DCS 540 Data Preparation (DSC540-T301 2225-1)

Bellevue University

Assignment: Weeks 3 & 4 Exercises

Author: Jake Meyer

Date: 04/08/2022

Assignment Instructions:

1. Data Wrangling with Python: Activity 5, page 116
2. Data Wrangling with Python: Activity 6, page 171
3. Create a series and practice basic arithmetic steps:

- a. Series 1 = 7.3, -2.5, 3.4, 1.5
- i. Index = 'a', 'c', 'd', 'e'
- b. Series 2 = -2.1, 3.6, -1.5, 4, 3.1
- i. Index = 'a', 'c', 'e', 'f', 'g'
- c. Add Series 1 and Series 2 together and print the results
- d. Subtract Series 1 from Series 2 and print the results

1. Data Wrangling with Python: Activity 5, page 116

In [617…

```
'''
Load the essential libraries required for reading a
.csv file for the Boston housing dataset, storing dataframes,
and plotting the data. Start with NumPy, Pandas, and matplotlib. Additional
libraries will be added later in the code if required.
'''
# Import NumPy as np similar to how Python for Data Analysis portrays on Page 89.
import numpy as np

# Import Pandas as pd as shown on Page 125 from Python for Data Analysis.
# Specifically from pandas, Series and DataFrame will be imported into the local name s
# (not required, but wanted to include in the code)
import pandas as pd
from pandas import Series, DataFrame

# Import matplotlib.pyplot as shown in Data Wrangling with Python on page 100.
import matplotlib.pyplot as plt
```

In [618…

```
'''
Locate the Boston_housing.csv file provided with the following
link: https://github.com/TrainingByPackt/Data-Wrangling-with-Python/tree/master/Chapter
Store the file in the same working repository as this assignment.
If unable to store in same working repository, then keep track of the directory path an
The read_csv() function from pandas will be utilized to read the data.
'''
# Store the Boston_housing.csv data as df.
df = pd.read_csv('Boston_housing.csv')
```

In [619…

```
'''
This section of code will check the first 10 records (index 0-9) across each column.
'''
# Use head() to view the first 10 records in df.
df.head(10)
```

Out[619…

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	PRIC
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.(
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.(
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2
5	0.02985	0.0	2.18	0	0.458	6.430	58.7	6.0622	3	222	18.7	394.12	5.21	28.7
6	0.08829	12.5	7.87	0	0.524	6.012	66.6	5.5605	5	311	15.2	395.60	12.43	22.9
7	0.14455	12.5	7.87	0	0.524	6.172	96.1	5.9505	5	311	15.2	396.90	19.15	27.
8	0.21124	12.5	7.87	0	0.524	5.631	100.0	6.0821	5	311	15.2	386.63	29.93	16.8
9	0.17004	12.5	7.87	0	0.524	6.004	85.9	6.5921	5	311	15.2	386.71	17.10	18.9

In [620…

```
'''
Find the total number of records present in Boston_housing.csv.
Use size for df.
'''
# Understand the size of df.
size = df.size

print('Total number of records present in Boston_housing is {} records.'.format(size))
```

```
Total number of records present in Boston_housing is 7084 records.
```

In [621…

```
'''
Find the total number of records present in Boston_housing.csv.
Use shape df.
'''
# Understand the shape of df.
shape = df.shape

print('Total number of records present in Boston_housing is {} records.'.format(shape))
```

Total number of records present in Boston_housing is (506, 14) records.

In [622…

```
'''
This section of code will drop the following columns: CHAS, NOX, B, and LSTAT.
The df.drop method was originally attempted, however specific columns from df ended up
'''
# df_reduced will include the original df columns minus CHAS, NOX, B, and LSTAT.
# The first 10 columns will be shown with head() to verify columns were removed.
# df_reduced = df.drop('CHAS', 'NOX', 'B', 'LSTAT') - could not get drop method to work
# ended up calling out all other column names from df.
df_reduced = df[['CRIM','ZN','INDUS','RM','AGE','DIS', 'RAD', 'TAX','PTRATIO','PRICE']]
df_reduced.head(10)
```

Out[622…

	CRIM	ZN	INDUS	RM	AGE	DIS	RAD	TAX	PTRATIO	PRICE
0	0.00632	18.0	2.31	6.575	65.2	4.0900	1	296	15.3	24.0
1	0.02731	0.0	7.07	6.421	78.9	4.9671	2	242	17.8	21.6
2	0.02729	0.0	7.07	7.185	61.1	4.9671	2	242	17.8	34.7
3	0.03237	0.0	2.18	6.998	45.8	6.0622	3	222	18.7	33.4
4	0.06905	0.0	2.18	7.147	54.2	6.0622	3	222	18.7	36.2
5	0.02985	0.0	2.18	6.430	58.7	6.0622	3	222	18.7	28.7
6	0.08829	12.5	7.87	6.012	66.6	5.5605	5	311	15.2	22.9
7	0.14455	12.5	7.87	6.172	96.1	5.9505	5	311	15.2	27.1
8	0.21124	12.5	7.87	5.631	100.0	6.0821	5	311	15.2	16.5
9	0.17004	12.5	7.87	6.004	85.9	6.5921	5	311	15.2	18.9

In [623…

```
'''
This section of code will check the last 7 entries for df_reduced using the tail() comm
'''
# Use tail(7) to get the last 7 entries of df_reduced.
df_reduced.tail(7)
```

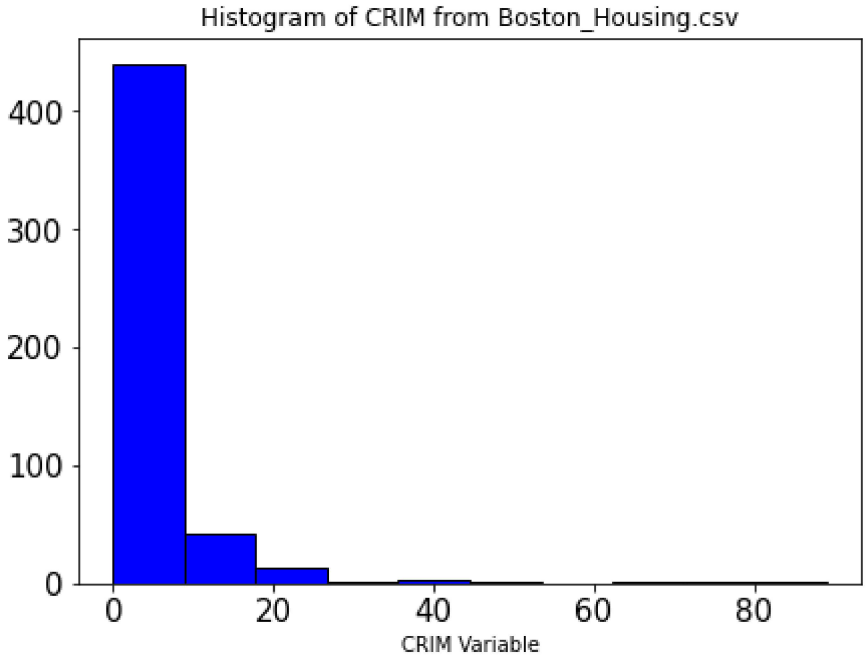
Out[623…

	CRIM	ZN	INDUS	RM	AGE	DIS	RAD	TAX	PTRATIO	PRICE
499	0.17783	0.0	9.69	5.569	73.5	2.3999	6	391	19.2	17.5
500	0.22438	0.0	9.69	6.027	79.7	2.4982	6	391	19.2	16.8
501	0.06263	0.0	11.93	6.593	69.1	2.4786	1	273	21.0	22.4
502	0.04527	0.0	11.93	6.120	76.7	2.2875	1	273	21.0	20.6
503	0.06076	0.0	11.93	6.976	91.0	2.1675	1	273	21.0	23.9
504	0.10959	0.0	11.93	6.794	89.3	2.3889	1	273	21.0	22.0
505	0.04741	0.0	11.93	6.030	80.8	2.5050	1	273	21.0	11.9

In [624…

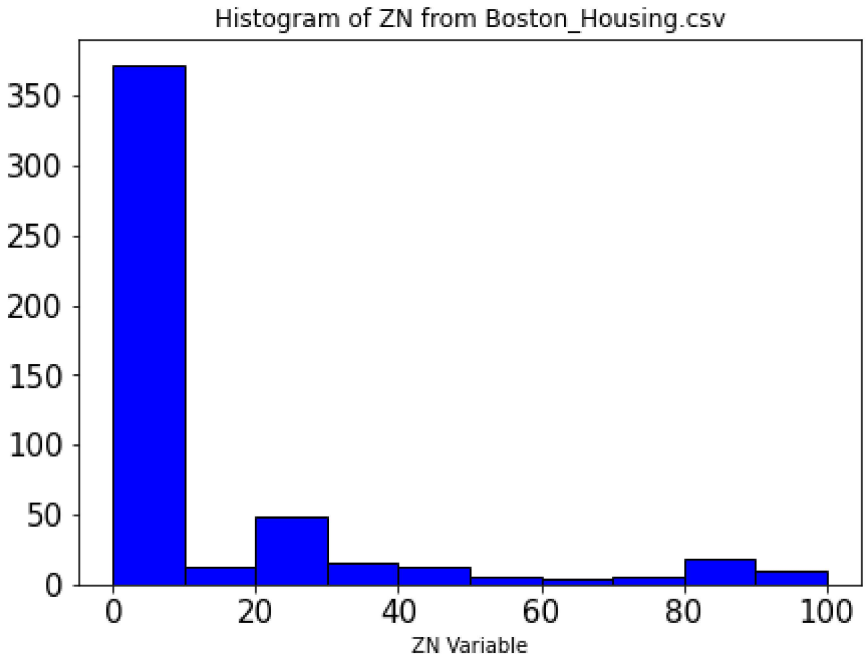
```
'''
Using the plt.hist() function, this section of code will plot each column in a histogra
This is specifically related to Step 6. of Activy 5
```

```
This section will plot column df_reduced['CRIM'].
'''
# Histogram plot of df_reduced column 'CRIM' as shown on page 111 of Data Wrangling wit
plt.figure(figsize=(7,5))
plt.hist(df_reduced['CRIM'], color = 'blue', edgecolor='k')
plt.title("Histogram of CRIM from Boston_Housing.csv")
plt.xlabel("CRIM Variable")
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.show()
```



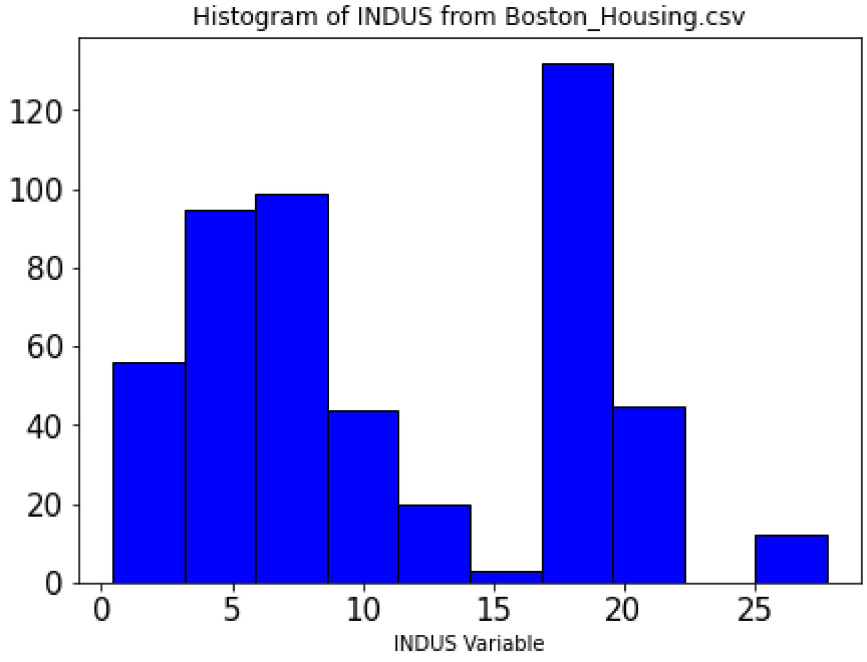
In [625…

```
'''
Using the plt.hist() function, this section of code will plot each column in a histogra
This is specifically related to Step 6. of Activy 5
This section will plot column df_reduced['ZN'].
'''
# Histogram plot of df_reduced column 'ZN' as shown on page 111 of Data Wrangling with
plt.figure(figsize=(7,5))
plt.hist(df_reduced['ZN'], color = 'blue', edgecolor='k')
plt.title("Histogram of ZN from Boston_Housing.csv")
plt.xlabel("ZN Variable")
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.show()
```



In [626…

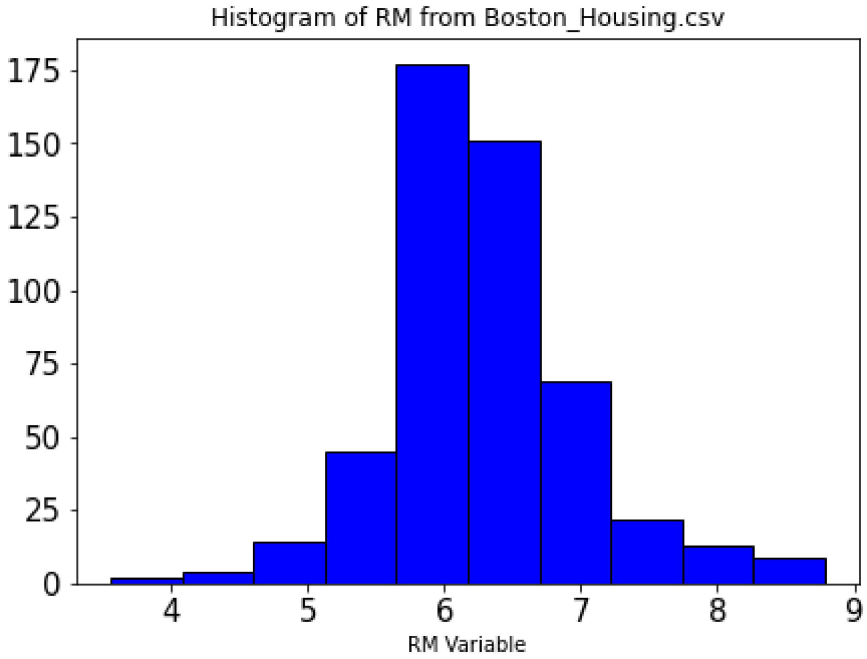
```
'''
Using the plt.hist() function, this section of code will plot each column in a histogra
This is specifically related to Step 6. of Activy 5
This section will plot column df_reduced['INDUS'].
'''
# Histogram plot of df_reduced column 'INDUS' as shown on page 111 of Data Wrangling wi
plt.figure(figsize=(7,5))
plt.hist(df_reduced['INDUS'], color = 'blue', edgecolor='k')
plt.title("Histogram of INDUS from Boston_Housing.csv")
plt.xlabel("INDUS Variable")
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.show()
```



In [627…

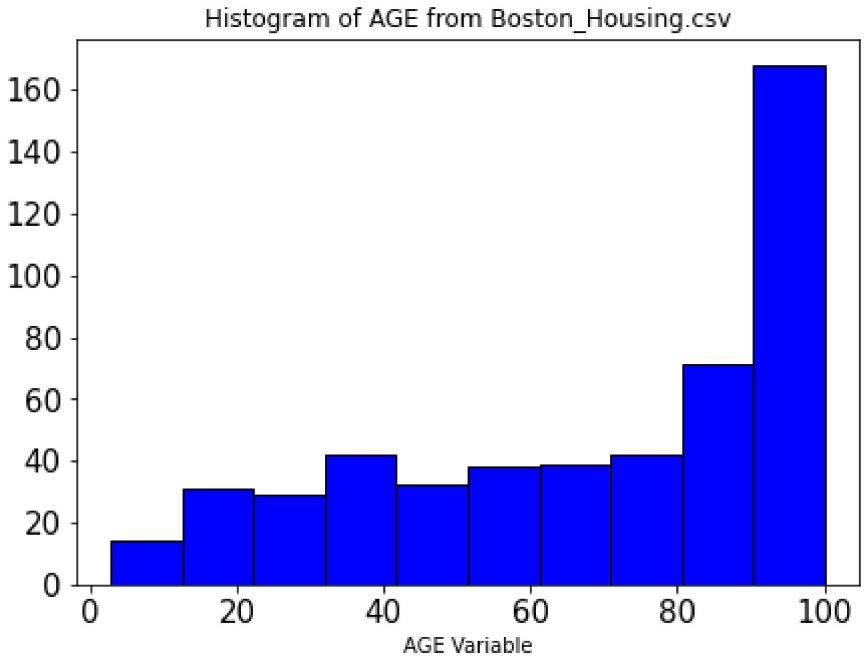
```
'''
```

```
Using the plt.hist() function, this section of code will plot each column in a histogra
This is specifically related to Step 6. of Activy 5
This section will plot column df_reduced['RM'].
'''
# Histogram plot of df_reduced column 'RM' as shown on page 111 of Data Wrangling with
plt.figure(figsize=(7,5))
plt.hist(df_reduced['RM'], color = 'blue', edgecolor='k')
plt.title("Histogram of RM from Boston_Housing.csv")
plt.xlabel("RM Variable")
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.show()
```



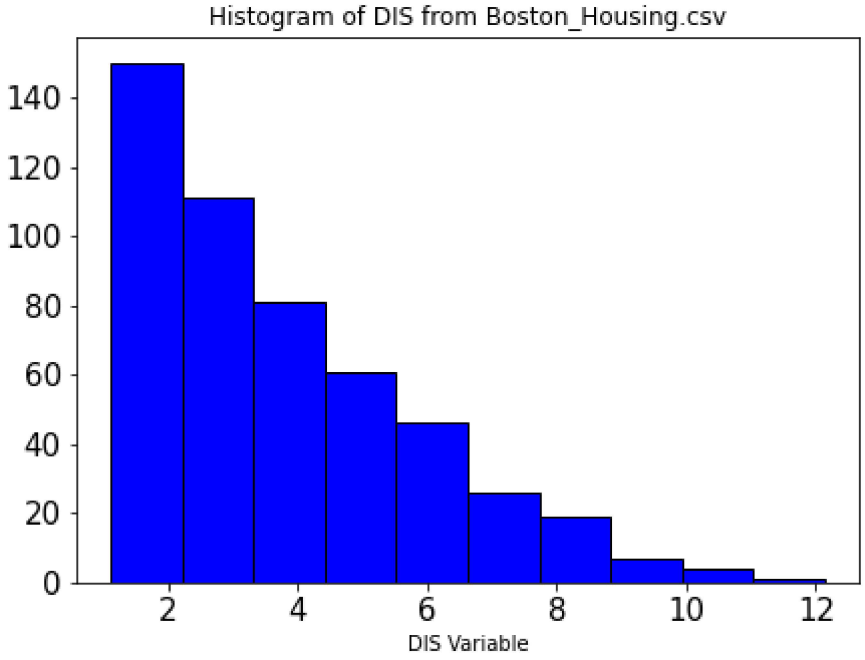
In [628…

```
'''
Using the plt.hist() function, this section of code will plot each column in a histogra
This is specifically related to Step 6. of Activy 5
This section will plot column df_reduced['AGE'].
'''
# Histogram plot of df_reduced column 'AGE' as shown on page 111 of Data Wrangling with
plt.figure(figsize=(7,5))
plt.hist(df_reduced['AGE'], color = 'blue', edgecolor='k')
plt.title("Histogram of AGE from Boston_Housing.csv")
plt.xlabel("AGE Variable")
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.show()
```



In [629…

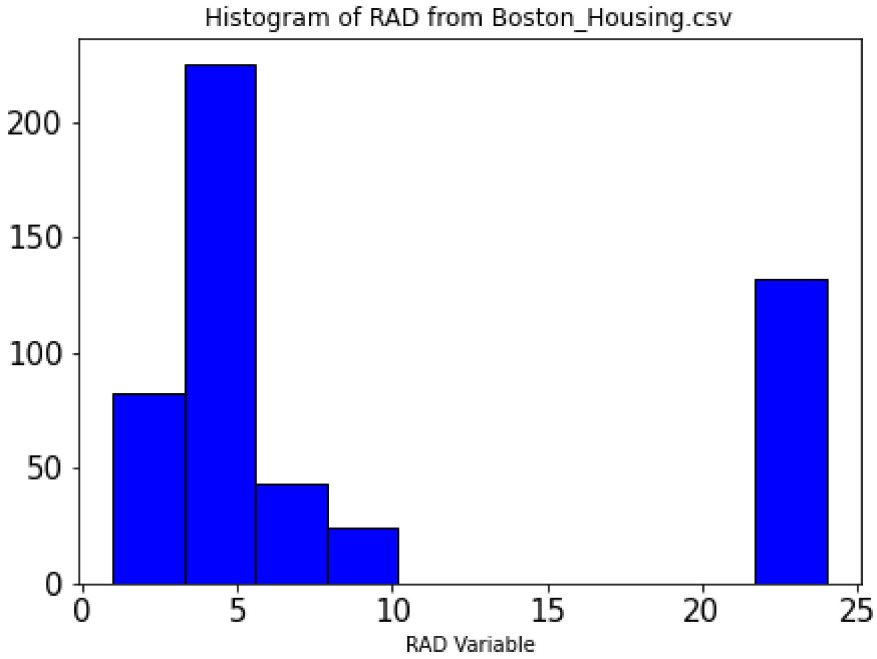
```
'''
Using the plt.hist() function, this section of code will plot each column in a histogra
This is specifically related to Step 6. of Activy 5
This section will plot column df_reduced['DIS'].
'''
# Histogram plot of df_reduced column 'DIS' as shown on page 111 of Data Wrangling with
plt.figure(figsize=(7,5))
plt.hist(df_reduced['DIS'], color = 'blue', edgecolor='k')
plt.title("Histogram of DIS from Boston_Housing.csv")
plt.xlabel("DIS Variable")
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.show()
```



In [630…

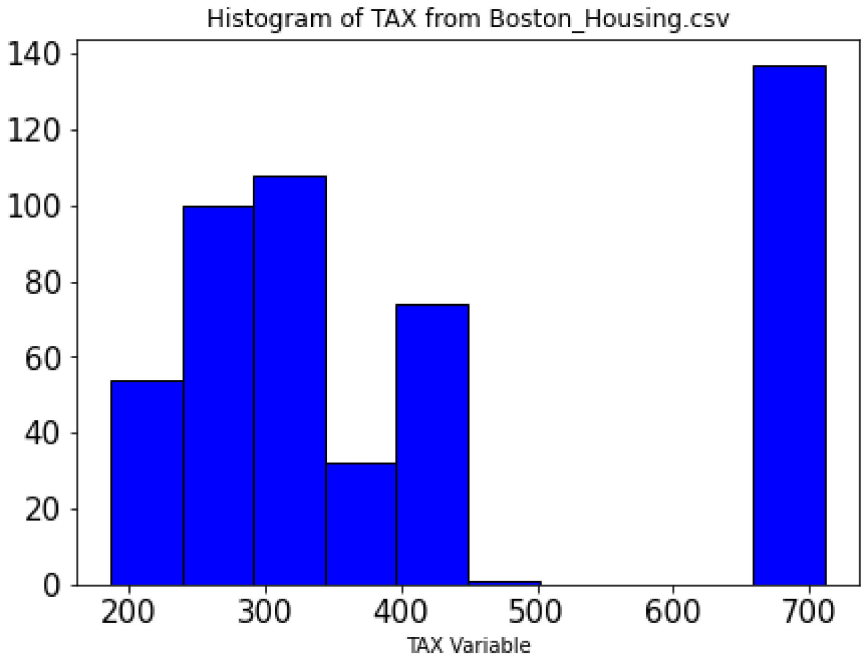
```
'''
```

```
Using the plt.hist() function, this section of code will plot each column in a histogra
This is specifically related to Step 6. of Activy 5
This section will plot column df_reduced['RAD'].
'''
# Histogram plot of df_reduced column 'RAD' as shown on page 111 of Data Wrangling with
plt.figure(figsize=(7,5))
plt.hist(df_reduced['RAD'], color = 'blue', edgecolor='k')
plt.title("Histogram of RAD from Boston_Housing.csv")
plt.xlabel("RAD Variable")
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.show()
```



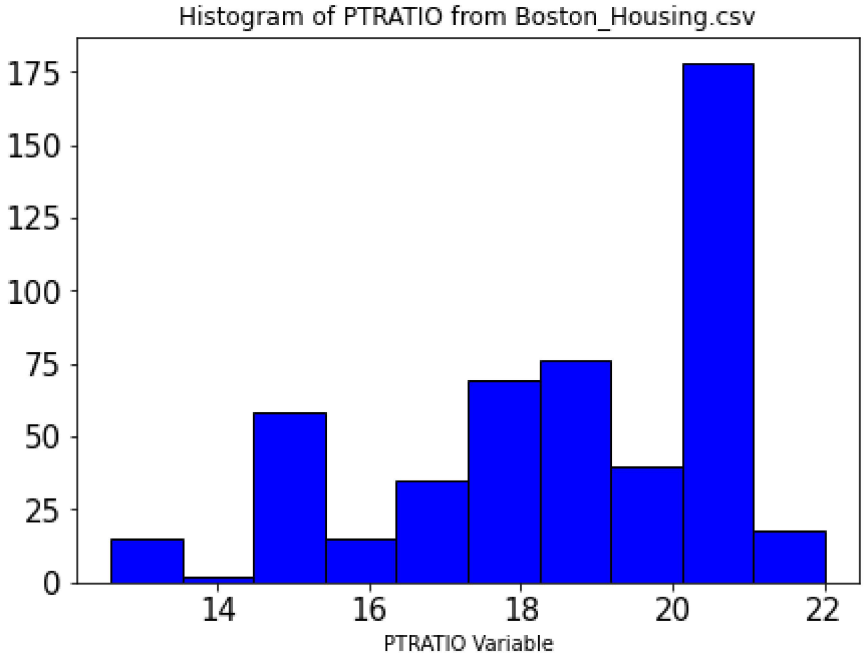
In [631…

```
'''
Using the plt.hist() function, this section of code will plot each column in a histogra
This is specifically related to Step 6. of Activy 5
This section will plot column df_reduced['TAX'].
'''
# Histogram plot of df_reduced column 'TAX' as shown on page 111 of Data Wrangling with
plt.figure(figsize=(7,5))
plt.hist(df_reduced['TAX'], color = 'blue', edgecolor='k')
plt.title("Histogram of TAX from Boston_Housing.csv")
plt.xlabel("TAX Variable")
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.show()
```



In [632…

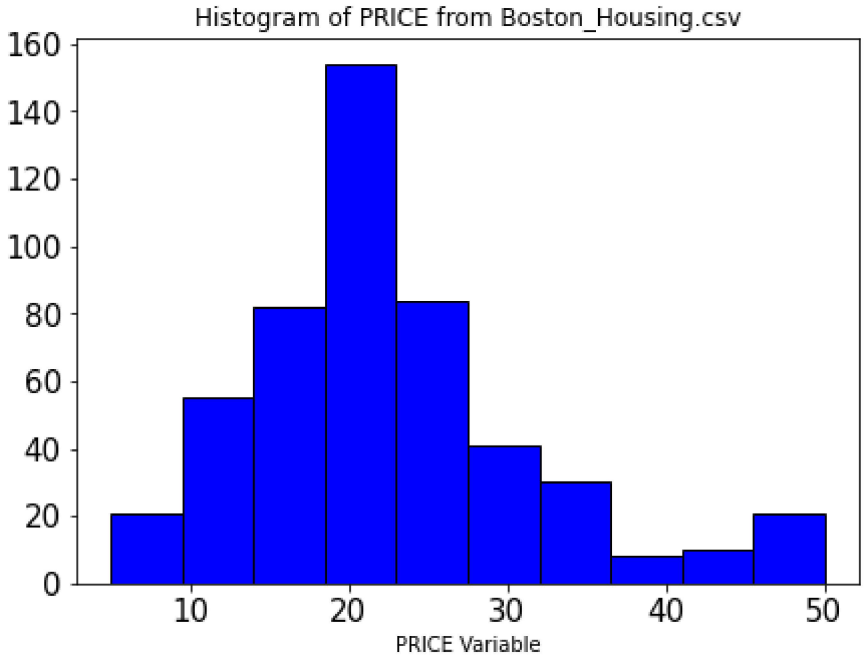
```
'''
Using the plt.hist() function, this section of code will plot each column in a histogra
This is specifically related to Step 6. of Activy 5
This section will plot column df_reduced['PTRATIO'].
'''
# Histogram plot of df_reduced column 'PTRATIO' as shown on page 111 of Data Wrangling
plt.figure(figsize=(7,5))
plt.hist(df_reduced['PTRATIO'], color = 'blue', edgecolor='k')
plt.title("Histogram of PTRATIO from Boston_Housing.csv")
plt.xlabel("PTRATIO Variable")
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.show()
```



In [633…

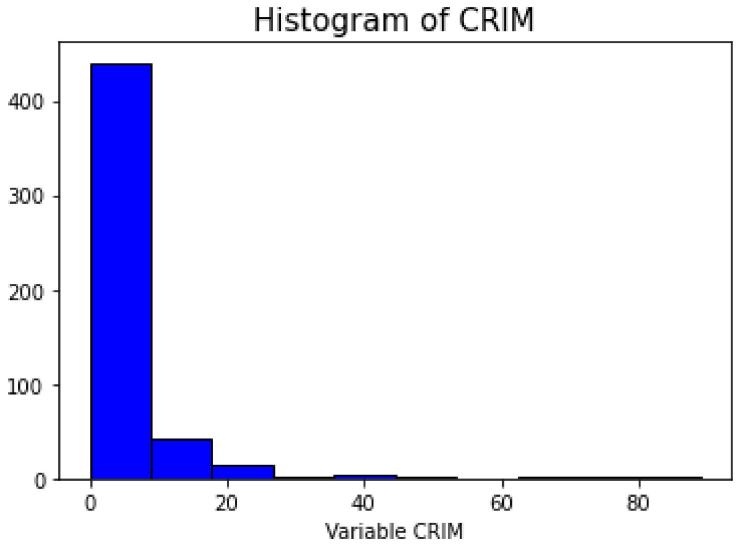
```
'''
```

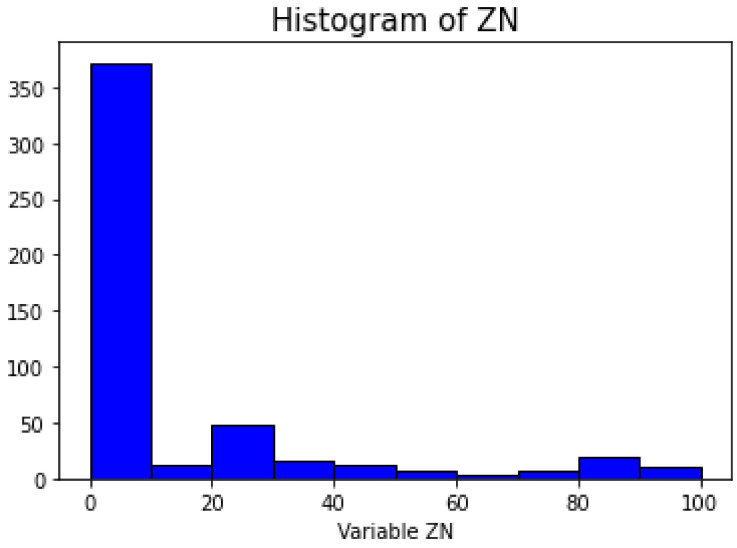
```
Using the plt.hist() function, this section of code will plot each column in a histogra
This is specifically related to Step 6. of Activy 5.
This section will plot column df_reduced['PRICE'].
'''
# Histogram plot of df_reduced column 'PRICE' as shown on page 111 of Data Wrangling wi
plt.figure(figsize=(7,5))
plt.hist(df_reduced['PRICE'], color = 'blue', edgecolor='k')
plt.title("Histogram of PRICE from Boston_Housing.csv")
plt.xlabel("PRICE Variable")
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.show()
```

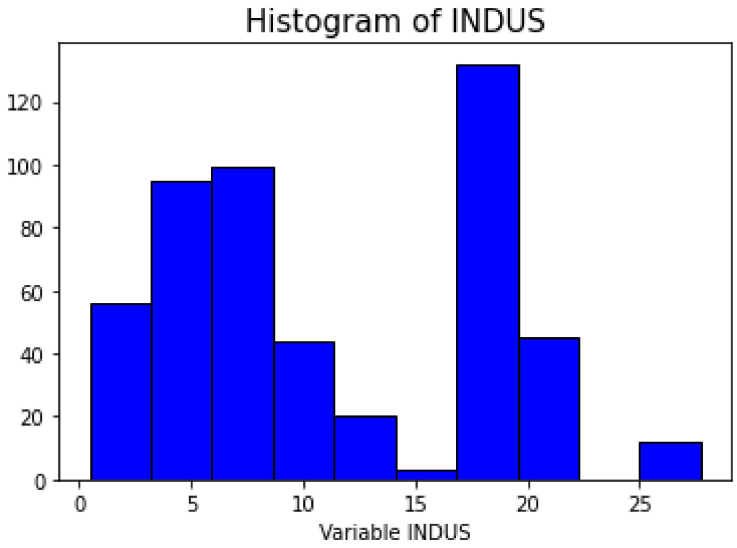


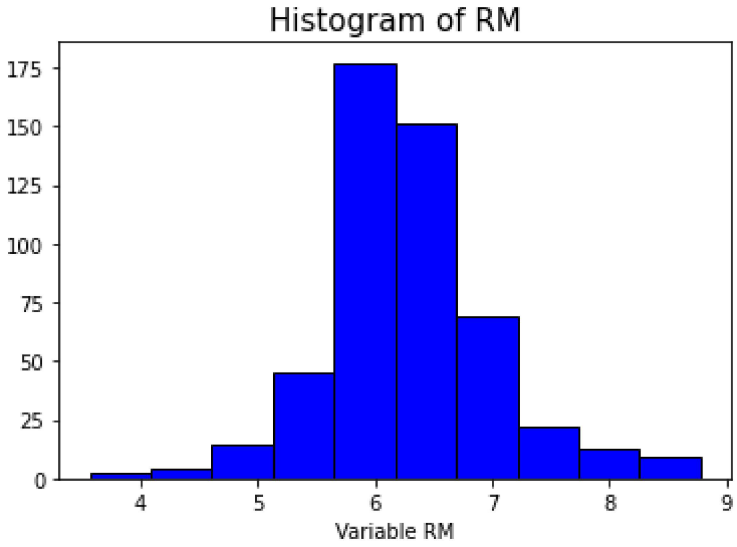
In [634…

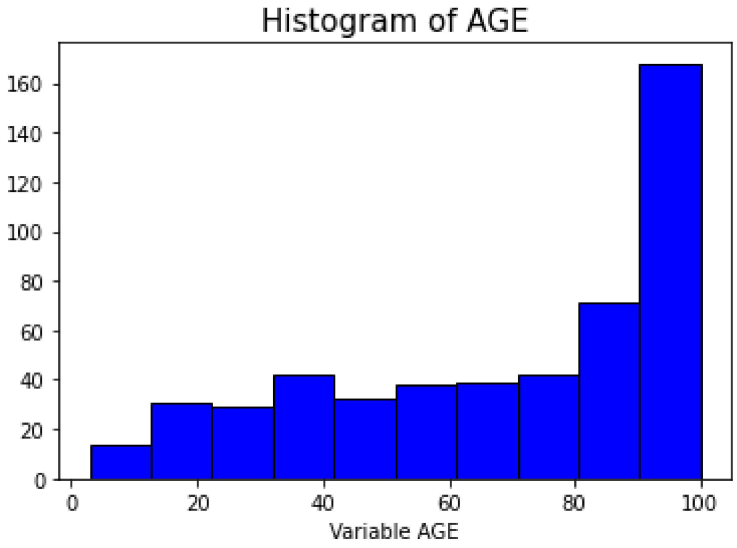
```
'''
Since ploting all the histograms separately is redundant, this section of the code will
all the histograms for the columns at once using the plt.hist() function.
This is specifically related to Step 7. of Activy 5.
'''
# For loop generation of histogram plot across all columns in Boston_housing.csv.
for col in df_reduced.columns:
    # Iterate across columns in df_reduced to plot a histogram of each col.
    plt.hist(df_reduced[col], color = 'blue', edgecolor='k') # plt.hist() function to g
    plt.title("Histogram of {}".format(col), fontsize=15) # Title added to each histogr
    plt.xlabel("Variable " +col, fontsize=10) # Xlabel added to each histogram plot.
    plt.xticks(fontsize=10) # Formatting of x tick marks for each histogram plot.
    plt.yticks(fontsize=10) # Formatting of y tick marks for each histogram plot
    plt.show() # plt.show() will provide the output histogram plots.
```

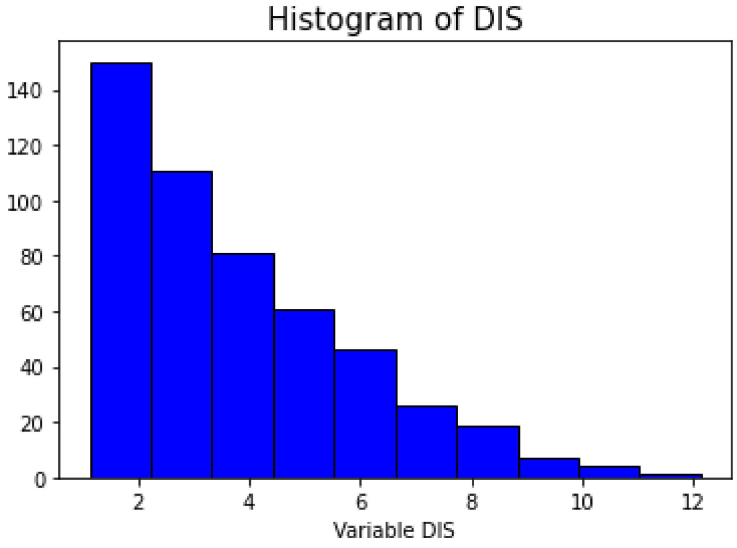

Histogram of CRIM
400
300
200
100
0
0
20
40
60
80
Variable CRIM

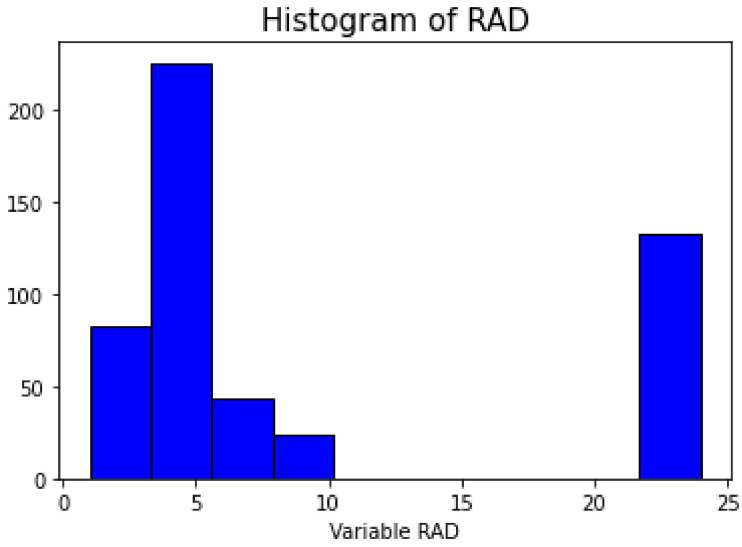

Histogram of ZN
350
300
250
200
150
100
50
0
0
20
40
60
80
100
Variable ZN

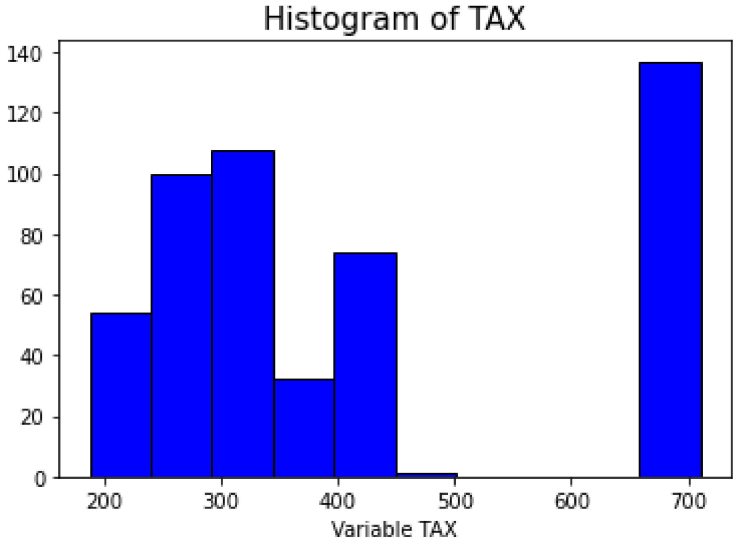

Histogram of INDUS
120
100
80
60
40
20
0
0
5
10
15
20
25
Variable INDUS

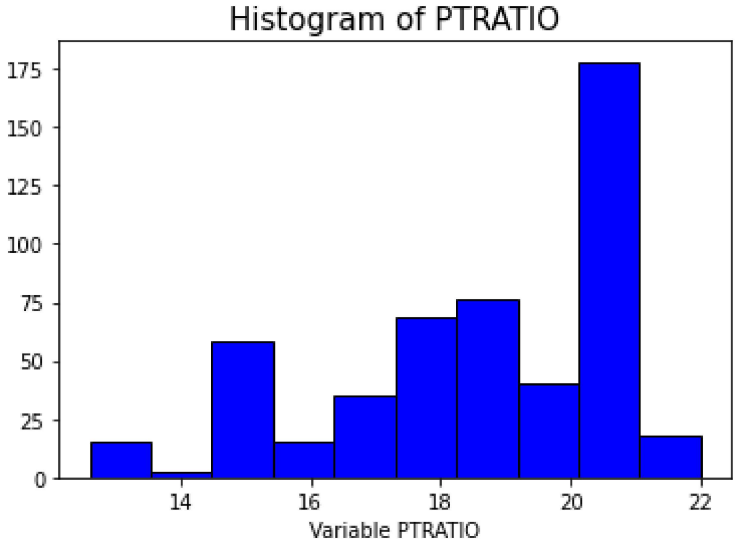

Histogram of RM
175
150
125
100
75
50
25
0
4
5
6
7
8
9
Variable RM

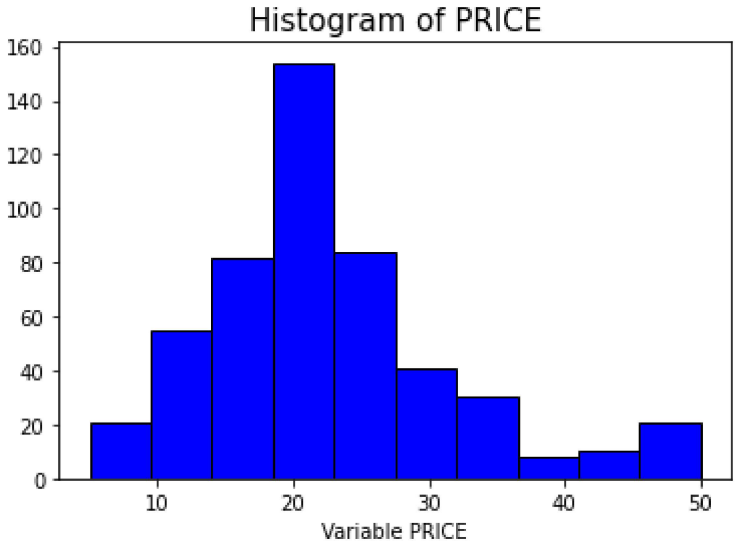

Histogram of AGE
160
140
120
100
80
60
40
20
0
0
20
40
60
80
100
Variable AGE


Histogram of DIS
140
120
100
80
60
40
20
0
2
4
6
8
10
12
Variable DIS


Histogram of RAD
200
150
100
50
0
0
5
10
15
20
25
Variable RAD

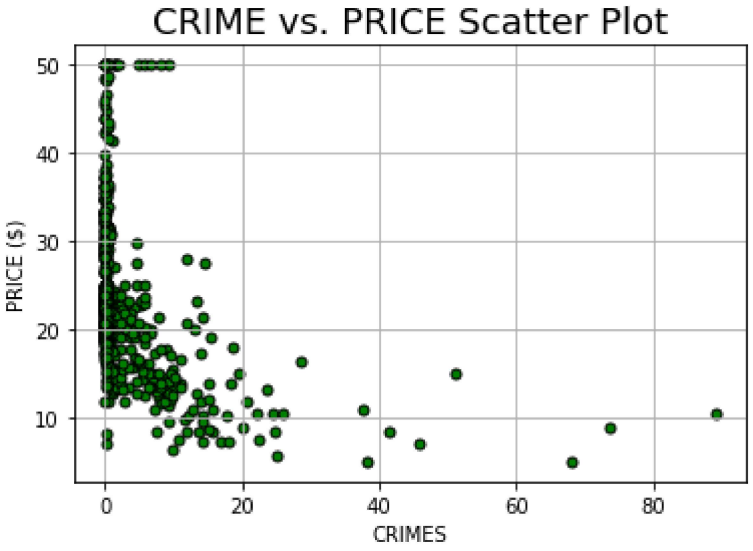

Histogram of TAX
140
120
100
80
60
40
20
0
200
300
400
500
600
700
Variable TAX


Histogram of PTRATIO
175
150
125
100
75
50
25
0
14
16
18
20
22
Variable PTRATIO



In [635…

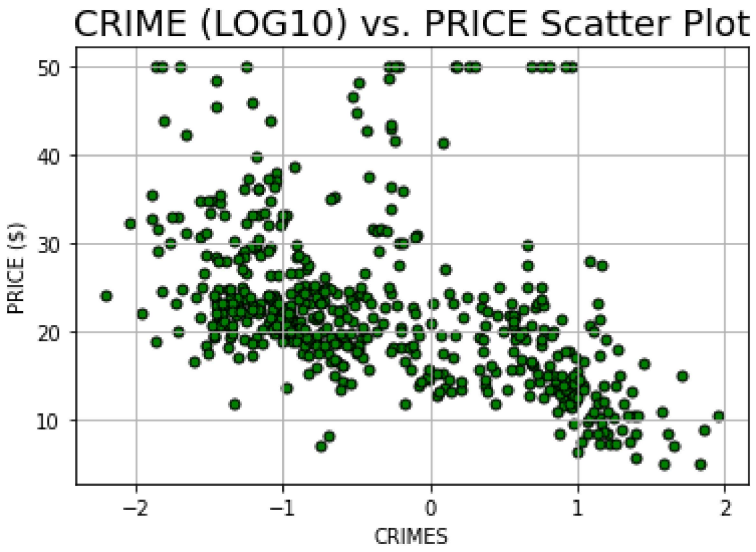
```
'''
This section of code will create a scatter plot showing the relationship between crime
The .plot.scatter() command will be utilized as shown on page 115 from Data Wrangling w
'''
# Use the .plot.scatter() with the df_reduced data frame and add plot details.
df_reduced.plot.scatter('CRIM','PRICE', s=25, c='green', edgecolor='k')
plt.grid(True) # Places a grid onto the plot.
plt.title("CRIME vs. PRICE Scatter Plot", fontsize=18)
plt.xlabel("CRIMES")
plt.ylabel("PRICE ($)")
plt.show()
```



In [636…

```
'''
This section of code will create a similar scatter plot (as seen above) showing the rel
This time the log10() will be applied to the CRIME Values and replotted.
The plt.scatter() command will be utilized to store a plot as log10_plot.
'''
# Use the plt.scatter() with the df_reduced data frame and add plot details.
log10_plot = plt.scatter(np.log10(df_reduced['CRIM']),df_reduced['PRICE'], s=25, c='gre
plt.grid(True) # Places a grid onto the plot.
plt.title("CRIME (LOG10) vs. PRICE Scatter Plot", fontsize=18)
plt.xlabel("CRIMES")
```

```
plt.ylabel("PRICE ($)")
plt.show(log10_plot)
```



In [637…

```
'''
Calculate the mean rooms per dwelling. Variable of interest for rooms per dwelling is '
'''
# Use the mean() command to calculate the mean rooms per dwelling.
# Round the value to the nearest hundredth with round().
RM_mean = df_reduced['RM'].mean()
print("Mean rooms per dwelling is:", round(RM_mean,2))
```

```
Mean rooms per dwelling is: 6.28
```

In [638…

```
'''
Calculate the median age. Variable of interest for age is 'AGE'.
'''
# Use the median() command to calculate median age.
age_median = df_reduced['AGE'].median()
print("Median age is:", age_median)
```

```
Median age is: 77.5
```

In [639…

```
'''
Calculate the mean distances to five Boston employment centers. Variable of interest fo
'''
# Use the mean() command to calculate the mean distance.

distance_mean = df_reduced['DIS'].mean()
print("Mean distance is:", round(distance_mean,3))
```

```
Mean distance is: 3.795
```

In [640…

```
'''
Calculate the percentages of houses with a lower price (<$20,000).
This section will show the quantiles with the describe() function to get an idea of whe
'''
# Utilize the describe() function for 'PRICE' which shows the value should reside betwe
df_reduced['PRICE'].describe()
```

Out[640…
```
count    506.000000
mean      22.532806
std        9.197104
min        5.000000
25%       17.025000
50%       21.200000
75%       25.000000
max       50.000000
Name: PRICE, dtype: float64
```

In [641…
```
'''
This section will calculate the percentage by finding the count for the number below $2
and dividing by the total count of values in df_reduced['PRICE'].
'''
# Store the 'PRICE' column under the variable price_total.
price_total = df_reduced['PRICE']

# Identify the values under $20,000 and store under lower_20 variable.
lower_20 = price_total[price_total < 20]

# Calculate the percentage of houses that fall under this $20,000 mark and round to the
# Take the count() for each of the price_total and lower_20 in the calculation.
percentage_house = ((lower_20.count()) / (price_total.count()))*100
print("Percentage of houses with lower price than $20,000: {}%".format(round(percentage
```

```
Percentage of houses with lower price than $20,000: 41.5%
```

2. Data Wrangling with Python: Activity 6, page 171

In [573…
```
'''
Load the essential libraries required for reading the adult income data set.
Additional libraries will be added later in the code if required.
'''
# NumPy was already imported as np from Activity 1. Code is included below for referenc
# import numpy as np

# Pandas was already imported as pd from Activity 1. Code is included below for referen
# import pandas as pd
# from pandas import Series, DataFrame

# matplotlib.pyplot was laready imported as plt from Activity 1. Code is included below
# import matplotlib.pyplot as plt
```

Out[573…
```
'\nLoad the essential libraries required for reading the adult income data set.\nAdditio
nal libraries will be added later in the code if required.\n'
```

In [574…
```
'''
Locate the adult_income_data.csv dataset file provided with the following
link: https://github.com/TrainingByPackt/Data-Wrangling-with-Python/tree/master/Chapter
Store the file in the same working repository as this assignment.
If unable to store in same working repository, then keep track of the directory path an
The read_csv() function from pandas will be utilized to read the data.
'''
# Store the adult_income_data.csv data as df_income.
df_income = pd.read_csv('adult_income_data.csv')
```

```
# Read the first 10 rows of df_income
df_income.head(10)
```

Out[574…

	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	Male	2174	0	40	United-States
0	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	Male	0	0	13	United-States
1	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	Male	0	0	40	United-States
2	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Male	0	0	40	United-States
3	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Female	0	0	40	Cuba
4	37	Private	284582	Masters	14	Married-civ-spouse	Exec-managerial	Wife	Female	0	0	40	United-States
5	49	Private	160187	9th	5	Married-spouse-absent	Other-service	Not-in-family	Female	0	0	16	Jamaica
6	52	Self-emp-not-inc	209642	HS-grad	9	Married-civ-spouse	Exec-managerial	Husband	Male	0	0	45	United-States
7	31	Private	45781	Masters	14	Never-married	Prof-specialty	Not-in-family	Female	14084	0	50	United-States
8	42	Private	159449	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	Male	5178	0	40	United-States
9	37	Private	280464	Some-college	10	Married-civ-spouse	Exec-managerial	Husband	Male	0	0	80	United-States

In [575…

```
'''
First thing that stands out for the data is the lack of column names. Locate the adult_
provided with the same link:
https://github.com/TrainingByPackt/Data-Wrangling-with-Python/tree/master/Chapter04/Act
Store the file in the same working repository as this assignment.
This section of code is specifically intended to understand what is in the .txt file.
'''
# Read the .txt file using "with open" syntax and r mode.
# Use a for loop to read each line in adult_income_names.txt.
with open("adult_income_names.txt", "r") as file:
    for line in file:
        print(line)
```

```
age: continuous.
```

workclass: Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked.

fnlwgt: continuous.

education: Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool.

education-num: continuous.

marital-status: Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse.

occupation: Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces.

relationship: Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.

sex: Female, Male.

capital-gain: continuous.

capital-loss: continuous.

hours-per-week: continuous.

native-country: United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holand-Netherlands.

In [576…

```
'''
```

```
Create a list of column names labeled column_names from the .txt file.
Utilize a for loop to iterate over the first words and store in a list.
'''
# Create an empty list to later store the column names.
column_names = []

# Open the .txt file.
with open("adult_income_names.txt", "r") as file:
    for line in file:
        # Create a for loop to iterate over each line and collect the first word to sto
        file.readline()
        name = line.split(":")[0]
        column_names.append(name)
    print(column_names)
```

```
['age', 'workclass', 'fnlwgt', 'education', 'education-num', 'marital-status', 'occupati
on', 'relationship', 'sex', 'capital-gain', 'capital-loss', 'hours-per-week', 'native-co
untry']
```

In [577…

```
'''
Add income as a column name to the list that was just generated.
If income is not added, then the column names are not aligned correctly.
'''
column_names.append('income')
print(column_names)
```

```
['age', 'workclass', 'fnlwgt', 'education', 'education-num', 'marital-status', 'occupati
on', 'relationship', 'sex', 'capital-gain', 'capital-loss', 'hours-per-week', 'native-co
untry', 'income']
```

In [578…

```
'''
Re-read the adult_income_data.csv file with the columns_name list as the names argument
'''
# Store the adult_income_data.csv data as df_income with column_names.
df_income = pd.read_csv('adult_income_data.csv', names = column_names)

# Read the first 20 rows of df_income.
df_income.head(20)
```

Out[578…

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	sex	capital-gain
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	Male	2174
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	Male	0
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	Male	0
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Male	0

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	sex	capital-gain
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Female	0
5	37	Private	284582	Masters	14	Married-civ-spouse	Exec-managerial	Wife	Female	0
6	49	Private	160187	9th	5	Married-spouse-absent	Other-service	Not-in-family	Female	0
7	52	Self-emp-not-inc	209642	HS-grad	9	Married-civ-spouse	Exec-managerial	Husband	Male	0
8	31	Private	45781	Masters	14	Never-married	Prof-specialty	Not-in-family	Female	14084
9	42	Private	159449	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	Male	5178
10	37	Private	280464	Some-college	10	Married-civ-spouse	Exec-managerial	Husband	Male	0
11	30	State-gov	141297	Bachelors	13	Married-civ-spouse	Prof-specialty	Husband	Male	0
12	23	Private	122272	Bachelors	13	Never-married	Adm-clerical	Own-child	Female	0
13	32	Private	205019	Assoc-acdm	12	Never-married	Sales	Not-in-family	Male	0
14	40	Private	121772	Assoc-voc	11	Married-civ-spouse	Craft-repair	Husband	Male	0
15	34	Private	245487	7th-8th	4	Married-civ-spouse	Transport-moving	Husband	Male	0
16	25	Self-emp-not-inc	176756	HS-grad	9	Never-married	Farming-fishing	Own-child	Male	0
17	32	Private	186824	HS-grad	9	Never-married	Machine-op-inspct	Unmarried	Male	0
18	38	Private	28887	11th	7	Married-civ-spouse	Sales	Husband	Male	0
19	43	Self-emp-not-inc	292175	Masters	14	Divorced	Exec-managerial	Unmarried	Female	0

In [579…
```
'''
Check each column in the df_income for any NaN values.
'''
df_income.isna().any()
```

Out[579…
```
age               False
workclass         False
fnlwgt            False
education         False
education-num     False
marital-status    False
occupation        False
relationship      False
sex               False
capital-gain      False
capital-loss      False
hours-per-week    False
native-country    False
income            False
dtype: bool
```

In [580…
```
'''
Check each column in the df_income for any null values.
'''
df_income.isnull().any()
```

Out[580…
```
age               False
workclass         False
fnlwgt            False
education         False
education-num     False
marital-status    False
occupation        False
relationship      False
sex               False
capital-gain      False
capital-loss      False
hours-per-week    False
native-country    False
income            False
dtype: bool
```

In [581…
```
'''
After further review, there are "?" that are present in the df_income['native-country',
I will remove the question mark and make these entries appear as null values.
'''
# df_income['native-country'].unique() was used to find the unique " ?" value. (leading
# Originally, the replace() command was not working because "?" vs " ?".
df_income = df_income.replace(" ?", "")

# Once the ? was removed, use the code below to specify NaN values in the empty cells.
df_income = df_income.replace(r'^\s*$', np.nan, regex=True)
```

In [582…
```
'''
Check the columns with known "?" to verify they were removed and the NaN (Not a Number)
This is more of a sanity check and is not required since the isnull().any will still be
'''
```

```
# df_income['native-country'][:30]

df_income['occupation'][:30]
```

Out[582…

```
0           Adm-clerical
1         Exec-managerial
2       Handlers-cleaners
3       Handlers-cleaners
4          Prof-specialty
5         Exec-managerial
6           Other-service
7         Exec-managerial
8          Prof-specialty
9         Exec-managerial
10        Exec-managerial
11         Prof-specialty
12           Adm-clerical
13                  Sales
14           Craft-repair
15       Transport-moving
16        Farming-fishing
17      Machine-op-inspct
18                  Sales
19        Exec-managerial
20         Prof-specialty
21          Other-service
22        Farming-fishing
23       Transport-moving
24           Tech-support
25           Tech-support
26           Craft-repair
27                    NaN
28        Exec-managerial
29           Craft-repair
Name: occupation, dtype: object
```

In [583…

```
'''
Recheck df_income to show the "?" was removed as stated above.
Confirmed with the output.
'''
df_income.head(20)
```

Out[583…

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	sex	capital-gain
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	Male	2174
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	Male	0
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	Male	0
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Male	0

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	sex	capital-gain
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Female	0
5	37	Private	284582	Masters	14	Married-civ-spouse	Exec-managerial	Wife	Female	0
6	49	Private	160187	9th	5	Married-spouse-absent	Other-service	Not-in-family	Female	0
7	52	Self-emp-not-inc	209642	HS-grad	9	Married-civ-spouse	Exec-managerial	Husband	Male	0
8	31	Private	45781	Masters	14	Never-married	Prof-specialty	Not-in-family	Female	14084
9	42	Private	159449	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	Male	5178
10	37	Private	280464	Some-college	10	Married-civ-spouse	Exec-managerial	Husband	Male	0
11	30	State-gov	141297	Bachelors	13	Married-civ-spouse	Prof-specialty	Husband	Male	0
12	23	Private	122272	Bachelors	13	Never-married	Adm-clerical	Own-child	Female	0
13	32	Private	205019	Assoc-acdm	12	Never-married	Sales	Not-in-family	Male	0
14	40	Private	121772	Assoc-voc	11	Married-civ-spouse	Craft-repair	Husband	Male	0
15	34	Private	245487	7th-8th	4	Married-civ-spouse	Transport-moving	Husband	Male	0
16	25	Self-emp-not-inc	176756	HS-grad	9	Never-married	Farming-fishing	Own-child	Male	0
17	32	Private	186824	HS-grad	9	Never-married	Machine-op-inspct	Unmarried	Male	0
18	38	Private	28887	11th	7	Married-civ-spouse	Sales	Husband	Male	0
19	43	Self-emp-not-inc	292175	Masters	14	Divorced	Exec-managerial	Unmarried	Female	0

In [584…
```
'''
Utilize the describe() method to understand the initial data set.
'''
df_income.describe()
```

Out[584…

	age	fnlwgt	education-num	capital-gain	capital-loss	hours-per-week
count	32561.000000	3.256100e+04	32561.000000	32561.000000	32561.000000	32561.000000
mean	38.581647	1.897784e+05	10.080679	1077.648844	87.303830	40.437456
std	13.640433	1.055500e+05	2.572720	7385.292085	402.960219	12.347429
min	17.000000	1.228500e+04	1.000000	0.000000	0.000000	1.000000
25%	28.000000	1.178270e+05	9.000000	0.000000	0.000000	40.000000
50%	37.000000	1.783560e+05	10.000000	0.000000	0.000000	40.000000
75%	48.000000	2.370510e+05	12.000000	0.000000	0.000000	45.000000
max	90.000000	1.484705e+06	16.000000	99999.000000	4356.000000	99.000000

In [585…
```
'''
Recheck each column in the df_income for any null values.
Turns out workclass, occupation, and native-country have null values that were "?".
'''
df_income.isnull().any()
```

Out[585…
```
age              False
workclass         True
fnlwgt           False
education        False
education-num    False
marital-status   False
occupation        True
relationship     False
sex              False
capital-gain     False
capital-loss     False
hours-per-week   False
native-country    True
income           False
dtype: bool
```

In [586…
```
'''
This section of code will find the number of null values in the three columns identifie
The columns with missing values have been identified along with the number of occurence
'''
df_income.isnull().sum()
```

Out[586…
```
age                 0
workclass        1836
fnlwgt              0
education           0
education-num       0
marital-status      0
occupation       1843
relationship        0
```

```
sex                0
capital-gain       0
capital-loss       0
hours-per-week     0
native-country   583
income             0
dtype: int64
```

In [587…

```
'''
Lastly, use the .isnull across the df_income DataFrame to identify each location as Tru
This wraps up the task to identify or find the missing values.
'''
df_income.isnull()
```

Out[587…

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	sex	capita gai
0	False	False	False	False	False	False	False	False	False	Fals
1	False	False	False	False	False	False	False	False	False	Fals
2	False	False	False	False	False	False	False	False	False	Fals
3	False	False	False	False	False	False	False	False	False	Fals
4	False	False	False	False	False	False	False	False	False	Fals
…	…	…	…	…	…	…	…	…	…	
32556	False	False	False	False	False	False	False	False	False	Fals
32557	False	False	False	False	False	False	False	False	False	Fals
32558	False	False	False	False	False	False	False	False	False	Fals
32559	False	False	False	False	False	False	False	False	False	Fals
32560	False	False	False	False	False	False	False	False	False	Fals

32561 rows × 14 columns

In [588…

```
'''
Per the assignment, create a subset DataFrame to include age, education, and occupation
'''
# This code will take all the rows under the three specified columns and store in df_su
df_subset = df_income.loc[:,['age','education','occupation']]
df_subset.head(10)
```

Out[588…

	age	education	occupation
0	39	Bachelors	Adm-clerical
1	50	Bachelors	Exec-managerial
2	38	HS-grad	Handlers-cleaners
3	53	11th	Handlers-cleaners

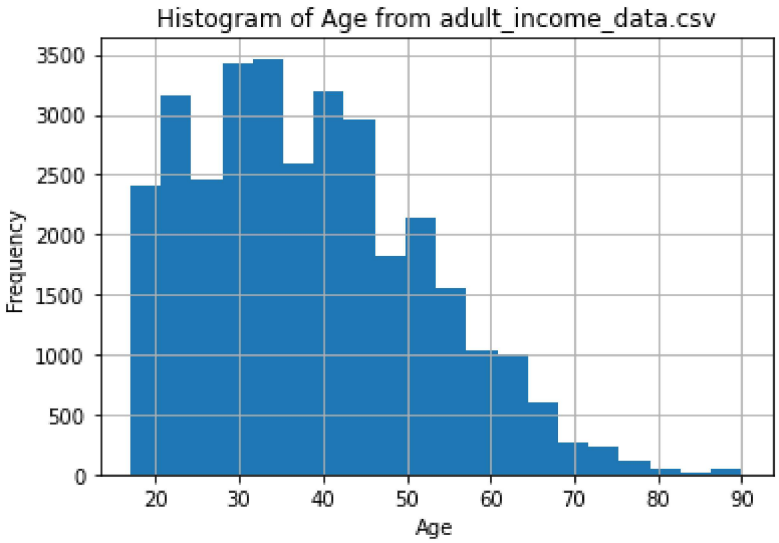
	age	education	occupation
4	28	Bachelors	Prof-specialty
5	37	Masters	Exec-managerial
6	49	9th	Other-service
7	52	HS-grad	Exec-managerial
8	31	Masters	Prof-specialty
9	42	Bachelors	Exec-managerial

In [589…

```
'''
Per the assignment, a histogram will be plotted with a bin size of 20.
'''
# plt.hist() will be used to create the plot
plt.hist(df_subset['age'], bins=20)
plt.grid(True) # Places a grid onto the plot.
plt.title("Histogram of Age from adult_income_data.csv", fontsize=12)
plt.xlabel("Age")
plt.ylabel("Frequency")
plt.show
```

Out[589…

```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [590…

```
'''
Strip the spaces at the beginning and ending of the subset DataFrame string columns.
Initially create a function that will perform the repetitive task.
'''
# Utilize the .strip() function to remove leading and trailing whitespaces.
def remove_whitespace(string):
    # Input argument will be a string and the return will be the string stripped of whi
    return string.strip()
```

In [591…

```
'''
Identify which columns are strings from the df_subset DataFrame.
'''
```

```
data_types = df_subset.dtypes
data_types
```

Out[591…
```
age            int64
education     object
occupation    object
dtype: object
```

In [592…
```
'''
Use the remove_whitespace() function to perform on the string columns from df_subset.
Education and occupation columns contain strings based on the return data_types and hea
Each string column will have the apply() method performed with the newly created.
'''
# Use apply() in conjunction with remove_whitespace() on education column.
# Create a new column called education_clean
df_subset['education_clean'] = df_subset['education'].apply(remove_whitespace)

# Understand the length of each string output using  apply() for two new columns.
# df_subset['education'] = df_subset['education'].apply(len) - Do not run, this code wa
# df_subset['education_clean_length'] = df_subset['education_clean'].apply(len) - Do no

# Move the cleaned string values over the original education column.
df_subset['education'] = df_subset['education_clean']

# Drop the "education_clean" column that was originally created to apply(remove_whitesp
df_subset.drop(labels = ['education_clean'], axis=1, inplace=True)

df_subset.head(5)
```

Out[592…

	age	education	occupation
0	39	Bachelors	Adm-clerical
1	50	Bachelors	Exec-managerial
2	38	HS-grad	Handlers-cleaners
3	53	11th	Handlers-cleaners
4	28	Bachelors	Prof-specialty

In [593…
```
'''
Use the remove_whitespace() function to perform on the string columns from df_subset.
Education and occupation columns contain strings based on the return data_types and hea
Each string column will have the apply() method performed with the newly created.
'''
# Fill the NaN values in the column with generic NoData string.
df_subset['occupation'].fillna('NoData', inplace=True)

# Use apply() in conjunction with remove_whitespace() on occupation column.
# Create a new column called occupation_clean
df_subset['occupation_clean'] = df_subset['occupation'].apply(remove_whitespace)

# Understand the length of each string output using  apply() for two new columns.
# df_subset['occupation'] = df_subset['occupation'].apply(len) - Do not run, this code
# df_subset['occupation'] = df_subset['occupation_clean'].apply(len) - Do not run, this

# Move the cleaned string values over the original education column.
```

```
df_subset['occupation'] = df_subset['occupation_clean']

# Drop the "occupation_clean" column that was originally created to apply(remove_whites,
df_subset.drop(labels = ['occupation_clean'], axis=1, inplace=True)

df_subset.head(10)
```

Out[593…

	age	education	occupation
0	39	Bachelors	Adm-clerical
1	50	Bachelors	Exec-managerial
2	38	HS-grad	Handlers-cleaners
3	53	11th	Handlers-cleaners
4	28	Bachelors	Prof-specialty
5	37	Masters	Exec-managerial
6	49	9th	Other-service
7	52	HS-grad	Exec-managerial
8	31	Masters	Prof-specialty
9	42	Bachelors	Exec-managerial

In [594…

```
'''
Utilize filtering on df_subset to find the data for ages between 30 and 50.
Store the filtered data under a new DataFrame labeled df_filter.
'''
# Use the & condition to filter for data that meets both above 30 and lower than 50 req
df_filter = df_subset[(df_subset['age']>=30) & (df_subset['age']<=50)]
df_filter.head(10)
```

Out[594…

	age	education	occupation
0	39	Bachelors	Adm-clerical
1	50	Bachelors	Exec-managerial
2	38	HS-grad	Handlers-cleaners
5	37	Masters	Exec-managerial
6	49	9th	Other-service
8	31	Masters	Prof-specialty
9	42	Bachelors	Exec-managerial
10	37	Some-college	Exec-managerial
11	30	Bachelors	Prof-specialty
13	32	Assoc-acdm	Sales

In [595…

```
'''
count() will be used to find the number of people between 30 and 50 for df_filter.
```

```
'''

print('Number of people between ages 30 & 50:', df_filter['age'].count())
```

```
Number of people between ages 30 & 50: 16390
```

In [596…

```
'''
Per the assignment, group the records based on education.
Utilize describe() to find how the mean age is distributed.
'''
# Use the goupby() function and store in a new DataFrame called df_group.
df_group = df_subset.groupby(['education'])

# Use describe() to find out how the mean age is distributed.
df_group.describe()['age']
```

Out[596…

education	count	mean	std	min	25%	50%	75%	max
10th	933.0	37.429796	16.720713	17.0	22.00	34.0	52.0	90.0
11th	1175.0	32.355745	15.545485	17.0	18.00	28.0	43.0	90.0
12th	433.0	32.000000	14.334625	17.0	19.00	28.0	41.0	79.0
1st-4th	168.0	46.142857	15.615625	19.0	33.00	46.0	57.0	90.0
5th-6th	333.0	42.885886	15.557285	17.0	29.00	42.0	54.0	84.0
7th-8th	646.0	48.445820	16.092350	17.0	34.25	50.0	61.0	90.0
9th	514.0	41.060311	15.946862	17.0	28.00	39.0	54.0	90.0
Assoc-acdm	1067.0	37.381443	11.095177	19.0	29.00	36.0	44.0	90.0
Assoc-voc	1382.0	38.553546	11.631300	19.0	30.00	37.0	46.0	84.0
Bachelors	5355.0	38.904949	11.912210	19.0	29.00	37.0	46.0	90.0
Doctorate	413.0	47.702179	11.784716	24.0	39.00	47.0	55.0	80.0
HS-grad	10501.0	38.974479	13.541524	17.0	28.00	37.0	48.0	90.0
Masters	1723.0	44.049913	11.068935	18.0	36.00	43.0	51.0	90.0
Preschool	51.0	42.764706	15.126914	19.0	31.00	41.0	53.5	75.0
Prof-school	576.0	44.746528	11.962477	25.0	36.00	43.0	51.0	90.0
Some-college	7291.0	35.756275	13.474051	17.0	24.00	34.0	45.0	90.0

In [597…

```
'''
Per the assignment, group the records based on occupation.
Utilize describe() to find how the mean age is distributed.
'''
# Use the goupby() function and store in a new DataFrame called df_group2.
df_group2 = df_subset.groupby(['occupation'])

# Use describe() to find out how the mean age is distributed.
df_group2.describe()['age']
```

Out[597…

occupation	count	mean	std	min	25%	50%	75%	max
Adm-clerical	3770.0	36.964456	13.362998	17.0	26.0	35.0	46.0	90.0
Armed-Forces	9.0	30.222222	8.089774	23.0	24.0	29.0	34.0	46.0
Craft-repair	4099.0	39.031471	11.606436	17.0	30.0	38.0	47.0	90.0
Exec-managerial	4066.0	42.169208	11.974548	17.0	33.0	41.0	50.0	90.0
Farming-fishing	994.0	41.211268	15.070283	17.0	29.0	39.0	52.0	90.0
Handlers-cleaners	1370.0	32.165693	12.372635	17.0	23.0	29.0	39.0	90.0
Machine-op-inspct	2002.0	37.715285	12.068266	17.0	28.0	36.0	46.0	90.0
NoData	1843.0	40.882800	20.336350	17.0	21.0	35.0	61.0	90.0
Other-service	3295.0	34.949621	14.521508	17.0	22.0	32.0	45.0	90.0
Priv-house-serv	149.0	41.724832	18.633688	17.0	24.0	40.0	57.0	81.0
Prof-specialty	4140.0	40.517633	12.016676	17.0	31.0	40.0	48.0	90.0
Protective-serv	649.0	38.953775	12.822062	17.0	29.0	36.0	47.0	90.0
Sales	3650.0	37.353973	14.186352	17.0	25.0	35.0	47.0	90.0
Tech-support	928.0	37.022629	11.316594	17.0	28.0	36.0	44.0	73.0
Transport-moving	1597.0	40.197871	12.450792	17.0	30.0	39.0	49.0	90.0

In [598…

```
'''
Per the assignment, find which profession has the oldest workers on average.
'''
# Based on the descriptive statistics for age by occupation seen above, Exec-managerial
# Exec-managerial has a mean age of 42.17 years or a median of 41.0 years.
```

Out[598…

```
'\nPer the assignment, find which profession has the oldest workers on average.\n'
```

In [599…

```
'''
Per the assignment, find which profession has the largest share of the workforce above
'''
# Find the 75th percentile for age across the entire dataset.
df_income.describe()['age']
```

Out[599…

```
count    32561.000000
mean        38.581647
std         13.640433
min         17.000000
25%         28.000000
50%         37.000000
75%         48.000000
max         90.000000
Name: age, dtype: float64
```

In [600…

```
'''
The 75th percentile for age is 48 years old across the entire dataset.
```

```
Use filtering to create another DataFrame (df_filter2) from df_subset for ages>=48.
'''
# df_filter2 will include values for ages above the 75th percentile for age which is 48
df_filter2 = df_subset[(df_subset['age']>=48)]
df_filter2.head(10)
```

Out[600…

	age	education	occupation
1	50	Bachelors	Exec-managerial
3	53	11th	Handlers-cleaners
6	49	9th	Other-service
7	52	HS-grad	Exec-managerial
21	54	HS-grad	Other-service
24	59	HS-grad	Tech-support
25	56	Bachelors	Tech-support
27	54	Some-college	NoData
29	49	HS-grad	Craft-repair
35	48	11th	Machine-op-inspct

In [601…

```
'''
Group the records based on occupation for df_filter2.
Utilize describe() to get a count for each occupation above 48 years old.
'''
# Use the goupby() function and store in a new DataFrame called df_group2.
df_group3 = df_filter2.groupby(['occupation'])

# Use describe() to find the count for age grouped by each occupation.
df_group3.describe()['age']
```

Out[601…

occupation	count	mean	std	min	25%	50%	75%	max
Adm-clerical	810.0	56.880247	7.540856	48.0	51.0	55.0	61.0	90.0
Craft-repair	992.0	55.128024	6.097157	48.0	50.0	54.0	59.0	90.0
Exec-managerial	1262.0	56.256735	7.562629	48.0	51.0	54.0	60.0	90.0
Farming-fishing	322.0	59.375776	8.041855	48.0	53.0	58.0	64.0	90.0
Handlers-cleaners	180.0	56.244444	7.651357	48.0	50.0	55.0	61.0	90.0
Machine-op-inspct	433.0	55.267898	6.590700	48.0	50.0	54.0	59.0	90.0
NoData	722.0	64.038781	8.470305	48.0	59.0	64.0	69.0	90.0
Other-service	672.0	57.944940	7.928045	48.0	52.0	57.0	62.0	90.0
Priv-house-serv	55.0	62.690909	9.093898	48.0	54.5	62.0	70.0	81.0
Prof-specialty	1075.0	56.332093	7.479733	48.0	51.0	55.0	60.0	90.0
Protective-serv	155.0	57.212903	8.178200	48.0	51.0	54.0	62.0	90.0

occupation	count	mean	std	min	25%	50%	75%	max
Sales	894.0	57.210291	7.564613	48.0	51.0	56.0	61.0	90.0
Tech-support	169.0	55.171598	5.566708	48.0	51.0	54.0	59.0	73.0
Transport-moving	441.0	56.328798	6.372406	48.0	51.0	55.0	60.0	90.0

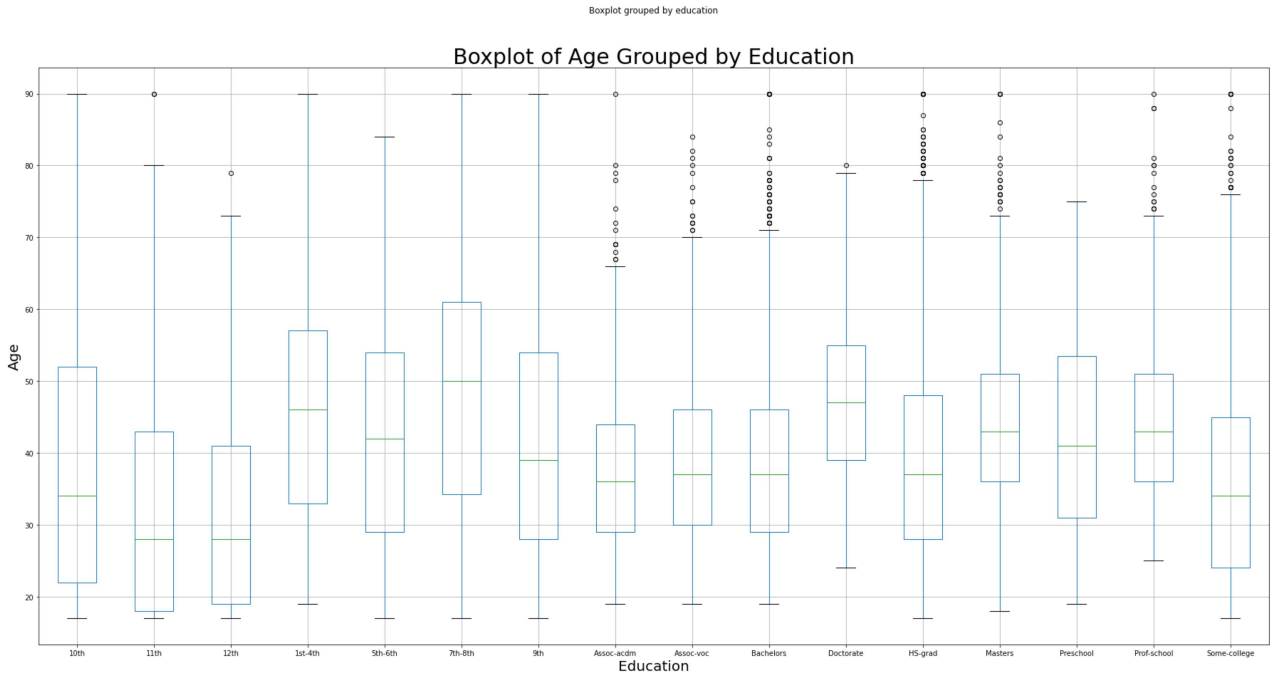
In [602…

```
print('The profession that has the largest share of the workforce above the 75th percen
```

```
The profession that has the largest share of the workforce above the 75th percentile for
age is Exec-managerial.
```

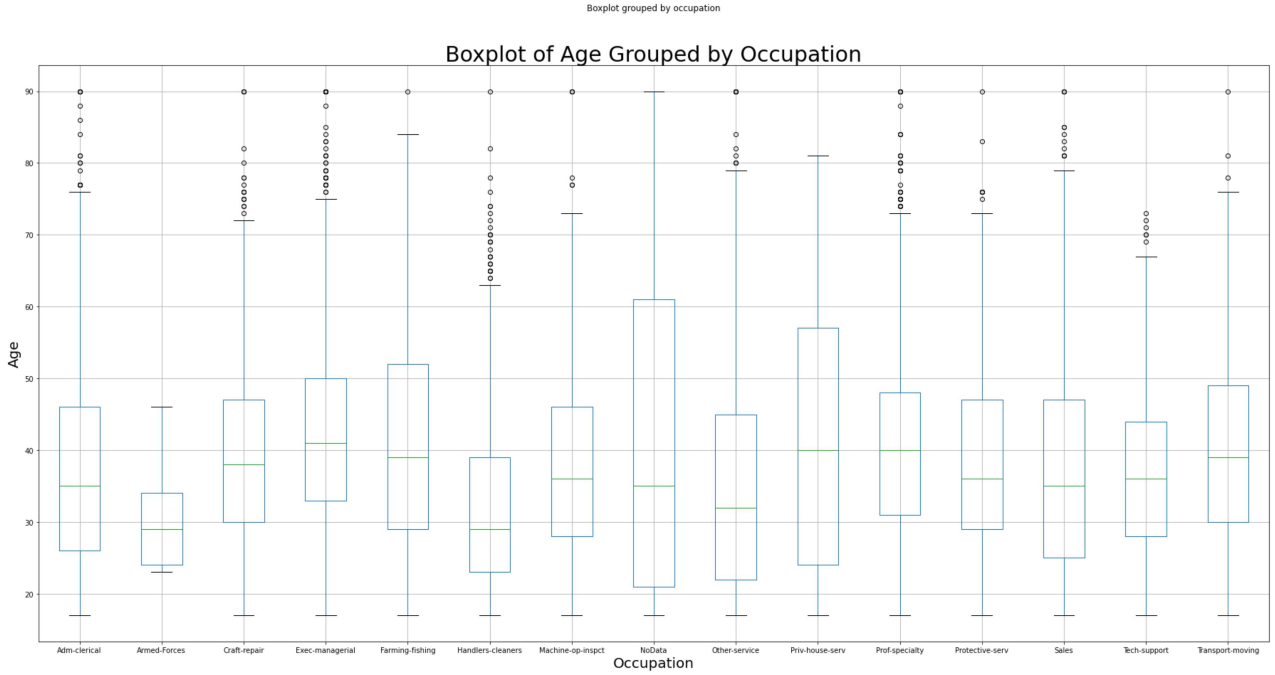
In [603…

```
'''
To identify outliers, use a box plot to understand age based on education.
'''
# Use boxplot() with age and education
df_subset.boxplot(column='age', by='education',figsize=(30,15))
plt.xlabel("Education", fontsize=20)
plt.ylabel("Age", fontsize=20)
plt.title("Boxplot of Age Grouped by Education", fontsize=30)
plt.show()
```



In [605…

```
'''
To identify outliers, use a box plot to understand age based on occupation.
'''
# Use boxplot() with age and education
df_subset.boxplot(column='age', by='occupation',figsize=(30,15))
plt.xlabel("Occupation", fontsize=20)
plt.ylabel("Age", fontsize=20)
plt.title("Boxplot of Age Grouped by Occupation", fontsize=30)
plt.show()
```

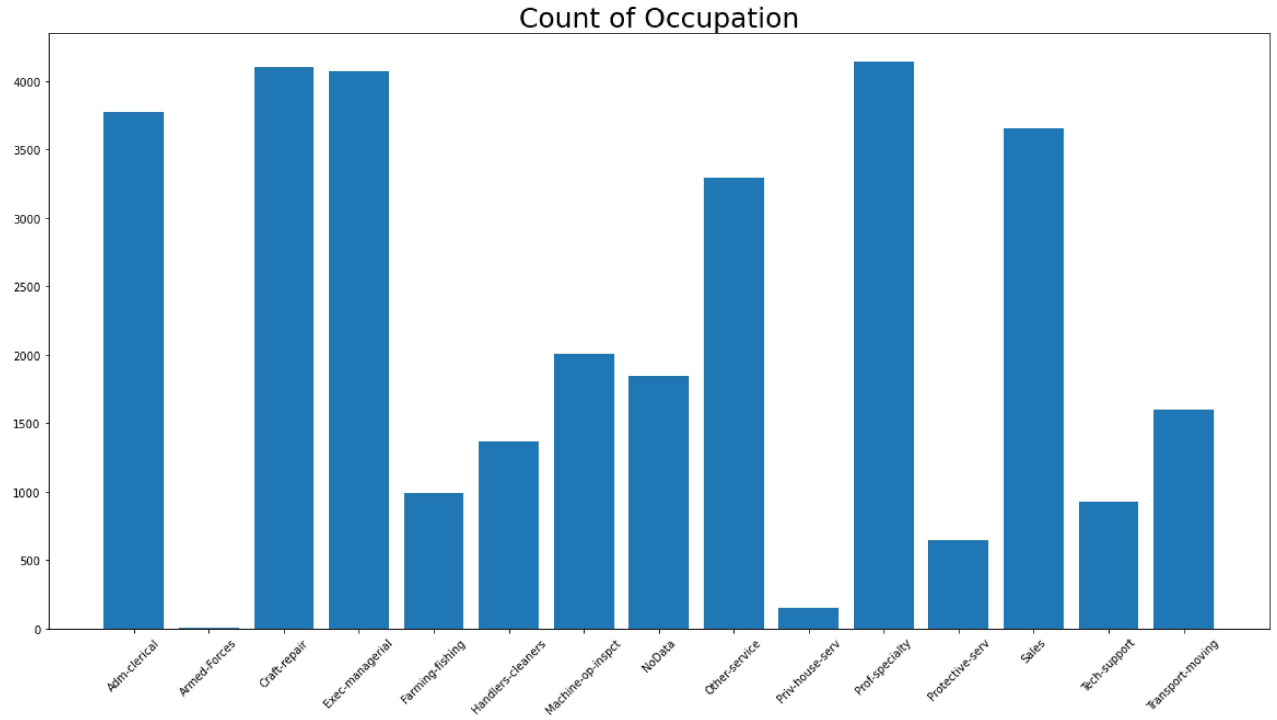


In [606…

```
# Store age descriptive statistics grouped by occupation in variable labeled stats.
stats=df_group2.describe()['age']
index = stats.index
```

In [607…

```
'''
Plot the count for each occupation on a bar chart.
'''
# Utilize the plt.bar() command for a bar chart. Use index and stats['count'] as the ar
plt.figure(figsize=(20,10))
plt.bar(x=index, height=stats['count'])
plt.xticks(rotation = 45, fontsize=10)
plt.yticks(fontsize=10)
plt.title('Count of Occupation', fontsize=25)
plt.show()
```



In [608…

```
'''
Per the assignment, merge data using common keys.
Create two data frames sampled from the initial data.
First data frame is created below with random sampling.
'''
# Use sample to randomly sample 25% of the subset data.
df_one = df_income[['occupation','age','education']].sample(frac=.25)

# df_one.shape - Overall number of rows = 8140, columns = 3 which are education occupat
# df_income.shape - Overall number of rows = 32561, columns = 14
df_one.head(10)
```

Out[608…

	occupation	age	education
244	Craft-repair	38	Some-college
17846	Sales	43	Bachelors
28185	Exec-managerial	32	Masters
19055	Other-service	57	HS-grad
14914	Sales	34	Bachelors
8547	Craft-repair	38	HS-grad
23941	Handlers-cleaners	18	11th
3317	Tech-support	45	Some-college
13459	Sales	35	Some-college
26778	Adm-clerical	65	Some-college

In [609…

```
'''
Second data frame is created below with random sampling and three columns.
```

```
Columns are occupation, relationship, and income.
'''
# Use sample to randomly sample 25% of the subset data.
df_two = df_income[['occupation','relationship','income']].sample(frac=.25)

#df_two.shape - Overall number of rows = 8140, columns = 3 which are occupation, relati
# df_income.shape - Overall number of rows = 32561, columns = 14
df_two.head(10)
```

Out[609…

	occupation	relationship	income
14551	Exec-managerial	Not-in-family	<=50K
15714	Other-service	Husband	<=50K
1674	Adm-clerical	Wife	>50K
15309	NaN	Other-relative	<=50K
24919	Sales	Wife	>50K
8710	Prof-specialty	Husband	>50K
20297	Prof-specialty	Not-in-family	>50K
6012	Prof-specialty	Not-in-family	<=50K
3072	NaN	Own-child	<=50K
5484	Prof-specialty	Husband	<=50K

In [610…

```
'''
Merge df_one and df_two into a single DataFrame df_merged.
'''
# Use the merge() command to join these two DataFrames together based on occupation.
# Drop the duplicates with drop_duplicates().
df_merged = pd.merge(df_one,df_two,on='occupation',how='inner').drop_duplicates()

# df_merged.head(20) - checked to see initial and ending data (20 rows in the merged da
df_merged.tail(20)
```

Out[610…

	occupation	age	education	relationship	income
6424950	Priv-house-serv	21	11th	Wife	<=50K
6424961	Priv-house-serv	21	11th	Other-relative	<=50K
6425011	Priv-house-serv	19	Some-college	Own-child	<=50K
6425012	Priv-house-serv	19	Some-college	Not-in-family	<=50K
6425014	Priv-house-serv	19	Some-college	Unmarried	<=50K
6425018	Priv-house-serv	19	Some-college	Wife	<=50K
6425029	Priv-house-serv	19	Some-college	Other-relative	<=50K
6425045	Priv-house-serv	71	9th	Own-child	<=50K
6425046	Priv-house-serv	71	9th	Not-in-family	<=50K
6425048	Priv-house-serv	71	9th	Unmarried	<=50K

	occupation	age	education	relationship	income
6425052	Priv-house-serv	71	9th	Wife	<=50K
6425063	Priv-house-serv	71	9th	Other-relative	<=50K
6425079	Priv-house-serv	27	5th-6th	Own-child	<=50K
6425080	Priv-house-serv	27	5th-6th	Not-in-family	<=50K
6425082	Priv-house-serv	27	5th-6th	Unmarried	<=50K
6425086	Priv-house-serv	27	5th-6th	Wife	<=50K
6425097	Priv-house-serv	27	5th-6th	Other-relative	<=50K
6425113	Armed-Forces	29	Some-college	Not-in-family	<=50K
6425114	Armed-Forces	29	Some-college	Husband	<=50K
6425115	Armed-Forces	29	Some-college	Own-child	<=50K

3. Create a series and practice basic arithmetic steps:

- a. Series 1 = 7.3, -2.5, 3.4, 1.5
- i. Index = ‘a’, ‘c’, ‘d’, ‘e’
- b. Series 2 = -2.1, 3.6, -1.5, 4, 3.1
- i. Index = ‘a’, ‘c’, ‘e’, ‘f’, ‘g’
- c. Add Series 1 and Series 2 together and print the results
- d. Subtract Series 1 from Series 2 and print the results

In [611…

```
'''
Following the instructions for Part 3a, the following code will create a series
Series 1 = 7.3, -2.5, 3.4, 1.5 with an index of Index = ‘a’, ‘c’, ‘d’, ‘e’
'''
# Create Series 1 with the corresponding index listed above.
Series_1 = pd.Series([7.3, -2.5, 3.4, 1.5], index=['a','c','d','e'])
Series_1
```

Out[611…

```
a    7.3
c   -2.5
d    3.4
e    1.5
dtype: float64
```

In [612…

```
'''
Following the instructions for Part 3b, the following code will create a series
Series 2 = -2.1, 3.6, -1.5, 4, 3.1 with an index of Index = ‘a’, ‘c’, ‘e’, ‘f’, ‘g’
'''
# Create Series 2 with the corresponding index listed above.
Series_2 = pd.Series([-2.1, 3.6, -1.5, 4, 3.1], index=['a','c','e','f','g'])
Series_2
```

Out[612…

```
a   -2.1
c    3.6
e   -1.5
f    4.0
```

```
g    3.1
dtype: float64
```

In [613…

```
'''
Following the instructions for Part 3c, Series_1 and Series_2 will be added together
and the results will be printed.
'''
# Perform the addition between Series_1 and Series_2.
add_Series = Series_1 + Series_2
print(add_Series)
```

```
a    5.2
c    1.1
d    NaN
e    0.0
f    NaN
g    NaN
dtype: float64
```

In [614…

```
'''
Following the instructions for Part 3d, Series_1 will be subtracted from Series_2
and the results will be printed.
'''
# Perform the subtraction of Series_1 from Series_2.
sub_Series = Series_2 - Series_1
print(sub_Series)
```

```
a   -9.4
c    6.1
d    NaN
e   -3.0
f    NaN
g    NaN
dtype: float64
```

In [615…

```
'''
Attempt to add Series_1 and Series_2 together by using the series.add() function mentio
'''
print(Series_1.add(Series_2, fill_value = 0))
```

```
a    5.2
c    1.1
d    3.4
e    0.0
f    4.0
g    3.1
dtype: float64
```

In [616…

```
'''
Attempt to subtract Series_1 from Series_2  by using the series.subtract() function men
'''
print(Series_2.subtract(Series_1, fill_value = 0))
```

```
a   -9.4
c    6.1
d   -3.4
e   -3.0
```

```
f    4.0
g    3.1
dtype: float64
```