# don't worry about this import! It just allows us to grab all the files in the countries/ directory

import os

import csv

from simpleimage import SimpleImage

# Dimensions of the final visualization. Change these if the

# image is too large for your screen

VISUALIZATION\_WIDTH = 1920

VISUALIZATION\_HEIGHT = 1080

# Setting the 'boundaries' for the visualization. By default, the

# visualization holds the entire world. If you want to zoom in on a

# specific country, you can find the corresponding latitudes and longitudes

# here: https://gist.github.com/graydon/11198540

MIN\_LONGITUDE = -180

MAX\_LONGITUDE = 180

MIN\_LATITUDE = -90

MAX\_LATITUDE = 90

# The folder in which all the country data can be found

COUNTRY\_DIRECTORY = "countries/"

def plot\_country(visualization, filename):

"""

Responsible for reading in geographic data from a file

about the cities in a particular country and plotting them

in the visualization.

Parameters:

- `visualization` is the SimpleImage that will eventually be

shown to the user

- `filename` is the file we want to read through

"""

with open(filename) as cities\_file:

reader = csv.DictReader(cities\_file)

for line in reader:

lat = float(line['Latitude'])

lon = float(line['Longitude'])

plot\_one\_city(visualization, lat, lon)

"""

DO NOT MODIFY THE CODE BELOW

(but you're welcome to read it 😀 )

"""

def main():

# create a blank image on which we'll plot cities

visualization = SimpleImage.blank(

VISUALIZATION\_WIDTH, VISUALIZATION\_HEIGHT

)

# get which countries should be plotted from the user

countries = get\_countries()

# iterate through each of the countries and plot it

for country in countries:

country\_filename = COUNTRY\_DIRECTORY + country + ".csv"

cities\_data = plot\_country(visualization, country\_filename)

# once we're done with all the countries, show the image

visualization.show()

def get\_countries():

"""

Gets the list of countries from the user, but doesn't check that

the user types in valid country names.

Returns a list of country names

"""

countries = []

while True:

country = input("Enter a country, or 'all'. Press enter to finish: ")

if country == "":

break

if country == "all":

# don't worry about this bit of code! It just looks inside

# `COUNTRY\_DIRECTORY` and returns a list of all the filenames

return [s.split(".")[0] for s in os.listdir(COUNTRY\_DIRECTORY)]

# if the user didn't press enter immediately or type all,

# store the country name

countries.append(country.strip())

return countries

def plot\_one\_city(visualization, latitude, longitude):

"""

Given the visualization image as well as a single city's latitude and longitude,

plot the city on the image

Parameters:

- `visualization` is the SimpleImage that will eventually be

shown to the user

- `latitude` is the latitude of the city (a float)

- `longitude` is the longitude of the city (a float)

"""

# convert the Earth coordinates to pixel coordinates

x = longitude\_to\_x(longitude)

y = latitude\_to\_y(latitude)

# if the pixel is in bounds of the window we specified through constants,

# plot it

if 0 < x < visualization.width and 0 < y < visualization.height:

plot\_pixel(visualization, x , y)

def plot\_pixel(visualization, x, y):

"""

Set a pixel at a particular coordinate to be blue. Pixels start off as

white, so all three color components have a value of 255. Setting the red

and green components to 0 makes the pixel appear blue.

Note that we don't return anything in this function because the Pixel is

'mutated' in place

Parameters:

- `visualization` is the SimpleImage that will eventually be

shown to the user

- `x` is the x coordinate of the pixel that we are turning blue

- `y` is the y coordinate of the pixel that we are turning blue

"""

pixel = visualization.get\_pixel(x, y)

pixel.red = 0

pixel.green = 0

def longitude\_to\_x(longitude):

"""

Scales a longitude coordinate to a coordinate in the visualization email

"""

return VISUALIZATION\_WIDTH \* (longitude - MIN\_LONGITUDE) / (MAX\_LONGITUDE - MIN\_LONGITUDE)

def latitude\_to\_y(latitude):

"""

Scales a latitude coordinate to a coordinate in the visualization email

"""

return VISUALIZATION\_HEIGHT \* (1.0 - (latitude - MIN\_LATITUDE) / (MAX\_LATITUDE - MIN\_LATITUDE))

if \_\_name\_\_ == "\_\_main\_\_":

main()