# **Automating my workflow with Python #1: File Organizer**

[#python](https://dev.to/t/python)[#100daysofcode](https://dev.to/t/100daysofcode)[#codenewbie](https://dev.to/t/codenewbie)[#productivity](https://dev.to/t/productivity)

This is my first article on Automating my workflow with Python.

In this (beginner-friendly) article, you will learn how you can make *your own Python 🐍 script* for organizing files for *your own workflow* just the way I did.

## **What problem did I fix? 🤔**

I tweet a lot of Python 🐍 snippets, make videos, and work on projects which require me to download a lot of *Images, Videos, and whatnot*!

I don't get time to organize my downloads 📂 and if I want to look for a file...

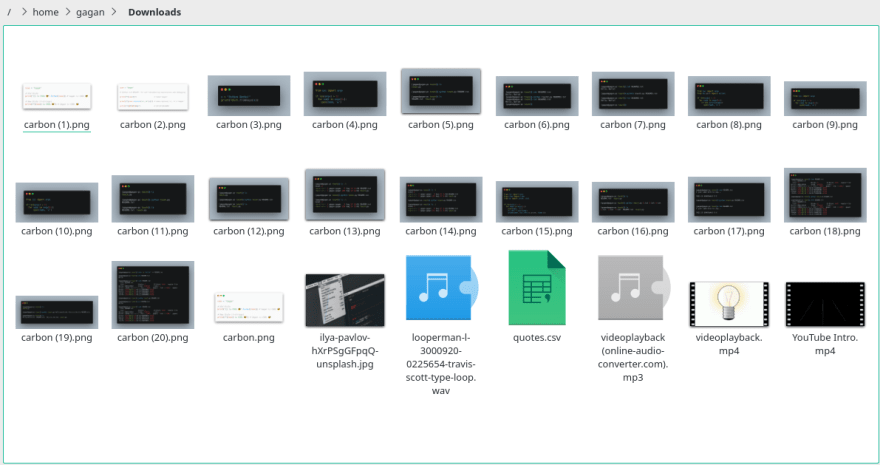


*Source: makeameme.org*

So, I wrote a Python script to automatically organize my files in one go. ✨✨

### **Before the script 😰**

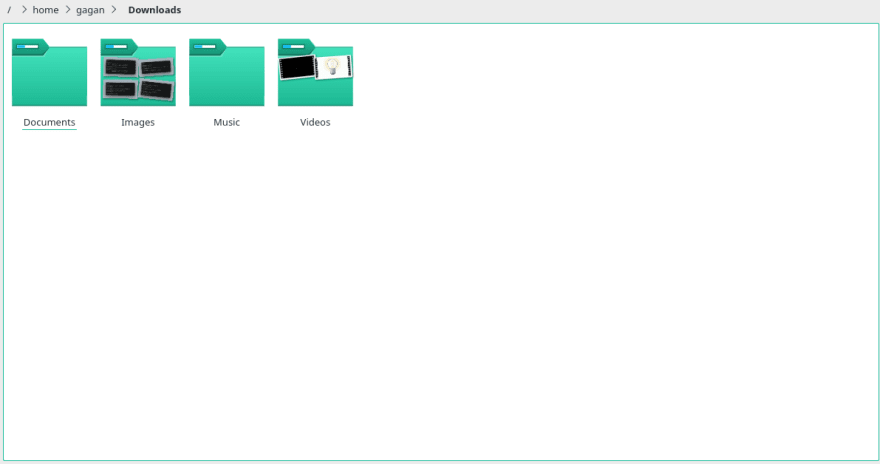
#### **This is how my Downloads folder usually looks like...**

****

*Yeah, it's an unorganized mess 💀*

### **After the (magical ✨) script 😄🥰**

#### **This is how my Downloads folder looks like now...**

****

*It looks cleaner than my room now 🤩*

As you can see that all files are now organized 📂✨!!

Cool, Right!? 😎😎

After writing the script, it's a lot easier for me to look for anything, cause every file is organized and belongs to their own 📂📂, which saves me a lot of time now.



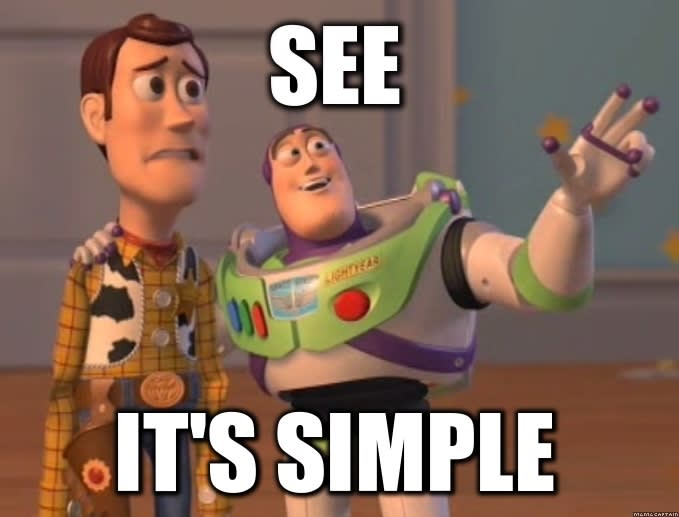
*Source: cheezburger.com*

# **The IDEA 🔥**

A Script which Moves files to their respective 📂✨

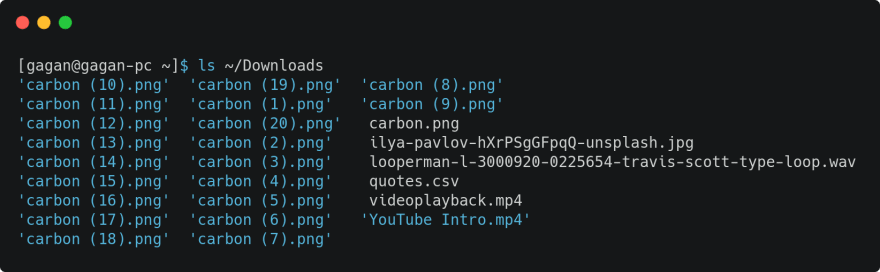
For implementing that, here's what I did (abstract view):

* Created a dictionary 📔 which contains the file types (extensions) and paired them with their respective 📂 names
* Took input (path) from the User via the command-line argument 💻
* Looked for files in that path and made a mechanism (function) for getting their respective 📂 names.
* Moved 🖼, 📽, 🎵,🗒️ etc to their 📂📂📂



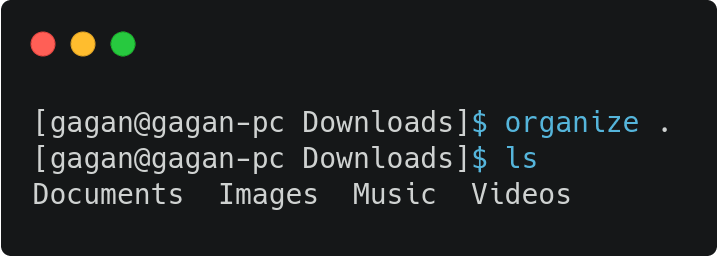
*Source: http://www.brendanconnolly.net/keep-it-simple-theres-more-to-it/*

### **So Let's go from this 😰...**

****

*ls command output ( before running the script )*

### **To this 😃✨✨**

****

*ls command output (after running the script)*

# **Steps for Writing our (Magical ✨) Script**

## **Step #1: Creating a dictionary 📔 of Extensions (suffix) and Directories 📂**

This part is the most time consuming (and the easiest) 😁, we just need to

Create a dictionary 📔 (Hashmap) where *keys are extensions* of the files and their corresponding *values are the name of Directories* we want to move them in.

Note: You can relate Dictionaries in Python 🐍 to Objects in JavaScript.

So, Every *multimedia* file has an extension:

* Image files have extensions *.jpeg, .png, .jpg, .gif*...
* Video files have extensions *.mp4, .mov, .mkv, .webm*...
* Document files have extensions *.doc, .docx, .csv, .pdf*...

*And so on...*

So, we can Google the extensions of *Images, Videos, Documents, Program Files*, and whatever kind of files we need to organize and store them in a dictionary in {"extension": "directory"} pair in a variable named dirs like this...

dirs = {

# Images

"jpeg": "Images",

"png": "Images",

"jpg": "Images",

"tiff": "Images",

"gif": "Images",

# Videos

"mp4": "Videos",

"mkv": "Videos",

"mov": "Videos",

"webm": "Videos",

"flv": "Videos",

# Music

"mp3": "Music",

"ogg": "Music",

"wav": "Music",

"flac": "Music",

# Program Files

"py": "Program Files",

"js": "Program Files",

"cpp": "Program Files",

"html": "Program Files",

"css": "Program Files",

"c": "Program Files",

"sh": "Program Files",

# Documents

"pdf": "Documents",

"doc": "Documents",

"docx": "Documents",

"txt": "Documents",

"ppt": "Documents",

"ods": "Documents",

"csv": "Documents"

}

## **Step #2: Taking input (path) from the user 🧑🏻‍💻👩🏻‍💻 via a command-line argument 💻**

### **What are command-line arguments?**

Here's a simplified version for those who don't know what a *command-line argument* is:

when you run cd command for changing directory and give it a path like this...

cd Documents/TOP\_SECRET\_STUFF

#### **OR**

ls Downloads

that path is called a command-line argument.

Basically, command-line arguments are values that are passed onto the programs or scripts being executed.

### **Now, getting back to our script 📃👇🏻**

Python has a module named sys which has a function argv which returns a list of the command-line arguments given while executing the script.

Here's how it works...

When you run a python script like this...

python helloWorld.py

argv returns a list that contains all the command line arguments given.

In the above example, the first (and only) command-line argument is "helloWorld.py" which is the name of the script file.

argv returns ['helloWorld.py'].

### **If we give command-line arguments like this...**

python add.py 2 3

argv will return ['add.py', '2', '3'].

### **Let's use it for taking a path from the user in our script file.**

from sys import argv

# argv = ['our\_script\_file.py', 'path/to/dir']

# Select second element (at index 1) as path

path = argv[1] # 'path/to/dir'

### **There's a tiny *bug* in it.**

If the user runs the script without providing the path, then our script will throw Index error (cause we are trying to access 2nd element but there won't be any 2nd element to access)



Though, we can easily fix that bug by

### **adding a *condition* before accessing the path from argv**

That simple condition is to check if the command-line arguments are exactly 2 (name of the script file and the path).

### **If this isn't the case then display the Error message along with Usage like this...**

from pathlib import Path

if len(argv) != 2:

print("=" \* 35)

print("[ERROR] Invalid number of arguments were given")

print(f"[Usage] python {Path(\_\_file\_\_).name} <dir\_path>")

print("=" \* 35)

exit(1) # terminate the execution of the script with code 1

# Path to Organize

PATH = argv[1]

And... It's fixed! 😁😁😁

Note:

* \_\_file\_\_ contains the absolute path of the script file (being executed)
* Path(\_\_file\_\_) returns a Path object (the next section digs more into it), and with this object, we can simply print the name of the script file by using the name data member with this Path(\_\_file\_\_).name

### **Now we know how we can take input from the command-line!! 🔥🔥🔥**

## **Step #3: Scan the path for files and move them into their folders**

Python has an AWESOME 😎 module called pathlib which provides Classes for working with paths.

It's so cool that it can handle paths from different Operating Systems! 🤯

### **For organizing files, we need to...**

* 🔍 Look for files in the path 🛣
* Check each file's extension 🧐 (file type)
* Select a destination 🗺️ based on that extension
* Move ✂️ files to their destination directory



### **🔍 Looking for files in the path 🛣**

For doing that, we gonna convert the path (provided via command-line argument) into a Path Object with the help of Path class.

For doing that, we import Path class like this...

from pathlib import Path

and then use the path stored in the 1st index of argv (2nd element) and call the Path class using that

from pathlib import Path

PATH = Path(argv[1]) # argv[1] = 'path/to/organize'

For looking up the files in that directory, we gonna iterate it with the help of the iterdir method.

from pathlib import Path

PATH = Path(argv[1]) # 'path/to/organize'

for filename in PATH.iterdir():

# Do Magic (Organize)

This is gonna help us iterate files and folders in any path! 😃

Cause we want our script to organize only files, we need to check if the filename is a file or a directory. We can easily do that with the is\_file method.

from pathlib import Path

PATH = Path(argv[1]) # argv[1] = 'path/to/organize'

for filename in PATH.iterdir():

if filename.is\_file(): # If filename is a file

# Do Magic (Organize)

### **Checking each file's extension 🧐 (file type)**

Cause we are working with Path objects, it makes it pretty simple for us to get the extension of a file.

Let's say filename is a Path object. To get the extension name, It's stored in the suffix member of the Path object.

# filename = Path object with path value "AWESOME.jpeg"

extension = filename.suffix # .jpeg

cause we need that extension without '.' as a prefix, we gonna remove that with string slicing.

# filename = Path object with path value "AWESOME.jpeg"

extension = filename.suffix[1:] # 'jpeg'

Now, we know how to get extension names! 😄🎉

### **Selecting a destination 🗺️ based on that extension**

The heading says it all.

For selecting a *destination*, we will use the dictionary we created earlier (long key: value pair thingy) and use it for getting the right *destination* name for every single file.

# filename = Path object with path value "AWESOME.jpeg"

extension = filename.suffix[1:] # 'jpeg'

directory\_name = dirs.get(extension) # 'Images'

Cool 😎! But what if the *extension* is not in our dictionary 😐 It's gonna return None and we don't want None. Do we?

So, let's add a default value "Miscellaneous" as a directory name for files that are not in our dictionary.

# filename = Path object with path value "New\_IMG.ppm"

extension = filename.suffix[1:] # 'ppm'

directory\_name = dirs.get(extension, 'Miscellaneous') # 'Miscellaneous'

### **We are almost done**

cause we gonna use this code, again and again, we can wrap it in a *function*.

Gonna name it get\_dir cause it's gonna return directory name for every file it receives as input. Makes sense. Right!?

### **Also, adding a little *docstring* never hurts...**

def get\_dir(filename):

"""

This function takes filename and returns name of the

parent directory of the respective filename

Returns Miscellaneous if Parent is not found

"""

ext = filename.suffix[1:]

return dirs.get(ext, "Miscellaneous")

### **Move ✂️ files to their destination directory (finally! 😃)**

We gonna use themove function from shutil (shell utilities..?) module (Yes, another dope module in Python) for moving files from one place to another.

Moving files is super easy!. Here's an example.

from shutil import move

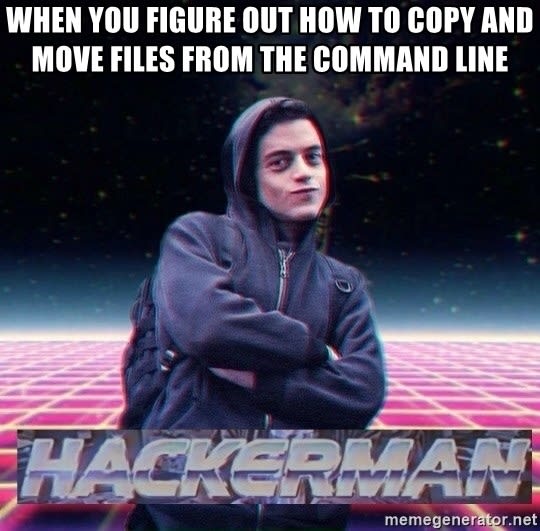
source\_path = '/home/gagan/Downloads/Awesome.jpeg'

destination\_path = '/home/gagan/Downloads/Images'

move(source\_path, destination\_path) # self explanatory

It's similar to executing this command in Bash

mv ~/Downloads/Awesome.jpeg ~/Downloads/Images



*Source: memegenerator.net*

For getting path from a Path object, we use str function which will return path in string instead of a Path object.

We gotta use it cause move function takes in

move(

str(source\_path),

str(destination\_path)

)

### **Create a destination directory for files to be moved in**

We gonna use the get\_dir function we defined earlier for creating a destination directory for each file by doing the following:

destination = PATH / get\_dir(filename)

if not destination.exists():

destination.mkdir()

We made a new Path object with the *path* we gave via the command-line and joined that with our *selected directory* name using PATH / get\_dir(filename) and stored it in destination variable.

Also, we need to make it if it doesn't exist, which is done by exists and mkdir method.

Note:

* "~" contains the path to the home directory (Mine is /home/gagan/)
* We need an absolute path for moving files (Absolute Path refers to the full path to a file/directory)
  + For getting an absolute path, we use the absolute method of Path object.
  + filename.absolute() returns Path object with absolute source path of the file
* pathlib (kinda) supports moving files too! But we used shutil cause it mimics the behavior of mv command and It doesn't have any issues doing its job.

Now, we know everything we need for implementing the (Magical ✨) Script! 😎🔥

### **🔥🔥🔥🔥**

### **LET'S PUT EVERYTHING TOGETHER**

### **WE HAVE LEARNED/DID SO FAR**

### **🔥🔥🔥🔥**

from pathlib import Path

from sys import argv

from shutil import move

def get\_dir(filename):

"""

This function takes filename and returns name of the

parent directory of the respective filename

Returns Miscellaneous if Parent is not found

"""

ext = filename.suffix[1:]

return dirs.get(ext, "Miscellaneous")

dirs = {

# Images

"jpeg": "Images",

"png": "Images",

"jpg": "Images",

"tiff": "Images",

"gif": "Images",

# Videos

"mp4": "Videos",

"mkv": "Videos",

"mov": "Videos",

"webm": "Videos",

"flv": "Videos",

# Music

"mp3": "Music",

"ogg": "Music",

"wav": "Music",

"flac": "Music",

# Program Files

"py": "Program Files",

"js": "Program Files",

"cpp": "Program Files",

"html": "Program Files",

"css": "Program Files",

"c": "Program Files",

"sh": "Program Files",

# Documents

"pdf": "Documents",

"doc": "Documents",

"docx": "Documents",

"txt": "Documents",

"ppt": "Documents",

"ods": "Documents",

"csv": "Documents"

}

if len(argv) != 2:

print("=" \* 35)

print("[ERROR] Invalid number of arguments were given")

print(f"[Usage] python {Path(\_\_file\_\_).name} <dir\_path>")

print("=" \* 35)

exit(1)

# Directory Path

PATH = Path(argv[1])

for filename in PATH.iterdir():

path\_to\_file = filename.absolute()

if path\_to\_file.is\_file():

destination = PATH / get\_dir(filename)

if not destination.exists():

destination.mkdir()

move(str(path\_to\_file), str(destination))

# **Finally, Running the (magical) script 🔥**

For organizing my downloads folder, I can now simply run the following command...

python organize.py ~/Downloads

### **Making this long command shorter with alias**

I created an alias that executes the above command so I don't have to write python ~/projects/File-Organizer/organize.py for executing the script from any directory.

### **Here's how I now organize files in a path**

organize ~/Downloads

### **SO CLEAN! 😁**

This is how you can create an alias 👇🏻👇🏻👇🏻

#### **Paste the following command in .bashrc file**

alias organize="python /path/to/script\_file.py $1"

Replace /path/to/script\_file.py with the path where you saved your script file. Make sure that it's an absolute path of your script file.

Note:

$1 is the first argument that will be passed to our script file.

#### **For activating it in your terminal, execute the following command**

source ~/.bashrc

# **And... we are done! 😎🎉🎉🎉🎉**

## **This was so simple! Right!?**

[Link to the script](https://github.com/gagangulyani/File-Organizer)



*cheezburger.com*

I hope you guys found this article (which I realized is some sort of memeful tutorial 🤨) useful.

If I missed something or you would like to add more to it, feel free to mention it in the comments.

### **If you guys wanna support me, Please give me some genuine feedback on how I can improve myself and my articles.**

Thanks for reading, Stay Vibrant! ✨