Lab Assignment 2 - Scott Souter

Student ID - 200017121

GitHub Repository URL: https://github.com/scottsouter/PY4SA23_Assignment

Python Basics

Task 1: Create an If...Else statement that will test whether a number is divisible by three. "YOUR VALUE is divisible by 3" should be printed if the value is divisible by three. "YOUR VALUE is not divisible by three" should be printed if it is not divisible by three. Test the statement on a numeric variable. Upper case text in the print statement should be replaced with the tested number.

In [310…

```
x = input("What is your input?")

if float (x) %3 == 0:
    print(str(x) + " is divisible by 3.")
else:
    print(str(x) + " is not divisible by 3.")
```

```
5 is not divisible by 3.
```

I used the input function allowing a value designated as 'x' to be entered. Using '%3 == 0' and the 'if' and 'else' statements, if the whole number can be divided by 3 to return a whole number then ('x' is divisible by 3) will be returned. Otherwise, ('x' is not divisible by 3) will be returned. The float function allows the string value to be converted to a float so it can work with the function '%3 == 0'

T2: Create an If...Else statement that will test whether a type of fruit, represented as a text string, is in a list of acceptable fruits (apple, orange, pear, kiwi, or strawberry). If the fruit is on the list, the following should be printed: "YOUR FRUIT is acceptable." If not, then the following should be printed: "YOUR FRUIT is not acceptable." Upper case text in the print statement should be replaced with the tested fruit.

In [312…

```
fruit_list = ['apple', 'orange', 'pear', 'kiwi', 'strawberry']

fruit = (input('Enter the fruit you want to select:'))

if fruit in fruit_list:
    print('fruit is acceptable')
else:
    print('fruit is not acceptable')
```

```
fruit is acceptable
```

I have created a list containing different fruits and also used the input function allowing a fruit to be selected. Using the 'if' and 'else' statements, if the fruit selected is in the fruit_list then (fruit is acceptable) will be returned. Otherwise, (fruit is not acceptable) will be returned.

T3: Create a function to calculate the distance between two coordinates using the haversine formula. Write the following formula where the input parameters are a pair of coordinates as two lists.

In [366…

```
import numpy as np

from math import *

#import latitude and longitude for coordinate 1
coordinate1 = [lat1, lon1]
coordinate1[0]=(float(input("Enter the latitude of your first coordinate:")))

#import longitude for cooridnate 1
coordinate1[1]=(float(input("Enter the longitude of your first coordinate:")))

#import latitude and longitiude for cooridnate 2
coordinate2 = [lat2, lon2]
coordinate2[0]=(float(input("Enter the latitude of your first coordinate:"))) #float turns the string into a numb

#import longitude for cooridnate 2
coordinate2[1]=(float(input("Enter the longitude of your first coordinate:")))

def haversine(coordinate1, coordinate2):

    coordinate1, coordinate2 = np.radians([coordinate1, coordinate2]) #converting a degree value into radians

    a = sin((coordinate2[0]-coordinate1[0])/2)**2 + cos(coordinate1[0]) * cos(coordinate2[0]) * sin((coordinate2
    c = 2 * asin(sqrt(a))

    return 6371 * c

d=(haversine(coordinate1, coordinate2))

print("The distance between the two coordinates is:", d,"km")
```

```
The distance between the two coordinates is: 2399.128640288973 km
```

Firstly I imported numpy as np and, from math imported '*'. I used the 'input' function to allow values of latitude and longitude to be entered for both coordinates so the distance between these points can be measured. These points are labelled 'lon1', 'lat1', 'lon2', 'lat2'. I used 'float' to convert real numbers or intigers into floats. I used the haversine formula to calculate the distance between the two coordinates. The value '6371' is the radius of the earth in 'km' which is multiplied by 'c', which is calcultaed using 'a'. Finally, I printed 'The distance between the two coordinates is' using the 'print' formula - which is the two coordinates multiplied by haversine.

Pandas and NumPy

Question 1: How many trees are of the Quercus or Acer genus?

In [362…
```
import numpy as np #importing numpy
import pandas as pd #importing pandas

park_trees_df = pd.read_csv('portland_park_trees.csv')
subQuerusAcer = park_trees_df[(park_trees_df["Genus"]=="Quercus") + (park_trees_df["Genus"]=="Acer")]
print("There are", len(subQuerusAcer), "trees of the Quercus or Acer genus")
```

```
There are 5675 trees of the Quercus or Acer genus
```

Firstly, I imported numpy as np and imported pandas as pd. To see how many trees were of the Quercus or Acer genus I first read in the data file 'portland_park_trees.csv' and called it 'park_trees_df'. I then created a subset of this data file called 'subQuercusAcer' which contained only trees from the Quercus and Acer genus. I then used the 'len' function to find out how many trees were within the subset. I used 'print' to show this.

Q2: How many trees are of the Quercus or Acer genus and have a DBH larger than 50 inches?

In [354…
```
QuAcfifty = subQuerusAcer[subQuerusAcer["DBH"]>50] #creating a sub-set where Querus and Acer trees have a DBH of
print("There are", len(QuAcfifty),"trees of the Quercus or Acer genus that have a DBH larger than 50 inches")
```

```
There are 124 trees of the Quercus or Acer genus that have a DBH larger than 50 inches
```

To find out how many trees of the Quercus and Acer genus have a DBH which is larger than 50 inches I created a subset called QuAcfifty. Within this subset 'subQuercusAcer' was filtered so that only trees which had a larger DBH than 50 appeared. I did this using the 'len' function and I printed this using the 'print'

Q3: Which genus has the highest mean DBH of the following genera: Quercus, Acer, or Fraxinus?

In [356…
```
Genus_list = ['Quercus', 'Acer', 'Fraxinus']
df = park_trees_df.loc[park_trees_df['Genus'].isin(Genus_list)]

genus_mean = df.groupby('Genus')['DBH'].mean()
greatest_mean = genus_mean.idxmax()
print("The genus with the greatest mean DBH is", greatest_mean)
```

```
The genus with the greatest mean DBH is Quercus
```

Firstly, I created a list of Quercus, Acer and Fraxinus by creating 'Genus_list'. I then created a subset which contained only the 'Genus' column for the genus in the 'Genus_list'. Then I grouped the rows by unique values in the 'Genus' column and calculated the DBH mean. Using '.idxmax' I then found the greatest mean value for this column and printed this value using 'print'.

Q4: How many different species of trees are recorded in the Acer genus?

In [287…
```
Species_tree = (park_trees_df[["Species", "Genus"]])
Species_tree
AcerSpecies = Species_tree[(Species_tree["Genus"]=="Acer")]
AcerSpecies
print("There are", len(AcerSpecies["Species"].unique()), "different species of trees recorded in the Acer genus")
```

```
There are 20 different species of trees recorded in the Acer genus
```

I have made a subset that just contains "Species" and "Genus" called 'Species_tree'. I have then made a second subset which just contains the Genus, Acer, called AcerSpecies. I finally printed how many different species of tree there are in the Acer Genus using the AcerSpecies subset. I did this using 'len' and the '.unique' function.

Question 5

Calculate a new column named "pop_M" (population in millions), by transforming the "pop" (population) column.

In [363…
```
cities_df = pd.read_csv('world_cities.csv')
```

I read in the dataset 'world_cities.csv' by using the '.read_csv' function and called this 'cities_df'

In [364…
```
cities_df["pop_M"] = cities_df["pop"] / 1000000 # creating new column pop_M
cities_df
```

Out[364]:

	city	country	pop	lat	lon	capital	pop_M
0	'Abasan al-Jadidah	Palestine	5629	31.31	34.34	0	0.005629
1	'Abasan al-Kabirah	Palestine	18999	31.32	34.35	0	0.018999
2	'Abdul Hakim	Pakistan	47788	30.55	72.11	0	0.047788
3	'Abdullah-as-Salam	Kuwait	21817	29.36	47.98	0	0.021817
4	'Abud	Palestine	2456	32.03	35.07	0	0.002456
...	...	...	...	...	...	...	...
43640	az-Zubayr	Iraq	124611	30.39	47.71	0	0.124611
43641	az-Zulfi	Saudi Arabia	54070	26.30	44.80	0	0.054070
43642	az-Zuwaytinah	Libya	21984	30.95	20.12	0	0.021984
43643	s-Gravenhage	Netherlands	479525	52.07	4.30	0	0.479525
43644	s-Hertogenbosch	Netherlands	135529	51.68	5.30	0	0.135529

43645 rows × 7 columns

I used the data stored in the 'pop' column to create a new column called 'pop_M' which derives from the function (pop_M = pop / 1000000) giving the population value by the million. I then printed the new 'cities_df' to get the 'pop_M' column

Remove the original "pop" column.

In [314…
```
#creating a subset to remove original 'pop' column
subset_query = cities_df.query("pop_M > 0")[["city", "country", "lat", "lon", "capital", "pop_M"]]
subset_query #printing new subset
```

Out[314]:

	city	country	lat	lon	capital	pop_M
0	'Abasan al-Jadidah	Palestine	31.31	34.34	0	0.005629
1	'Abasan al-Kabirah	Palestine	31.32	34.35	0	0.018999
2	'Abdul Hakim	Pakistan	30.55	72.11	0	0.047788
3	'Abdullah-as-Salam	Kuwait	29.36	47.98	0	0.021817
4	'Abud	Palestine	32.03	35.07	0	0.002456
...	...	...	...	...	...	...
43640	az-Zubayr	Iraq	30.39	47.71	0	0.124611
43641	az-Zulfi	Saudi Arabia	26.30	44.80	0	0.054070
43642	az-Zuwaytinah	Libya	30.95	20.12	0	0.021984
43643	s-Gravenhage	Netherlands	52.07	4.30	0	0.479525
43644	s-Hertogenbosch	Netherlands	51.68	5.30	0	0.135529

43628 rows × 6 columns

To remove the original 'pop' column I created a new subset called 'subset_query'. Within this subset I made a new dataset in which all values of the 'pop_M' were above 0, but I have also sub-setted this to include everything apart from the original 'pop' column. I then printed this new subset, 'subset_query'

Choose/subset a city that starts with the same letter as your first name (for example, "Mexico City" if your first name is Michael).

In [302…
```
#creating a subset of cities starting with letter 'S' using startwith function
subset_query[subset_query["city"].str.startswith("S")]
```

Out[302]:

	city	country	lat	lon	capital	pop_M
27919	San Antonio de Pale	Equatorial Guinea	-1.41	5.61	0	0.004279
31942	Sa Dec	Vietnam	10.30	105.76	0	0.061430
31943	Sa Kaeo	Thailand	13.82	102.07	0	0.029821
31944	Sa'abat	Algeria	36.63	3.70	0	0.019440
31945	Sa'idah	Algeria	34.84	0.14	0	0.129821
...	...	...	...	...	...	...
36897	Szprotawa	Poland	51.57	15.52	0	0.012760
36898	Sztum	Poland	53.85	18.96	0	0.010022
36899	Szubin	Poland	53.01	17.72	0	0.009145
36900	Szurdokpuspoki	Hungary	47.86	19.70	0	0.001937
36901	Szydlowiec	Poland	51.23	20.86	0	0.012099

4960 rows × 6 columns

To create a subset of cities that starts with the same letter as my firstname I used the new subset 'subset_query' and filtered this subset so that only cities starting with the letter 'S' appeared. I did this by using '.str.startswith'.

Subset the five biggest (i.e., largest population sizes) cities from the country where your selected city is.

In [306…
```
Poland_subset = subset_query[subset_query["country"] == "Poland"] #creating a country subset of just Poland
Poland_subset #printing this subset

Poland_pop = Poland_subset.sort_values(by="pop_M").tail() #printing the 5 largest population sizes using the tai
```

To select the five biggest populations in a city from Poland I first created a new subset 'Poland_subset' and filtered the subset 'subset_query' inside of this subset to only contain the country Poland. I then created a second subset 'Poland_pop' and within this subset, it was filtered that the previous subset created 'Poland_subset' only contained the cities with the five largest populations in Poland. This was done by using '.sort values' and '.tail'.

Print the result

In [305…
```
Poland_pop #printing this subset
```

Out[305]:

	city	country	lat	lon	capital	pop_M
29756	Poznan	Poland	52.40	16.90	0	0.567957
41895	Wroclaw	Poland	51.11	17.03	0	0.633276
8382	Cracow	Poland	50.06	19.96	0	0.753829
21253	Lodz	Poland	51.77	19.46	0	0.762615
41441	Warsaw	Poland	52.26	21.02	1	1.634441

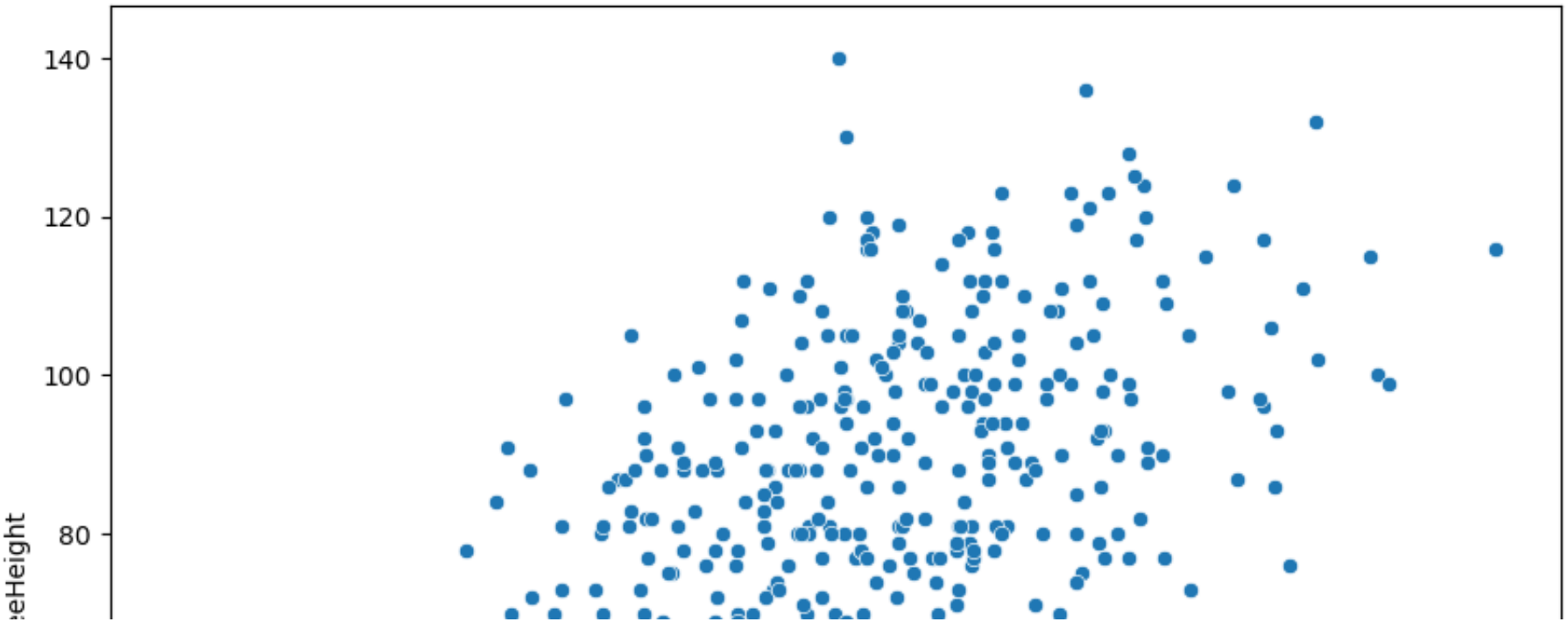
I then printed this reuslt by typing 'Poland_pop'

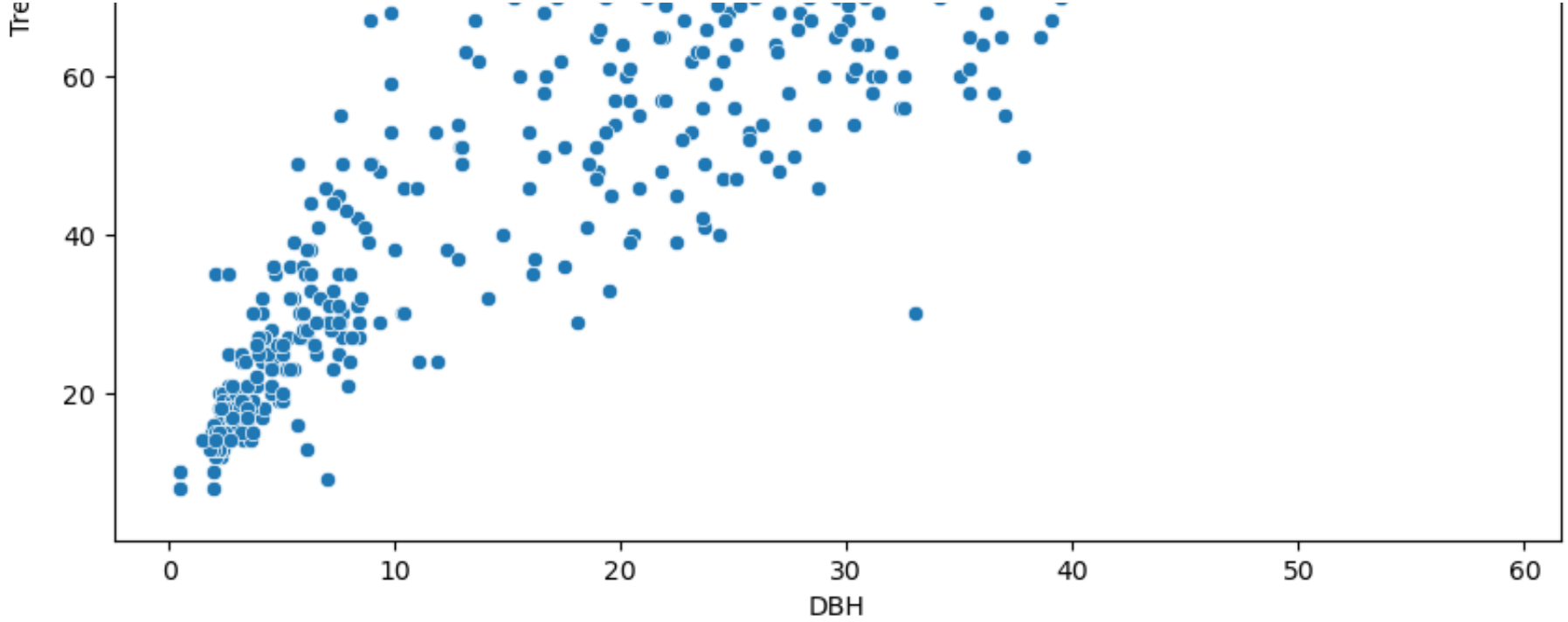
Python Data Visualization

Graph 1: Create a scatterplot for just trees in the Ulmus genus with DBH mapped to the x-axis and tree height mapped to the y-axis (Hint: You will need to use the “Genus”, “DBH”, and “TreeHeight” attributes.).

In [328…
```
#impprting mathplot
import matplotlib.pyplot as plt
#importing seaborn
import seaborn as sns
#creating scatterplot of trees in Ulmus genus
sns.scatterplot(data=park_trees_df.loc[park_trees_df["Genus"]=="Ulmus"], x='DBH', y='TreeHeight')
```

Out[328]: <AxesSubplot: xlabel='DBH', ylabel='TreeHeight'>



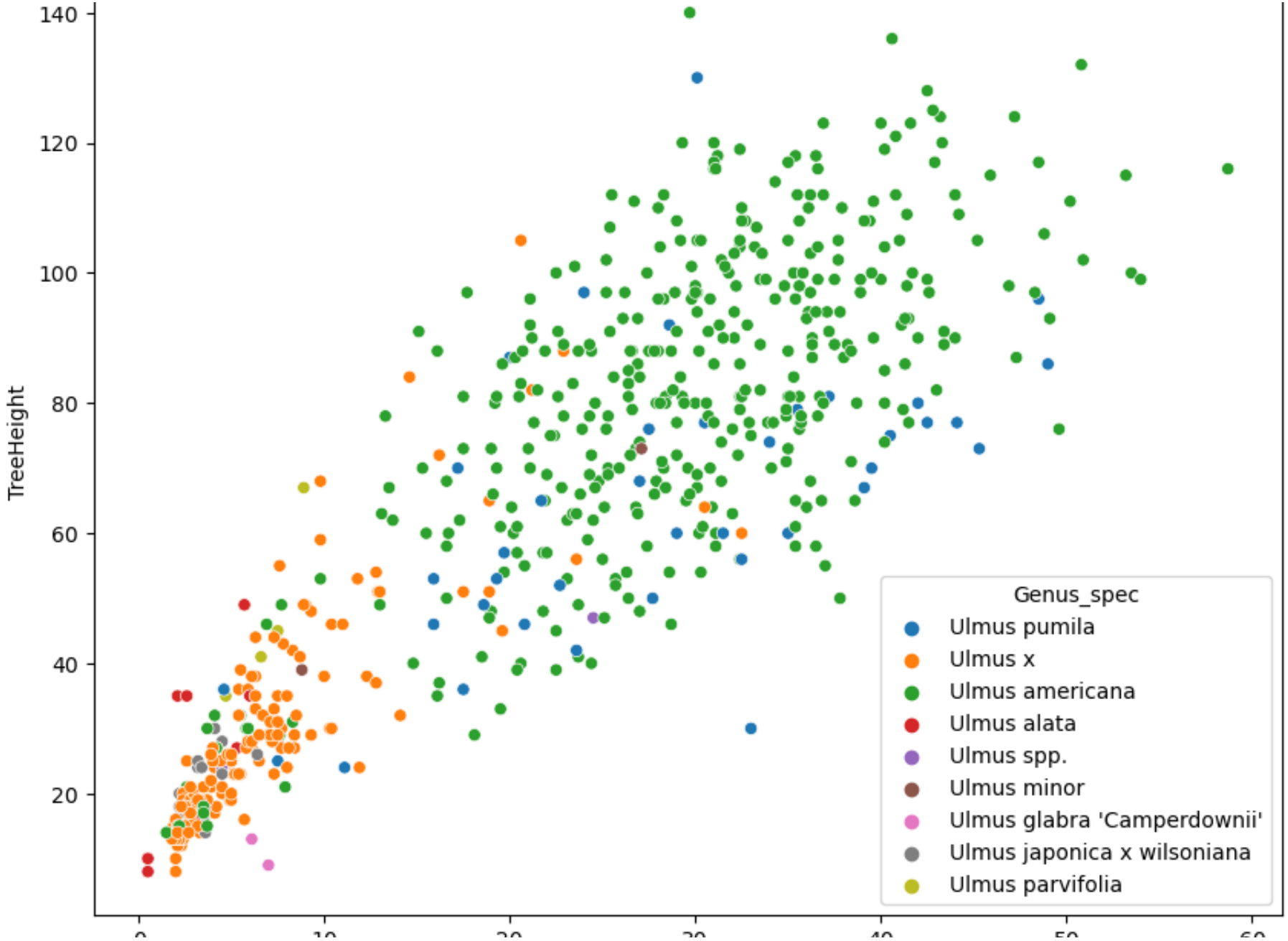


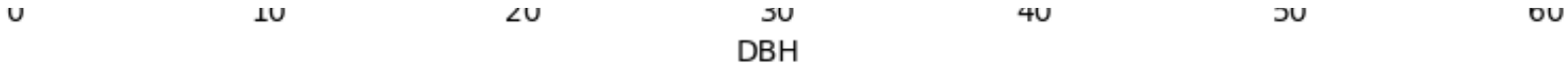
I imported mathplotlib and seaborn. I then created a scatterplot using 'loc' (allowing the return of specified columns in the dataframe) for just trees in the Ulmus genus with DBH on the 'x' axis and tree height on the 'y' axis. I did this by using the '.scatterplot' function.

G2: Create a scatterplot for just trees in the Ulmus genus with DBH mapped to the x-axis, tree height mapped to the y-axis, and tree species mapped to hue (Hint: You will need to use the "Genus", "Genus_spec", "DBH", and "TreeHeight" attributes.)

In [329…
```
#creating scatterplot of trees in Ulmus genus
sns.scatterplot(data=park_trees_df.loc[park_trees_df["Genus"]=="Ulmus"], x='DBH', y='TreeHeight', hue='Genus_spe
```

Out[329]: <AxesSubplot: xlabel='DBH', ylabel='TreeHeight'>


TreeHeight
140
120
100
80
60
40
20
Genus_spec
Ulmus pumila
Ulmus x
Ulmus americana
Ulmus alata
Ulmus spp.
Ulmus minor
Ulmus glabra 'Camperdownii'
Ulmus japonica x wilsoniana
Ulmus parvifolia

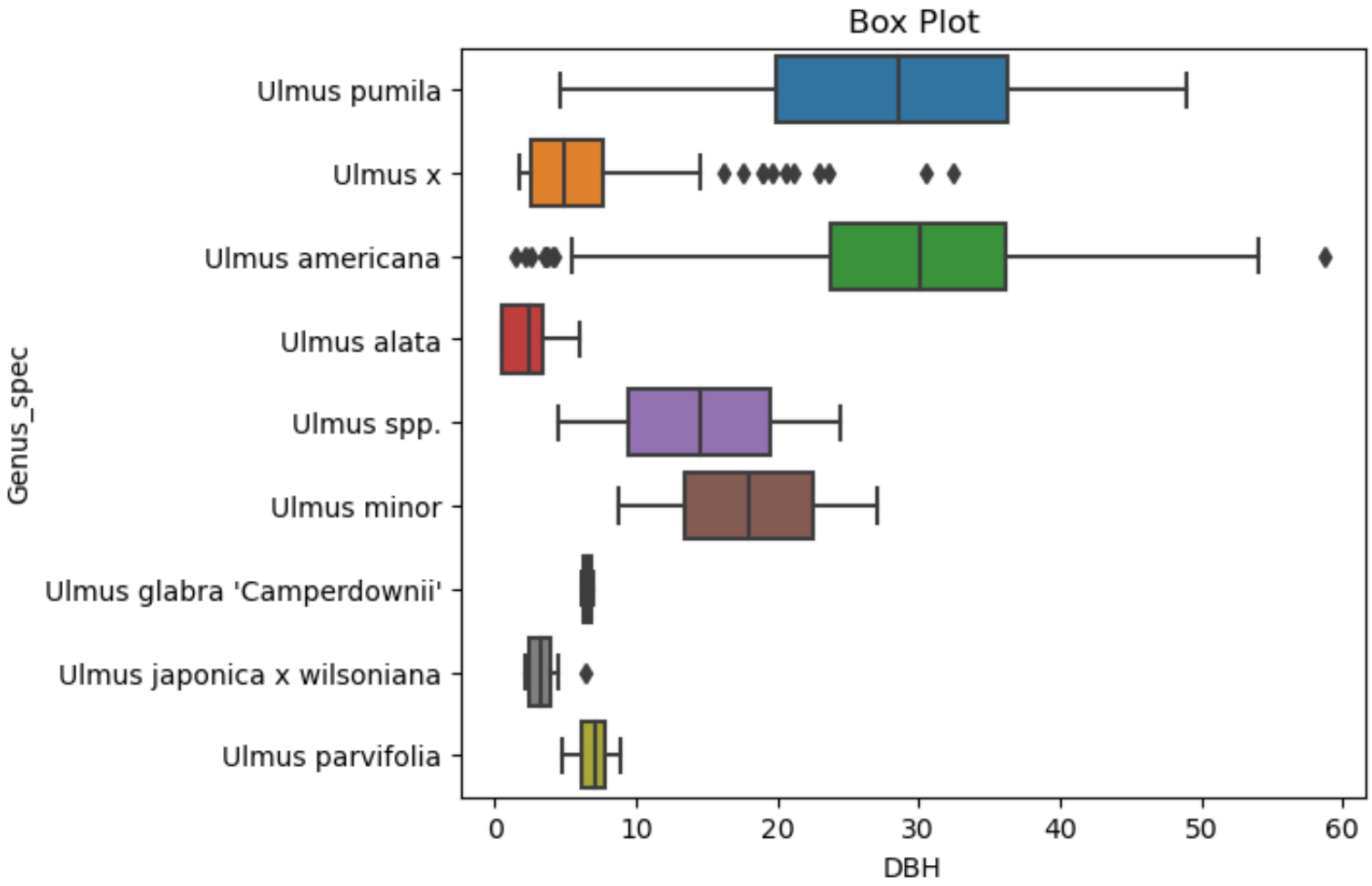


I created a scatterplot using the 'loc' (allowing the return of specified columns in the dataframe) for just trees in the Ulmus genus with DBH on the 'x' axis and tree height on the 'y' axis and I also added a third variable, tree species, mapped to hue. I did this by using the '.scatterplot' function.

G3: Create a boxplot of DBH for just the Ulmus genus differentiated by species (or, each species should have its own boxplot).

In [330…

```
#setting plot bounds
plt.figure(figsize=(6, 5))
#creating boxplot of DBH for just the Ulmus Genus
sns.boxplot(data=park_trees_df.loc[park_trees_df["Genus"]=="Ulmus"], x='DBH', y='Genus_spec')
#setting a title for the boxplot
plt.title('Box Plot')
#printing the boxlot
plt.show()
```

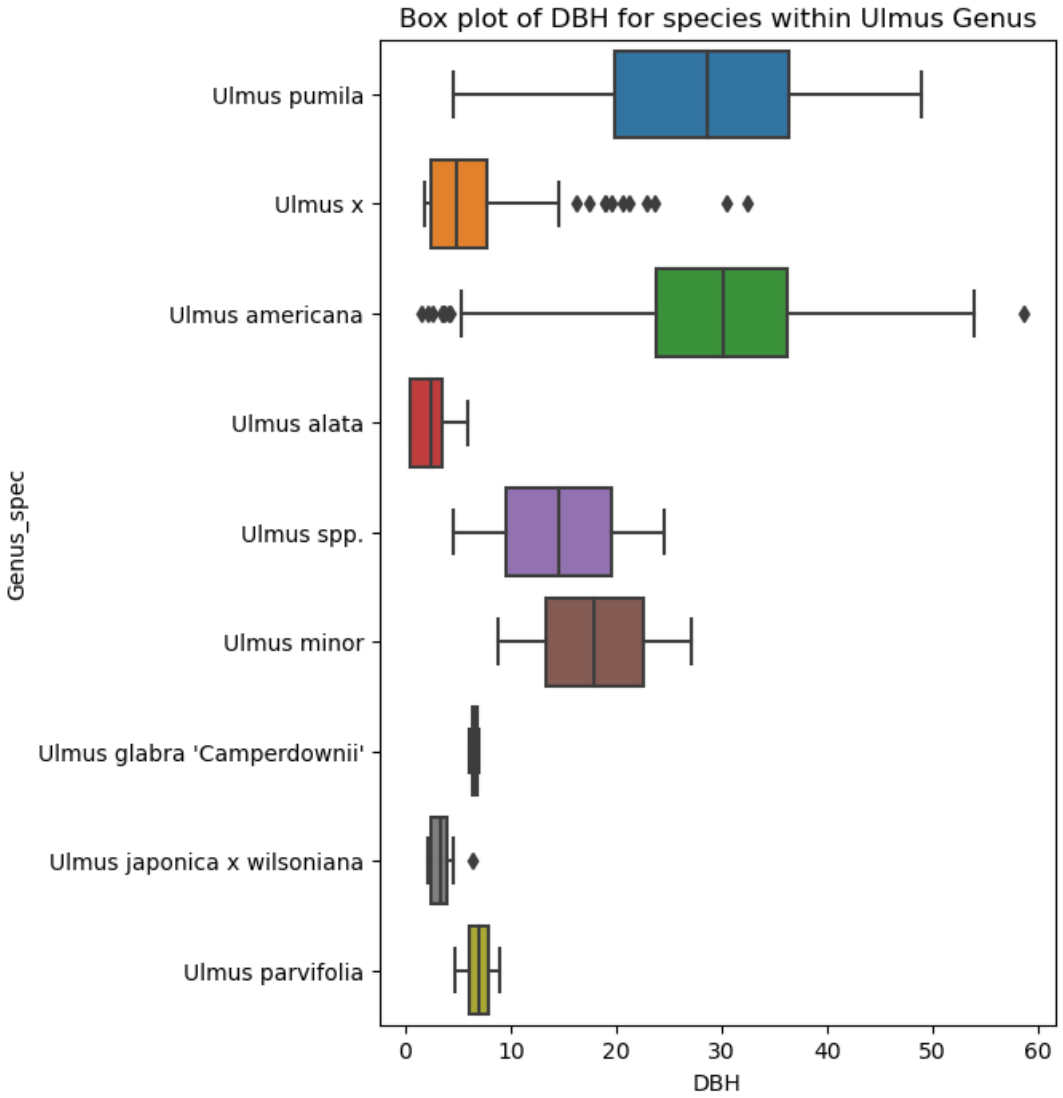


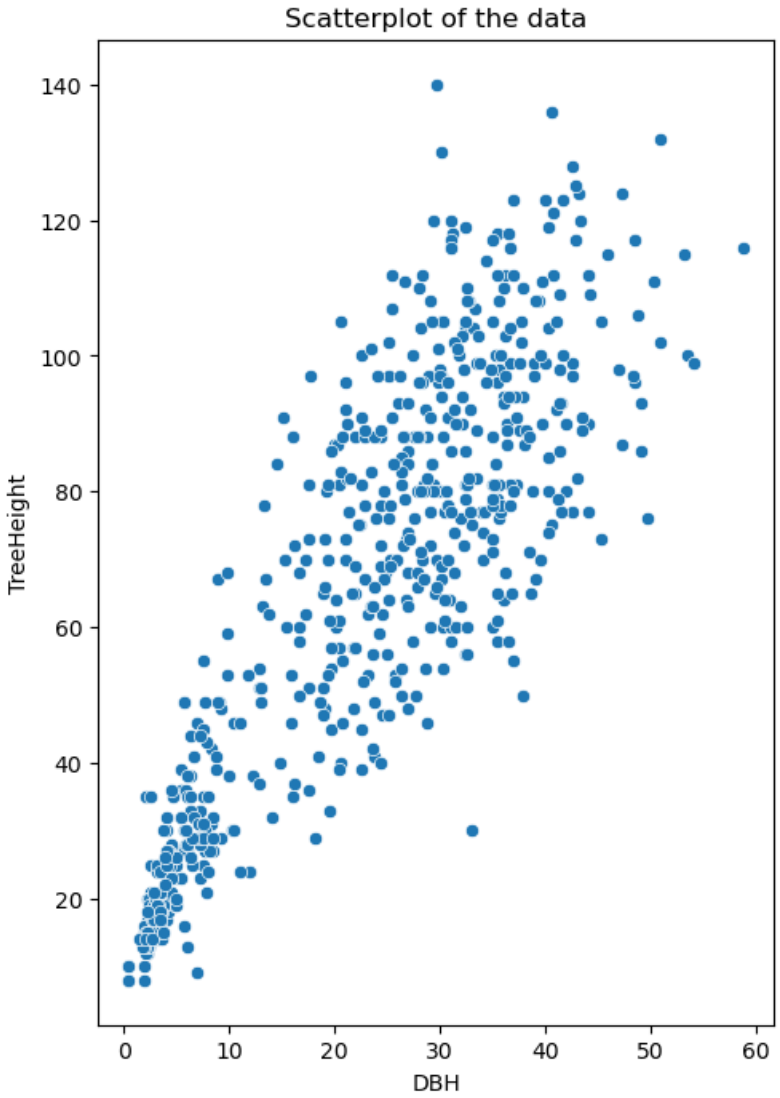
I firstly set the plot bounds for the boxplot. I then created a boxplot, using 'loc', of DBH for just the Ulmus genus differentiated by species using the '.boxplot' function. I added 'DBH' to the 'x' variable, and 'Genus_spec' to the 'y' variable. I then added a title to the boxplot using the '.title' function and then printed this boxplot by using 'plt.show()'.

G4: Combine Graphs 1 and 3 into a single figure. Do not plot a legend for any of the graphs.

In [357…

```
import matplotlib.pyplot as plt
import seaborn as sns

#creating a figure with two sub-plots beside each other
fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(12, 8))

#plotting boxplot on the left subplot
sns.boxplot(data=park_trees_df.loc[park_trees_df["Genus"]=="Ulmus"], x='DBH', y='Genus_spec', ax=ax1)
ax1.set_title('Box plot of DBH for species within Ulmus Genus')

#plotting scatterplot on the right subplot
sns.scatterplot(data=park_trees_df.loc[park_trees_df["Genus"]=="Ulmus"], x='DBH', y='TreeHeight', ax=ax2)
ax2.set_title('Scatterplot of the data')

#print sub-plots
plt.show()
```


Box plot of DBH for species within Ulmus Genus
Genus_spec
Ulmus pumila
Ulmus x
Ulmus americana
Ulmus alata
Ulmus spp.
Ulmus minor
Ulmus glabra 'Camperdownii'
Ulmus japonica x wilsoniana
Ulmus parvifolia
0
10
20
30
40
50
60
DBH


Scatterplot of the data
TreeHeight
140
120
100
80
60
40
20
0
10
20
30
40
50
60
DBH

I combined graphs 1 and 3 into a single figure firstly by creating a figure with two sub-plots beside each other (ax1 and ax2) by using the '.subplots' function and I also added suitable figure sizes. I then designated the number of subplot columns to 2 using the 'ncols=2' function. Next I plotted the boxplot created previously to the left side of the subplot (ax1) and set a title to this by using the '.set_title' function. Then I plotted the scatterplot created previously to the right side of the subplot (ax2) and set a title to this by using the '.set_title' function. Finally I printed these graphs combined into one figure using 'plt.show'.

Python GeoPandas

In [332…
```
import geopandas as gpd #importing geopandas
import pandas as pd #importing pandas
```

I imported geopandas as gpd and imported pandas as pd

Task 1: Read the selected dataset as GeoPandas DataFrame

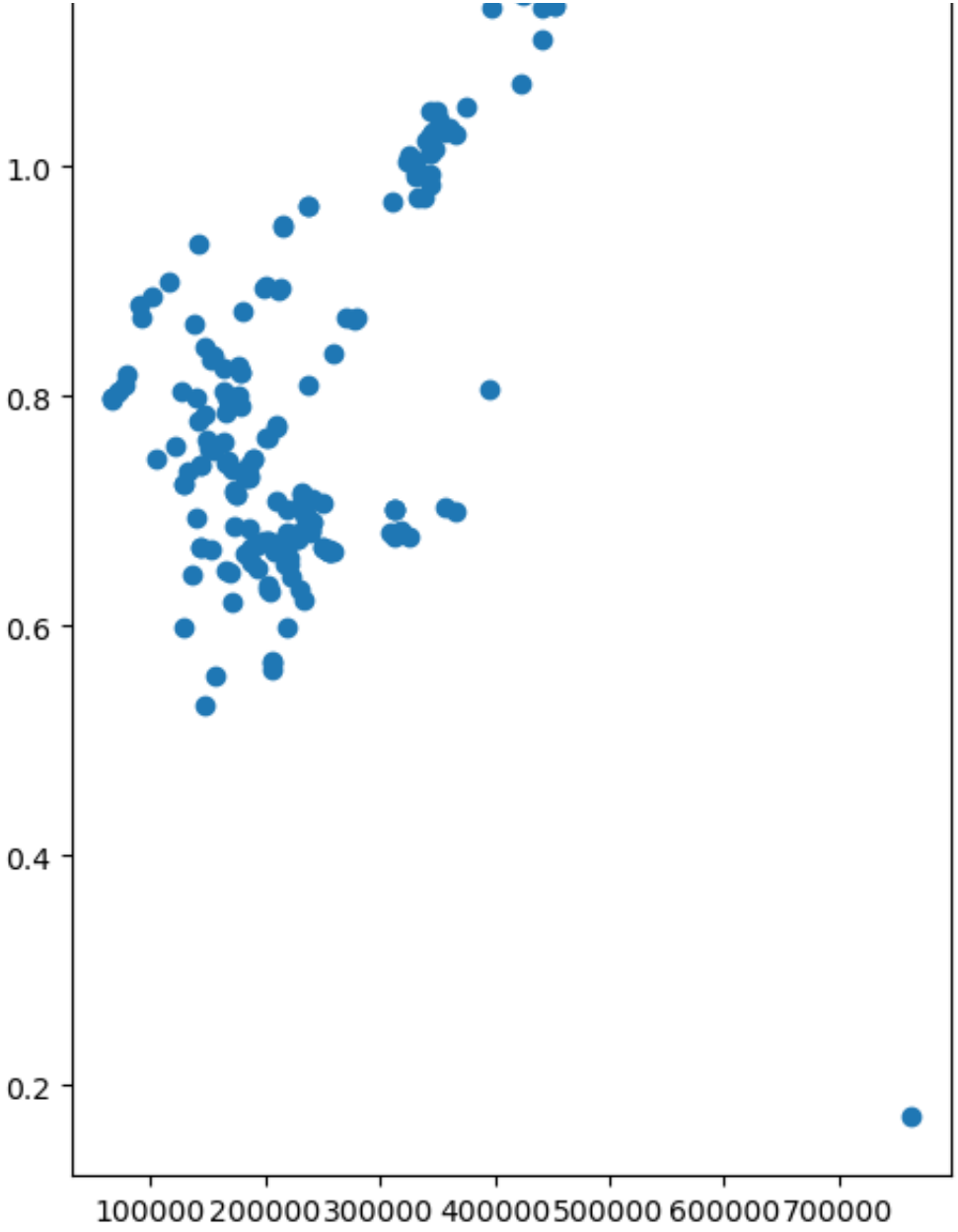
In [333…
```
ferry_geo = gpd.read_file('SG_FerryPorts_2021/SG_FerryPorts_2021.shp') #reading in FerryPorts file
```

I read in the 'SG_FerryPorts_2021.shp' file from the Spatial Data portal of Scotland which shows the passenger and vehicle ferry ports in Scotland and called this 'ferry_geo'

T2: Use the correct code to plot the first 5 and the last 5 sets of records in your selected dataset.

In [334…
```
ferry_geo.plot() #plotting the FerryPorts dataset
```

Out[334]: <AxesSubplot: >


1.0
0.8
0.6
0.4
0.2
100000
200000
300000
400000
500000
600000
700000

I plotted the 'SG_FerryPorts_2021.shp' file using the '.plot()' function

In [335…
```
ferry_geo #plotting the first and last 5 sets of records for FerryPorts dataset
```

Out[335]:

	PortCode	PortName	LocalAutho	MarineRegi	ServiceTyp	Status	ZElev	geometry
0	NaN	Anstruther	Fife	Forth and Tay	Commercial	Active	0	POINT Z (356873.187 703448.126 0.000)
1	NaN	Isle of May	Fife	Forth and Tay	Commercial	Active	0	POINT Z (365975.139 699070.039 0.000)
2	NaN	Raasay (Suisnish)	Highland	West Highlands	Lifeline	Inactive	0	POINT Z (155502.835 834146.438 0.000)
3	NaN	Larne	International	NaN	Commercial	Active	0	POINT Z (155907.432 557263.208 0.000)
4	NaN	Ballycastle	International	NaN	Commercial	Active	0	POINT Z (130063.698 598238.289 0.000)
...	...	...	...	...	...	...	...	...
202	WBU	West Burrafirth	Shetland Islands	Shetland Isles	Lifeline	Active	0	POINT Z (425571.648 1156856.275 0.000)
203	WLS	Walls	Shetland Islands	Shetland Isles	Lifeline	Active	0	POINT Z (423966.216 1148924.330 0.000)
204	WMB	Wemyss Bay	Inverclyde	Clyde	Lifeline	Active	0	POINT Z (219248.709 668511.680 0.000)
205	WYR	Wyre	Orkney Islands	Orkney Islands	Lifeline	Active	0	POINT Z (344478.668 1026808.944 0.000)
206	YOK	Yoker	Glasgow City	Clyde	Lifeline	Active	0	POINT Z (251223.184 668571.302 0.000)

207 rows × 8 columns

I plotted the first 5 and last 5 sets of records for the 'SG_FerryPorts_2021.shp' dataset by typing in the dataset (ferry_geo) I created when I read in the 'SG_FerryPorts_2021.shp' file.

T3. Create a map where you can explore the selected dataset. Try to plot the map using some categorical attribute. Include a ToolTip.

In [358…

```
ferry_geo.explore(column='PortName', cmap='RdYlBu', legend = False)
```

Out[358]: Make this Notebook Trusted to load map: File -> Trust Notebook

I created a map where you can explore the Ferry Ports dataset (ferry_geo) by using the '.explore' function. I selected a suitable column to explore using the 'column=' function, in this case 'PortName'. I also selected a suitable colour for the map using the 'cmap=' function. the legend was removed because it is too big.

T4. What is the Coordinate Reference System of the selected dataset?

In [315…
```
print(ferry_geo.crs)
```

```
EPSG:27700
```

I used 'print()' and then the name of the data set, in this case 'ferry_geo'. I then put .crs after the name of the data set to find out the Coordinate Refernce System of this selected data set.

T5. How many features does the selected dataset contain?

In [229…
```
print(len(ferry_geo))
```

```
207
```

I used 'print()' and then 'len()' and then the name of the dataset ('ferry_geo') to find out how many features the dataset 'ferry_geo' contains.

T6. Define a sub-setting criterion to create a new geopandas dataframe where you filter the selected dataset based on a categorical attribute.

In [316…
```
cat_ferry_geo = ferry_geo[ferry_geo["PortName"]=="Walls"] #creating a sub-set by filtering the PortName to just
cat_ferry_geo #printing this subset
```

Out[316]:

	PortCode	PortName	LocalAutho	MarineRegi	ServiceTyp	Status	ZElev	geometry
203	WLS	Walls	Shetland Islands	Shetland Isles	Lifeline	Active	0	POINT Z (423966.216 1148924.330 0.000)

I defined a sub-set criterion 'cat_ferry_geo' to create a new geopandas dataframe where I filtered the selcted dataset 'ferry_geo' based on the categorical variable 'PortName' to just 'Walls'. I then printed this subset 'cat_ferry_geo'.

T7. Define a sub-setting criterion to create a new geopandas dataframe where you filter the selected dataset based on a numerical attribute.

In [317…
```
num_ferry_geo = ferry_geo[ferry_geo["ZElev"] >= 0] #creating a sub-set by filtering the ZElev to greater than 0
num_ferry_geo #printing this subset
```

Out[317]:

	PortCode	PortName	LocalAutho	MarineRegi	ServiceTyp	Status	ZElev	geometry
0	NaN	Anstruther	Fife	Forth and Tay	Commercial	Active	0	POINT Z (356873.187 703448.126 0.000)
1	NaN	Isle of May	Fife	Forth and Tay	Commercial	Active	0	POINT Z (365975.139 699070.039 0.000)
2	NaN	Raasay (Suisnish)	Highland	West Highlands	Lifeline	Inactive	0	POINT Z (155502.835 834146.438 0.000)
3	NaN	Larne	International	NaN	Commercial	Active	0	POINT Z (155907.432 557263.208 0.000)
4	NaN	Ballycastle	International	NaN	Commercial	Active	0	POINT Z (130063.698 598238.289 0.000)
...	...	...	...	...	...	...	...	...
202	WBU	West Burrafirth	Shetland Islands	Shetland Isles	Lifeline	Active	0	POINT Z (425571.648 1156856.275 0.000)
203	WLS	Walls	Shetland Islands	Shetland Isles	Lifeline	Active	0	POINT Z (423966.216 1148924.330 0.000)
204	WMB	Wemyss Bay	Inverclyde	Clyde	Lifeline	Active	0	POINT Z (219248.709 668511.680 0.000)
205	WYR	Wyre	Orkney Islands	Orkney Islands	Lifeline	Active	0	POINT Z (344478.668 1026808.944 0.000)
206	YOK	Yoker	Glasgow City	Clyde	Lifeline	Active	0	POINT Z (251223.184 668571.302 0.000)

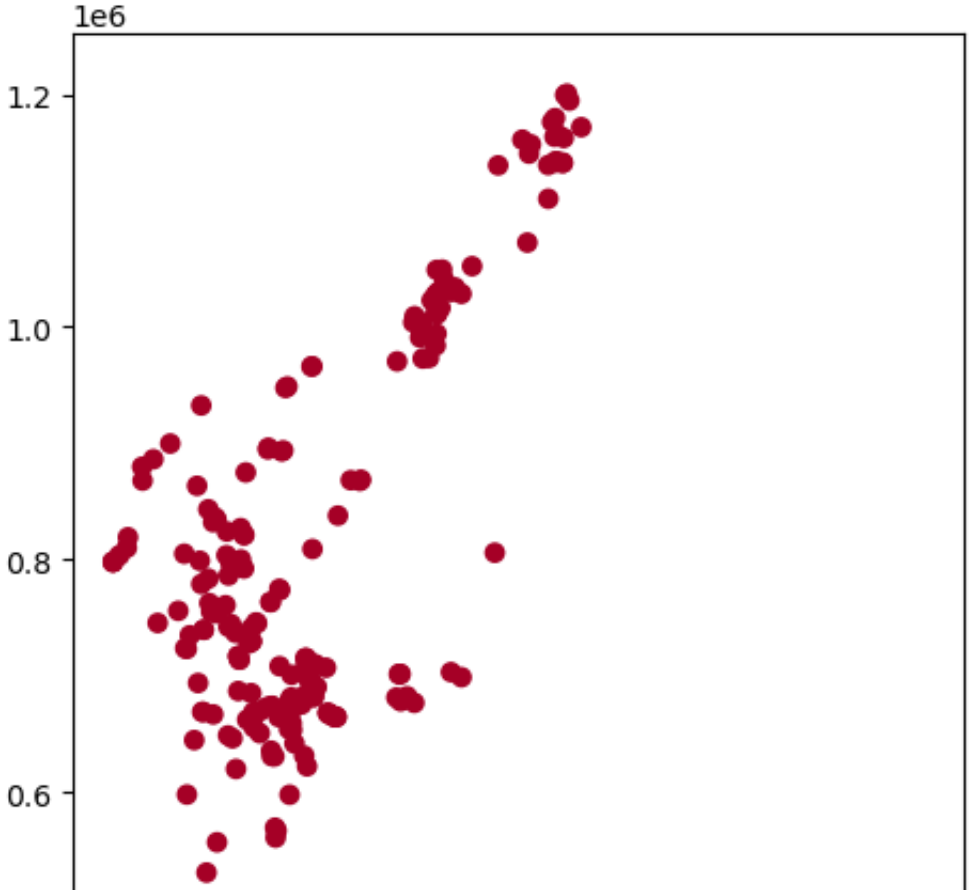
207 rows × 8 columns

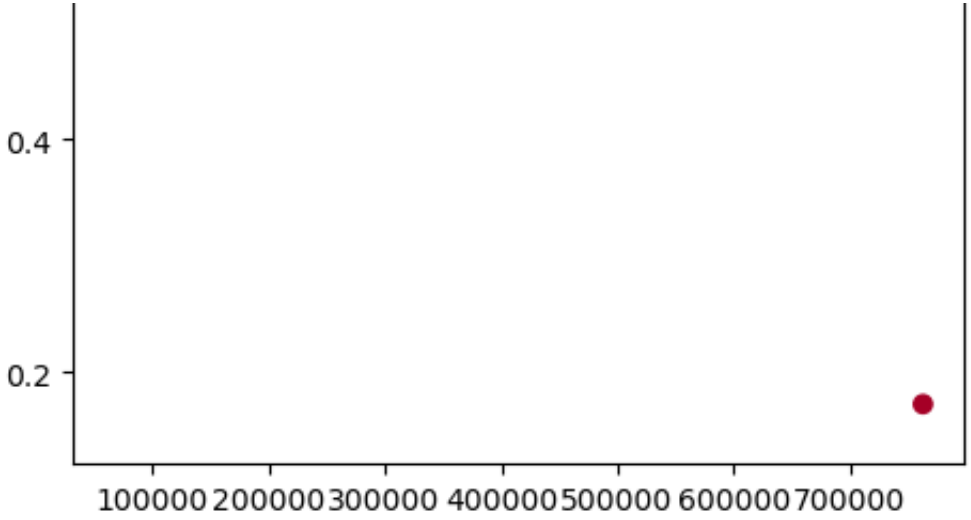
I defined a sub-set criterion 'num_ferry_geo' to create a new geopandas dataframe where I filtered the selcted dataset 'ferry_geo' based on the numerical variable 'ZElev' to '>=0'. I then printed this subset 'num_ferry_geo'

T8. Plot the new/filtered geopandas dataframe using one of the attributes to create a choropleth map.

In [360…
```
#plotting the dataframe to a choropleth map using the catergorical attribute using the plot function
num_ferry_geo.plot(column='ZElev' ,cmap='RdYlBu')
```

Out[360]: <AxesSubplot: >





I plotted the new numerical geopandas dataframe using the '.plot' function on the new dataframe to create a choropleth map. Within this '.plot()' function I used 'column' to select the appropriate column, 'ZElev', and I also selected a suitbale colour for this map using the 'cmap' function. I also had to remove the legend from this chlorepleth map as it was too big.

Python Rasterrio

In [361…
```
import rasterio as rio
from rasterio import plot
```

I imported rasterio as rio, and from rasterio imported plot.

Task 1. Read the file as a rasterio dataset

In [338…
```
# reading the file as a rasterio dataset
elev_geo = rio.open('elev.tif')
elev_array = elev_geo.read(1)
elev_array
```

Out[338]:
```
array([[-32768, -32768, -32768, ..., -32768, -32768, -32768],
       [-32768, -32768, -32768, ..., -32768, -32768, -32768],
       [-32768, -32768, -32768, ..., -32768, -32768, -32768],
       ...,
       [-32768, -32768, -32768, ..., -32768, -32768, -32768],
       [-32768, -32768, -32768, ..., -32768, -32768, -32768],
       [-32768, -32768, -32768, ..., -32768, -32768, -32768]], dtype=int16)
```

Using rio.open(), I opened the 'elev.tif' file and called it 'elev_geo'. I read the file using the '.read()' method coinciding with the array function to produce the array which shows the elevation vlaues.

T2: What is the CRS of the dataset?

In [321…
```
print(elev_geo.crs)
```

```
EPSG:32617
```

I used 'print()' and then the name of the data set, in this case 'elev_geo'. I then put .crs after the name of the data set to find out the Coordinate Refernce System of this selected data set.

T3: Describe the raster dataset regarding the raster extent (bounds), the reference system, and how many bands are in this dataset.

In [322…
```
print(elev_geo.bounds)
print(elev_geo.crs)
print(elev_geo.count)
```

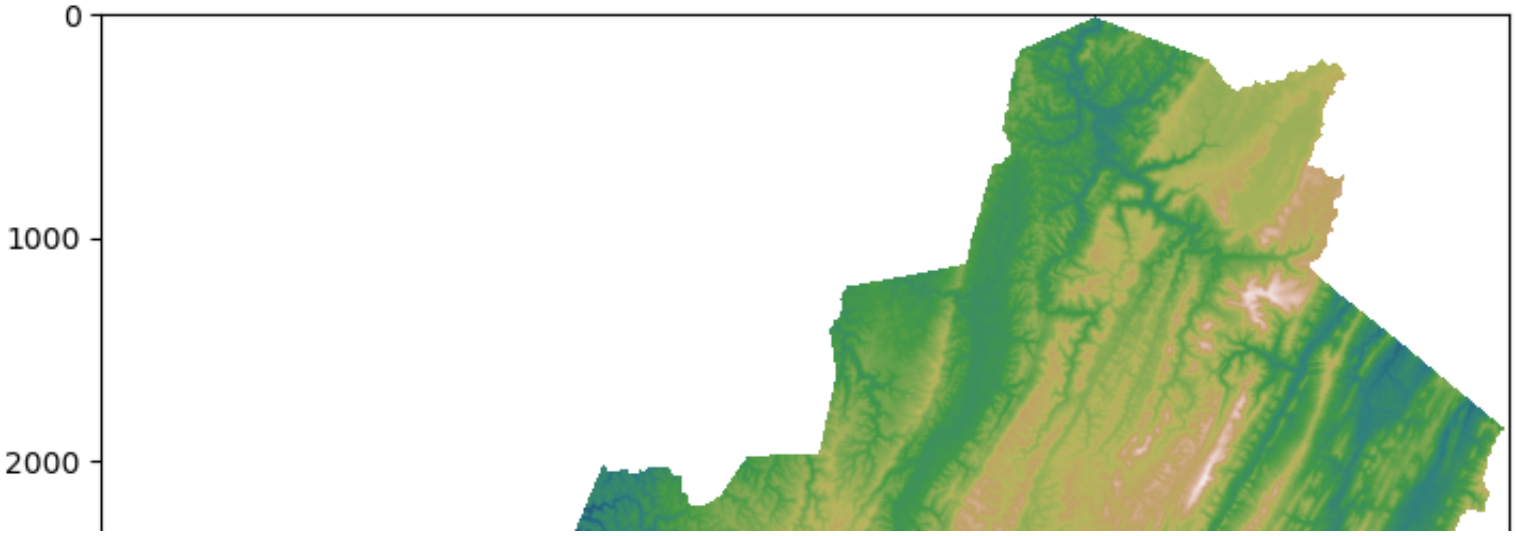
```
BoundingBox(left=479753.39945587853, bottom=4170823.2037591375, right=668843.3994558785, top=4347733.203759138)
EPSG:32617
1
```

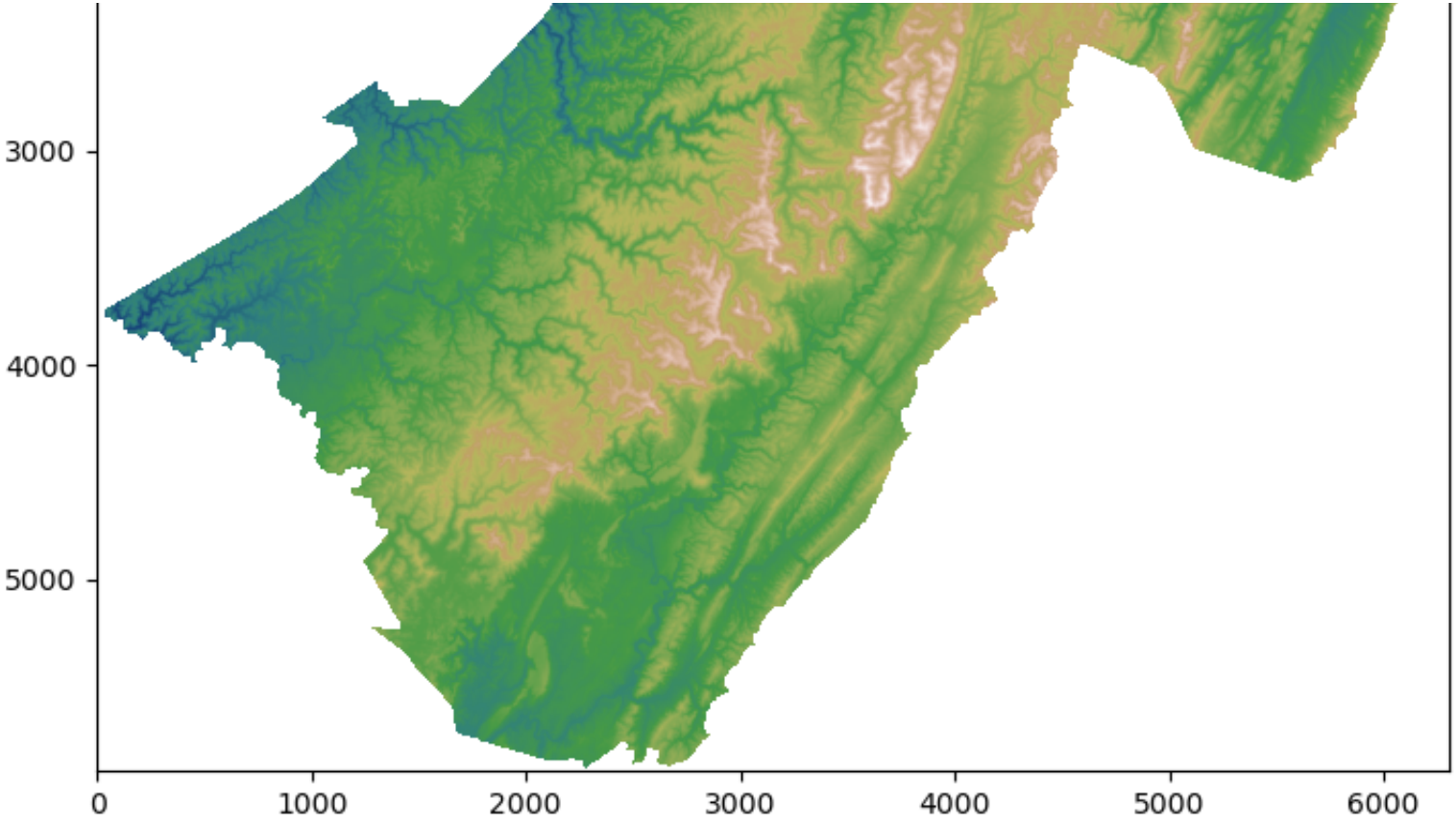
I used 'print()' and then the name of the data set, 'elev_geo'. I then put the .bounds function after the name of the dataset to describe the raster dataset regarding the raster extent (bounds). I used 'print()' and then the name of the data set, 'elev_geo'. I then put the .crs function after the name of the data set to describe the reference system. I used 'print()' and then the name of the data set, 'elev_geo'. I then put the .count function after the name of the data set to describe how many bands are in this selected data set.

T4: Create a plot/map of the raster dataset.

In [323…
```
rasterio_read_masked = np.ma.masked_array(elev_array, mask=(elev_array == -32768))
rasterio_read_masked

#display masked raster using the gist_grey colour
plt.rcParams['figure.figsize'] = [10, 8]
plt.imshow(rasterio_read_masked, cmap="gist_earth")
```

Out[323]: `<matplotlib.image.AxesImage at 0x1865ecb80>`



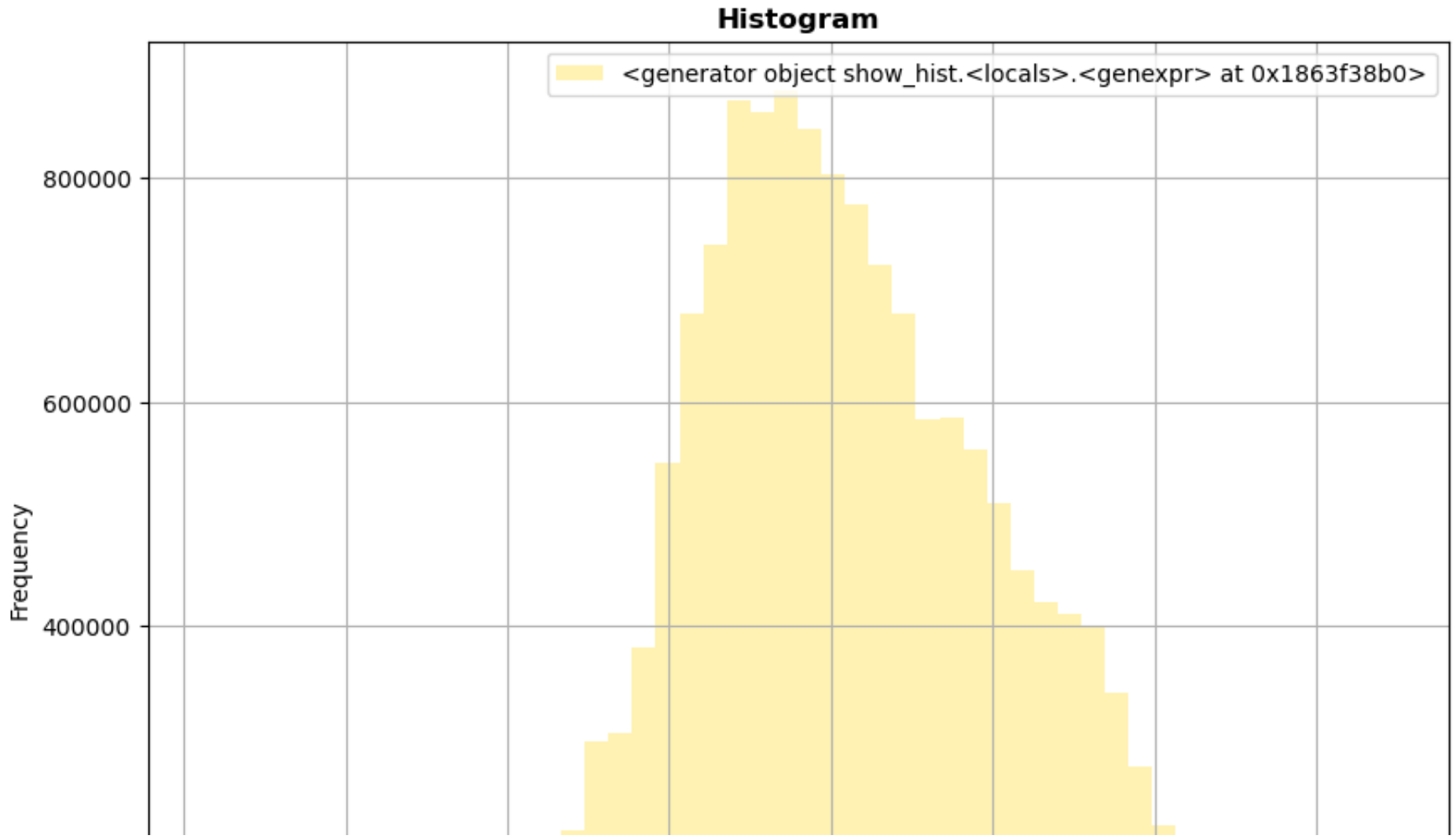


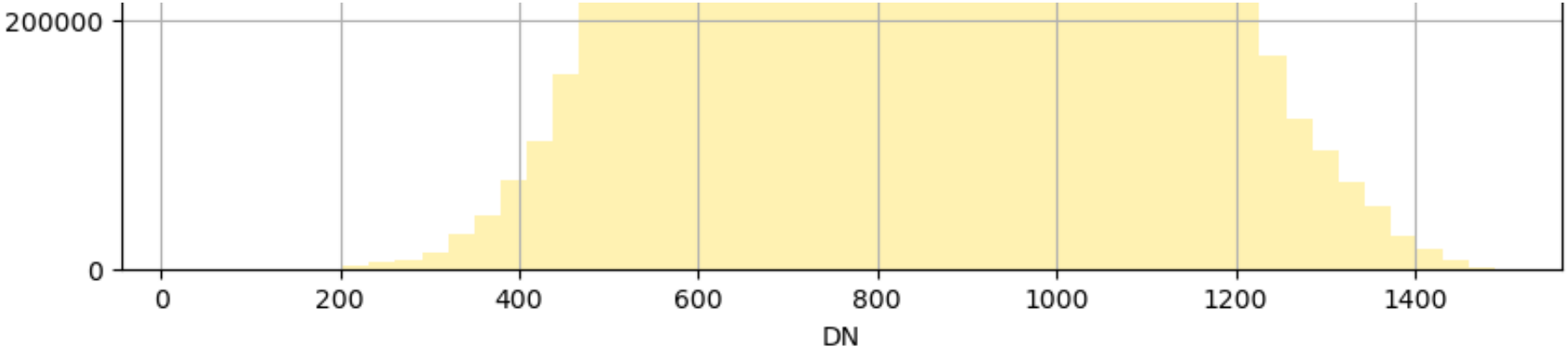
T5: Create Histograms from the raster.

I have firstly masked the 'elev_array' dataset and called this 'rasterio_read_masked' using 'ma.masked_array'. I set the figure size at [10, 8] using '.rcParams' and then using the 'imshow' method I have mapped the raster data. I selected a suitable colour using 'cmap'.

In [324…

```
from rasterio.plot import show_hist

#creating histogram of 50 bins using show_hist with the set paramters
show_hist(rasterio_read_masked, bins=50, lw=0.0, stacked=False, alpha = 0.3,
         histtype='stepfilled', title='Histogram')
```





I firstly imported the 'show_hist' function from rasterio.plot. I then used the 'show_hist' function to create a histogram highlighting the changes of frequency in elevation. I set the number of bins to be 50, the lw=0.0, (linewidth), stacked = False, alpha = 0.3, the 'histtype'='stepfilled. I also set the 'title'='Histogram'.