#Question 1:

#Function to validate user input: return ture if valid, return false if invalid

def validate(inputStr):

    #Remove space and test the input string with isalpha method

    testStr = inputStr.replace(' ','')

    if testStr.isalpha():

        return True

    else:

        #Print case 4 invalid input message

        print("Invalid input. Please enter English letters only.")

        return False

try:

    #Open menu file

    with open("C:/food/menu.csv", "r") as fob:

        #Empty dictionary foodDict to store food and price

        foodDict = {}

        #Read and discard the first header line

        fob.readline()

        #Read the rest of the centent of the file

        for line in fob:

            #On each line, store the name and the price to the dictionary.

            #Remove the ending new line character on the price.

            item = line.split(',')

            foodDict[item[0]] = item[2].removesuffix('\n')

        #Get user input and convert to upper case for case insensitive comparison

        userInput = input("Enter a character or characters: ").upper()

        #Test user input against the validate function and ask user to re-enter if input is invalid

        while not validate(userInput):

            userInput = input("Enter a character or characters: ").upper()

        #Boolean variable found to store rather input is found.

        #Default is not found.

        found = False

        #Check every keys in foodDict

        for key in foodDict.keys():

            #If matching item is found

            if key.upper().startswith(userInput):

                #Print the heading line if this is the first matching item:

                if not found:

                    print("Matching items:")

                    found = True

                #Normal case, print the matching item:

                print("- " + key + ": $" + foodDict[key])

        #Print case 5 when no match items found

        if not found:

            print("No matching items found.")

#Handle case 6 when menu.csv does not exist

except (FileNotFoundError):

    print ("The Menu cannot be found!")

#Handle other exceptions for debug

except Exception as e:

    print("Error: " + type(e).\_\_name\_\_ + " at line " + str(e.\_\_traceback\_\_.tb\_lineno))

#Question 2:

import os

import glob

import shutil

#Define spath, mpath, lpath for global use

spath = "C:/photos/small/"

mpath = "C:/photos/medium/"

lpath = "C:/photos/large/"

#Function for move photos and print the summary at the end

def movePhotos():

    #Define number of bytes for a megabytes

    megabytes = 1048576

    #fcount is used to count total number of files

    #scount is used to count number of small files

    #mcount is used to count number of medium files

    #lcount is used to count number of large files

    #Initialize all counter to 0

    fcount = scount = mcount = lcount = 0

    #Get number of jpg files in C:\photos

    fcount = len(glob.glob("C:/photos/\*.jpg"))

    #For each file in C:\photos, move files to correspoding folders and increment the correspoding counters

    for filename in glob.glob("C:/photos/\*.jpg"):

        out = os.path.getsize(filename)

        if out < 3 \* megabytes:

            shutil.move(filename, spath + os.path.basename(filename))

            scount += 1

        elif out < 7 \* megabytes:

            shutil.move(filename, mpath + os.path.basename(filename))

            mcount += 1

        else:

            shutil.move(filename, lpath + os.path.basename(filename))

            lcount += 1

    #Print the summary at the end

    print("Categorized " + str(fcount) + " photos:")

    print("- " + str(scount) + " photos moved to 'small' folder")

    print("- " + str(mcount) + " photos moved to 'medium' folder")

    print("- " + str(lcount) + " photos moved to 'large' folder")

#Main program

try:

    #Create the 3 destination folders

    os.mkdir(spath)

    os.mkdir(mpath)

    os.mkdir(lpath)

#If the destination folders exists, ignore the exception and continue to call movePhotos()

except (FileExistsError):

    movePhotos()

#Halt the program and print debug message for other exceptions

except Exception as e:

    print("Error: " + type(e).\_\_name\_\_ + " at line " + str(e.\_\_traceback\_\_.tb\_lineno))

#Call movePhotos if no exceptions

else:

    movePhotos()