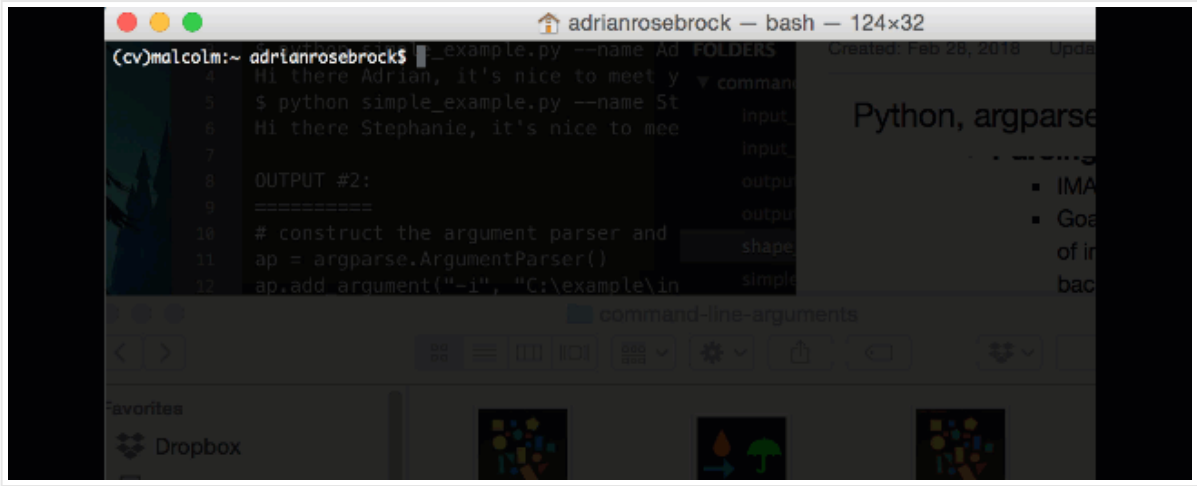
pyimagesearch
be awesome at building image search engines

Python, argparse, and command line arguments

by **Adrian Rosebrock** on March 12, 2018 in **Resources**, **Tutorials**



Today we are going to discuss a fundamental developer, engineer, and computer scientist skill — **command line arguments.**

Specifically, we'll be discussing:

- What are command line arguments
- Why we use command line arguments
- How to parse command line arguments with Python

Command line arguments are an elementary skill that you must learn how to use, especially if you are trying to apply more advanced computer vision, image processing, or deep learning concepts.

If you are new to command line arguments or do not know how to use them *that's okay!* **But you still need to take the time to educate yourself on how to use them** — this post will help you do exactly that.

By the end of today's post you will have a strong understanding of command line arguments, how they work, and how to use them.

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

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

Interested in computer vision and image search engines, but don't know where to start? Let me help. I've created a free, 21-day crash course that is hand-tailored to give you the best possible introduction to computer vision. Sound good? Enter your email below to start your journey to becoming a computer vision master.

Email Address

LET'S DO IT!

> *$ python detect_faces.py*
> *usage: detect_faces.py [-h] -i IMAGE -p PROTOTXT -m MODEL [-c CONFIDENCE]*
> *detect_faces.py: error: the following arguments are required: -i/–image, -p/–prototxt, -m/–model*
>
> *Help!*

Arjun is far from alone in struggling with this error.

Many other readers run into similar problems when working with command line arguments — **but the honest truth is that nearly *all* of these errors can be avoided by taking the time to educate yourself on command line arguments.**

Inside the rest of today's post you'll learn that command line arguments are *a lot* easier to work with than they seem (even if you have never used them before).

You'll find that you do not have to modify a single line of code to work with them. And by the end of the post you'll be able to work with command line arguments like a pro.

Let's get started.

What are command line arguments?

Command line arguments are flags given to a program/script at runtime. They contain additional information for our program so that it can execute.

Not all programs have command line arguments as not all programs need them. That being said, on this blog we make extensive use of command line arguments in our Python scripts and I'd even go so far to say that 98% of the articles on this blog make use of them.

Why do we use command line arguments?

As stated, command line arguments give additional information to a program at runtime.

This allows us to give our program different input on the fly *without changing the code*.

You can draw the analogy that a command line argument is similar to a function parameter. If you know how functions are declared and called in various programming languages, then you'll

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

Interested in computer vision and image search engines, but don't know where to start? Let me help. I've created a free, 21-day crash course that is hand-tailored to give you the best possible introduction to computer vision. Sound good? Enter your email below to start your journey to becoming a computer vision master.

Email Address

LET'S DO IT!

The argparse Python library

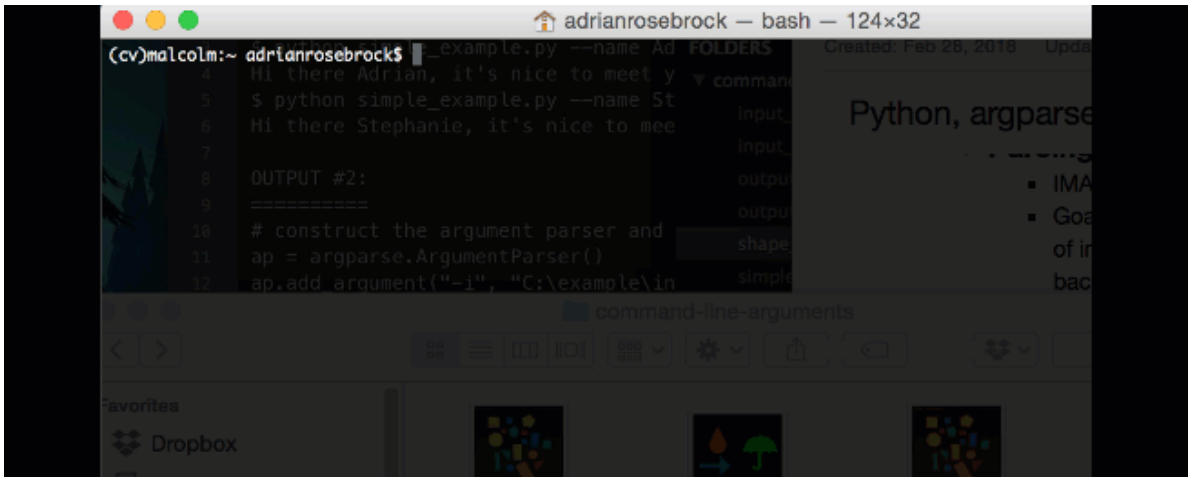


Figure 1: My terminal screen is used to run a Python script with command line arguments.

First, let's make a new script, naming it `simple_example.py` :

Python, argparse, and command line arguments — Python

```python
# import the necessary packages
import argparse

# construct the argument parse and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-n", "--name", required=True,
    help="name of the user")
args = vars(ap.parse_args())

# display a friendly message to the user
print("Hi there {}, it's nice to meet you!".format(args["name"]))
```

First, we need the `argparse` package, so we go ahead and import it on **Line 2.**

On **Line 5** we instantiate the `ArgumentParser` object as `ap` .

Then on **Lines 6 and 7** we add our only argument, `--name` . We must specify both *shorthand* (`-n`) and *longhand* versions (`--name`) where either flag could be used in the command line. This is a required argument as is noted by `required=True` .

The `help` string from **Line 7** will give additional information in the terminal if you need it. To view the command usage help you may enter the following in the terminal (output directly below):

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

Interested in computer vision and image search engines, but don't know where to start? Let me help. I've created a free, 21-day crash course that is hand-tailored to give you the best possible introduction to computer vision. Sound good? Enter your email below to start your journey to becoming a computer vision master.

Email Address

LET'S DO IT!

Line 8 of the script instructs Python and the `argparse` library to parse the command line arguments. I also call `vars` on the object to turn the parsed command line arguments into a Python dictionary where the *key* to the dictionary is the name of the command line argument and the value is *value* of the dictionary supplied for the command line argument. To see this in action I would suggest inserting a `print(args)` statement into the code.

While optional, I prefer converting the arguments object to a dictionary so I can execute the script via my command line or in a Jupyter Notebook. When using a Jupyter Notebook I can simply delete the command line arguments parsing code and insert a dictionary named `args` with any hardcoded values.

Now you're probably wondering: **How can I access the value from the command line argument argument?**

That's simple, and there's an example on **Line 11** of the script.

In the format string, we specify `args["name"]` . As you'll see shortly, we'll print our name to the screen dynamically with this command line argument.

Here are the steps we need to take in order to execute our `simple_example.py` script:

1. **Step 1:** Download the zip accompanying this blog post from the ***"Downloads"*** section into a location on your machine of your choosing (I would suggest the Desktop for sake of simplicity).
2. **Step 2:** Open a terminal and change directory to where the zip lives.
3. **Step 3:** Unzip.
4. **Step 4:** Change directory again, this time into the new directory that was just extracted.
5. **Step 5:** Execute the program (with command line arguments) and view the output.

On this blog, I show commands and their arguments in a "shell" codeblock. The `$` at the beginning of the prompt is your queue that this is a terminal command and you should enter the command after the `$` character along with your preferred arguments similar to or exactly as written.

I've taken care of **Step 1** by downloading the code to this lesson to my Desktop in my PyImageSearch directory for easy access.

From there, I entered the following commands and generated the accompanying output:

```
Python, argparse, and command line arguments                    Shell
1  $ cd ~/Desktop/PyImageSearch
2  $
```


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

Interested in computer vision and image search engines, but don't know where to start? Let me help. I've created a free, 21-day crash course that is hand-tailored to give you the best possible introduction to computer vision. Sound good? Enter your email below to start your journey to becoming a computer vision master.

Email Address

LET'S DO IT!

Step 2:

I *changed directory* to where I downloaded the zip for this lesson (**Line 1**).

I pressed the enter/return button on **Line 2** to make the output easier to read. This is optional.

Step 3:

I *unzipped the .zip file* associated with this lesson (**Line 3**).

The `...` on **Line 4** signifies that there was output from the unzipping process but I am not showing it here. Note that output doesn't have a preceding `$` .

Step 4:

Next I need to *change directory* into the folder that I just unzipped (**Line 6**).

Just to be sure I'm where I need to be, I *print my working directory* on **Line 7** with the output being shown on **Line 8**.

Step 5:

I *execute the command with argument* on **Line 10**. I'm specifying my name after the `--name` flag. As long as my name doesn't have any spaces, it will be displayed properly in the output.

The output is displayed on **Line 11.** Notice how the script *dynamically* shows my name exactly as I entered it in the command. Command line arguments are powerful and allow you to test your program with different input without changing a line of code.

Lines 13-17 demonstrate two additional examples that my script with no code changes will print a given name. Try it yourself with your own name…or maybe that of your nemesis.

***Note:** If I execute **Step 5** without command line arguments (or with incorrect ones), I'll see usage/error information printed as is shown.*

```
Python, argparse, and command line arguments                    Python
```

```
$ python simple_example.py
usage: simple_example.py [-h] -n NAME
simple_example.py: error: argument -n/--name is required
```


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

Interested in computer vision and image search engines, but don't know where to start? Let me help. I've created a free, 21-day crash course that is hand-tailored to give you the best possible introduction to computer vision. Sound good? Enter your email below to start your journey to becoming a computer vision master.

Email Address

LET'S DO IT!

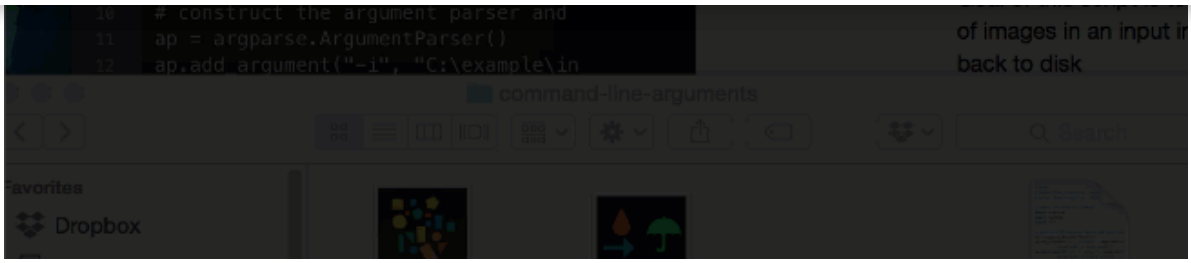


Figure 2: Using the argparse Python package you can easily parse command line arguments in the terminal/command line.

In this next example we'll be counting shapes in any given input image while annotating an output image that gets written to disk.

We'll be making use of command line arguments again to specify the input image path and the output image path.

The image processing technicals will be light for this explanation — after all we're just making this example for the purposes of command line arguments.

So let's create a new file called `shape_counter.py` and start coding:

Python, argparse, and command line arguments — Python

```
Codeblock #1: Lines 1-20# import the necessary packages
import argparse
import imutils
import cv2

# construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--input", required=True,
    help="path to input image")
ap.add_argument("-o", "--output", required=True,
    help="path to output image")
args = vars(ap.parse_args())

# load the input image from disk
image = cv2.imread(args["input"])

# convert the image to grayscale, blur it, and threshold it
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
blurred = cv2.GaussianBlur(gray, (5,5), 0)
thresh = cv2.threshold(blurred, 60, 255, cv2.THRESH_BINARY)[1]
```

We import `argparse` on **Line 2** — this is the package that will help us parse and access our command line arguments.

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

Interested in computer vision and image search engines, but don't know where to start? Let me help. I've created a free, 21-day crash course that is hand-tailored to give you the best possible introduction to computer vision. Sound good? Enter your email below to start your journey to becoming a computer vision master.

Email Address

LET'S DO IT!

Similarly, on **Lines 10 and 11**, we specify our `--output` argument which is also required.

From there we load the image using the path. Remember, the input image path is contained in `args["input"]`, so that is the parameter to `cv2.imread`.

Simple, right?

The rest of the lines are image processing specific, so if you've landed on this blog without any OpenCV or image processing skills, you might want to poke around in the archives for further explanations on these concepts.

On **Lines 18-20** we complete three operations:

1. Convert the `image` to grayscale.
2. Blur the grayscale image.
3. Threshold the `blurred` image.

We're ready to find and draw shape contours:

Python, argparse, and command line arguments — Python

```
22 # extract contours from the image
23 cnts = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL,
24     cv2.CHAIN_APPROX_SIMPLE)
25 cnts = cnts[0] if imutils.is_cv2() else cnts[1]
26
27 # loop over the contours and draw them on the input image
28 for c in cnts:
29     cv2.drawContours(image, [c], -1, (0, 0, 255), 2)
30
31 # display the total number of shapes on the image
32 text = "I found {} total shapes".format(len(cnts))
33 cv2.putText(image, text, (10, 20), cv2.FONT_HERSHEY_SIMPLEX, 0.5,
34         (0, 0, 255), 2)
35
36 # write the output image to disk
37 cv2.imwrite(args["output"], image)
```

On **Lines 23-25** we're finding shape contours in the `thresh` image.

From there, we draw the contours on the input image (**Lines 28 and 29**).

To learn more about contours, please see *Finding Shapes in Images using Python and OpenCV* and the contours tag archives. I also discuss contours and other image processing fundamentals in my

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

Interested in computer vision and image search engines, but don't know where to start? Let me help. I've created a free, 21-day crash course that is hand-tailored to give you the best possible introduction to computer vision. Sound good? Enter your email below to start your journey to becoming a computer vision master.

Email Address

LET'S DO IT!

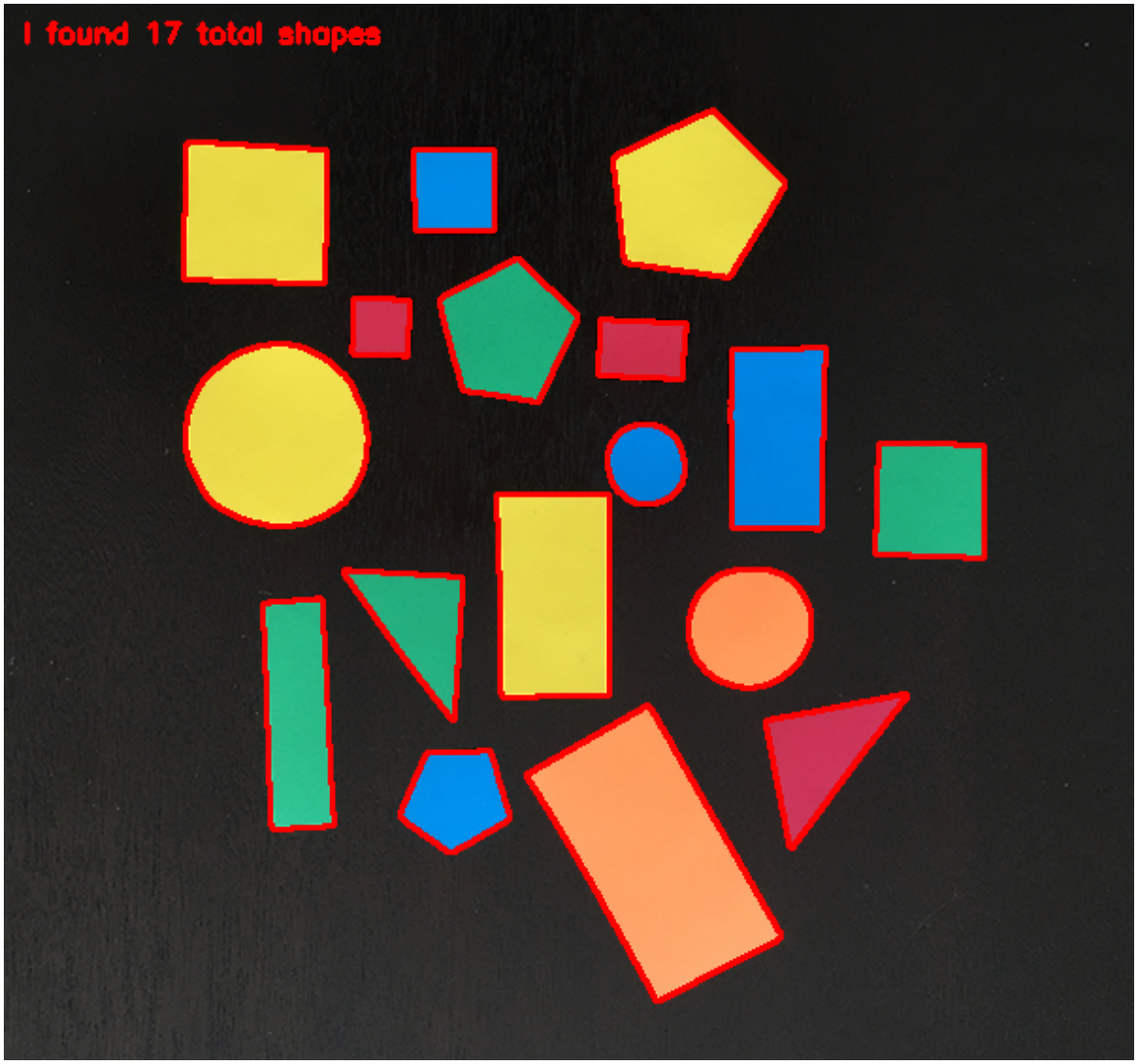


Figure 3: Shapes have been counted with our Python + OpenCV script which takes in command line arguments.

Let's execute the command again with different arguments:

```
Python, argparse, and command line arguments                    Python
1 $ python shape_counter.py --input input_02.png --output output_02.png
```

Again, you'll notice a new output file in your directory: `output_02.png` .

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

Interested in computer vision and image search engines, but don't know where to start? Let me help. I've created a free, 21-day crash course that is hand-tailored to give you the best possible introduction to computer vision. Sound good? Enter your email below to start your journey to becoming a computer vision master.

Email Address

LET'S DO IT!

Figure 4: Three shapes have been detected with OpenCV and Python by simply changing the command line arguments.

Now, take a step back. and consider what we have done from a command line arguments perspective.

What we did here is use *one script* with *no changes* and provided it *different arguments*. The `--input` argument contained the path/filename of the input image and likewise with `--output` .

The concept is extremely simple and I hope this has cleared up how to use command line arguments. Before we wrap up this post, let's take a look at what *not* to do.

How to *not* parse command line arguments

Every now and then I see readers who attempt to modify the code itself to accept command line arguments.

A great example of how not to parse command line arguments can be seen by starting with our command line arguments on **Lines 6-12** from the previous section:

Python, argparse, and command line arguments — Python

```
6  # construct the argument parser and parse the arguments
7  ap = argparse.ArgumentParser()
8  ap.add_argument("-i", "--input", required=True,
9      help="path to input image")
10 ap.add_argument("-o", "--output", required=True,
11     help="path to output image")
12 args = vars(ap.parse_args())
```

I've seen readers mistakenly try to update the argument parsing code to include the actual path to the input image:

Python, argparse, and command line arguments — Python

```
# construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-i", "C:\example\input_image.png", required=True,
    help="path to input image")
ap.add_argument("-o", "C:\example\output_image.png", required=True,
    help="path to output image")
args = vars(ap.parse_args())
```

Or in a list ditch effort, try to use the `help` parameter to include the file path:

```
Python, argparse, and command line arguments                    Python
1 # construct the argument parser and parse the arguments
2 ap = argparse.ArgumentParser()
```


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

Interested in computer vision and image search engines, but don't know where to start? Let me help. I've created a free, 21-day crash course that is hand-tailored to give you the best possible introduction to computer vision. Sound good? Enter your email below to start your journey to becoming a computer vision master.

Email Address

LET'S DO IT!

This is demonstrated in this excerpt from the PyImageSearch Gurus course Content Based Image Retrieval Module:

```
Python, argparse, and command line arguments                    Python
12 # construct the argument parser and parse the arguments
13 ap = argparse.ArgumentParser()
14 ap.add_argument("-d", "--dataset", required=True, help="Path to the directory of indexed ir
15 ap.add_argument("-f", "--features-db", required=True, help="Path to the features database")
16 ap.add_argument("-c", "--codebook", required=True, help="Path to the codebook")
17 ap.add_argument("-o", "--output", required=True, help="Path to output directory")
18 args = vars(ap.parse_args())
19
20 # load the codebook and open the features database
21 vocab = pickle.loads(open(args["codebook"], "rb").read())
22 featuresDB = h5py.File(args["features_db"], mode="r")
23 print("[INFO] starting distance computations...")
```

Notice on the highlighted lines that I've defined the argument as `--features-db` (with a dash), but I reference it by `args["features_db"]` (with an underscore). This is because the `argparse` Python library replaces dashes with underscores during the parsing.

You can do it!

Command line arguments may take some practice or getting used to if you are new to working in a terminal — **but I have faith in you!**

Don't get discouraged.

Take your time.

And keep practicing.

Before you know it, you'll have mastered command line arguments — **take the time now to invest in your education and this valuable skill!**

Setting your command line arguments via an IDE

From time to time I receive emails and blog comments asking how to run Python scripts from within their IDE.

About 90% of the time the question is similar to:

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

Interested in computer vision and image search engines, but don't know where to start? Let me help. I've created a free, 21-day crash course that is hand-tailored to give you the best possible introduction to computer vision. Sound good? Enter your email below to start your journey to becoming a computer vision master.

Email Address

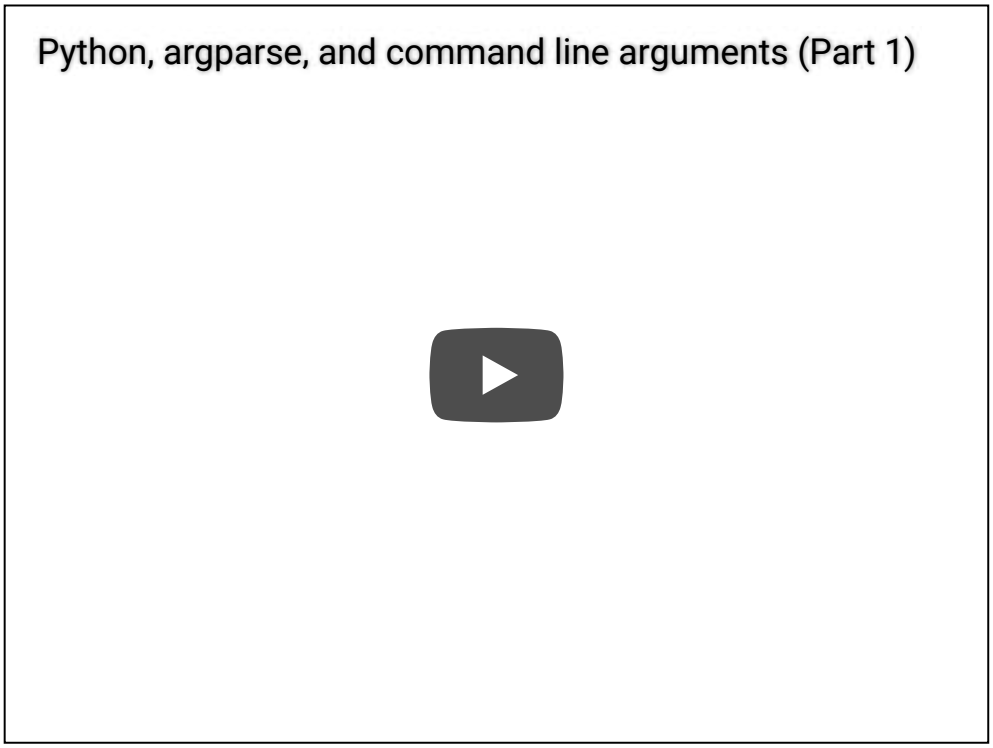
LET'S DO IT!

saves time to use the terminal rather than to click around the GUI of your IDE and set up the arguments.

In an effort to serve the community, I'm providing two screencasts that were captured by David Hoffman — you'll hear my familiar voice as I recorded the voiceover to explain the steps.

Step 1 is to grab the code to this tutorial (or another one on this blog if you are so inclined) via the ***"Downloads"*** section.

As you'll see in the video, David downloaded the code into a folder residing on his desktop. Once you've put the download somewhere convenient for you, press play and follow along:



David and I took it a step further to demonstrate that you can select a virtual environment for your PyCharm run configuration.

The second example in this blog post requires that you have OpenCV installed.

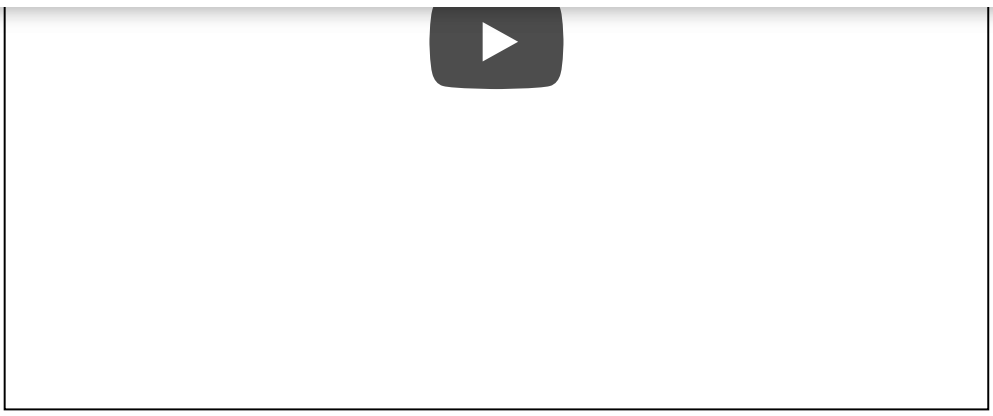
Since David was a student of the PyImageSearch Gurus course, he used his `gurus` virtual environment. You can select an environment on your own system that you've set up with OpenCV. If

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

Interested in computer vision and image search engines, but don't know where to start? Let me help. I've created a free, 21-day crash course that is hand-tailored to give you the best possible introduction to computer vision. Sound good? Enter your email below to start your journey to becoming a computer vision master.

Email Address

LET'S DO IT!



As you can see, both examples are relatively easy. PyCharm provides a convenient way to test code without using your terminal. I'm still partial to using the terminal as I've always got it open anyway for the many tasks that PyCharm can't handle.

Did your virtual environment not show up in PyCharm?

Don't worry!

You might just need to browse to the `~/.virtualenvs` folder and select your interpreter (all from within the PyCharm Run Configuration screen).

So, what's next?

I hope this tutorial helped you get up to speed with command line arguments and how to use them. Chances are, if you're reading this tutorial you may be a bit new to the Python programming language and the OpenCV library.

If so, that's totally okay!

I was in the same boat as you when I first started working with Python years and years ago. I would spend hours trying new examples, running code, re-coding the entire script, re-running it, and trying to reproduce the results.

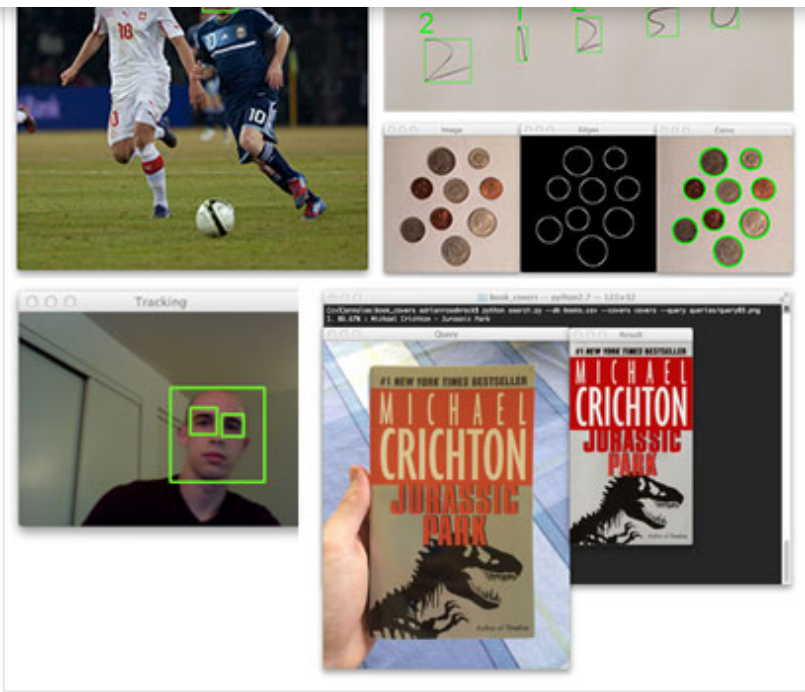
As you might imagine, I'm a big fan of **learning by example**, so a good next step would be to have some fun and read this blog post on detecting cats in images/videos. This tutorial is meant to be *very*

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

Interested in computer vision and image search engines, but don't know where to start? Let me help. I've created a free, 21-day crash course that is hand-tailored to give you the best possible introduction to computer vision. Sound good? Enter your email below to start your journey to becoming a computer vision master.

Email Address

LET'S DO IT!



Thousands of PyImageSearch readers have used *Practical Python and OpenCV* to learn *both* the fundamentals of the Python programming language *and* the OpenCV library at the same time. Each and every line of code in the book is thoroughly documented, ensuring you understand *exactly* what each piece of code is doing.

To learn more about my book, and how it can help you on your path to Python + OpenCV mastery, ***just click here.***

Summary

In today's blog post you learned how to utilize command line arguments using the Python programming language.

You also learned how to utilize the `argparse` library for parsing command line arguments.

At this point you should now understand the fundamentals of command line arguments and the role they play in software, programs, and scripts.

As you saw, when used correctly, you do not have to modify *any* code to work with command line arguments — just open up a terminal, navigate to your Python script, and execute it!

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

Interested in computer vision and image search engines, but don't know where to start? Let me help. I've created a free, 21-day crash course that is hand-tailored to give you the best possible introduction to computer vision. Sound good? Enter your email below to start your journey to becoming a computer vision master.

Email Address

LET'S DO IT!

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:

zhql0907@sjtu.edu.cn

DOWNLOAD THE CODE!

Resource Guide (it's totally free).

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!

basics, **command line arguments**, **contours**, **python**

‹ The 7 best deep learning books you should be reading right now

Reading barcodes with Python and OpenMV ›

28 Responses to *Python, argparse, and command line arguments*

Mansoor Nasir March 12, 2018 at 10:54 am #

REPLY

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

Interested in computer vision and image search engines, but don't know where to start? Let me help. I've created a free, 21-day crash course that is hand-tailored to give you the best possible introduction to computer vision. Sound good? Enter your email below to start your journey to becoming a computer vision master.

Email Address

LET'S DO IT!

code:

```python
# construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--input", required=True,
    help="path to input image")
ap.add_argument("-o", "--output", required=True,
    help="path to output image")
args = vars(ap.parse_args())
```

And replacing it with:

```python
args = {
    "input": "path/to/input_image.jpg",
    "output": "path/to/output_image.jpg"
}
```

This will replace the command line arguments with a hardcoded dictionary. The downside is that you will need to (1) ensure your keys to the dictionary matchup throughout the code and (2) you will need to edit the code whenever you want to change the file paths (which pretty much gets us back to why we are using command line arguments in the first place).

Also, if you are using Windows or want the code to be cross-platform with file paths, make sure you use “os.path.sep.join”.

Ahmadreza March 12, 2018 at 11:39 am #

REPLY

I think it’s will be good if you explain findContours arguments and hierarchy.

Can you please ?

Adrian Rosebrock March 12, 2018 at 3:10 pm #

REPLY

I’ve explained the cv2.findContours function in a good many PyImageSearch posts. Here’s a link to all posts that mention the function. It’s a good many as I frequently use the function in

posts. I would suggest starting there. The OpenCV docs has a good explanation of the hierarchy as well.

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

Interested in computer vision and image search engines, but don't know where to start? Let me help. I've created a free, 21-day crash course that is hand-tailored to give you the best possible introduction to computer vision. Sound good? Enter your email below to start your journey to becoming a computer vision master.

Email Address

LET'S DO IT!

Adrian Rosebrock March 12, 2018 at 3:12 pm #

REPLY

I used to use sys.argv as well but I found that the code would become too hard to maintain. At first you think you only need one argument. Then it turns out you need another. And another. And another. It snowballs. I prefer to use argparse out of the gate. It gives me the flexibility of extendible code from a command line perspective.

I think the other issue to address is that if you already know how to use sys.argv you can likely transition over to argparse without any real issue. If you don't have experience with command line arguments in general I think you would struggle either way. I use argparse here on PyImageSearch which is why I went with it instead of sys.argv. I hope that makes sense and thank you for the comment!

Đăng March 12, 2018 at 12:17 pm #

REPLY

Hi Adrian,
I have just read your email, and then go to this blog post.
I see this topic is very useful. I will try to learn how to adapt with this style of coding.
In fact, I didn't face some problems with Argument Parser since I comment out these part.
And change your code by my way. I had filled out all input path directly. ^^"
Anyway, I really appreciate your help and your sharing. For a beginner like me, I can't go around this huge world without some guideline from some experts like you.

Adrian Rosebrock March 12, 2018 at 3:13 pm #

REPLY

If you prefer to hardcode arguments and file paths, that's totally acceptable. However, I would encourage you, as well as other readers, to learn how to use the command line arguments as you'll find them to be quite powerful 🙂

Cipri_tom March 12, 2018 at 3:46 pm #

REPLY

Hey! Thanks for posting such nice examples! I've been a long time reader of your blog. Now I may have something to contribute back. Do you know Docopt? I find it much friendlier to use than argparse, although the latter has kind of become a standard.

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

Interested in computer vision and image search engines, but don't know where to start? Let me help. I've created a free, 21-day crash course that is hand-tailored to give you the best possible introduction to computer vision. Sound good? Enter your email below to start your journey to becoming a computer vision master.

Email Address

LET'S DO IT!

said, I totally agree there are better command line argument libraries (click, for example, is my favorite).

As for the comment box, I'm not sure why that would be. Perhaps it's the browser on your phone?

Tony March 12, 2018 at 5:23 pm #

REPLY

As usual, an excellent post.

Here are two additional examples of useful argparse idoms taken from some of my code, that illustrate a couple useful features.

First is the boolean setting, useful for options that are normally false, but you want to occasionally turn on. Using a boolean option is preferable to using an option that you pass in a "1" to to enable. For example,

parser.add_argument("-t", "–test", help="turn on test mode", action="store_true")

Since you create a dictionary from the output of ap.parse_args(), use it like this:

if args["test"]: # was -t/–test used?

(If you had used the output of ap.parse_args() directly, the code would look like this:

args = ap.parse_args()
if args.text: # was -t/–test used?
)

The other example is often used in a similar fashion, but lets you use it multiple times to provide more than one level of that option.

parser.add_argument("-v", "–verbose", help="Verbose mode, may be specified multiple times for additional verbosity", action="count", default=0)

Use it like this:

if args["verbose"]: # was -v/–verbose used?

or

if args["verbose"] > 1: # is extra verbosity needed?

or

if args[“verbose”] > 5: # give them everything possible

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

Interested in computer vision and image search engines, but don't know where to start? Let me help. I've created a free, 21-day crash course that is hand-tailored to give you the best possible introduction to computer vision. Sound good? Enter your email below to start your journey to becoming a computer vision master.

Email Address

LET'S DO IT!

creepy for me, maybe some OOP tutorial in near future?

Adrian Rosebrock March 13, 2018 at 6:36 am #

REPLY

OOP in Python can be a bit tricky depending on what you are doing. We don’t use too much OOP programming here, other than defining classes and methods inside them. We don’t do much in terms of inheritance which is where OOP gets tricky with Python. I would recommend you take a look at the list of resources I recommend to learn Python here as well as Dan Bader’s blog which includes many Python tutorials (his book is also really good).

Ian Walker March 12, 2018 at 11:57 pm #

REPLY

wow, just this morning I was pondering a way to wrapper commands and command line options (maybe via TCL/TK) either in a gui (graphical) or a text based gui (like sidekick [an olde TSR program for DOS] there that ages me!) the idea being reflection back from the help and man pages into a much better and forgiving way of configuring commands. With trenchant examples of phased intensity. The current (and has been so for 50 years) way of man pages calls to mind a passive aggressive crypto-sadistic neckbeard (of any sex) with a trunk full of BDSM porgs under their desk, squealing in unrelenting agony; you can never be smart enough to work out just wtf is going on. This is what powershell was meant to be, sort of a modern riff on DCL, a very consistent command language indeed. Alas the aforementioned dread neckbeards of unices of yore skull dragged it through the laboratories on the island of Dr Moreau, hitting all the ugly on way through… In actual fact, I think I shall return to a pet project of mine (no porg abuse intended) – the best IDE I ever used was ms Word, just the ability to use styles alone was worth it. And it got me away from the ridiculous old ASCII spaghetti paradigm that has cursed terminals and compilers since before dot …

Adrian Rosebrock March 13, 2018 at 6:37 am #

REPLY

wow, just this morning I was pondering a way to wrapper commands and command line options (maybe via TCL/TK) either in a gui (graphical)

I think you might be interested in Gooey which does exactly that.

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

Interested in computer vision and image search engines, but don't know where to start? Let me help. I've created a free, 21-day crash course that is hand-tailored to give you the best possible introduction to computer vision. Sound good? Enter your email below to start your journey to becoming a computer vision master.

Email Address

LET'S DO IT!

John W March 13, 2018 at 8:46 am #

REPLY

Very nice article. I have a minimal knowledge of Python compared to R but a goal is to boost my knowledge of Python and deep dive into the world of machine learning with Python. However, I just need to pull myself away from R. In other words, I need to diversify my knowledge of machine learning with other languages.

Rex Haw March 18, 2018 at 11:28 am #

REPLY

Greetings. As always your articles are amazing! I was experimenting with Pycharm and found that it's possible to enter command line arguments within Pycharm's Terminal window. I used the simple name example you provided and ran the code via Pycharm's Terminal window (directory of .py code:/ python –name somename) and it worked like a "charm." I was also able to make new changes to the code and run the code via Terminal and the output with changes was instant. This method seemed a little faster without having to go in the configuration menu and changing the parameters. Hope this makes sense and once again thank you for your excellent articles and sharing you knowledge with all of us.

Adrian Rosebrock March 19, 2018 at 5:10 pm #

REPLY

Awesome, thank you for sharing, Rex!

Azizhan March 26, 2018 at 4:44 pm #

REPLY

Hi Adrian

Im quite new in Python. I installed already imutils with sudo pip install imutils code. But I'm taking this kind of Error;

Traceback (most recent call last):
File “shape_counter.py”, line 32, in
import imutils

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

Interested in computer vision and image search engines, but don't know where to start? Let me help. I've created a free, 21-day crash course that is hand-tailored to give you the best possible introduction to computer vision. Sound good? Enter your email below to start your journey to becoming a computer vision master.

Email Address

LET'S DO IT!

2. Install “imutils” via “pip”

Using “sudo” will install the package into the system Python virtual environment which you do not want. Try this instead:

```python
$ workon your_env_name
$ pip install imutils
```

Guido April 6, 2018 at 1:25 pm #

REPLY

Hello Adrian,
I am trying to run a python script from the command line following your tutorial. My script load an image, perform some opencv methods and then write a csv file with some lines of text. The issue is that I am receiving an error:
[Errno 13] Permission denied: ‘D:\\aaa’
I am running the python script from the command line and from my conda environment. If I run the script (without the argparse) from Spyder everything works just fine. I guess there is a problem with permission to write a file ina directory. I am in Windows. Can you tell me how to solve this?

Guido April 6, 2018 at 1:51 pm #

REPLY

Sorry Adrian,
my mistake. I forgot to add “.csv” to the file path,
SOrry

Adrian Rosebrock April 10, 2018 at 12:47 pm #

REPLY

Congrats on resolving the issue, Guido!

Ahmad April 9, 2018 at 2:27 am #

REPLY

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

Interested in computer vision and image search engines, but don't know where to start? Let me help. I've created a free, 21-day crash course that is hand-tailored to give you the best possible introduction to computer vision. Sound good? Enter your email below to start your journey to becoming a computer vision master.

Email Address

LET'S DO IT!


```
2 $ pip install imutils
```

You should also double-check that imutils was installed using the `pip freeze` command.

Leave a Reply

Name (required)

Email (will not be published) (required)

Website

SUBMIT COMMENT

Search...

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.

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

Interested in computer vision and image search engines, but don't know where to start? Let me help. I've created a free, 21-day crash course that is hand-tailored to give you the best possible introduction to computer vision. Sound good? Enter your email below to start your journey to becoming a computer vision master.

Email Address

LET'S DO IT!

You're interested in deep learning and computer vision, *but you don't know how to get started.* Let me help. **My new book will teach you all you need to know about deep learning.**

CLICK HERE TO MASTER DEEP LEARNING

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

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

Interested in computer vision and image search engines, but don't know where to start? Let me help. I've created a free, 21-day crash course that is hand-tailored to give you the best possible introduction to computer vision. Sound good? Enter your email below to start your journey to becoming a computer vision master.

Email Address

LET'S DO IT!

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

Interested in computer vision and image search engines, but don't know where to start? Let me help. I've created a free, 21-day crash course that is hand-tailored to give you the best possible introduction to computer vision. Sound good? Enter your email below to start your journey to becoming a computer vision master.

Email Address

LET'S DO IT!

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

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

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

Ubuntu 16.04: How to install OpenCV
OCTOBER 24, 2016

Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry Pi
SEPTEMBER 4, 2017

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

Find me on **Twitter**, **Facebook**, **Google+**, and **LinkedIn**.

© 2018 PyImageSearch. All Rights Reserved.

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

Interested in computer vision and image search engines, but don't know where to start? Let me help. I've created a free, 21-day crash course that is hand-tailored to give you the best possible introduction to computer vision. Sound good? Enter your email below to start your journey to becoming a computer vision master.

Email Address

LET'S DO IT!