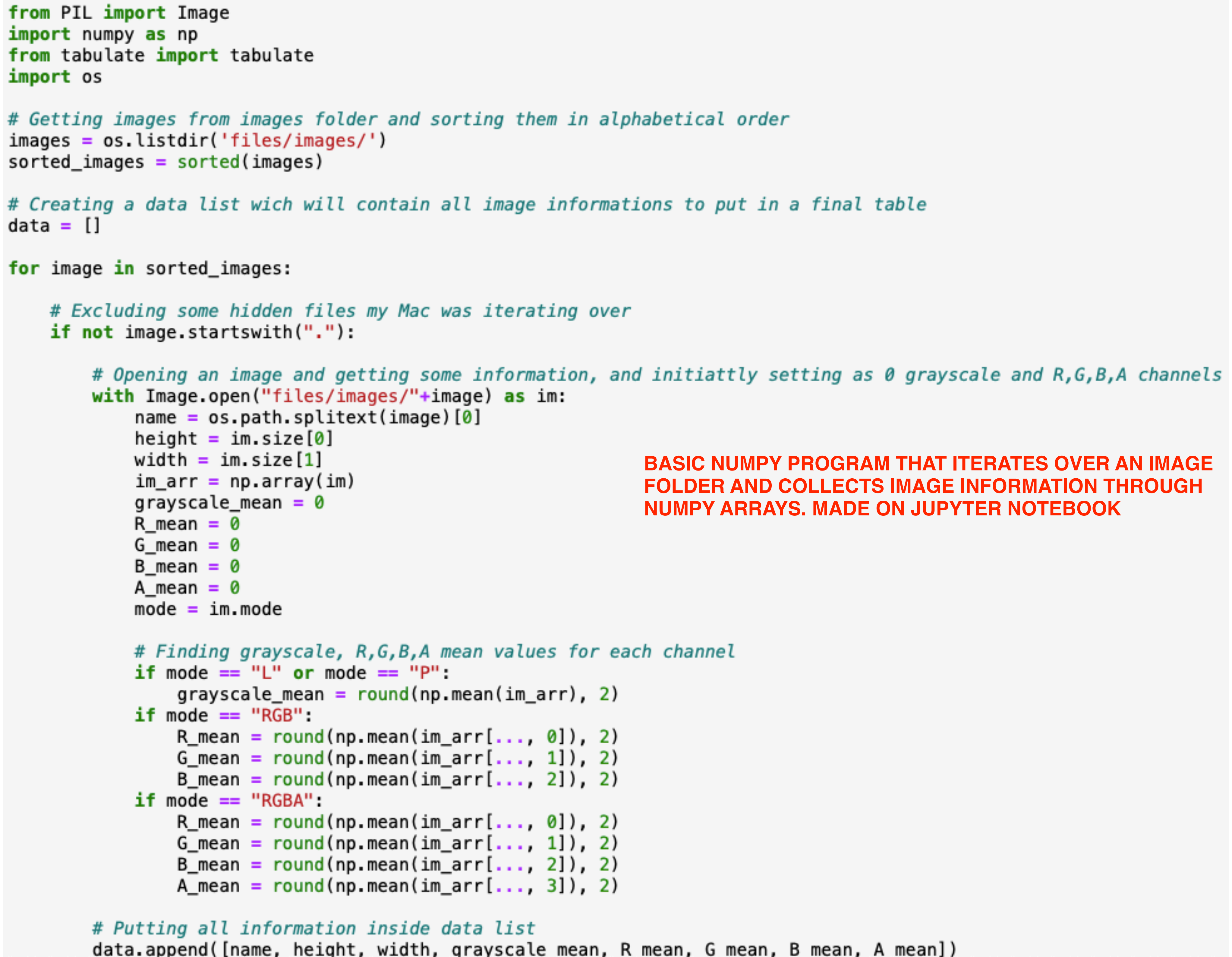
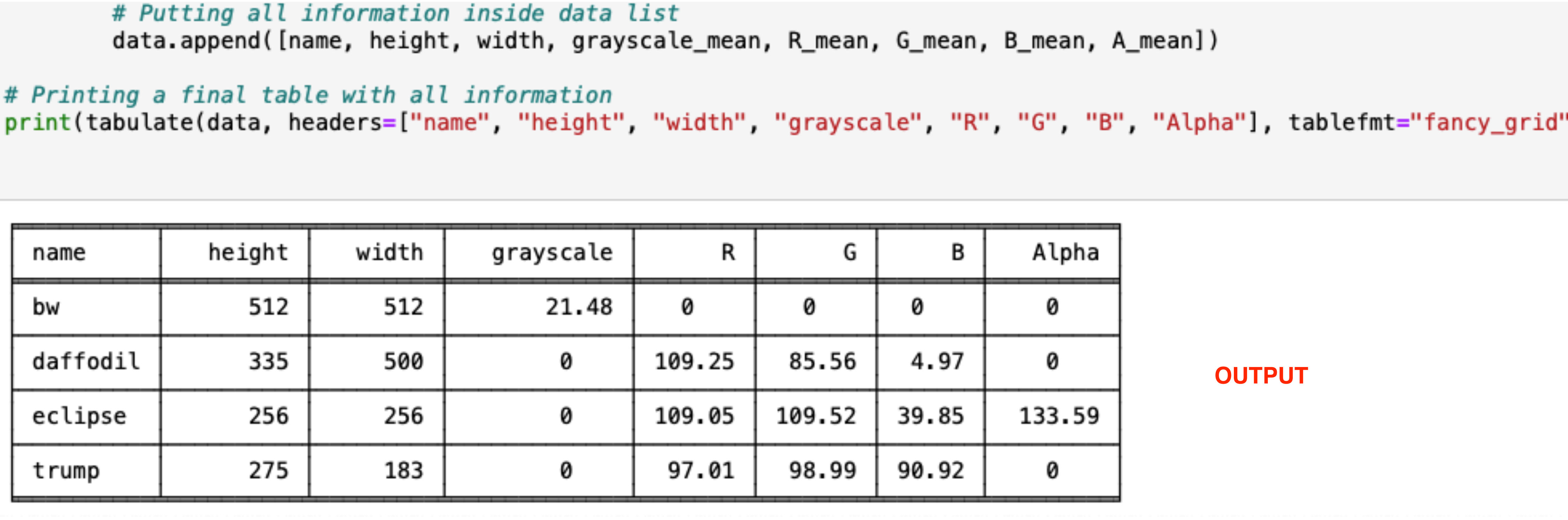


```python
from PIL import Image
import numpy as np
from tabulate import tabulate
import os

# Getting images from images folder and sorting them in alphabetical order
images = os.listdir('files/images/')
sorted_images = sorted(images)

# Creating a data list wich will contain all image informations to put in a final table
data = []

for image in sorted_images:

    # Excluding some hidden files my Mac was iterating over
    if not image.startswith("."):

        # Opening an image and getting some information, and initiattly setting as 0 grayscale and R,G,B,A channels
        with Image.open("files/images/"+image) as im:
            name = os.path.splitext(image)[0]
            height = im.size[0]
            width = im.size[1]
            im_arr = np.array(im)
            grayscale_mean = 0
            R_mean = 0
            G_mean = 0
            B_mean = 0
            A_mean = 0
            mode = im.mode

            # Finding grayscale, R,G,B,A mean values for each channel
            if mode == "L" or mode == "P":
                grayscale_mean = round(np.mean(im_arr), 2)
            if mode == "RGB":
                R_mean = round(np.mean(im_arr[..., 0]), 2)
                G_mean = round(np.mean(im_arr[..., 1]), 2)
                B_mean = round(np.mean(im_arr[..., 2]), 2)
            if mode == "RGBA":
                R_mean = round(np.mean(im_arr[..., 0]), 2)
                G_mean = round(np.mean(im_arr[..., 1]), 2)
                B_mean = round(np.mean(im_arr[..., 2]), 2)
                A_mean = round(np.mean(im_arr[..., 3]), 2)

        # Putting all information inside data list
        data.append([name, height, width, grayscale_mean, R_mean, G_mean, B_mean, A_mean])
```

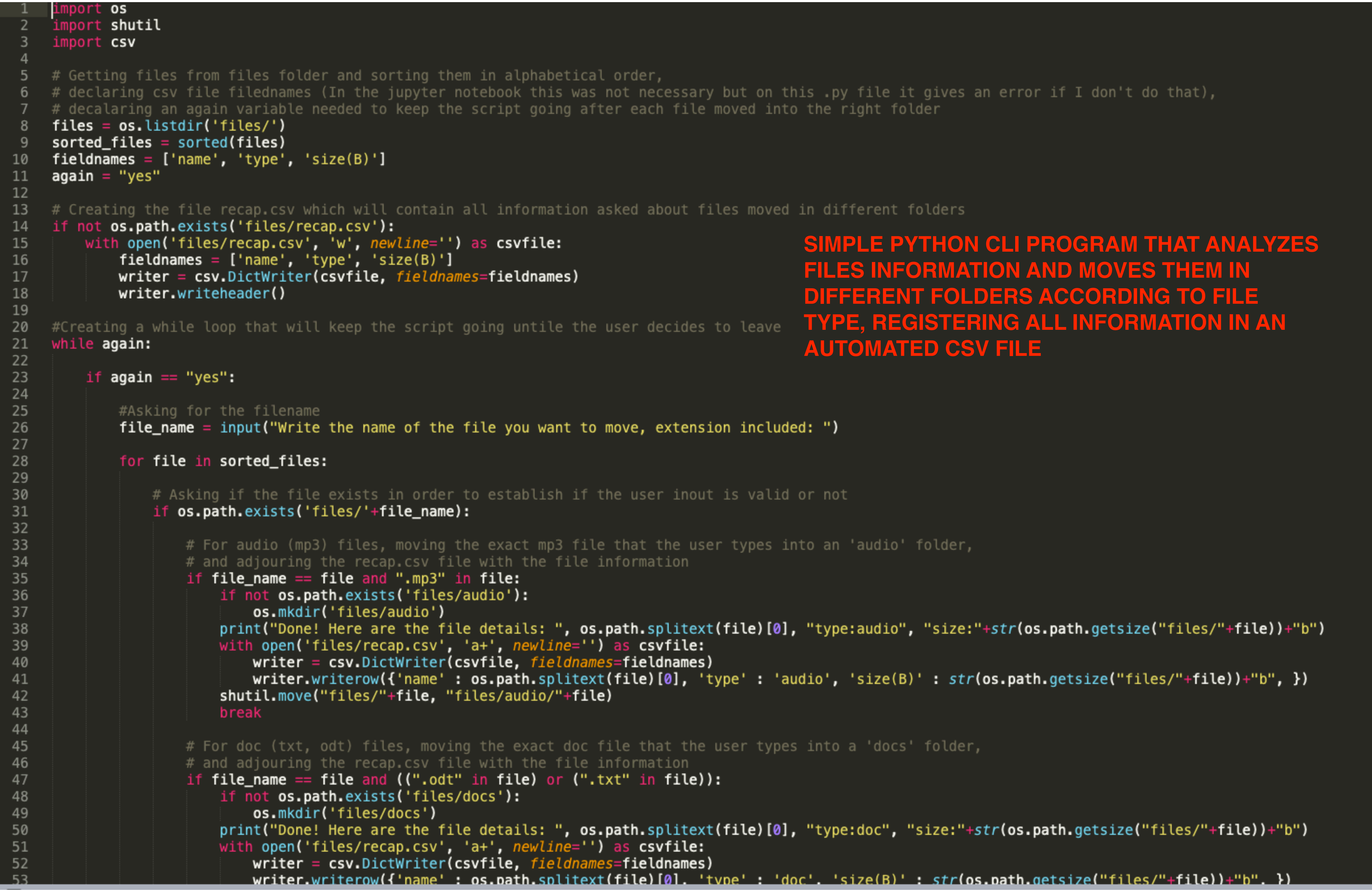
BASIC NUMPY PROGRAM THAT ITERATES OVER AN IMAGE FOLDER AND COLLECTS IMAGE INFORMATION THROUGH NUMPY ARRAYS. MADE ON JUPYTER NOTEBOOK



```python
        # Putting all information inside data list
        data.append([name, height, width, grayscale_mean, R_mean, G_mean, B_mean, A_mean])

# Printing a final table with all information
print(tabulate(data, headers=["name", "height", "width", "grayscale", "R", "G", "B", "Alpha"], tablefmt="fancy_grid"
```

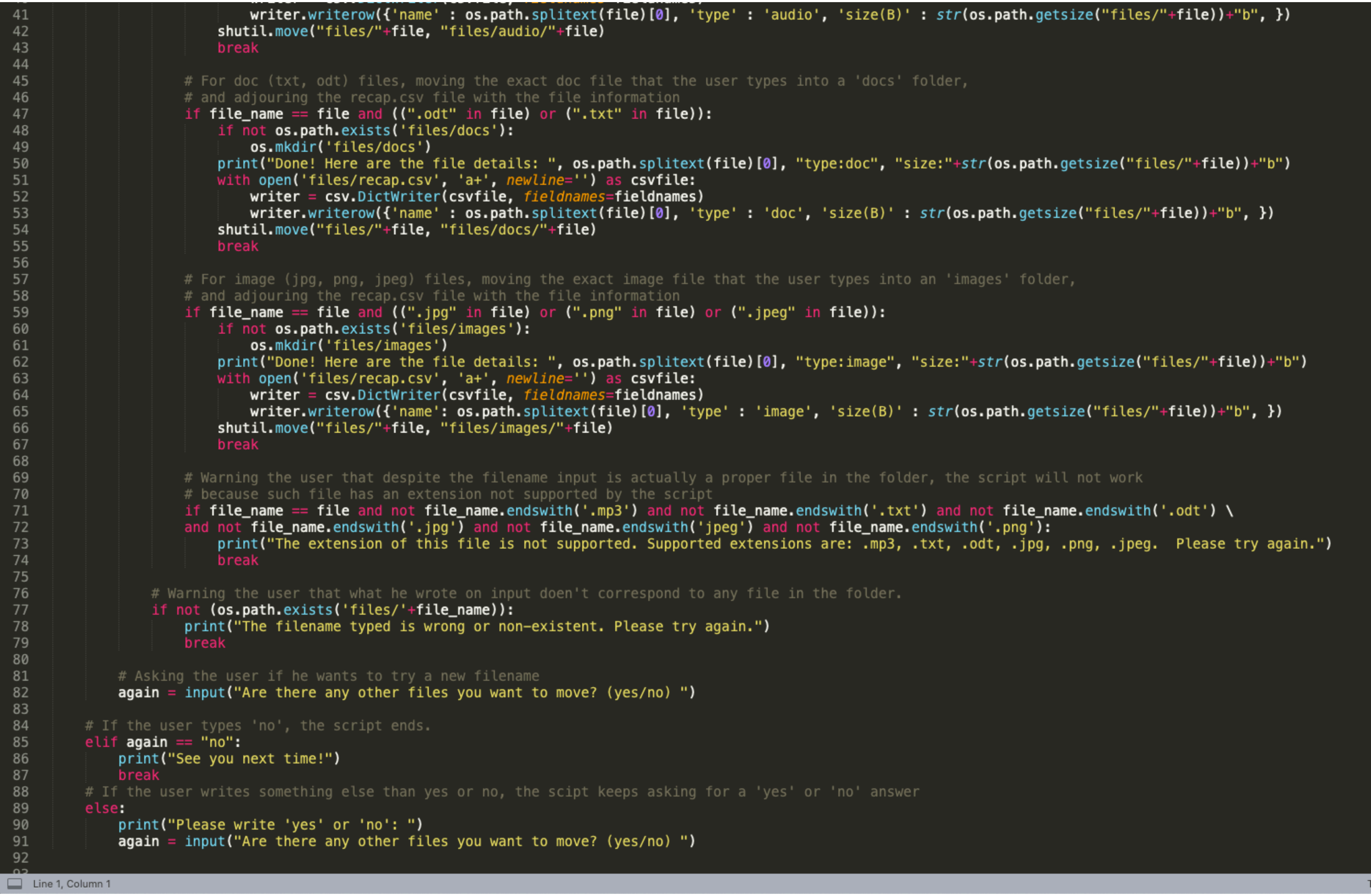
```
╒══════════╤══════════╤═════════╤═════════════╤════════╤════════╤═══════╤═════════╕
│ name     │   height │   width │   grayscale │      R │      G │     B │   Alpha │
╞══════════╪══════════╪═════════╪═════════════╪════════╪════════╪═══════╪═════════╡
│ bw       │      512 │     512 │       21.48 │   0    │   0    │  0    │    0    │
├──────────┼──────────┼─────────┼─────────────┼────────┼────────┼───────┼─────────┤
│ daffodil │      335 │     500 │        0    │ 109.25 │  85.56 │  4.97 │    0    │
├──────────┼──────────┼─────────┼─────────────┼────────┼────────┼───────┼─────────┤
│ eclipse  │      256 │     256 │        0    │ 109.05 │ 109.52 │ 39.85 │  133.59 │
├──────────┼──────────┼─────────┼─────────────┼────────┼────────┼───────┼─────────┤
│ trump    │      275 │     183 │        0    │  97.01 │  98.99 │ 90.92 │    0    │
╘══════════╧══════════╧═════════╧═════════════╧════════╧════════╧═══════╧═════════╛
```

OUTPUT



SIMPLE PYTHON CLI PROGRAM THAT ANALYZES FILES INFORMATION AND MOVES THEM IN DIFFERENT FOLDERS ACCORDING TO FILE TYPE, REGISTERING ALL INFORMATION IN AN AUTOMATED CSV FILE

```python
import os
import shutil
import csv

# Getting files from files folder and sorting them in alphabetical order,
# declaring csv file filednames (In the jupyter notebook this was not necessary but on this .py file it gives an error if I don't do that),
# decalaring an again variable needed to keep the script going after each file moved into the right folder
files = os.listdir('files/')
sorted_files = sorted(files)
fieldnames = ['name', 'type', 'size(B)']
again = "yes"

# Creating the file recap.csv which will contain all information asked about files moved in different folders
if not os.path.exists('files/recap.csv'):
    with open('files/recap.csv', 'w', newline='') as csvfile:
        fieldnames = ['name', 'type', 'size(B)']
        writer = csv.DictWriter(csvfile, fieldnames=fieldnames)
        writer.writeheader()

#Creating a while loop that will keep the script going untile the user decides to leave
while again:

    if again == "yes":

        #Asking for the filename
        file_name = input("Write the name of the file you want to move, extension included: ")

        for file in sorted_files:

            # Asking if the file exists in order to establish if the user inout is valid or not
            if os.path.exists('files/'+file_name):

                # For audio (mp3) files, moving the exact mp3 file that the user types into an 'audio' folder,
                # and adjouring the recap.csv file with the file information
                if file_name == file and ".mp3" in file:
                    if not os.path.exists('files/audio'):
                        os.mkdir('files/audio')
                    print("Done! Here are the file details: ", os.path.splitext(file)[0], "type:audio", "size:"+str(os.path.getsize("files/"+file))+"b")
                    with open('files/recap.csv', 'a+', newline='') as csvfile:
                        writer = csv.DictWriter(csvfile, fieldnames=fieldnames)
                        writer.writerow({'name' : os.path.splitext(file)[0], 'type' : 'audio', 'size(B)' : str(os.path.getsize("files/"+file))+"b", })
                    shutil.move("files/"+file, "files/audio/"+file)
                    break

                # For doc (txt, odt) files, moving the exact doc file that the user types into a 'docs' folder,
                # and adjouring the recap.csv file with the file information
                if file_name == file and ((".odt" in file) or (".txt" in file)):
                    if not os.path.exists('files/docs'):
                        os.mkdir('files/docs')
                    print("Done! Here are the file details: ", os.path.splitext(file)[0], "type:doc", "size:"+str(os.path.getsize("files/"+file))+"b")
                    with open('files/recap.csv', 'a+', newline='') as csvfile:
                        writer = csv.DictWriter(csvfile, fieldnames=fieldnames)
                        writer.writerow({'name' : os.path.splitext(file)[0], 'type' : 'doc', 'size(B)' : str(os.path.getsize("files/"+file))+"b", })
```



```python
                        writer.writerow({'name' : os.path.splitext(file)[0], 'type' : 'audio', 'size(B)' : str(os.path.getsize("files/"+file))+"b", })
                    shutil.move("files/"+file, "files/audio/"+file)
                    break

                # For doc (txt, odt) files, moving the exact doc file that the user types into a 'docs' folder,
                # and adjouring the recap.csv file with the file information
                if file_name == file and ((".odt" in file) or (".txt" in file)):
                    if not os.path.exists('files/docs'):
                        os.mkdir('files/docs')
                    print("Done! Here are the file details: ", os.path.splitext(file)[0], "type:doc", "size:"+str(os.path.getsize("files/"+file))+"b")
                    with open('files/recap.csv', 'a+', newline='') as csvfile:
                        writer = csv.DictWriter(csvfile, fieldnames=fieldnames)
                        writer.writerow({'name' : os.path.splitext(file)[0], 'type' : 'doc', 'size(B)' : str(os.path.getsize("files/"+file))+"b", })
                    shutil.move("files/"+file, "files/docs/"+file)
                    break

                # For image (jpg, png, jpeg) files, moving the exact image file that the user types into an 'images' folder,
                # and adjouring the recap.csv file with the file information
                if file_name == file and ((".jpg" in file) or (".png" in file) or (".jpeg" in file)):
                    if not os.path.exists('files/images'):
                        os.mkdir('files/images')
                    print("Done! Here are the file details: ", os.path.splitext(file)[0], "type:image", "size:"+str(os.path.getsize("files/"+file))+"b")
                    with open('files/recap.csv', 'a+', newline='') as csvfile:
                        writer = csv.DictWriter(csvfile, fieldnames=fieldnames)
                        writer.writerow({'name': os.path.splitext(file)[0], 'type' : 'image', 'size(B)' : str(os.path.getsize("files/"+file))+"b", })
                    shutil.move("files/"+file, "files/images/"+file)
                    break

                # Warning the user that despite the filename input is actually a proper file in the folder, the script will not work
                # because such file has an extension not supported by the script
                if file_name == file and not file_name.endswith('.mp3') and not file_name.endswith('.txt') and not file_name.endswith('.odt') \
                and not file_name.endswith('.jpg') and not file_name.endswith('jpeg') and not file_name.endswith('.png'):
                    print("The extension of this file is not supported. Supported extensions are: .mp3, .txt, .odt, .jpg, .png, .jpeg.  Please try again.")
                    break

            # Warning the user that what he wrote on input doen't correspond to any file in the folder.
            if not (os.path.exists('files/'+file_name)):
                print("The filename typed is wrong or non-existent. Please try again.")
                break

        # Asking the user if he wants to try a new filename
        again = input("Are there any other files you want to move? (yes/no) ")

    # If the user types 'no', the script ends.
    elif again == "no":
        print("See you next time!")
        break
    # If the user writes something else than yes or no, the scipt keeps asking for a 'yes' or 'no' answer
    else:
        print("Please write 'yes' or 'no': ")
        again = input("Are there any other files you want to move? (yes/no) ")
```