pyimagesearch
be awesome at building image search engines

Building a Pokedex in Python: Scraping the Pokemon Sprites (Step 2 of 6)

by **Adrian Rosebrock** on March 24, 2014 in **Building a Pokedex**, **Examples of Image Search Engines**, **Tutorials**

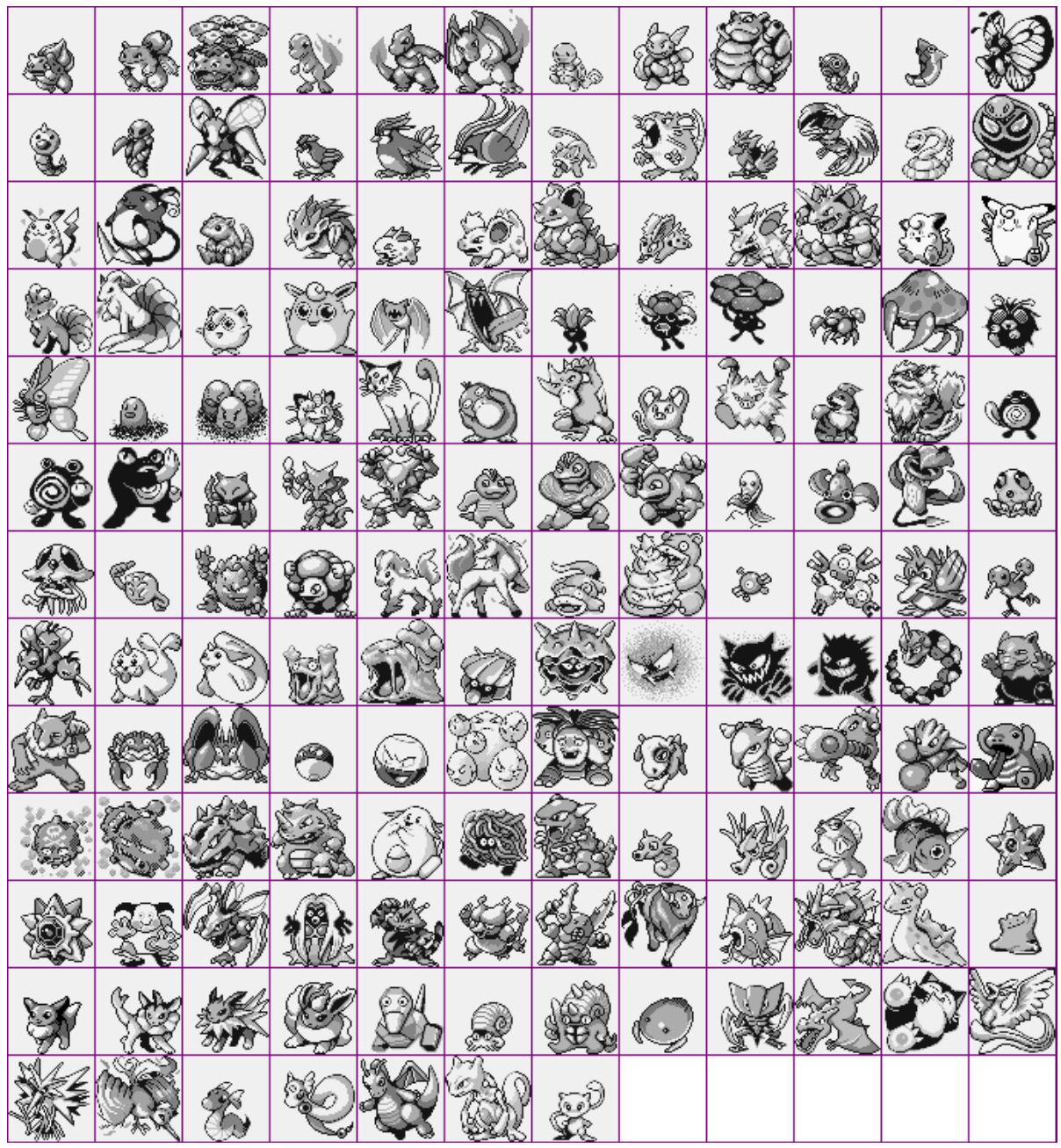


Figure 1: Our database of Pokemon Red, Blue, and Green sprites.

What if we could build a real life Pokedex?

You know, just like Ash Ketchum — point your Pokedex at a Pokemon (or in this case, snap a photo of a Pokemon), identify it, and get its stats.

> **Looking for the source code to this post?**
> **Jump right to the downloads section.**

While this idea has its roots in the Pokemon TV show, I'm going to show you how to make it a reality.

Previous Posts:

Before we get too far into detail, here are some previous posts you can look over for context and more detail on building our Pokedex:

- **Step 1:** Building a Pokedex in Python: Getting Started (Step 1 of 6)

Step 2: Scraping our Pokemon Database

Prior to even starting to build our Pokemon search engine, we first need to gather the data. And this post is dedicated to exactly that —

1 2

scraping tutorial; by the time you have finished

Our Data Source

I ended up deciding to scrape Pokemon DB because they have the some of the highest quality sprites that are easily accessible. And their HTML is nicely formatted and made it easy to download the Pokemon sprite images.

However, I cheated a little bit and copied and pasted the relevant portion of the webpage into a plaintext file. Here is a sample of some of the HTML:

Sample of Pokemon Database HTML — XHTML

```
<span class="infocard"><span class="infocard-img"><i class="pki" data-sprite="pkiAll n2"></i></span><span class="infocard-data
<span class="infocard"><span class="infocard-img"><i class="pki" data-sprite="pkiAll n3"></i></span><span class="infocard-data
<span class="infocard"><span class="infocard-img"><i class="pki" data-sprite="pkiAll n4"></i></span><span class="infocard-data
<span class="infocard"><span class="infocard-img"><i class="pki" data-sprite="pkiAll n5"></i></span><span class="infocard-data
<span class="infocard"><span class="infocard-img"><i class="pki" data-sprite="pkiAll n6"></i></span><span class="infocard-data
<span class="infocard"><span class="infocard-img"><i class="pki" data-sprite="pkiAll n7"></i></span><span class="infocard-data
<span class="infocard"><span class="infocard-img"><i class="pki" data-sprite="pkiAll n8"></i></span><span class="infocard-data
<span class="infocard"><span class="infocard-img"><i class="pki" data-sprite="pkiAll n9"></i></span><span class="infocard-data
<span class="infocard"><span class="infocard-img"><i class="pki" data-sprite="pkiAll n10"></i></span><span class="infocard-dat
<span class="infocard"><span class="infocard-img"><i class="pki" data-sprite="pkiAll n11"></i></span><span class="infocard-dat
<span class="infocard"><span class="infocard-img"><i class="pki" data-sprite="pkiAll n12"></i></span><span class="infocard-dat
<span class="infocard"><span class="infocard-img"><i class="pki" data-sprite="pkiAll n13"></i></span><span class="infocard-dat
...
```

You can download the full HTML file using the form at the bottom of this post.

Scraping and Downloading

Now that we have our raw HTML, we need to parse it and download the sprite for each Pokemon.

I'm a big fan of lots of examples, lots of code, so let's jump right in and figure out how we are going to do this:

Scraping and Downloading Pokemon Sprites using Python — Python

```
1  # import the necessary packages
2  from BeautifulSoup import BeautifulSoup
3  import argparse
4  import requests
5
6  # construct the argument parser and parse the arguments
7  ap = argparse.ArgumentParser()
8  ap.add_argument("-p", "--pokemon-list", required = True,
9      help = "Path to where the raw Pokemon HTML file resides")
10 ap.add_argument("-s", "--sprites", required = True,
11     help = "Path where the sprites will be stored")
12 args = vars(ap.parse_args())
```

Lines 2-4 handle importing the packages we will be using. We'll use BeautifulSoup to parse our HTML and requests to download the Pokemon images. Finally, argparse is used to parse our command line arguments.

Then, on **Lines 7-12** we parse our command line arguments. The switch --pokemon-list is the path to our HTML file that we are going to parse, while --sprites is the path to the directory where our Pokemon sprites will be downloaded and stored.

Now, let's extract the Pokemon names from the HTML file:

Scraping and Downloading Pokemon Sprites using Python — Python

```
14 # construct the soup and initialize the list of pokemon
15 # names
16 soup = BeautifulSoup(open(args["pokemon_list"]).read())
17 names = []
18
19 # loop over all link elements
20 for link in soup.findAll("a"):
21     # update the list of pokemon names
22     names.append(link.text)
```

On **Line 16** we use BeautifulSoup to parse our HTML — we simply load our HTML file off disk and then pass it into the constructor. BeautifulSoup takes care of the rest. **Line 17** then initializes the list to store our Pokemon names.

Then, we start to loop over all link elements on **Line 20**. The href attributes of these links point to a specific Pokemon. However, we do not need to follow each link. Instead, we just grab the inner text of the element. This text contains the name of our Pokemon.

Scraping and Downloading Pokemon Sprites using Python — Python

```
24 # loop over the pokemon names
25 for name in names:
26     # initialize the parsed name as just the lowercase
27     # version of the pokemon name
28     parsedName = name.lower()
29
30     # if the name contains an apostrophe (such as in
```

Free 21-day crash course on computer vision & image search engines

```
                                1                   2
34      # if the name contains a period followed by a space
35      # (as is the case with Mr. Mime), then replace it
36      # with a dash
37      parsedName = parsedName.replace(". ", "-")
38
39      # handle the case for Nidoran (female)
40      if name.find(u'\u2640') != -1:
41          parsedName = "nidoran-f"
42
43      # and handle the case for Nidoran (male)
44      elif name.find(u'\u2642') != -1:
45          parsedName = "nidoran-m"
```

Now that we have a list of Pokemon names, we need to loop over them (**Line 25**) and format the name correctly so we can download the file. Ultimately, the formatted and sanitized name will be used in a URL to download the sprite.

Let's examine each of these steps:

- **Line 28:** The first step to sanitizing the Pokemon name is to convert it to lowercase.
- **Line 32:** The first special case we need to handle is removing the apostrophe character. The apostrophe occurs in the name "Farfetch'd".
- **Line 37:** Then, we need to replace the occurrence of a period and space. This happens in the name "Mr. Mime". Notice the ". " in the middle of the name. This needs to be removed.
- **Lines 40-45:** Now, we need to handle unicode characters that occur in the Nidoran family. The symbols for "male" and "female" are used in the actual game, but in order to download the sprite for the Nidorans, we need to manually construct the filename.

Now, we can finally download the Pokemon sprite:

Scraping and Downloading Pokemon Sprites using Python — Python

```
47      # construct the URL to download the sprite
48      print "[x] downloading %s" % (name)
49      url = "http://img.pokemondb.net/sprites/red-blue/normal/%s.png" % (parsedName)
50      r = requests.get(url)
51
52      # if the status code is not 200, ignore the sprite
53      if r.status_code != 200:
54          print "[x] error downloading %s" % (name)
55          continue
56
57      # write the sprite to file
58      f = open("%s/%s.png" % (args["sprites"], name.lower()), "wb")
59      f.write(r.content)
60      f.close()
```

Line 49 constructs the URL of the Pokemon sprite. The base of the URL is http://img.pokemondb.net/sprites/red-blue/normal/ — we finish building the URL by appending the name of the Pokemon plus the ".png" file extension.

Downloading the actual image is handled on a single line (**Line 50**) using the requests package.

Lines 53-55 check the status code of the request. If the status code is not 200, indicating that the download was not successful, then we handle the error and continue looping over the Pokemon names.

Finally **Lines 58-60** saves the sprite to file.

Running Our Scrape

Now that our code is complete, we can execute our scrape by issuing the following command:

Scraping and Downloading Pokemon Sprites using Python — Shell

```
1 $ python parse_and_download.py --pokemon-list pokemon_list.html --sprites sprites
```

This script assumes that the file that containing the Pokemon HTML is stored in pokemon_list.html and the downloaded Pokemon sprites will be stored in the sprites directory.

After the script has finished running, you should have a directory full of Pokemon sprites:

Free 21-day crash course on computer vision & image search engines

1 2

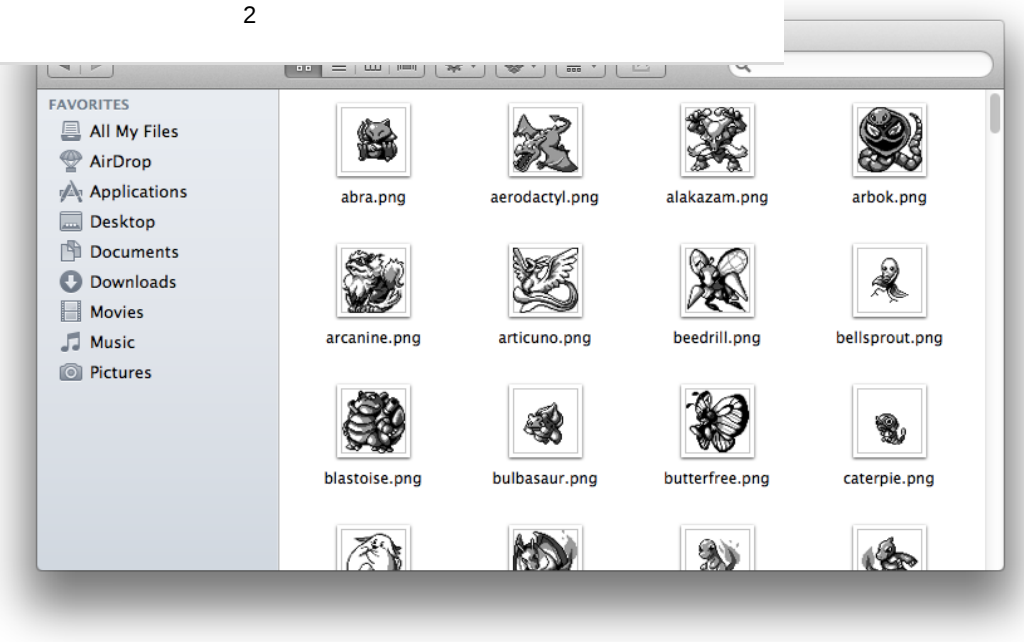


Figure 1: After parse_and_download.py has finished running, you should have a directory filled with Pokemon sprites, like this.

It's that simple! Just a little bit of code and some knowledge on how to scrape images, we can build a Python script to scrape Pokemon sprites in under 75 lines of code.

Note: *After I wrote this blog post, thegatekeeper07 suggested using the Veekun Pokemon Database. Using this database allows you to skip the scraping step and you can download a tarball of the Pokemon sprites. If you decide to take this approach, this is a great option; however, you might have to modify my source code a little bit to use the Veekun database. Just something to keep in mind!*

Summary

This post served as a Python web scraping tutorial: we downloaded sprite images for the original 151 Pokemon from the Red, Blue, and Green versions.

We made use of the BeautifulSoup and requests packages to download our Pokemon. These packages are essential to making scraping easy and simple, and keeping headaches to a minimum.

Now that we have our database of Pokemon, we can index them and characterize their shape using shape descriptors. We'll cover that in the next blog post.

If you would like to receive an email update when posts in this series are released, please enter your email address in the form below:

Downloads:

If you would like to download the code and images used in this post, please enter your email address in the form below. Not only will you get a .zip of the code, I'll also send you a **FREE 11-page Resource Guide** on Computer Vision and Image Search Engines, including **exclusive techniques** that I don't post on this blog! Sound good? If so, enter your email address and I'll send you the code immediately!

Email address:

Your email address

DOWNLOAD THE CODE!

Free 21-day crash course on computer vision & image search engines

1 2

Enter your email address below to get my **free 11-page Image Search Engine Resource Guide PDF**. Uncover **exclusive techniques** that I don't publish on this blog and start building image search engines of your own!

Your email address

DOWNLOAD THE GUIDE!

BeautifulSoup, **blue version**, **database**, **green version**, **pokedex**, **pokemon**, **red version**, **requests**, **scraping**

‹ Building a Pokedex in Python: Getting Started (Step 1 of 6) Building a Pokedex in Python: Indexing our Sprites using Shape Descriptors (Step 3 of 6) ›

11 Responses to *Building a Pokedex in Python: Scraping the Pokemon Sprites (Step 2 of 6)*

Matt Gathu March 26, 2014 at 12:25 pm # REPLY

Great post!! I did the scraping like a boss!! □

Looking forward to the next post.

Adrian Rosebrock March 26, 2014 at 1:58 pm # REPLY

Glad you liked it!

Divyansh May 24, 2016 at 9:00 am # REPLY

Brilliant.

Jeni February 12, 2017 at 7:41 am # REPLY

Thanks a lot!! Very informative article.

Adrian Rosebrock February 13, 2017 at 1:43 pm # REPLY

Thanks Jeni!

Rohit May 9, 2017 at 5:38 am # REPLY

Thanks Adrian

Finally learnt a bit of BeautifulSoup after reading this blog.The website structure has changed I guess. Hence took me a while to figure out how to scrape the website

Thanks again

Free 21-day crash course on computer vision & image search engines

1 2 REPLY

BeautifulSoup is a great package, I definitely encourage readers to play with and use it. Great job re-scraping the website!

Trackbacks/Pingbacks

Building a Pokedex in Python: Indexing our Sprites using Shape Descriptors (Step 3 of 6) - PyImageSearch - April 7, 2014
[…] Step 2: Building a Pokedex in Python: Scraping the Pokemon Sprites (Step 2 of 6) […]
Building a Pokedex in Python: Finding the Game Boy Screen (Step 4 of 6) - PyImageSearch - April 22, 2014
[…] Step 2: Building a Pokedex in Python: Scraping the Pokemon Sprites (Step 2 of 6) […]
Python and OpenCV Example: Warp Perspective and Transform - May 5, 2014
[…] Step 2: Building a Pokedex in Python: Scraping the Pokemon Sprites (Step 2 of 6) […]
Comparing Shape Descriptors for Similarity using Python and OpenCV - May 19, 2014
[…] explored what it takes to build a Pokedex using computer vision. Then we scraped the web and built up a database of Pokemon. We've indexed our database of Pokemon sprites using Zernike moments. We've analyzed […]

Leave a Reply

Name (required)

Email (will not be published) (required)

Website

SUBMIT COMMENT

Resource Guide (it's totally free).

Click the button below to get my **free 11-page Image Search Engine Resource Guide PDF**. Uncover **exclusive techniques** that I don't publish on this blog and start building image search engines of your own.

Download for Free!

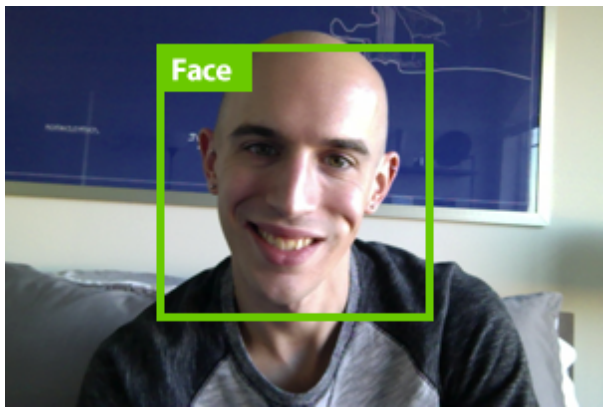
Deep Learning for Computer Vision with Python Book

You're interested in deep learning and computer vision, *but you don't know how to get started.* Let me h

Free 21-day crash course on computer vision & image search engines

CLICK HERE TO PRE-ORDER MY NEW BOOK

You can detect faces in images & video.



Are you interested in **detecting faces in images & video?** But **tired of Googling for tutorials** that *never work?* Then let me help! I guarantee that my new book will turn you into a **face detection ninja** by the end of this weekend. Click here to give it a shot yourself.

CLICK HERE TO MASTER FACE DETECTION

PyImageSearch Gurus: NOW ENROLLING!

The PyImageSearch Gurus course is *now enrolling!* Inside the course you'll learn how to perform:

- Automatic License Plate Recognition (ANPR)
- Deep Learning
- Face Recognition
- *and much more!*

Click the button below to learn more about the course, take a tour, and get 10 (FREE) sample lessons.

TAKE A TOUR & GET 10 (FREE) LESSONS

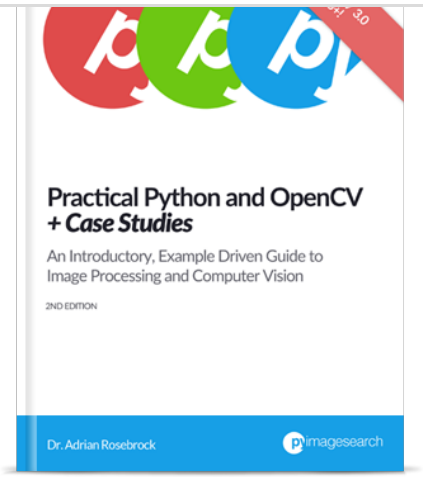
Hello! I'm Adrian Rosebrock.

I'm an entrepreneur and Ph.D who has launched two successful image search engines, ID My Pill and Chic Engine. I'm here to share my tips, tricks, and hacks I've learned along the way.

Learn computer vision in a single weekend.

Free 21-day crash course on computer vision & image search engines

1 2



Want to learn computer vision & OpenCV? I can teach you in a **single weekend**. I know. It sounds crazy, but it's no joke. My new book is your **guaranteed, quick-start guide** to becoming an OpenCV Ninja. So why not give it a try? Click here to become a computer vision ninja.

CLICK HERE TO BECOME AN OPENCV NINJA

Subscribe via RSS

Never miss a post! Subscribe to the PyImageSearch RSS Feed and keep up to date with my image search engine tutorials, tips, and tricks

POPULAR

Install OpenCV and Python on your Raspberry Pi 2 and B+
FEBRUARY 23, 2015

Home surveillance and motion detection with the Raspberry Pi, Python, OpenCV, and Dropbox
JUNE 1, 2015

Install guide: Raspberry Pi 3 + Raspbian Jessie + OpenCV 3
APRIL 18, 2016

How to install OpenCV 3 on Raspbian Jessie
OCTOBER 26, 2015

Basic motion detection and tracking with Python and OpenCV
MAY 25, 2015

Accessing the Raspberry Pi Camera with OpenCV and Python
MARCH 30, 2015

Install OpenCV 3.0 and Python 2.7+ on Ubuntu
JUNE 22, 2015

Search

Search...

Find me on **Twitter**, **Facebook**, **Google+**, and **LinkedIn**.
© 2017 PyImageSearch. All Rights Reserved.

Free 21-day crash course on computer vision & image search engines