

---

# **quickdraw Documentation**

***Release 1.0.0***

**Martin O'Hanlon**

**May 16, 2023**



---

## Contents:

---

<b>1</b>	<b>Getting started</b>	<b>3</b>
<b>2</b>	<b>Use</b>	<b>5</b>
<b>3</b>	<b>Examples</b>	<b>7</b>
<b>4</b>	<b>Documentation</b>	<b>9</b>
<b>5</b>	<b>Warning</b>	<b>11</b>
<b>6</b>	<b>Status</b>	<b>13</b>
<b>7</b>	<b>Table of Contents</b>	<b>15</b>
7.1	quickdraw API . . . . .	15
7.2	Change log . . . . .	20
	<b>Python Module Index</b>	<b>23</b>
	<b>Index</b>	<b>25</b>



Quick Draw is a drawing game which is training a neural network to recognise doodles.



## Can a neural network learn to recognize doodling?

quickdraw is a Python API for accessing the [Quick Draw data](#) - it downloads the data files as and when needed, caches them locally and interprets them so they can be used.



Created by [Martin O'Hanlon](#) ([@martinohanlon](#), [stuffaboutco.de](#)).



# CHAPTER 1

---

## Getting started

---

Install the *quickdraw* python library using *pip*.

- Windows

```
pip install quickdraw
```

- macOS

```
pip3 install quickdraw
```

- Linux / Raspberry Pi

```
sudo pip3 install quickdraw
```





## CHAPTER 2

---

### Use

---

Here are some examples of how to use quickdraw but be sure to also checkout the [API documentation](#) for more information.

Open the Quick Draw data using [QuickDrawData](#) and pull back a drawing of an **anvil**.

```
from quickdraw import QuickDrawData
qd = QuickDrawData()
anvil = qd.get_drawing("anvil")

print(anvil)
```

quickdraw will download the `anvil.bin` data file and return the data for a random drawing of an anvil (well a doodle of an anvil anyway).

Drawings are returned as [QuickDrawing](#) objects which exposes the properties of the drawing.

```
print(anvil.name)
print(anvil.key_id)
print(anvil.countrycode)
print(anvil.recognized)
print(anvil.timestamp)
print(anvil.no_of_strokes)
print(anvil.image_data)
print(anvil.strokes)
```

You can save the drawing using the `image` property.

```
anvil.image.save("my_anvil.gif")
```

You can save an animation of the drawing using the `animation` property.

```
anvil.animation.save("my_anvil_animation.gif")
```

You can open a group of Quick Draw drawings using `QuickDrawDataGroup` passing the name of the drawing (“anvil”, “aircraft”, “baseball”, etc).

```
from quickdraw import QuickDrawDataGroup

anvils = QuickDrawDataGroup("anvil")
print(anvils.drawing_count)
print(anvils.get_drawing())
```

By default only 1000 drawings are opened, you can change this by modifying the `max_drawings` parameter of `QuickDrawDataGroup`, setting it to `None` will open all the drawings in that group.

```
from quickdraw import QuickDrawDataGroup

anvils = QuickDrawDataGroup("anvil", max_drawings=None)
print(anvils.drawing_count)
```

To iterate through all the drawings in a group use the `drawings` generator.

```
from quickdraw import QuickDrawDataGroup

qdg = QuickDrawDataGroup("anvil")
for drawing in qdg.drawings:
    print(drawing)
```

You can get a list of all the drawing names using the `drawing_names` property of `QuickDrawData`.

```
from quickdraw import QuickDrawData

qd = QuickDrawData()
print(qd.drawing_names)
```

## CHAPTER 3

---

### Examples

---

Code examples can be found in the [quickdraw GitHub repository](#).



## CHAPTER 4

---

### Documentation

---

API documentation can be found at [quickdraw.readthedocs.io](https://quickdraw.readthedocs.io)



## CHAPTER 5

---

### Warning

---

The drawings have been moderated but there is no guarantee it'll actually be a picture of what you are asking it for (although in my experience they are)!





## CHAPTER 6

---

### Status

---

**Stable.**

Raise any [issues](#) in the [github repository](#).



## 7.1 quickdraw API

### 7.1.1 QuickDrawData

**class** quickdraw.**QuickDrawData** (*recognized=None, max\_drawings=1000, refresh\_data=False, jit\_loading=True, print\_messages=True, cache\_dir='./quickdrawcache'*)

Allows interaction with the Google Quick, Draw! data set, downloads Quick Draw data from [https://storage.googleapis.com/quickdraw\\_dataset/full/binary/](https://storage.googleapis.com/quickdraw_dataset/full/binary/) and loads it into memory for easy access and processing.

The following example will load the anvil drawings and get a single drawing:

```
from quickdraw import QuickDrawData

qd = QuickDrawData()

anvil = qd.get_drawing("anvil")
anvil.image.save("my_anvil.gif")
```

#### Parameters

- **recognized** (*bool*) – If `True` only recognized drawings will be loaded, if `False` only unrecognized drawings will be loaded, if `None` (the default) both recognized and unrecognized drawings will be loaded.
- **max\_drawings** (*int*) – The maximum number of drawings to be loaded into memory, defaults to 1000.
- **refresh\_data** (*bool*) – If `True` forces data to be downloaded even if it has been downloaded before, defaults to `False`.
- **jit\_loading** (*bool*) – If `True` (the default) only downloads and loads data into memory when it is required (`jit` = just in time). If `False` all drawings will be downloaded and loaded into memory.

- **print\_messages** (*bool*) – If `True` (the default), status messages will be printed stating when data is being downloaded or loaded.
- **cache\_dir** (*string*) – Specify a cache directory to use when downloading data files, defaults to `./quickdrawcache`.

**get\_drawing** (*name*, *index=None*)

Get a drawing.

Returns an instance of `QuickDrawing` representing a single Quick, Draw drawing.

#### Parameters

- **name** (*string*) – The name of the drawing to get (anvil, ant, aircraft, etc).
- **index** (*int*) – The index of the drawing to get.

If `None` (the default) a random drawing will be returned.

**get\_drawing\_group** (*name*)

Get a group of drawings by name.

Returns an instance of `QuickDrawDataGroup`.

**Parameters** **name** (*string*) – The name of the drawings (anvil, ant, aircraft, etc).

**load\_all\_drawings** ()

Loads (and downloads if required) all drawings into memory.

**load\_drawings** (*list\_of\_drawings*)

Loads (and downloads if required) all drawings into memory.

**Parameters** **list\_of\_drawings** (*list*) – A list of the drawings to be loaded (anvil, ant, aircraft, etc).

**search\_drawings** (*name*, *key\_id=None*, *recognized=None*, *countrycode=None*, *timestamp=None*)

Search the drawings.

Returns an list of `QuickDrawing` instances representing the matched drawings.

Note - search criteria are a compound.

Search for all the drawings with the `countrycode` “PL”

```
from quickdraw import QuickDrawDataGroup

anvils = QuickDrawDataGroup("anvil")
results = anvils.search_drawings(countrycode="PL")
```

#### Parameters

- **name** (*string*) – The name of the drawings (anvil, ant, aircraft, etc) to search.
- **key\_id** (*int*) – The `key_id` to such for. If `None` (the default) the `key_id` is not used.
- **recognized** (*bool*) – To search for drawings which were recognized. If `None` (the default) recognized is not used.
- **countrycode** (*int*) – To search for drawings which with the `countrycode`. If `None` (the default) `countrycode` is not used.
- **countrycode** – To search for drawings which with the `timestamp`. If `None` (the default) `timestamp` is not used.

**drawing\_names**

Returns a list of all the potential drawing names.

**loaded\_drawings**

Returns a list of drawing which have been loaded into memory.

## 7.1.2 QuickDrawDataGroup

```
class quickdraw.QuickDrawDataGroup(name, recognized=None, max_drawings=1000,  
                                   refresh_data=False, print_messages=True,  
                                   cache_dir='./quickdrawcache')
```

Allows interaction with a group of Quick, Draw! drawings.

The following example will load the ant group of drawings and get a single drawing:

```
from quickdraw import QuickDrawDataGroup

ants = QuickDrawDataGroup("ant")
ant = ants.get_drawing()
ant.image.save("my_ant.gif")
```

### Parameters

- **name** (*string*) – The name of the drawings to be loaded (anvil, ant, aircraft, etc).
- **recognized** (*bool*) – If `True` only recognized drawings will be loaded, if `False` only unrecognized drawings will be loaded, if `None` (the default) both recognized and unrecognized drawings will be loaded.
- **max\_drawings** (*int*) – The maximum number of drawings to be loaded into memory, defaults to 1000.
- **refresh\_data** (*bool*) – If `True` forces data to be downloaded even if it has been downloaded before, defaults to `False`.
- **print\_messages** (*bool*) – If `True` (the default), status messages will be printed stating when data is being downloaded or loaded.
- **cache\_dir** (*string*) – Specify a cache directory to use when downloading data files, defaults to `./quickdrawcache`.

### **get\_drawing** (*index=None*)

Get a drawing from this group.

Returns an instance of `QuickDrawing` representing a single Quick, Draw drawing.

Get a single anvil drawing:

```
from quickdraw import QuickDrawDataGroup

anvils = QuickDrawDataGroup("anvil")
anvil = anvils.get_drawing()
```

**Parameters** **index** (*int*) – The index of the drawing to get.

If `None` (the default) a random drawing will be returned.

**search\_drawings** (*key\_id=None, recognized=None, countrycode=None, timestamp=None*)

Searches the drawings in this group.

Returns an list of *QuickDrawing* instances representing the matched drawings.

Note - search criteria are a compound.

Search for all the drawings with the countrycode “PL”

```
from quickdraw import QuickDrawDataGroup

anvils = QuickDrawDataGroup("anvil")
results = anvils.search_drawings(countrycode="PL")
```

#### Parameters

- **key\_id** (*int*) – The key\_id to such for. If None (the default) the key\_id is not used.
- **recognized** (*bool*) – To search for drawings which were recognized. If None (the default) recognized is not used.
- **countrycode** (*int*) – To search for drawings which with the countrycode. If None (the default) countrycode is not used.
- **timestamp** – To search for drawings which with the timestamp. If None (the default) timestamp is not used.

**drawing\_count**

Returns the number of drawings loaded.

**drawings**

An iterator of all the drawings loaded in this group. Returns a *QuickDrawing* object.

Load the anvil group of drawings and iterate through them:

```
from quickdraw import QuickDrawDataGroup

anvils = QuickDrawDataGroup("anvil")
for anvil in anvils.drawings:
    print(anvil)
```

### 7.1.3 QuickDrawing

**class** quickdraw.**QuickDrawing** (*name, drawing\_data*)

Represents a single Quick, Draw! drawing.

**get\_animation** (*stroke\_color=(0, 0, 0), stroke\_width=2, bg\_color=(255, 255, 255)*)

Returns a *QuickDrawAnimation* instance representing the an animation of the QuickDrawing being created.

#### Parameters

- **stroke\_color** (*int*) – A list of RGB (red, green, blue) values for the stroke color, defaults to (0,0,0).
- **stroke\_color** – A width of the stroke, defaults to 2.
- **bg\_color** (*list*) – A list of RGB (red, green, blue) values for the background color, defaults to (255,255,255).

**get\_image** (*stroke\_color*=(0, 0, 0), *stroke\_width*=2, *bg\_color*=(255, 255, 255))

Get a [PIL Image](#) object of the drawing.

#### Parameters

- **stroke\_color** (*int*) – A list of RGB (red, green, blue) values for the stroke color, defaults to (0,0,0).
- **stroke\_color** – A width of the stroke, defaults to 2.
- **bg\_color** (*list*) – A list of RGB (red, green, blue) values for the background color, defaults to (255,255,255).

#### animation

Returns a [QuickDrawAnimation](#) instance representing the an animation of the QuickDrawing on a white background with a black drawing. Alternative image parameters can be set using `get_animation()`.

To save the animation you would use the `save` method:

```
from quickdraw import QuickDrawData

qd = QuickDrawData()

anvil = qd.get_drawing("anvil")
anvil.animation.save("my_anvil_animation.gif")
```

#### countrycode

Returns the country code for the drawing.

#### image

Returns a [PIL Image](#) object of the drawing on a white background with a black drawing. Alternative image parameters can be set using `get_image()`.

To save the image you would use the `save` method:

```
from quickdraw import QuickDrawData

qd = QuickDrawData()

anvil = qd.get_drawing("anvil")
anvil.image.save("my_anvil.gif")
```

#### image\_data

Returns the raw image data as list of strokes with a list of X co-ordinates and a list of Y co-ordinates.

Co-ordinates are aligned to the top-left hand corner with values from 0 to 255.

See <https://github.com/googlecreativelab/quickdraw-dataset#simplified-drawing-files-ndjson> for more information regarding how the data is represented.

#### key\_id

Returns the id of the drawing.

#### name

Returns the name of the drawing (anvil, aircraft, ant, etc).

#### no\_of\_strokes

Returns the number of pen strokes used to create the drawing.

#### recognized

Returns a boolean representing whether the drawing was recognized.

**strokes**

Returns a list of pen strokes containing a list of (x,y) coordinates which make up the drawing.

To iterate though the strokes data use:

```
from quickdraw import QuickDrawData

qd = QuickDrawData()

anvil = qd.get_drawing("anvil")
for stroke in anvil.strokes:
    for x, y in stroke:
        print("x={} y={}".format(x, y))
```

**timestamp**

Returns the time the drawing was created (in seconds since the epoch).

## 7.1.4 QuickDrawAnimation

**class** quickdraw.**QuickDrawAnimation** (*quick\_drawing, stroke\_color, stroke\_width, bg\_color*)

Represents an animation of a *QuickDrawing*.

While an instance can be created directly it is typically returned by *QuickDrawing.get\_animation()* or *QuickDrawing.animation()*.

To save the animation you would use the *save* method:

```
from quickdraw import QuickDrawData

qd = QuickDrawData()

anvil = qd.get_drawing("anvil")
anvil.animation.save("my_anvil_animation.gif")
```

**save** (*filename, frame\_length=0.1, loop\_times=0*)

Saves the animation to a given filename.

**Parameters**

- **filename** (*string*) – The filename or path to save the animation. The filetype must be gif.
- **frame\_length** (*int*) – The time in seconds between each frame, defaults to 0.1.
- **loop\_times** (*int*) – The number of times the animation should loop. A value of 0 will result in the animation looping forever. A value of None will result in the animation not looping. The default is 0.

**frames**

Returns a list [PIL Image](#) objects of the animation.

## 7.2 Change log

### 7.2.1 1.0.0 - 2022-05-16

- Added QuickDrawAnimation



- Made the project version 1.0.0 and set as Stable

### 7.2.2 0.2.0 - 2021-03-10

- Python 3.7+ compatibility fix

### 7.2.3 0.1.0

- Beta
- Bug fixes
- Additional properties methods and stuff
- Tests

### 7.2.4 0.0.1 > 0.0.4

- Alpha - setting up, working out the api, sorting out the bugs



**q**

quickdraw, [15](#)



## A

animation (*quickdraw.QuickDrawing attribute*), 19

## C

countrycode (*quickdraw.QuickDrawing attribute*), 19

## D

drawing\_count (*quickdraw.QuickDrawDataGroup attribute*), 18

drawing\_names (*quickdraw.QuickDrawData attribute*), 16

drawings (*quickdraw.QuickDrawDataGroup attribute*), 18

## F

frames (*quickdraw.QuickDrawAnimation attribute*), 20

## G

get\_animation() (*quickdraw.QuickDrawing method*), 18

get\_drawing() (*quickdraw.QuickDrawData method*), 16

get\_drawing() (*quickdraw.QuickDrawDataGroup method*), 17

get\_drawing\_group() (*quickdraw.QuickDrawData method*), 16

get\_image() (*quickdraw.QuickDrawing method*), 18

## I

image (*quickdraw.QuickDrawing attribute*), 19

image\_data (*quickdraw.QuickDrawing attribute*), 19

## K

key\_id (*quickdraw.QuickDrawing attribute*), 19

## L

load\_all\_drawings() (*quickdraw.QuickDrawData method*), 16

load\_drawings() (*quickdraw.QuickDrawData method*), 16

loaded\_drawings (*quickdraw.QuickDrawData attribute*), 17

## N

name (*quickdraw.QuickDrawing attribute*), 19

no\_of\_strokes (*quickdraw.QuickDrawing attribute*), 19

## Q

quickdraw (*module*), 15

QuickDrawAnimation (*class in quickdraw*), 20

QuickDrawData (*class in quickdraw*), 15

QuickDrawDataGroup (*class in quickdraw*), 17

QuickDrawing (*class in quickdraw*), 18

## R

recognized (*quickdraw.QuickDrawing attribute*), 19

## S

save() (*quickdraw.QuickDrawAnimation method*), 20

search\_drawings() (*quickdraw.QuickDrawData method*), 16

search\_drawings() (*quickdraw.QuickDrawDataGroup method*), 17

strokes (*quickdraw.QuickDrawing attribute*), 19

## T

timestamp (*quickdraw.QuickDrawing attribute*), 20