphSpace

```
gs.del ete _gr aph (gr aph =graph)
```

Delete graph from GraphSpace



Update Graph if it Exists, Otherwise Post it

```
def post(G, gs):
    try:
        graph = gs.update_graph(G)
    except:
        graph = gs.post_graph(G)
    return graph
```

Fetch Posted Graph

```
graph = gs.get_graph(graph_name='My Sample Graph')
```

```
graph = gs.get_graph(graph_id=12345)
```

```
graph.name
```

```
graph.id
```

Make Graph Public

```
gs.publish_graph(graph_id=graph_id) Make graph public
```

```
gs.unpublish_graph(graph_id=graph_id) Make graph private
```

Groups

```
mygroup = GSGroup(name='My group', description='sample group')
```

```
group = gs.post_group(mygroup)
```

```
group = gs.get_group(group_name='My group')
```

```
gs.share_graph(graph=graph, group_name='My group')
```

Groups (cont)

```
gs.unshare_graph(graph_id=graph_id, group_name='My group')
```

Layouts

```
L = GSLayout()
```

Fetch

```
by L.set_node_position('a', y=38.5, x=67.3)
```

graph

```
name.add_node_style(),L.add_edge_style()
```

Fetch

by

```
graph.set_name('My Sample Layout')
```

id

```
graph.set_is_shared(1)
```

of

graph

ID of

graph

```
layout = graphspace.post_graph_layout(L, graph_id)
```

```
layout = graphspace.get_graph_layout(layout_id, graph_id)
```

```
layout1 = graphspace.update_graph_layout(layout_id, graph_id)
```

Graphs usually have manually-determined layouts from GraphSpace, which are saved through the user interface.

Add

group

Fetch

group

Share

graph

with

group