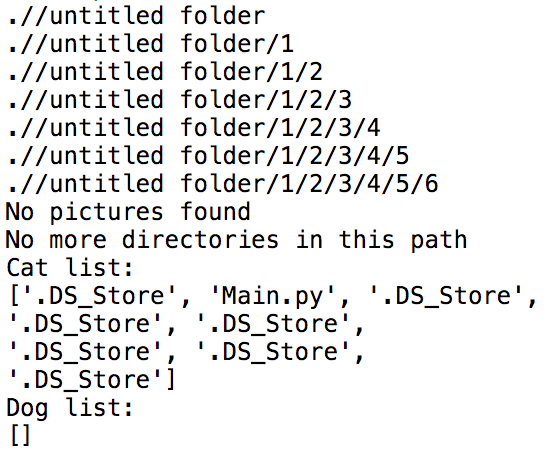
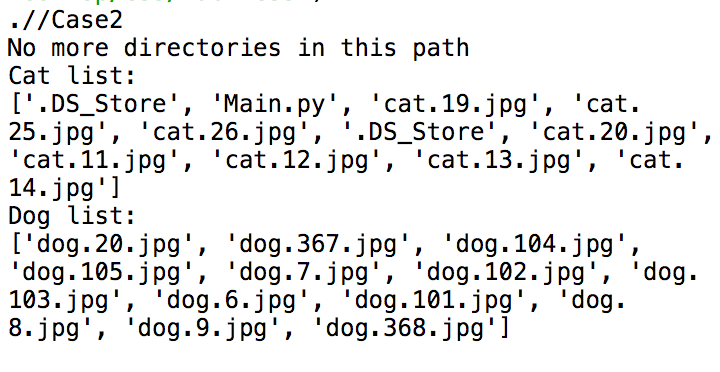
In this problem we are trying to sort the dog and cat pictures found within the folders provided in a list. I attempted to solve this problem by implementing the method ﻿classify\_pic to classify the pictures found within the files inside my process\_dir. In order to implement this method, I had to update the path passed as parameter after every method call, which after entering each folder becomes longer. The path is the “address” within the folders of the computer. Using the method ﻿get\_dirs\_and\_files I was able to create a list of both the files and the new directories found inside the folder. The first action of my method ﻿process\_dir was to, once inside the current directory, check the file list and classify each picture using the classify\_pic method. Using the length of the directory as my base case, because once there are no more directories within a folder the program should reach the end, I began my recursion part of my method. After the method checks for base case, if not true, the else executes and checks every directory found in the folder and calls itself with the new path added. After the recursion execution of the method, the method returns two lists which are added to the final returned lists for both cats and dogs. At the end of my main, I printed the directories paths found in the files, and then the dog and cat lists.

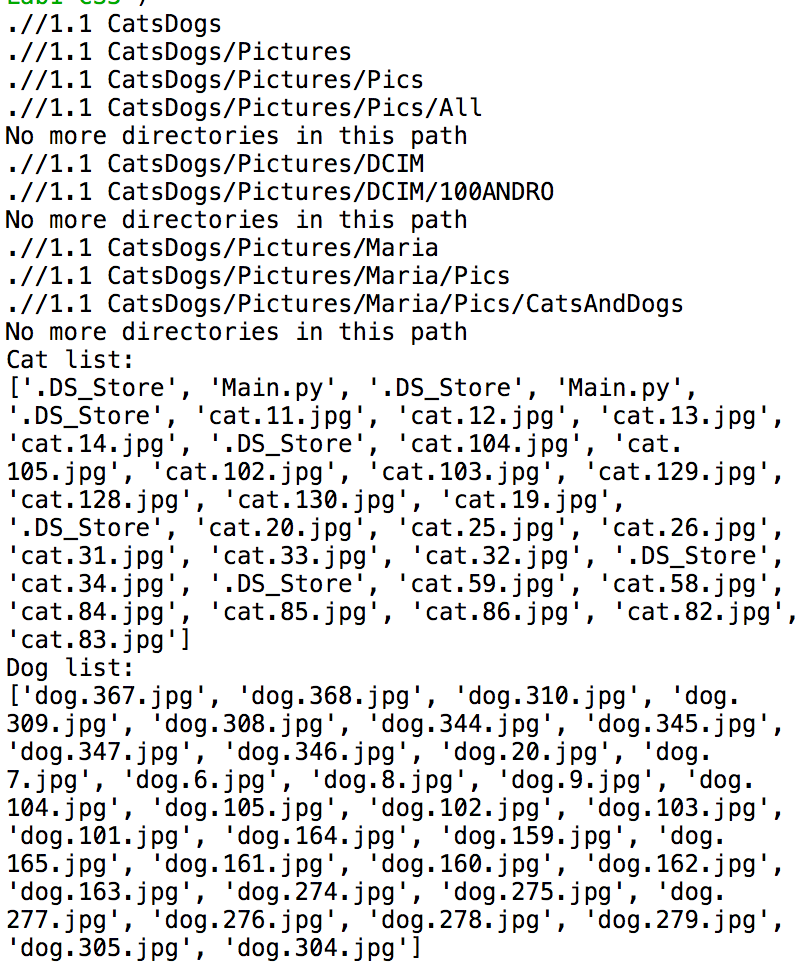
For my first test case, I created a folder with many folders inside folders but no pictures. As shown below, the program prints the paths but since I have a mac it classified the .DS\_Store files in the cat list, since there was no “dog” in path. The dog list remains empty.



For my second base case, I created a folder with no more folders inside only cat or dog pictures. As shown below, the path only prints once but the method still appends the pictures to the cat and dog lists.



As my third test case, I used the folder provided by the instructor and tested it. The program printed all the paths and both cat and dog lists.



The time complexity of my method process\_dir is dependent on the length of both lists the file\_list and dir\_list. Their lengths can be said to be n and m, which would make the big Oh (m \* n). Solving this project, I learned that you can have for loops and still use recursion. In my previous CS classes I was told that recursion and for loops don’t go together, but now I understand it was for sake of time and initial comprehension of the concept. Since I initially thought for loops and recursion don’t combine I had a little bit of trouble understanding how I was supposed to access all the elements in the directory list. I also learned a little bit about using python as a command line for my OS.

Appendix A

import os

import random

def get\_dirs\_and\_files(path):

dir\_list = [directory for directory in os.listdir(path) if os.path.isdir(path + "/" + directory)]

file\_list = [directory for directory in os.listdir(path) if not os.path.isdir(path + "/" + directory)]

return dir\_list, file\_list

def classify\_pic(path):

# To be implemented by Diego: Replace with ML model

if "dog" in path:

return 0.5 + random.random() / 2

return random.random() / 2

def process\_dir(path):

dir\_list, file\_list = get\_dirs\_and\_files(path)

cat\_list = []

dog\_list = []

##no more pictures left, if not check pictures

##has to be done regardless of recursion and base cases

if len(file\_list) == 0:

print("No pictures found")

else:

##checks all files in file\_list with classify\_pic method, and appends to cat or dog list based on

##returned probability

for i in range(len(file\_list)):

if classify\_pic(file\_list[i]) >= .5:

dog\_list.append(file\_list[i])

else:

cat\_list.append(file\_list[i])

## no more directories left in the path, base case

if len(dir\_list) == 0:

print("No more directories in this path")

##recursion case

else:

for i in range(len(dir\_list)):

##print the directories that we are in

print(path + "/"+ dir\_list[i])

##merge 2 lists

cat2\_list, dog2\_list = process\_dir(path + "/" + dir\_list[i])

cat\_list += cat2\_list

dog\_list += dog2\_list

# Your code goes here

return cat\_list, dog\_list

def main():

start\_path = './' # current directory

get\_dirs\_and\_files(start\_path)

cat2\_list,dog2\_list = process\_dir(start\_path)

print("Cat list:")

print(cat2\_list)

print("Dog list:")

print(dog2\_list)

main()

I certify that this project is entirely my own work. I wrote, debugged, and tested the code being presented, performed the experiments, and wrote the report. I also certify that I did not share my code or report or provided inappropriate assistance to any student in the class.