python-pexels

Release 1.0

Joker Hacker

Jan 03, 2023

CONTENTS:

CHAPTER

ONE

INTRODUCTION

A unofficial python library to support all of Pexels API features.

- Supports all of the endpoints, which are available in API v1.0
- Return data are python objects.
- User friendly library.
- Easy to use.

The Pexels API enables programmatic access to the full Pexels content library, including photos, videos. All content is available free of charge, and you are welcome to use Pexels content for anything you'd like, as long as it is within our Guidelines.

I just wrapped whole API into a Python library, others can use it with ease. All of the credits goes to the Pexels team for making such big place where free stock photos, royalty free images & videos shared by creators.

CHAPTER

TWO

GUIDELINES

2.1 Credits

- Whenever you are doing an API request make sure to show a prominent link to Pexels. You can use a text link (e.g. "Photos provided by Pexels") or a link with our logo.
- Always credit our photographers when possible (e.g. "Photo by John Doe on Pexels" with a link to the photo page on Pexels).

2.2 Warning

- You may not copy or replicate core functionality of Pexels (including making Pexels content available as a wallpaper app).
- Abuse of the Pexels API, including but not limited to attempting to work around the rate limit, will lead to termination of your API access.

2.3 API Usage

Do not abuse the API. By default, the API is rate-limited to 200 requests per hour and 20,000 requests per month. You may contact us to request a higher limit, but please include examples, or be prepared to give a demo, that clearly shows your use of the API with attribution. If you meet our API terms, you can get unlimited requests for free.

Here are some basic rules to follow when using Pexels content:

- Identifiable people may not appear in a bad light or in a way that is offensive. This includes, for example, portraying people pictured engaging in criminal activities, suffering from a medical ailment, or in a pornographic context.
- Don't sell unaltered copies of a photo, e.g. don't sell it as a stock photo, poster, print, or on a physical product without adding any value.
- Don't imply endorsement of your product by people or brands on the image.
- Don't redistribute or sell the photos on other stock photo or wallpaper platforms.

CHAPTER

THREE

INSTALLATION

You can install the library via *pip*

```
pip install python-pexels
```

Or using source code, which is not recommended.

```
git clone https://github.com/jk6521/py-pexels-api.git

cd py-pexels-api

python setup.py install
```

Caution: Make sure you have latest version of pip installed, else install using following commands.

Linux

```
sudo apt install python-pip
```

Windows

```
curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py

python get-pip.py
```

CHAPTER

FOUR

USAGE AND EXAMPLE

You need API token to access Pexels API, you can get it here

Once library is installed using pip, create a new python file named main.py

main.py

```
from Pexels import Client

client = Client(token="abcd1223")

#search photos
photos = client.search_photo(query="Joker")

#you can access return data like
print(photos.total_results)

>>> 230
```

CHAPTER

FIVE

FREQUENTLY ASKED QUESTIONS

You can find most of the common FAQ's here, some are from Pexels help.

5.1 What is Pexels?

Pexels is a free stock photo and video website and app that helps designers, bloggers, and everyone who is looking for visuals to find great photos and videos that can be downloaded and used for free. If you see a photo or video you like, simply download it for free (no strings attached!).

5.2 What steps can I take to avoid hitting the rate limit?

All API keys come with a default limit of 20,000 requests per month. This is sufficient for most use cases but here are a few tips to making the most of your requests.

- Make requests that return more results at once. All the methods that return multiple objects can be configured with *page* and *per_page* parameters for pagination. The maximum value of *per_page* is 80. You should request for as much data as necessary for your use case.
- Implement your own cache of responses from the Pexels API. URLs to our content returned from the API should not change over the short term, so serving them from your own cache would allow you to use Pexels content without having to use up your quota. 24 hours is a good amount of time to cache responses for.
- If you are making searches based on your users' input, normalize the searches. A search for *cat* will return the same response as a search for *cat*. Use this in conjunction with the caching described above.

5.3 Do I have to pay for higher limits?

Nope! The API is free of charge. If you're likely to hit the default request limits and you're able to provide and show acceptable attribution to Pexels and our contributors, limits can be lifted free of charge.

5.4 Got more questions, where to ask?

I have made a telegram group where you can ask your queries. Telegram Channel

CHAPTER

SIX

API REFERENCE

A unofficial python wrapper library for Pexels API

6.1 Client module

Client class for the Pexels Unofficial API Wrapper Author: Joker Hacker

class Pexels.client.**Client**(*token: str*, *base_endpoint: str = 'https://api.pexels.com/v1/'*, *video_endpoint: str = 'https://api.pexels.com/videos'*)

Bases: `object`

This object represents Client.

```
client = Client(token="abcde12345")
```

Note:

- Whenever you are doing an API request make sure to show a prominent link to Pexels.

You can use a text link (e.g. "Photos provided by Pexels") or a link with our logo.

- Always credit our photographers when possible (e.g. "Photo by John Doe on Pexels" with a link to the photo page on Pexels).
- Most Pexels API requests return multiple records at once. All of these endpoints are paginated,

and can return a maximum of 80 requests at one time. Each paginated request accepts the same parameters and returns the same pagination data in the response.

- The *prev_page* and *next_page* response attributes will only be returned if there is a corresponding page.

Warning:

- You may not copy or replicate core functionality of Pexels (including making Pexels content available as a wallpaper app).
- Do not abuse the API. By default, the API is rate-limited to 200 requests per hour and 20,000 requests per month.

Parameters

- **token** (`str`) – Unique authentication token.

- **base_endpoint** (str, optional) – Base endpoint of the API, defaults to https://api.pexels.com/v1/
- **video_endpoint** (str, optional) – Video endpoint of the API, defaults to https://api.pexels.com/videos

get_collection_media(*id: str*, *type: Optional[str] = ''*, *page: Optional[int] = 1*, *per_page: Optional[int] = 15*, ***kwargs*) → *CollectionMediaResponse*

This method returns all featured collections on Pexels.

Parameters

- **type** (str, optional) – The type of media you are requesting, If not given or if given with an invalid value, all media will be returned. Supported values are *photos* and *videos*
- **page** (int, optional) – The page number you are requesting. Default: 1
- **per_page** (int, optional) – The number of results you are requesting per page. Default: 15 Max: 80

Returns

Pexels.types.CollectionResponse

Raises

PexelsError – When *per_page* is above 80.

get_featured_collections(*page: Optional[int] = 1*, *per_page: Optional[int] = 15*, ***kwargs*) → *CollectionResponse*

This method returns all featured collections on Pexels.

Parameters

- **page** (int, optional) – The page number you are requesting. Default: 1
- **per_page** (int, optional) – The number of results you are requesting per page. Default: 15 Max: 80

Returns

Pexels.types.CollectionResponse

Raises

PexelsError – When *per_page* is above 80.

get_my_collections(*page: Optional[int] = 1*, *per_page: Optional[int] = 15*, ***kwargs*) → *CollectionResponse*

This method returns all of your collections.

Parameters

- **page** (int, optional) – The page number you are requesting. Default: 1
- **per_page** (int, optional) – The number of results you are requesting per page. Default: 15 Max: 80

Returns

Pexels.types.CollectionResponse

Raises

PexelsError – When *per_page* is above 80.

get_photo(*id: int*) → *Photo*

Retrieve a specific *Photo* from its id.

Parameters
id (int) – The id of the photo you are requesting.

Returns
Pexels.types.Photo

get_popular_videos(*min_width: Optional[int]*, *min_height: Optional[int]*, *min_duration: Optional[int]*, *max_duration: Optional[int]*, *page: Optional[int] = 1*, *per_page: Optional[int] = 15*, ***kwargs*) → *VideoResponse*

This method enables you to receive the current popular Pexels videos.

Parameters

- **min_width** (int, optional) – The minimum width in pixels of the returned videos.
- **min_height** (int, optional) – The maximum height in pixels of the returned videos.
- **min_duration** (int, optional) – The minimum duration in seconds of the returned videos.
- **max_duration** (int, optional) – The maximum duration in seconds of the returned videos.
- **page** (int, optional) – The page number you are requesting. Default: 1
- **per_page** (int, optional) – The number of results you are requesting per page. Default: 15 Max: 80

Returns
Pexels.types.VideoResponse

Raises
PexelsError – When *per_page* is above 80.

get_video(*id: int*) → *Video*

Retrieve a specific *Video* from its id.

Parameters
id (int) – The id of the video you are requesting.

Returns
Pexels.types.Video

search_curated_photo(*page: Optional[int] = 1*, *per_page: Optional[int] = 15*) → *PhotoResponse*

This method enables you to receive real-time photos curated by the Pexels team. We add at least one new photo per hour to our curated list so that you always get a changing selection of trending photos.

Parameters

- **page** (int, optional) – The page number you are requesting. Default: 1
- **per_page** (int, optional) – The number of results you are requesting per page. Default: 15 Max: 80

Returns
Pexels.types.PhotoResponse

Raises
PexelsError – When *per_page* is above 80.

search_photos(*query: str*, *orientation: Optional[str] = ''*, *size: Optional[str] = ''*, *color: Optional[str] = ''*, *locale: Optional[str] = ''*, *page: Optional[int] = 1*, *per_page: Optional[int] = 15*, ***kwargs*) → *PhotoResponse*

This method enables you to search photos for any topic that you would like. For example your query could

be something broad like *Nature*, *Tigers*, *People*. Or it could be something specific like Group of people working.

Parameters

- **query** (str) – The search query. *Ocean*, *Tigers*, *Pears*, etc.
- **orientation** (str, optional) – Desired photo orientation. list of supported orientations are available at *Pexels.constants.ORIENTATION*.
- **size** (str, optional) – Minimum photo size. list of supported sizes are available at *Pexels.constants.SIZE*.
- **color** (str, optional) – Desired photo color. list of supported colors are available at *Pexels.constants.COLOR*.
- **locale** (str, optional) – The locale of the search you are performing. list of supported locales are available at *Pexels.constants.LOCALE_SUPPORTED*.
- **page** (int, optional) – The page number you are requesting. Default: 1
- **per_page** (int, optional) – The number of results you are requesting per page. Default: 15 Max: 80

Returns

Pexels.types.PhotoResponse

Raises

PexelsError – When invalid *orientation* or *color* or *size* or *locale* given or when *per_page* is above 80.

search_videos(*query: str*, *orientation: str = ''*, *size: str = ''*, *locale: str = ''*, *page: Optional[int] = 3*, *per_page: Optional[int] = 15*, ***kwargs*) → *VideoResponse*

This method enables you to search Videos for any topic that you would like. For example your query could be something broad like *Nature*, *Tigers*, *People*. Or it could be something specific like Group of people working.

Parameters

- **query** (str) – The search query. *Ocean*, *Tigers*, *Pears*, etc.
- **orientation** (str, optional) – Desired Video orientation. list of supported orientations are available at *Pexels.constants.ORIENTATION*.
- **size** (str, optional) – Minimum Video size. list of supported sizes are available at *Pexels.constants.SIZE*.
- **locale** (str, optional) – The locale of the search you are performing. list of supported locales are available at *Pexels.constants.LOCALE_SUPPORTED*.
- **page** (int, optional) – The page number you are requesting. Default: 1
- **per_page** (int, optional) – The number of results you are requesting per page. Default: 15 Max: 80

Returns

Pexels.types.VideoResponse

Raises

PexelsError – When invalid *orientation* or *size* or *locale* given or when *per_page* is above 80.

6.2 types module

class Pexels.types.**Collection**(*id: str*, *title: str*, *description: str*, *private: bool*, *media_count: int*, *photos_count: int*, *videos_count: int*, ***kwargs*)

Bases: *PexelsType*

description: str

The description of the collection.

id: str

The id of the collection.

media_count: int

The total number of media included in this collection.

photos_count: int

The total number of photos included in this collection.

private: bool

Whether or not the collection is marked as private.

title: str

The name of the collection.

videos_count: int

The total number of videos included in this collection.

class Pexels.types.**CollectionMediaResponse**(*id: str*, *media: List[Union[*Photo, Video*]]*, *page: int*, *per_page: int*, *total_results: int*, *prev_page: Optional[str] = ''*, *next_page: Optional[str] = ''*, ***kwargs*)

Bases: *PexelsType*

id: str

The id of the collection you are requesting.

media: List[Union[*Photo*, *Video*]]

A list of media objects. Each object has an extra type attribute to indicate the type of object.

next_page: Optional[str]

URL for the next page of results, if applicable.

page: int

The current page number.

per_page: int

The number of results returned with each page.

prev_page: Optional[str]

URL for the previous page of results, if applicable.

total_results: int

The total number of results for the request.

class Pexels.types.**CollectionResponse**(*collections: List[*Collection*]*, *page: int*, *per_page: int*, *total_results: int*, *prev_page: Optional[str] = ''*, *next_page: Optional[str] = ''*, ***kwargs*)

Bases: *PexelsType*

collections: List[*Collection*]

A list of collection objects.

next_page: Optional[str]

URL for the next page of results, if applicable.

page: int

The current page number.

per_page: int

The number of results returned with each page.

prev_page: Optional[str]

URL for the previous page of results, if applicable

total_results: int

The total number of results for the request.

class Pexels.types.**PexelsType**

Bases: object

Base class for all pexels objects

class Pexels.types.**Photo**(*id: int*, *width: int*, *height: int*, *url: str*, *photographer: str*, *photographer_url: str*, *photographer_id: int*, *avg_color: str*, *src:* Src, *alt: str*, *type: str = 'Photo'*, ***kwargs*)

Bases: *PexelsType*

alt: str

Text description of the photo for use in the alt attribute.

avg_color: str

The average color of the photo. Useful for a placeholder while the image loads

height: int

The real height of the photo in pixels.

id: int

The id of the photo.

photographer: str

The name of the photographer who took the photo.

photographer_id: str

The id of the photographer.

photographer_url: str

The URL of the photographer's Pexels profile.

src: *Src*

An assortment of different image sizes that can be used to display this Photo.

type: str

The type of media to be shown in collections.

url: str

The Pexels URL where the photo is located.

width: int

The real width of the photo in pixels.

class Pexels.types.**PhotoResponse**(*photos: List[Photo]*, *page: int*, *per_page: int*, *total_results: int*, *prev_page: str = ''*, *next_page: str = ''*, ***kwargs*)

Bases: *PexelsType*

next_page

URL for the next page of results, if applicable

alias of `str`

page

The current page number

alias of `int`

per_page

The number of results returned with each page

alias of `int`

photos

A list of *Photo* object

alias of `List[Photo]`

prev_page

URL for the previous page of results, if applicable

alias of `str`

total_results

The total number of results for the request

alias of `int`

class Pexels.types.**Src**(*original: str*, *large: str*, *large2x: str*, *medium: str*, *small: str*, *portrait: str*, *landscape: str*, *tiny: str*, ***kwargs*)

Bases: *PexelsType*

landscape: str

The image cropped to W 1200px X H 627px.

large: str

The image resized to W 940px X H 650px DPR 1.

large2x: str

The image resized W 940px X H 650px DPR 2.

medium: str

The image resized W 940px X H 650px DPR 2.

original: str

The image without any size changes. It will be the same as the width and height attributes.

portrait: str

The image cropped to W 800px X H 1200px.

small: str

The image scaled proportionally so that it's new height is 130px.

tiny: str

The image cropped to W 280px X H 200px.

class Pexels.types.**User**(*id: int*, *name: str*, *url: str*, ***kwargs*)

Bases: *PexelsType*

id: int

The id of the videographer.

name: str

The name of the videographer.

url: str

The URL of the videographer's Pexels Profile.

class Pexels.types.**Video**(*id: int*, *width: int*, *height: int*, *url: str*, *image: str*, *duration: int*, *user: User*, *video_files: List[VideoFiles]*, *video_pictures: List[VideoPicture]*, *type: str = 'Video'*, ***kwargs*)

Bases: object

duration: int

The duration of the video in seconds.

height: int

The real height of the video in pixels.

id: int

The id of the video.

image: str

URL to a screenshot of the video.

type: str

The type of this media to be shown collections.

url: str

The pexels URL where the video is located.

user: *User*

The videographer who shot the video.

video_files: List[*VideoFiles*]

A list of different sized versions of the video.

video_pictures: List[*VideoPicture*]

A list of preview pictures of the video.

width: int

The real width of the video in pixels.

class Pexels.types.**VideoFiles**(*id: int*, *quality: str*, *file_type: str*, *width: int*, *height: int*, *link: str*, ***kwargs*)

Bases: *PexelsType*

file_type: str

The video format of the *video_file*

height: int

The height of the *video_file* in pixels.

id: int

The id of the *video_file*

link: str

A link to where the *video_file* is hosted.

quality: str

The video quality of the *video_file*

width: int

The width of the *video_file* in pixels.

class Pexels.types.**VideoPicture**(*id: int*, *picture: str*, *nr: int*, ***kwargs*)

Bases: *PexelsType*

id: int

The id of the *video_picture*

nr: int

picture: str

A link to the preview image.

class Pexels.types.**VideoResponse**(*videos: List[*Video*]*, *url: str*, *page: int*, *per_page: int*, *total_results: int*, *prev_page: Optional[str] = ''*, *next_page: Optional[str] = ''*, ***kwargs*)

Bases: *PexelsType*

next_page: str

URL for the next page of results, if applicable.

page: int

The current page number.

per_page: int

The number of results returned with each page.

prev_page: str

URL for the previous page of results, if applicable.

total_results: int

The total number of results for the request.

url: str

The Pexels URL for the current search query.

videos: List[*Video*]

A list of *Video* objects.

6.3 constants module

All constants to be used are available here

Pexels.constants.**COLOR: List[str] = ['red', 'orange', 'yellow', 'green', 'turquoise', 'blue', 'violet', 'pink', 'brown', 'black', 'gray', 'white']**

Desired Photo color.

Pexels.constants.**LOCALE_SUPPORTED: List[str] = ['en-US', 'pt-BR', 'es-ES', 'ca-ES', 'de-DE', 'it-IT', 'fr-FR', 'sv-SE', 'id-ID', 'pl-PL', 'ja-JP', 'zh-TW', 'zh-CN', 'ko-KR', 'th-TH', 'nl-NL', 'hu-HU', 'vi-VN', 'cs-CZ', 'da-DK', 'fi-FI', 'uk-UA', 'el-GR', 'ro-RO', 'nb-NO', 'sk-SK', 'tr-TR', 'ru-RU']**

The locale of the search you are performing.

Pexels.constants.**ORIENTATION: List[str] = ['landscape', 'portrait', 'square']**

Desired photo orientation.

Pexels.constants.**SIZE: List[str] = ['large', 'medium', 'small']**

Minimum photo size.

6.4 errors module

Errors for PexelsAPI

exception Pexels.errors.**APIError**

Bases: *PexelsError*

Raises when an error occurs on the API

exception Pexels.errors.**InvalidTokenError**

Bases: *PexelsError*

Raises when given API Token is invalid

exception Pexels.errors.**PexelsError**

Bases: Exception

Base exception for this module

exception Pexels.errors.**QuotaExceedError**

Bases: *PexelsError*

Raises when total request limit for the monthly period ends

CHAPTER

SEVEN

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

p

INDEX

P

Q

S

T

U

V

W