Week 9 In-Class Exercise

Using the given data file, cities.csv, to work in this In-Class Exercise.

1. Use the following code to read and partially print the data file.

1.1 Reader	1.2 DictReader

`import csv`
`cities_data = []`
`with open('Cities.csv','r') as f:`
`    rows = csv.reader(f)`
`    for r in rows:`
`        cities_data.append(r)`
`print(cities_data[0:10])`
`print(cities_data[8:10])`
	`import csv`
	`cities_data = []`
	`with open('Cities.csv','r') as f:`
	`    rows = csv.DictReader(f)`
	`    for r in rows:`
	`        cities_data.append(r)`
	`print(cities_data[0:10])`
	`print(cities_data[8:10])`

   - Compare the given result and specify the difference.
   - Determine which one do you prefer to use?

2. Use `cities_data` from Reader and DictReader to create the tuple of all cities and their temperatures. Print the tuple at the end.

Partial Program

`# … Continue from 1.1 or 1.2`
 	
`city_temp_tuple = []`
 	
`# fill in the code yourself`
 	
`print(city_temp_tuple)`

Expected output:
[('Aalborg', 57.03), ('Aberdeen', 57.17), ('Abisko', 63.35), ('Adana', 36.99), ('Albacete', 39.0), ('Algeciras', 36.13), ('Amiens', 49.9), ('Amsterdam', 52.35), ('Ancona', 43.6), ('Andorra', 42.5), …, ('Yevpatoriya', 45.2), ('Zaragoza', 41.65), ('Zhytomyr', 50.25), ('Zonguldak', 41.43), ('Zurich', 47.38)]

See OrderedDict reference at: *https://www.geeksforgeeks.org/ordereddict-in-python/*

3. Write a function called *list_countries* to construct a list of unique countries in the data file.
   Hint: Use *in* operator to help out with membership test. (If you do not know what *in* operator is about, read moe in Google Classroom, Pre-midterm VDOs, Topic 4: List, Slides, page 14, Slide 27.)

Partial Program for Main

`# … Continue from 1.2 + Remove last two print lines in 1.2`
 	
`countries = list_countries(cities_data)`
`print(len(countries))`
`print(countries)`

Expected output:
37 ['Denmark', 'United Kingdom', 'Sweden', 'Turkey', 'Spain', 'France', 'Netherlands', 'Italy', 'Andorra', 'Romania', 'Greece', 'Germany', 'Moldova', 'Switzerland', 'Serbia', 'Norway', 'Poland', 'Ukraine', 'Portugal', 'Slovakia', 'Belarus', 'Czech Republic', 'Belgium', 'Hungary', 'Bulgaria', 'Ireland', 'Latvia', 'Albania', 'Austria', 'Finland', 'Lithuania', 'Slovenia', 'Montenegro', 'Croatia', 'Bosnia and Herzegovina', 'Macedonia', 'Estonia']

4. Write a function called *compute_ave_country_temp* to compute average temperature, based on cities from each country.
   The function will return a dictionary whose key:value pair = country name:its average temp. The size of the returned dictionary must equal the number of countries represented in Exercise 3.
   Hint: Make use of function *list_countries* from Exercise 3.

Partial Program for Main

`# … Continue from 1.2 + Remove last two print lines in 1.2`
 	
`country_temps = compute_ave_country_temp(cities_data)`
`print(len(country_temps))`
`print(country_temps)`

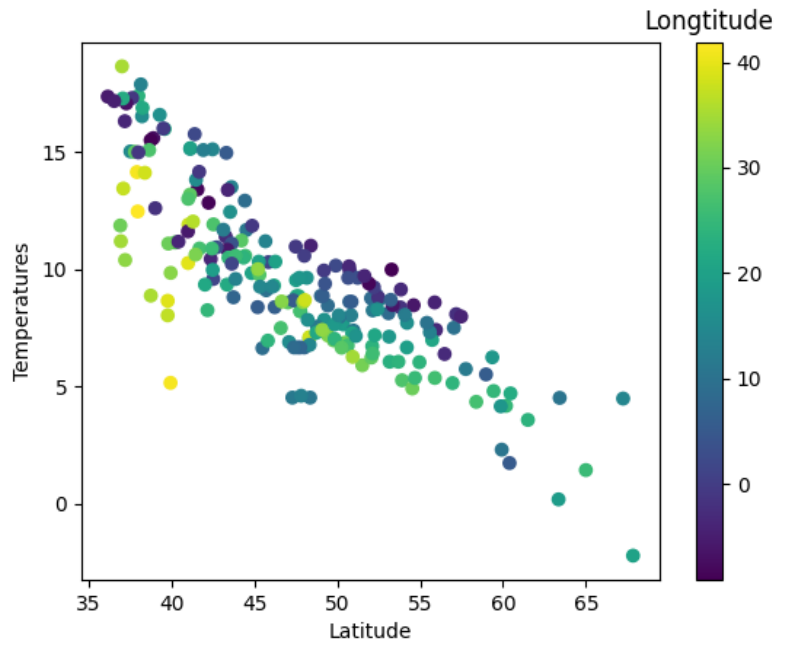
Expected output:
37 {'Denmark': 7.625, 'United Kingdom': 8.649999999999999, 'Sweden': 3.5866666666666673, 'Turkey': 11.726666666666665, 'Spain': 14.238333333333332, 'France': 10.151111111111112, 'Netherlands': 8.756666666666668, 'Italy': 13.474666666666668, 'Andorra': 9.6, 'Romania': 9.224444444444444, 'Greece': 16.9025, 'Germany': 7.869285714285714, 'Moldova': 8.415, 'Switzerland': 7.253333333333333, 'Serbia': 9.85, 'Norway': 3.7260000000000004, 'Poland': 7.250000000000002, 'Ukraine': 7.420000000000002, 'Portugal': 14.469999999999999, 'Slovakia': 8.48, 'Belarus': 5.946666666666666, 'Czech Republic': 7.8566666666666665, 'Belgium': 9.65, 'Hungary': 9.6025, 'Bulgaria': 10.44, 'Ireland': 9.299999999999999, 'Latvia': 5.27, 'Albania': 15.18, 'Austria': 6.144, 'Finland': 3.4875, 'Lithuania': 6.1433333333333335, 'Slovenia': 9.27, 'Montenegro': 9.99, 'Croatia': 10.865, 'Bosnia and Herzegovina': 9.6, 'Macedonia': 9.36, 'Estonia': 4.59}

5. Create a function *read_file* by using code from DictReader (Exercise 1.2) and pass in filename as function input parameter.
   In addition, create *extract_to_list* function to extract data from all rows of specific value. For example, lat in the code below contain latitude of all cities in the file. Try to write *extract_to_list* using list comprehension

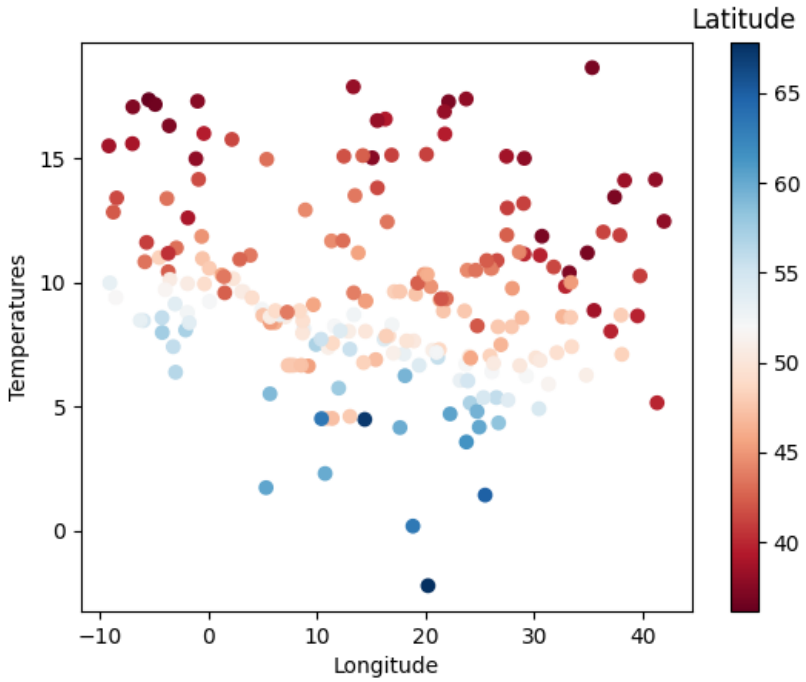
Code for Main

```
import csv
import matplotlib.pyplot as plt

cities_data = read_file('cities.csv')
lat = extract_to_list(cities_data,'latitude')
long = extract_to_list(cities_data,'longitude')
temps = extract_to_list(cities_data,'temperature')
high = extract_to_list(cities_data,'highest')

# Plot scatter plot using x = latitude,
#                         y = temperature,
#                         color=longitude
plt.scatter(lat,temps,c=long)
# Add x-axis label
plt.xlabel('Latitude')
# Add y-axis label
plt.ylabel('Temperatures')
# Add label for color bar
plt.colorbar().ax.set_title('Longtitude')
# Save plot as image file
plt.savefig('lat_temps_long.png')
# Show plot
plt.show()

plt.scatter(long,temps,c=lat)
plt.xlabel('Longitude')
plt.ylabel('Temperatures')
plt.colorbar().ax.set_title('Latitude')
# Set colormap to the selected one
# See more colormap selection in the reference at the end of
Exercise 5
plt.set_cmap('RdBu')
plt.savefig('long_temps_lat.png')
plt.show()
```

Expected output: `lat_temps_long.png`
(See next page)



Expected output: `long_temps_lat.png`



See Scatterplot reference at:
https://matplotlib.org/3.3.2/api/_as_gen/matplotlib.pyplot.scatter.html

See colormap reference at:

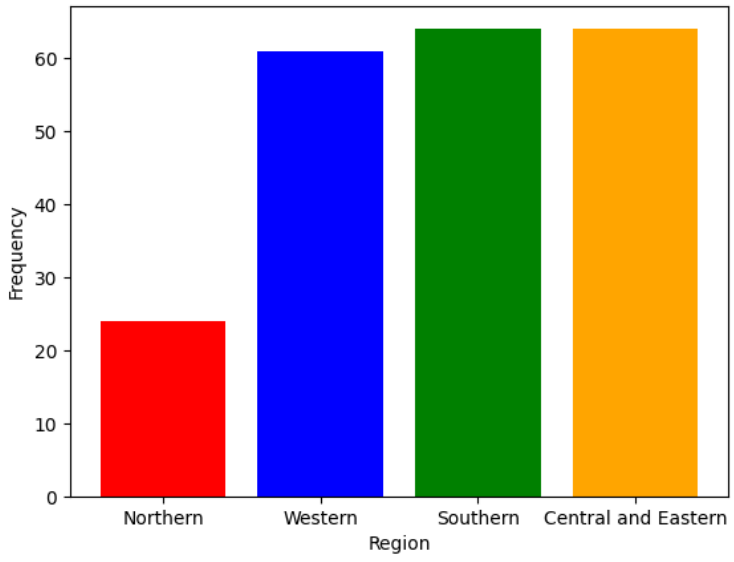
https://matplotlib.org/3.2.1/tutorials/colors/colormaps.html

6. Create a function *count_region_freq* to find unique region and the region frequency. Region frequency = Number of countries in such region. This function returns 2 output values: 1) list of unique regions, and 2) list of region frequencies.

Code for Main

```
import csv
import matplotlib.pyplot as plt

cities_data = read_file('cities.csv')

region_list, region_freq_list = count_region_freq(cities_data)
# Set up bar colors in bar graph
# See more color names in the reference at the end of Exercise 6
my_colors = ['red','blue','green','orange']
# Plot bar graph using x = unique region list
#                      y = region frequency
# Bar graph color is set to my_colors list
plt.bar(region_list, region_freq_list, color=my_colors)
plt.xlabel('Region')
plt.ylabel('Frequency')
plt.savefig('region_freq.png')
plt.show()
```

Expected output: region_freq.png



See Bar plot reference at:
https://matplotlib.org/3.3.2/api/_as_gen/matplotlib.pyplot.bar.html

See color name reference at:
https://matplotlib.org/3.1.0/gallery/color/named_colors.html

7. Create a function *find_lowest_highest_avg_city_temp* to find countries with lowest and highest average city temperature. This function returns 2 output values: 1) country with lowest city temperature and 2) country with highest city temperature.

Sample outputs:

```
>>> find_lowest_highest_avg_city_temp(cities_data[:11])
['Sweden', 'Turkey']
>>> find_lowest_highest_avg_city_temp(cities_data[-10:])
['Lithuania', 'Spain']
>>> find_lowest_highest_avg_city_temp(cities_data)
['Finland', 'Greece']
```