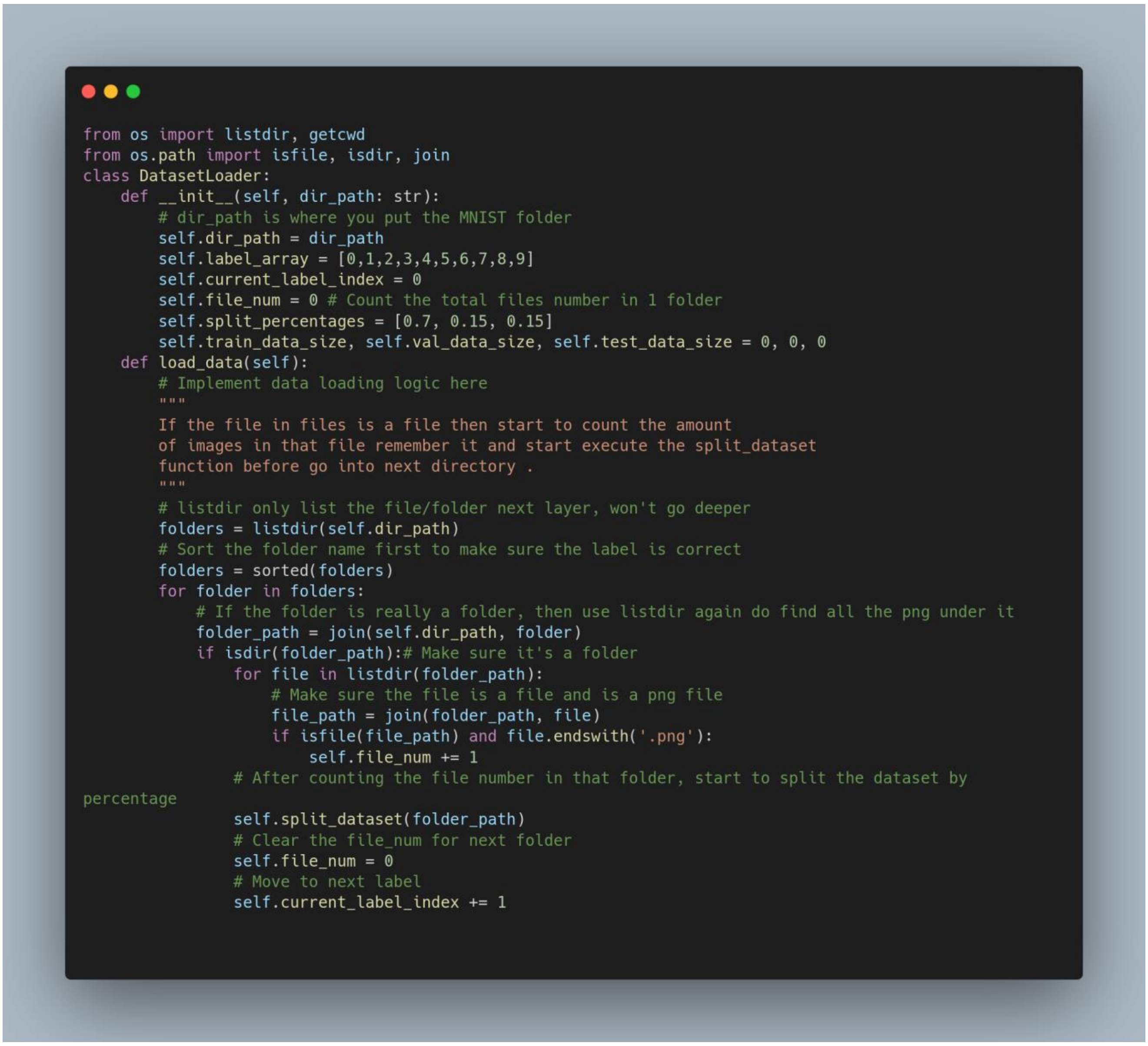
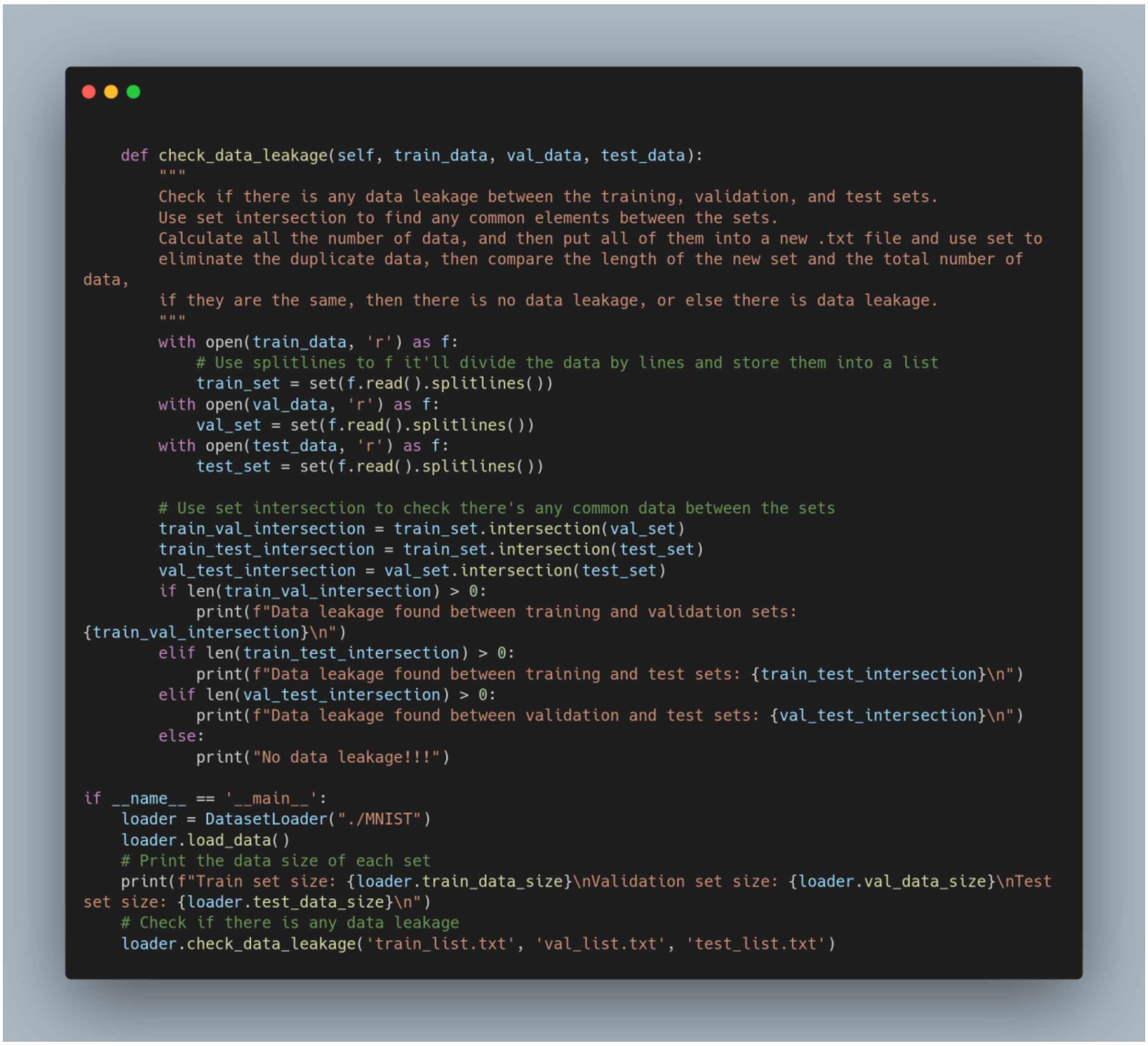
Intro-to-ML HW1 110612025 魏于翔

- Code block:



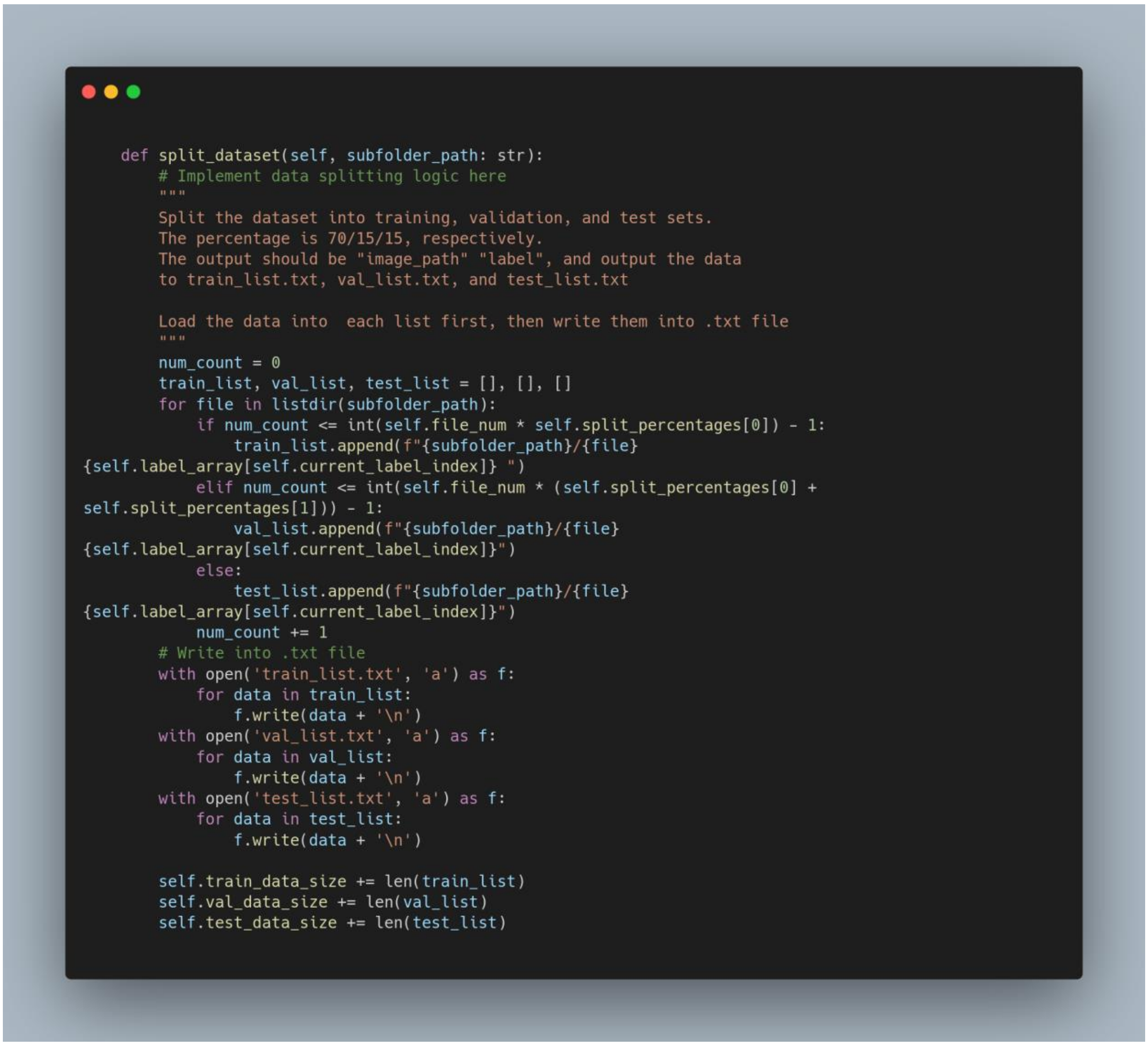
```
from os import listdir, getcwd
from os.path import isfile, isdir, join
class DatasetLoader:
    def __init__(self, dir_path: str):
        # dir_path is where you put the MNIST folder
        self.dir_path = dir_path
        self.label_array = [0,1,2,3,4,5,6,7,8,9]
        self.current_label_index = 0
        self.file_num = 0 # Count the total files number in 1 folder
        self.split_percentages = [0.7, 0.15, 0.15]
        self.train_data_size, self.val_data_size, self.test_data_size = 0, 0, 0
    def load_data(self):
        # Implement data loading logic here
        """
        If the file in files is a file then start to count the amount
        of images in that file remember it and start execute the split_dataset
        function before go into next directory .
        """
        # listdir only list the file/folder next layer, won't go deeper
        folders = listdir(self.dir_path)
        # Sort the folder name first to make sure the label is correct
        folders = sorted(folders)
        for folder in folders:
            # If the folder is really a folder, then use listdir again do find all the png under it
            folder_path = join(self.dir_path, folder)
            if isdir(folder_path):# Make sure it's a folder
                for file in listdir(folder_path):
                    # Make sure the file is a file and is a png file
                    file_path = join(folder_path, file)
                    if isfile(file_path) and file.endswith('.png'):
                        self.file_num += 1
                # After counting the file number in that folder, start to split the dataset by
percentage
                self.split_dataset(folder_path)
                # Clear the file_num for next folder
                self.file_num = 0
                # Move to next label
                self.current_label_index += 1
```



```
    def check_data_leakage(self, train_data, val_data, test_data):
        """
        Check if there is any data leakage between the training, validation, and test sets.
        Use set intersection to find any common elements between the sets.
        Calculate all the number of data, and then put all of them into a new .txt file and use set to
        eliminate the duplicate data, then compare the length of the new set and the total number of
data,
        if they are the same, then there is no data leakage, or else there is data leakage.
        """
        with open(train_data, 'r') as f:
            # Use splitlines to f it'll divide the data by lines and store them into a list
            train_set = set(f.read().splitlines())
        with open(val_data, 'r') as f:
            val_set = set(f.read().splitlines())
        with open(test_data, 'r') as f:
            test_set = set(f.read().splitlines())

        # Use set intersection to check there's any common data between the sets
        train_val_intersection = train_set.intersection(val_set)
        train_test_intersection = train_set.intersection(test_set)
        val_test_intersection = val_set.intersection(test_set)
        if len(train_val_intersection) > 0:
            print(f"Data leakage found between training and validation sets:
{train_val_intersection}\n")
        elif len(train_test_intersection) > 0:
            print(f"Data leakage found between training and test sets: {train_test_intersection}\n")
        elif len(val_test_intersection) > 0:
            print(f"Data leakage found between validation and test sets: {val_test_intersection}\n")
        else:
            print("No data leakage!!!")

if __name__ == '__main__':
    loader = DatasetLoader("./MNIST")
    loader.load_data()
    # Print the data size of each set
    print(f"Train set size: {loader.train_data_size}\nValidation set size: {loader.val_data_size}\nTest
set size: {loader.test_data_size}\n")
    # Check if there is any data leakage
    loader.check_data_leakage('train_list.txt', 'val_list.txt', 'test_list.txt')
```



```
    def split_dataset(self, subfolder_path: str):
        # Implement data splitting logic here
        """
        Split the dataset into training, validation, and test sets.
        The percentage is 70/15/15, respectively.
        The output should be "image_path" "label", and output the data
        to train_list.txt, val_list.txt, and test_list.txt

        Load the data into  each list first, then write them into .txt file
        """
        num_count = 0
        train_list, val_list, test_list = [], [], []
        for file in listdir(subfolder_path):
            if num_count <= int(self.file_num * self.split_percentages[0]) - 1:
                train_list.append(f"{subfolder_path}/{file}
{self.label_array[self.current_label_index]} ")
            elif num_count <= int(self.file_num * (self.split_percentages[0] +
self.split_percentages[1])) - 1:
                val_list.append(f"{subfolder_path}/{file}
{self.label_array[self.current_label_index]}")
            else:
                test_list.append(f"{subfolder_path}/{file}
{self.label_array[self.current_label_index]}")
            num_count += 1
        # Write into .txt file
        with open('train_list.txt', 'a') as f:
            for data in train_list:
                f.write(data + '\n')
        with open('val_list.txt', 'a') as f:
            for data in val_list:
                f.write(data + '\n')
        with open('test_list.txt', 'a') as f:
            for data in test_list:
                f.write(data + '\n')

        self.train_data_size += len(train_list)
        self.val_data_size += len(val_list)
        self.test_data_size += len(test_list)
```

Brief explanation:

1. load_data() 是去讀./MNIST 資料夾下所有的 file and folder 的名稱，當用 isdir()確定讀取的是 folder 後變進去那個 folder 裡算裡面到底有幾個.png 檔案以便作為分割 data 的依據，當檢查完一個 folder 裡有幾個檔案後傳遞該資料夾相對路徑給 split_dataset()來進行資料分割。
2. split_dataset()是根據 70/15/15 的比例來分個 data 給 train/validation/test data 的，用 listdir()讀取 folder 裡所有的.png 檔的檔名，將其加上 folder 路徑便可已得到與程式檔案的相對路經，由於作業規定輸出格式必須”image_path label”，我們必須先從 folder digit_0 照順序下去搜索，因此在 listdir()後就要先 sort folders，接著根據比例將輸出內容分別寫入對應.txt file 並紀錄各個 data 的總數
3. check_data_leakage() 則是檢查有沒有資料洩漏的發生，在這個 case 裡就是 train, validation, test data 是否有重複，我使用的方法是 set，先將對應的.txt 用 read().splitline()讀取並以一行為單位將其儲存成 list，並將其轉成 set，之後在以

兩個為一組的 set，用 intersection 來檢查內容是否重複，若有重複則將重複的文檔印出在 terminal 上。