#Question 1:

import re

#Function to validate user input: return True if valid, return False if invalid

def validate(inputStr):

    #any string contains alphabets or spaces

    regex = r'[A-Za-z ]\*'

    #The requirement does not indicate how to handle empty string input. Assume invalid

    if len(inputStr) > 0 and re.fullmatch(regex, inputStr):

        #return True if valid

        return True

    else:

        #Print case 4 invalid input message and return false

        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:

            #Remove the ending new line character if any

            if line[-1] == '\n':

                line = line[:-1]

            #Store the name and the price to the dictionary.

            item = line.split(',')

            foodDict[item[0]] = item[2]

        #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

#Function md() to create a single folder by the given parameter.

#Return true if successful.

#Return false if failed

def md(path):

    try:

        os.mkdir(path)

    #Considered as successful if the destination folder exists

    except (FileExistsError):

        return True

    #Print debug message and return false for other exceptions

    except Exception as e:

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

        print("Failed to create folder: " + path )

        return False

    #Return true if successful

    else:

        return True

#Define spath, mpath, lpath for small, medium, and large files

spath = "C:/photos/small/"

mpath = "C:/photos/medium/"

lpath = "C:/photos/large/"

#Create folder by calling function md.

#Proceed only when folders are created successfully.

if md(spath) and md(mpath) and md(lpath):

    #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")