PyPexels Documentation

Release 1.0.0rc1

Salvatore Ventura <salvoventura@gmail.com>

Jul 25, 2020

Contents

An open source Python wrapper for the Pexels REST API.

The source code is available on GitHub at https://github.com/salvoventura/pypexels.

CHAPTER 1

Installation

PyPexels is available on PyPI and thus can be installed with pip on most platforms.

```
$ pip install pypexels
```

CHAPTER 2

Dependencies

This library depends on Requests to make - well - requests to the Pexels API. This additional package should be automatically installed at installation time, or you can simply install it by:

```
$ pip install requests
```

CHAPTER 3

Usage

3.1 Creating an instance

```
import pypexels
pp = pypexels.PyPexels(api_key='YOUR_API_KEY')
```

API keys can be obtained from Pexels API page.

3.2 Library logging

The PyPexels library internal logging subsystem is driven by the application. The default logging level of the library is set to be **logging.ERROR**. If you want to access the library logging subsystem, you can fine-tune the logger with id **PyPexels.logger_name** as per the following example:

```
from pypexels import PyPexels

# Initialize app logging
logger = logging.getLogger()
logging.basicConfig(filename='app.log', level=logging.DEBUG)

# pypexels logger defaults to level logging.ERROR
# If you need to change that, use getLogger/setLevel
# on the module logger, like this:
logging.getLogger(PyPexels.logger_name).setLevel(logging.DEBUG)

py_pexels = PyPexels(api_key='YOUR_API_KEY')
# ... continue
```

CHAPTER 4

Examples

These example shows how the interaction with the paging functionality of the Pexels API is greatly abstracted and simplified.

4.1 Popular

The code below will iterate through all popular photos in pages of 30 items each, retrieve each photo in there, and print some of their attributes.

```
from pypexels import PyPexels
api_key = 'YOUR_API_KEY'

# instantiate PyPexels object
py_pexel = PyPexels(api_key=api_key)

popular_photos = py_pexel.popular(per_page=30)
while True:
    for photo in popular_photos.entries:
        print(photo.id, photo.photographer, photo.url)
    if not popular_photos.has_next:
        break
    popular_photos = popular_photos.get_next_page()
```

4.2 Search

The code below will perform a search based on a query string, then iterate through all the results pages of 40 photos each, retrieve each photo in there, and print some of their attributes.

```
from pypexels import PyPexels
api_key = 'YOUR_API_KEY'
```

(continues on next page)

(continued from previous page)

```
# instantiate PyPexels object
py_pexel = PyPexels(api_key=api_key)

search_results = py_pexel.search(query='red flowers', per_page=40)
while True:
    for photo in search_results.entries:
        print(photo.id, photo.photographer, photo.url)
    if not search_results.has_next:
        break
    search_results = search_results.get_next_page()
```

4.3 Random

The code below will return random images. Use the parameter `per_page` to specify how many random images the iterator will allow.

```
from pypexels import PyPexels
api_key = 'YOUR_API_KEY'

# instantiate PyPexels object
py_pexel = PyPexels(api_key=api_key)

random_photos_page = py_pexel.random(per_page=3)
for photo in random_photos_page.entries:
    print(photo.id, photo.photographer, photo.url)
```

4.4 Single Photo

The code below will return an image object. Use the parameter `photo_id` to specify the ID of the image.

```
from pypexels import PyPexels
api_key = 'YOUR_API_KEY'

# instantiate PyPexels object
py_pexel = PyPexels(api_key=api_key)

# access single photo by its ID
photo = py_pexel.single_photo(photo_id=415071)
print(photo.id, photo.photographer, photo.url)
```

4.5 Popular Videos

The code below will iterate through all popular videos in pages of 40 items each, retrieve each video in there, and print some of their attributes.

```
from pypexels import PyPexels
api_key = 'YOUR_API_KEY'
```

(continues on next page)

(continued from previous page)

```
# instantiate PyPexels object
py_pexel = PyPexels(api_key=api_key)

popular_videos_page = py_pexel.videos_popular(per_page=40)
while True:
    for video in popular_videos_page.entries:
        print(video.id, video.user.get('name'), video.url)
    if not popular_videos_page.has_next:
        break
    popular_videos_page = popular_videos_page.get_next_page()
```

4.6 Search Videos

The code below will perform a search of videos based on a query string, then iterate through all the results pages of 40 videos each, retrieve each video in there, and print some of their attributes.

```
from pypexels import PyPexels
api_key = 'YOUR_API_KEY'

# instantiate PyPexels object
py_pexel = PyPexels(api_key=api_key)

search_videos_page = py_pexel.videos_search(query="red+flower", per_page=40)
while True:
    for video in search_videos_page.entries:
        print(video.id, video.user.get('name'), video.url)
    if not search_videos_page.has_next:
        break
    search_videos_page = search_videos_page.get_next_page()
```

4.7 Single Video

The code below will return a video object. Use the parameter `video_id` to specify the ID of the video.

```
from pypexels import PyPexels
api_key = 'YOUR_API_KEY'

# instantiate PyPexels object
py_pexel = PyPexels(api_key=api_key)

# access single video by its ID
video = py_pexel.single_video(video_id=1448735)
print(video.id, video.user.get('name'), video.url)
```

CHAPTER 5

API: Class PyPexels

This is the main class used to interact with the `Pexels` REST API.

5.1 PyPexels(api_key)

Create an instance of class `PyPexels`. The `api_key` parameter is required. API keys can be obtained from Pexels API page.

Parameters

Argument	Type	Optional/Required	Notes
api_key	string	required	Your API key
api_version	string	optional	API version to use. Default: 'v1'

Returns

Object	Instance of class `PyPexels`

Example

```
import pypexels
py_pexel = pypexels.PyPexels(api_key='YOUR_API_KEY')
```

5.2 Methods

Methods exposed by the `PyPexels` class.

5.2.1 Methods related to Photos

5.2.1.1 PyPexels.popular(page, per_page)

Allows to list `popular` images from Pexels.

Parameters

Argument	Type	Optional/Required	Notes
page	number	optional (default: 1)	Page number to retrieve.
per_page	number	optional (default: 15)	Number of items per page (max: 40)

Returns

Object	Instance of class `Popular`

Example

```
import pypexels
py_pexel = pypexels.PyPexels(api_key='YOUR_API_KEY')

#
#
popular_photos = py_pexel.popular(per_page=40)
for photo in popular_photos.entries:
    print(photo.id, photo.photographer, photo.url, photo.get_attribution('txt
↪'))
```

5.2.1.2 PyPexels.curated(page, per_page)

Allows to list `curated` images from Pexels.

Parameters

Argument	Type	Optional/Required	Notes
page	number	optional (default: 1)	Page number to retrieve.
per_page	number	optional (default: 15)	Number of items per page (max: 40)

Returns

Object	Instance of class `Curated`

Example

```
import pypexels
py_pexel = pypexels.PyPexels(api_key='YOUR_API_KEY')

#
#
curated_photos = py_pexel.curated(per_page=40)
for photo in curated_photos.entries:
    print(photo.id, photo.photographer, photo.url)
```

5.2.1.3 PyPexels.search(query, page, per_page)

Allows to perform `search` based on a string query.

Parameters

Argument	Type	Optional/Required	Notes
query	string	required	Search terms
page	number	optional (default: 1)	Page number to retrieve.
per_page	number	optional (default: 15)	Number of items per page (max: 40)

Returns

Object	Instance of class `Search`

Example

```
import pypexels
py_pexel = pypexels.PyPexels(api_key='YOUR_API_KEY')

#
#
search_photos = py_pexel.search(query='red flowers', per_page=40)
for photo in search_photos.entries:
    print(photo.id, photo.photographer, photo.url)
```

5.2.1.4 PyPexels.random(page, per_page)

Retrieves `random` images from Pexels.

Parameters

Argument	Type	Optional/Required	Notes
page	number	optional (default: 1)	Page number to retrieve.
per_page	number	optional (default: 15)	Number of items per page (max: 40)

Returns

Object	Instance of class `Random`

Example

```
import pypexels
py_pexel = pypexels.PyPexels(api_key='YOUR_API_KEY')

#
#
random_photos_page = py_pexel.random(per_page=7)
for photo in random_photos_page.entries:
    print(photo.id, photo.photographer, photo.url)
```

5.2.1.5 PyPexels.single_photo(photo_id)

Retrieve a single photo, known by its ID.

Parameters

Argument	Type	Optional/Required	Notes
photo_id	str	required	Image to retrieve.

Returns

Object	Instance of class `Photo`

Example

```
import pypexels
py_pexel = pypexels.PyPexels(api_key='YOUR_API_KEY')

# Retrieve a single photo, known by its ID
photo = py_pexel.single_photo(photo_id=415071)
print(photo.id, photo.photographer, photo.url)
print(photo.get_attribution('txt'))
print(photo.get_attribution('html'))
```

—

5.2.2 Methods related to Videos

5.2.2.1 PyPexels.videos_popular(page, per_page)

Allows to list `popular` videos from Pexels.

Parameters

Argument	Type	Optional/Required	Notes
page	number	optional (default: 1)	Page number to retrieve.
per_page	number	optional (default: 15)	Number of items per page (max: 40)

Returns

Object	Instance of class `VideosPopular`

Example

```
import pypexels
py_pexel = pypexels.PyPexels(api_key='YOUR_API_KEY')

popular_videos_page = py_pexel.videos_popular(per_page=40)
while True:
    for video in popular_videos_page.entries:
        print(video.id, video.user.get('name'), video.url)
    if not popular_videos_page.has_next:
        break
    popular_videos_page = popular_videos_page.get_next_page()
```

5.2.2.2 PyPexels.videos_search(query, page, per_page)

Allows to perform `search` based on a string query.

Parameters

Argument	Type	Optional/Required	Notes
query	string	required	Search terms
page	number	optional (default: 1)	Page number to retrieve.
per_page	number	optional (default: 15)	Number of items per page (max: 40)

Returns

Object	Instance of class `VideosSearch`

Example

```
import pypexels
py_pexel = pypexels.PyPexels(api_key='YOUR_API_KEY')

while True:
    for video in search_videos_page.entries:
        print(video.id, video.user.get('name'), video.url)
    if not search_videos_page.has_next:
        break
    search_videos_page = search_videos_page.get_next_page()
```

5.2.2.3 PyPexels.single_video(video_id)

Retrieve a single video, known by its ID.

Parameters

Argument	Type	Optional/Required	Notes
video_id	str	required	Video to retrieve.

Returns

Object	Instance of class `Video`

Example

```
import pypexels
py_pexel = pypexels.PyPexels(api_key='YOUR_API_KEY')

# Retrieve a single video, known by its ID
video = py_pexel.single_video(video_id=415071)
print(video.id, video.user.get('name'), video.url)
print(video.get_attribution('txt'))
print(video.get_attribution('html'))
```

CHAPTER 6

API: Class Popular

This class is used to access the **popular** photos on the `PyPexels /popular/` REST API.

It is returned as result from a call to `PyPexels.popular(page, per_page)`

6.1 Properties

Properties exposed by the `Popular` class.

6.1.1 Popular.entries

Iterator for the returned objects contained in this `Popular` instance. Each entry will be an instance of class `Photo`.

iterator	each time an instance of class `Photo`

Example

```
import pypexels
py_pexel = pypexels.PyPexels(api_key='YOUR_API_KEY')

#
#
popular_photos_page = py_pexel.popular(per_page=40)
for photo in popular_photos_page.entries:
    print(photo.id, photo.photographer, photo.url)
    # no need to specify per_page: will take from original object
    popular_photos_page = popular_photos_page.get_next_page()
popular_results = py_pexel.popular(per_page=40)
for photo in popular_results.entries:
    print(photo.id, photo.photographer, photo.url)
```

6.1.2 Popular.page

Current popular photos page number.

int	current popular photos page number

6.1.3 Popular.per_page

Current popular photos per_page value.

int	current popular photos per_page value

6.1.4 Popular.has_previous

Returns boolean **True** or **False** depending on whether the current results page has a previous page to navigate to or not.

boolean	presence of previous page results

6.1.5 Popular.has_next

Returns boolean **True** or **False** depending on whether the current results page has a next page to navigate to or not.

boolean	presence of next page results

6.1.6 Popular.body

Returns JSON body of popular photos page.

JSON	JSON converted body of results page

6.1.7 Popular.headers

Returns response headers of popular photos page.

dict	headers of results page

6.1.8 Popular.link_self

Returns URL to current results page

str	URL to current results page

6.1.9 Popular.link_next

Returns URL to next results page

str	URL to next results page

6.1.10 Popular.link_previous

Returns URL to previous results page

str	URL to previous results page

6.1.11 Popular.link_first

Returns URL to first results page

str	URL to first results page

Note: `Popular.total_results` always returns 0 (zero). `Popular.link_last` always points to the first page.

6.2 Methods

Methods exposed by the `Popular` class.

6.2.1 Popular.get_page()

Returns the requested popular photos page with the current *query* and *per_page* parameters. The returned page may not contain *entries* if the page is out of boundaries.

Parameters

Argument	Type	Optional/Required	Notes
page	number	required	Page number to retrieve.

Returns

Object	Instance of class `Popular`

6.2.2 Popular.get_next_page()

Returns next available popular photos page with the current *query*, *page*, and *per_page* parameters. Returns *None* if no page is available.

Returns

Object	Instance of class `Popular` or *None*

Example

```
import pypexels
py_pexel = pypexels.PyPexels(api_key='YOUR_API_KEY')

#
#
search_results = py_pexel.popular(query='red flowers', per_page=40)
while search_results is not None:
    print 'Current page number %s' % search_results.page
    search_results = search_results.get_next_page()
```

6.2.3 Popular.get_previous_page()

Returns previous available popular photos page with the current *query*, *page*, and *per_page* parameters. Returns *None* if no page is available.

Returns

Object	Instance of class `Popular` or *None*

Example

```
import pypexels
py_pexel = pypexels.PyPexels(api_key='YOUR_API_KEY')

#
#
search_results = py_pexel.popular(query='red flowers', page=3, per_page=40)
while search_results is not None:
    print 'Current page number %s' % search_results.page
    search_results = search_results.get_previous_page()
```

6.2.4 Popular.get_first_page()

Returns first popular photos page with the current *query*, *page*, and *per_page* parameters. Returns *None* if no page is available.

Returns

Object	Instance of class `Popular` or *None*

Example

```
import pypexels
py_pexel = pypexels.PyPexels(api_key='YOUR_API_KEY')

#
#
search_results = py_pexel.popular(query='red flowers', page=3, per_page=40)
print 'Current page number %s' % search_results.page
# To something with search_results

# Go back to first page
search_results = search_results.get_first_page():
print 'Current page number %s' % search_results.page
```

Note: `Popular.get_last_page()` always returns the first page.

CHAPTER 7

API: Class Search

This class is used to access the results of a **search** on the `PyPexels /search/` REST API.

It is returned as result from a call to `PyPexels.search(query, page, per_page)`

7.1 Properties

Properties exposed by the `Search` class.

7.1.1 Search.entries

Returns an iterator for the returned objects contained in this `Search` instance. Each entry will be an instance of class `Photo`.

iterator	each time an instance of class `Photo`

Example

```
import pypexels
py_pexel = pypexels.PyPexels(api_key='YOUR_API_KEY')

#
#
search_results = py_pexel.search(query='red flowers', per_page=40)
for photo in search_results.entries:
    print(photo.id, photo.photographer, photo.url)
```

7.1.2 Search.page

Current search results page number.

int	current search results page number

7.1.3 Search.per_page

Current search results per_page value.

int	current search results per_page value

7.1.4 Search.total_results

Total results count, number of photos found

int	total results count, number of photos found

7.1.5 Search.has_previous

Returns boolean **True** or **False** depending on whether the current results page has a previous page to navigate to or not.

boolean	presence of previous page results

7.1.6 Search.has_next

Returns boolean **True** or **False** depending on whether the current results page has a next page to navigate to or not.

boolean	presence of next page results

7.1.7 Search.body

Returns JSON body of search results page.

JSON	JSON converted body of results page

7.1.8 Search.headers

Returns response headers of search results page.

dict	headers of results page

7.1.9 Search.link_self

Returns URL to current results page

str	URL to current results page

7.1.10 Search.link_next

Returns URL to next results page

str	URL to next results page

7.1.11 Search.link_previous

Returns URL to previous results page

str	URL to previous results page

7.1.12 Search.link_first

Returns URL to first results page

str	URL to first results page

7.1.13 Search.link_last

Returns URL to last results page

str	URL to last results page

7.2 Methods

Methods exposed by the `Search` class.

7.2.1 Search.get_page()

Returns the requested search results page with the current *query* and *per_page* parameters. The returned page may not contain *entries* if the page is out of boundaries.

Parameters

Argument	Type	Optional/Required	Notes
page	number	required	Page number to retrieve.

Returns

Object	Instance of class `Search`

7.2.2 Search.get_next_page()

Returns next available search results page with the current *query*, *page*, and *per_page* parameters. Returns *None* if no page is available.

Returns

Object	Instance of class `Search` or *None*

Example

```
import pypexels
py_pexel = pypexels.PyPexels(api_key='YOUR_API_KEY')

#
#
search_results = py_pexel.search(query='red flowers', per_page=40)
while search_results is not None:
    print 'Current page number %s' % search_results.page
    search_results = search_results.get_next_page()
```

7.2.3 Search.get_previous_page()

Returns previous available search results page with the current *query*, *page*, and *per_page* parameters. Returns *None* if no page is available.

Returns

Object	Instance of class `Search` or *None*

Example

```
import pypexels
py_pexel = pypexels.PyPexels(api_key='YOUR_API_KEY')

#
#
search_results = py_pexel.search(query='red flowers', page=3, per_page=40)
while search_results is not None:
    print 'Current page number %s' % search_results.page
    search_results = search_results.get_previous_page()
```

7.2.4 Search.get_first_page()

Returns first search results page with the current *query*, *page*, and *per_page* parameters. Returns *None* if no page is available.

Returns

Object	Instance of class `Search` or *None*

Example

```
import pypexels
py_pexel = pypexels.PyPexels(api_key='YOUR_API_KEY')

#
#
search_results = py_pexel.search(query='red flowers', page=3, per_page=40)
print 'Current page number %s' % search_results.page
# To something with search_results

# Go back to first page
search_results = search_results.get_first_page():
print 'Current page number %s' % search_results.page
```

7.2.5 Search.get_last_page()

Returns last search results page with the current *query*, *page*, and *per_page* parameters. Returns *None* if no page is available.

Returns

Object	Instance of class `Search` or *None*

Example

```
import pypexels
py_pexel = pypexels.PyPexels(api_key='YOUR_API_KEY')

#
#
search_results = py_pexel.search(query='red flowers', per_page=40)

# Go to last results page
search_results = search_results.get_last_page():
print 'Current page number %s' % search_results.page
```

CHAPTER 8

API: Class Curated

This class is used to access the **curated** photos on the `PyPexels /curated/` REST API.

It is returned as result from a call to `PyPexels.curated(page, per_page)`

8.1 Properties

Properties exposed by the `Curated` class.

8.1.1 Curated.entries

Iterator for the returned objects contained in this `Curated` instance. Each entry will be an instance of class `Photo`.

iterator	each time an instance of class `Photo`

Example

```
import pypexels
py_pexel = pypexels.PyPexels(api_key='YOUR_API_KEY')

#
#
curated_photos_page = py_pexel.curated(per_page=40)
for photo in curated_photos_page.entries:
    print(photo.id, photo.photographer, photo.url)
    # no need to specify per_page: will take from original object
    curated_photos_page = curated_photos_page.get_next_page()
curated_results = py_pexel.curated(per_page=40)
for photo in curated_results.entries:
    print(photo.id, photo.photographer, photo.url)
```

8.1.2 Curated.page

Current curated photos page number.

int	current curated photos page number

8.1.3 Curated.per_page

Current curated photos per_page value.

int	current curated photos per_page value

8.1.4 Curated.has_previous

Returns boolean **True** or **False** depending on whether the current results page has a previous page to navigate to or not.

boolean	presence of previous page results

8.1.5 Curated.has_next

Returns boolean **True** or **False** depending on whether the current results page has a next page to navigate to or not.

boolean	presence of next page results

8.1.6 Curated.body

Returns JSON body of curated photos page.

JSON	JSON converted body of results page

8.1.7 Curated.headers

Returns response headers of curated photos page.

dict	headers of results page

8.1.8 Curated.link_self

Returns URL to current results page

str	URL to current results page

8.1.9 Curated.link_next

Returns URL to next results page

str	URL to next results page

8.1.10 Curated.link_previous

Returns URL to previous results page

str	URL to previous results page

8.1.11 Curated.link_first

Returns URL to first results page

str	URL to first results page

Note: `curated.total_results` always returns 0 (zero). `curated.link_last` always points to the first page.

8.2 Methods

Methods exposed by the `Curated` class.

8.2.1 Curated.get_page()

Returns the requested curated photos page with the current *query* and *per_page* parameters. The returned page may not contain *entries* if the page is out of boundaries.

Parameters

Argument	Type	Optional/Required	Notes
page	number	required	Page number to retrieve.

Returns

Object	Instance of class `Curated`

8.2.2 Curated.get_next_page()

Returns next available curated photos page with the current *query*, *page*, and *per_page* parameters. Returns *None* if no page is available.

Returns

Object	Instance of class `Curated` or *None*

Example

```
import pypexels
py_pexel = pypexels.PyPexels(api_key='YOUR_API_KEY')

#
#
search_results = py_pexel.curated(query='red flowers', per_page=40)
while search_results is not None:
    print 'Current page number %s' % search_results.page
    search_results = search_results.get_next_page()
```

8.2.3 Curated.get_previous_page()

Returns previous available curated photos page with the current *query*, *page*, and *per_page* parameters. Returns *None* if no page is available.

Returns

Object	Instance of class `Curated` or *None*

Example

```
import pypexels
py_pexel = pypexels.PyPexels(api_key='YOUR_API_KEY')

#
#
search_results = py_pexel.curated(query='red flowers', page=3, per_page=40)
while search_results is not None:
    print 'Current page number %s' % search_results.page
    search_results = search_results.get_previous_page()
```

8.2.4 Curated.get_first_page()

Returns first curated photos page with the current *query*, *page*, and *per_page* parameters. Returns *None* if no page is available.

Returns

Object	Instance of class `Curated` or *None*

Example

```
import pypexels
py_pexel = pypexels.PyPexels(api_key='YOUR_API_KEY')

#
#
search_results = py_pexel.curated(query='red flowers', page=3, per_page=40)
print 'Current page number %s' % search_results.page
# To something with search_results

# Go back to first page
search_results = search_results.get_first_page():
print 'Current page number %s' % search_results.page
```

Note: `curated.get_last_page()` always returns the first page.

CHAPTER 9

API: Class Random

This class is used to emulate a **random** API using the `PyPexels /curated/` REST API.

It is returned as result from a call to `PyPexels.random(per_page)`

9.1 Properties

Properties exposed by the `Random` class.

9.1.1 Random.entries

Returns an iterator for the returned objects contained in this `Random` instance. Each entry will be an instance of class `Photo`.

iterator	each time an instance of class `Photo`

Example

```
import pypexels
py_pexel = pypexels.PyPexels(api_key='YOUR_API_KEY')

#
#
random_results = py_pexel.random(per_page=10)
for photo in random_results.entries:
    print(photo.id, photo.photographer, photo.url)
```

9.1.2 Random.per_page

Current random results per_page value.

int	current random results per_page value

9.1.3 Random.has_previous

Returns always boolean **False**

boolean	presence of previous page results

9.1.4 Random.has_next

Returns always boolean **True**

boolean	presence of next page results

9.1.5 IMPORTANT NOTE

Although this class will expose additional methods and properties from the `PyPexels.Curated` class, you should only rely upon and make use of the methods and properties listed above. Remember, this is a *convenience* class that provides some uniformity in behavior while emulating a random image generator. If you need to fully control the content and behavior of the classes, then revert to use `PyPexels.Curated` class, with `per_page=1` and `page=randint()` value directly.

CHAPTER 10

API: Class SinglePhoto

This class is used to create an individual `SinglePhoto` objects from its ID.

10.1 Properties

Properties exposed by the `SinglePhoto` class.

'entries'

10.1.1 SinglePhoto.entries

Iterator for the returned objects contained in this `SinglePhoto` instance. Each entry will be an instance of class `Photo`.

Note You should not use this class directly. Use PyPexels.single_photo() instead.

iterator	each time an instance of class `Photo`

Example

```
import pypexels
py_pexel = pypexels.PyPexels(api_key='YOUR_API_KEY')

#
#
photo = py_pexel.single_photo(photo_id=<ID>)
print(photo.id, photo.photographer, photo.url)
```

CHAPTER 11

API: Class Photo

This class is used to interact with individual `Photo` objects typically returned by `PyPexels` as part of either `Search` or `Popular` object entries.

11.1 Properties

Properties exposed by the `Photo` class.

'id', 'width', 'height', 'url', 'photographer', 'photographer_url', 'photographer_id', 'src', 'liked'

11.1.1 Photo.id

Unique identifier for this photo

int	Unique identifier for this photo

11.1.2 Photo.width

Original image size width

int	Original image size width

11.1.3 Photo.height

Original image size height

int	Original image size height

11.1.4 Photo.url

URL location of Pexels web page for this photo

str	Pexels.com page for this photo

11.1.5 Photo.photographer

Name of photographer or photo source

str	Name of photographer or photo source

11.1.6 Photo.photographer_url

URL of photographer page on Pexels

str	URL of photographer page on Pexels

11.1.7 Photo.photographer_id

Unique identifier for this photographer

int	Unique identifier for this photographer

11.1.8 Photo.src

Dictionary with direct links to the image in different sizes and resolutions.

dict	links to the image in different sizes and resolutions

The next table details the key and their corresponding image size attributes.

Image formats

Key	Description
original	The size of the original image is given with the attributes width and height.
large2x	This image has a maximum width of 1880px and a maximum height of 1300px. It has the aspect ratio of the original image.
large	This image has a maximum width of 940px and a maximum height of 650px. It has the aspect ratio of the original image.
medium	This image has a height of 350px and a flexible width. It has the aspect ratio of the original image.
small	This image has a height of 130px and a flexible width. It has the aspect ratio of the original image.
portrait	This image has a width of 800px and a height of 1200px.
landscape	This image has a width of 1200px and height of 627px.
tiny	This image has a width of 280px and height of 200px.

Example

```
import pypexels
py_pexel = pypexels.PyPexels(api_key='YOUR_API_KEY')

search_results = py_pexel.search(query='red flowers', per_page=40)
for photo in search_results.entries:
    print(photo.id, photo.photographer, photo.url)
    print photo.src.get('large')
    print photo.src.get('tiny')
```

11.1.9 Photo.liked

Unique identifier for this photographer

bool	Whether liked or not by this API user

11.2 Methods

Methods exposed by the `Photo` class.

'get_attribution()'

11.2.1 Photo.get_attribution(_format='str')

Generate and return a standard attribution string according to '_format' parameter.

Parameters

Argument	Type	Optional/Required	Notes
_format	string	optional	Valid values: 'txt', 'html'

Returns

string	Text or HTML standard attribution string.

Example

```
import pypexels
py_pexel = pypexels.PyPexels(api_key='YOUR_API_KEY')

# Retrieve a single photo, known by its ID
photo = py_pexel.single_photo(photo_id=<ID>)
print(photo.get_attribution('txt'))
print(photo.get_attribution('html'))
```

CHAPTER 12

API: Class VideoPopular

This class is used to access the **popular** videos on the `PyPexels` `/videos/popular/` REST API.

It is returned as result from a call to `PyPexels.videos_popular(page, per_page)`

12.1 Properties

Properties exposed by the `VideosPopular` class.

12.1.1 VideosPopular.entries

Iterator for the returned objects contained in this `VideosPopular` instance. Each entry will be an instance of class `Video`.

iterator	each time an instance of class `Videos`

Example

```
import pypexels
py_pexel = pypexels.PyPexels(api_key='YOUR_API_KEY')

#
#
popular_videos_page = py_pexel.videos_popular(per_page=40)
while True:
    for video in popular_videos_page.entries:
        print(video.id, video.user.get('name'), video.url)
    if not popular_videos_page.has_next:
        break
    popular_videos_page = popular_videos_page.get_next_page()
```

12.1.2 VideosPopular.page

Current popular photos page number.

int	current popular photos page number

12.1.3 VideosPopular.per_page

Current popular photos per_page value.

int	current popular photos per_page value

12.1.4 VideosPopular.has_previous

Returns boolean **True** or **False** depending on whether the current results page has a previous page to navigate to or not.

boolean	presence of previous page results

12.1.5 VideosPopular.has_next

Returns boolean **True** or **False** depending on whether the current results page has a next page to navigate to or not.

boolean	presence of next page results

12.1.6 VideosPopular.body

Returns JSON body of popular photos page.

JSON	JSON converted body of results page

12.1.7 VideosPopular.headers

Returns response headers of popular photos page.

dict	headers of results page

12.1.8 VideosPopular.link_self

Returns URL to current results page

str	URL to current results page

12.1.9 VideosPopular.link_next

Returns URL to next results page

str	URL to next results page

12.1.10 VideosPopular.link_previous

Returns URL to previous results page

str	URL to previous results page

12.1.11 VideosPopular.link_first

Returns URL to first results page

str	URL to first results page

Note: `VideosPopular.total_results` always returns 0 (zero). `VideosPopular.link_last` always points to the first page.

12.2 Methods

Methods exposed by the `VideosPopular` class.

12.2.1 VideosPopular.get_page()

Returns the requested popular photos page with the current *query* and *per_page* parameters. The returned page may not contain *entries* if the page is out of boundaries.

Parameters

Argument	Type	Optional/Required	Notes
page	number	required	Page number to retrieve.

Returns

Object	Instance of class `VideosPopular`

12.2.2 VideosPopular.get_next_page()

Returns next available popular photos page with the current *query*, *page*, and *per_page* parameters. Returns *None* if no page is available.

Returns

Object	Instance of class `VideosPopular` or *None*

Example

```
import pypexels
py_pexel = pypexels.PyPexels(api_key='YOUR_API_KEY')

#
#
search_results = py_pexel.videos_popular(query='red flowers', per_page=40)
while search_results is not None:
    print 'Current page number %s' % search_results.page
    search_results = search_results.get_next_page()
```

12.2.3 VideosPopular.get_previous_page()

Returns previous available popular photos page with the current *query*, *page*, and *per_page* parameters. Returns *None* if no page is available.

Returns

Object	Instance of class `VideosPopular` or *None*

Example

```
import pypexels
py_pexel = pypexels.PyPexels(api_key='YOUR_API_KEY')

#
#
search_results = py_pexel.videos_popular(query='red flowers', page=3, per_
↪page=40)
while search_results is not None:
    print 'Current page number %s' % search_results.page
    search_results = search_results.get_previous_page()
```

12.2.4 VideosPopular.get_first_page()

Returns first popular photos page with the current *query*, *page*, and *per_page* parameters. Returns *None* if no page is available.

Returns

Object	Instance of class `VideosPopular` or *None*

Example

```
import pypexels
py_pexel = pypexels.PyPexels(api_key='YOUR_API_KEY')

#
#
search_results = py_pexel.videos_popular(query='red flowers', page=3, per_
↪page=40)
print 'Current page number %s' % search_results.page
# To something with search_results

# Go back to first page
search_results = search_results.get_first_page():
print 'Current page number %s' % search_results.page
```

Note: `VideosPopular.get_last_page()` always returns the first page.

CHAPTER 13

API: Class VideosSearch

This class is used to access the results of a **search** on the `PyPexels /videos/search/` REST API.

It is returned as result from a call to `PyPexels.videos_search(query, page, per_page)`

13.1 Properties

Properties exposed by the `VideosSearch` class.

13.1.1 VideoSearch.entries

Returns an iterator for the returned objects contained in this `VideosSearch` instance. Each entry will be an instance of class `Video`.

iterator	each time an instance of class `Video`

Example

```
import pypexels
py_pexel = pypexels.PyPexels(api_key='YOUR_API_KEY')

#
#
search_results = py_pexel.video_search(query='red flowers', per_page=40)
for video in search_results.entries:
    print(video.id, video.user.get('name'), video.url)
```

13.1.2 VideoSearch.page

Current search results page number.

int	current search results page number

13.1.3 VideoSearch.per_page

Current search results per_page value.

int	current search results per_page value

13.1.4 VideoSearch.total_results

Total results count, number of photos found

int	total results count, number of photos found

13.1.5 VideoSearch.has_previous

Returns boolean **True** or **False** depending on whether the current results page has a previous page to navigate to or not.

boolean	presence of previous page results

13.1.6 VideoSearch.has_next

Returns boolean **True** or **False** depending on whether the current results page has a next page to navigate to or not.

boolean	presence of next page results

13.1.7 VideoSearch.body

Returns JSON body of search results page.

JSON	JSON converted body of results page

13.1.8 VideoSearch.headers

Returns response headers of search results page.

dict	headers of results page

13.1.9 VideoSearch.link_self

Returns URL to current results page

str	URL to current results page

13.1.10 VideoSearch.link_next

Returns URL to next results page

str	URL to next results page

13.1.11 VideoSearch.link_previous

Returns URL to previous results page

str	URL to previous results page

13.1.12 VideoSearch.link_first

Returns URL to first results page

str	URL to first results page

13.1.13 VideoSearch.link_last

Returns URL to last results page

str	URL to last results page

13.2 Methods

Methods exposed by the `VideosSearch` class.

13.2.1 VideoSearch.get_page()

Returns the requested search results page with the current *query* and *per_page* parameters. The returned page may not contain *entries* if the page is out of boundaries.

Parameters

Argument	Type	Optional/Required	Notes
page	number	required	Page number to retrieve.

Returns

Object	Instance of class `VideosSearch`

13.2.2 VideoSearch.get_next_page()

Returns next available search results page with the current *query*, *page*, and *per_page* parameters. Returns *None* if no page is available.

Returns

Object	Instance of class `VideosSearch` or *None*

Example

```
import pypexels
py_pexel = pypexels.PyPexels(api_key='YOUR_API_KEY')

#
#
search_results = py_pexel.video_search(query='red flowers', per_page=40)
while search_results is not None:
    print 'Current page number %s' % search_results.page
    search_results = search_results.get_next_page()
```

13.2.3 VideoSearch.get_previous_page()

Returns previous available search results page with the current *query*, *page*, and *per_page* parameters. Returns *None* if no page is available.

Returns

Object	Instance of class `VideosSearch` or *None*

Example

```
import pypexels
py_pexel = pypexels.PyPexels(api_key='YOUR_API_KEY')

#
#
search_results = py_pexel.video_search(query='red flowers', page=3, per_
↪page=40)
while search_results is not None:
    print 'Current page number %s' % search_results.page
    search_results = search_results.get_previous_page()
```

13.2.4 VideoSearch.get_first_page()

Returns first search results page with the current *query*, *page*, and *per_page* parameters. Returns *None* if no page is available.

Returns

Object	Instance of class `VideosSearch` or *None*

Example

```
import pypexels
py_pexel = pypexels.PyPexels(api_key='YOUR_API_KEY')

#
#
search_results = py_pexel.video_search(query='red flowers', page=3, per_
↪page=40)
print 'Current page number %s' % search_results.page
# To something with search_results

# Go back to first page
search_results = search_results.get_first_page():
print 'Current page number %s' % search_results.page
```

13.2.5 VideoSearch.get_last_page()

Returns last search results page with the current *query*, *page*, and *per_page* parameters. Returns *None* if no page is available.

Returns

Object	Instance of class `VideosSearch` or *None*

Example

```
import pypexels
py_pexel = pypexels.PyPexels(api_key='YOUR_API_KEY')

#
#
search_results = py_pexel.video_search(query='red flowers', per_page=40)

# Go to last results page
search_results = search_results.get_last_page():
print 'Current page number %s' % search_results.page
```

CHAPTER 14

API: Class SingleVideo

This class is used to create an individual `SingleVideo` objects from its ID.

14.1 Properties

Properties exposed by the `SingleVideo` class.

'entries'

14.1.1 SingleVideo.entries

Iterator for the returned objects contained in this `SingleVideo` instance. Each entry will be an instance of class `Video`.

Note You should not use this class directly. Use PyPexels.single_video() instead.

iterator	each time an instance of class `Video`

Example

```
import pypexels
py_pexel = pypexels.PyPexels(api_key='YOUR_API_KEY')

#
#
video = py_pexel.single_video(video_id=<ID>)
print(video.id, video.user.get('name'), video.url)
```

CHAPTER 15

API: Class Video

This class is used to interact with individual `Video` objects typically returned by `PyPexels` as part of either `VideosSearch` or `VideosPopular` object entries.

15.1 Properties

Properties exposed by the `Video` class.

'id', 'width', 'height', 'url', 'image', 'full_res', 'tags', 'duration', 'user', 'video_files', 'video_pictures'.

15.1.1 Video.id

Unique identifier for this photo

int	Unique identifier for this video

15.1.2 Video.width

Original video size width

int	Original video size width

15.1.3 Video.height

Original video size height

int	Original video size height

15.1.4 Video.url

URL location of Pexels web page for this video

str	Pexels.com page for this video

15.1.5 Video.image

URL location of Pexels front image for this video

str	Pexels.com front image for this video

15.1.6 Video.full_res

Full resolution

null	Undocumented/unused

15.1.7 Video.tags

Tags (unused?)

list	List of tags (str)

15.1.8 Video.duration

Duration of video in seconds

int	Duration of video in seconds

15.1.9 Video.user

Information about the user uploading this video.

dict	Information of user uploading the video

15.1.10 Video.video_files

List of videos generated from resampling/resizing the original. Each entry is a dict with keys (id, quality, file_type, width, height, link).

list	List of resampled video files

15.1.11 Video.video_pictures

DEPRECATED

15.2 Methods

Methods exposed by the `Video` class.

'get_attribution()'

15.2.1 Video.get_attribution(_format='str')

Generate and return a standard attribution string according to '_format' parameter.

Parameters

Argument	Type	Optional/Required	Notes
_format	string	optional	Valid values: 'txt', 'html'

Returns

string	Text or HTML standard attribution string.

Example

```
import pypexels
py_pexel = pypexels.PyPexels(api_key='YOUR_API_KEY')

# Retrieve a single video, known by its ID
video = py_pexel.single_video(video_id=<ID>)
print(video.get_attribution('txt'))
print(video.get_attribution('html'))
```

CHAPTER 16

Version

PyPexels v1.0.0rc1 (release candidate, v1)

This is the first release candidate for production ready 1.0.0 of PyPexels. Thanks to all the early adopters, there have been a number of improvements and bugfixes and now should be a good time to start the rc process.

This release introduces some good functionality, in particular:

- Support for Videos API: search, popular
- Introduction of SinglePhoto and SingleVideo * This allows instantiating a *Photo* or *Video* object using a *photo_id* or *video_id*
- Introduction of *get_attribution()* method on *Photo* and *Video* classes * This simplifies crediting the author
- Fix some documentation issues
- Extend Travis and Tox test coverage to include Python 3.6, 3.7 and 3.8

Note that using this library you still need to abide to Pexels Guidelines, which are explained on Pexels API page

CHAPTER 17

License

PyPexels is released under the MIT License.