Lab 05: Data Visualizations, Basic Data Structures and Data Frames

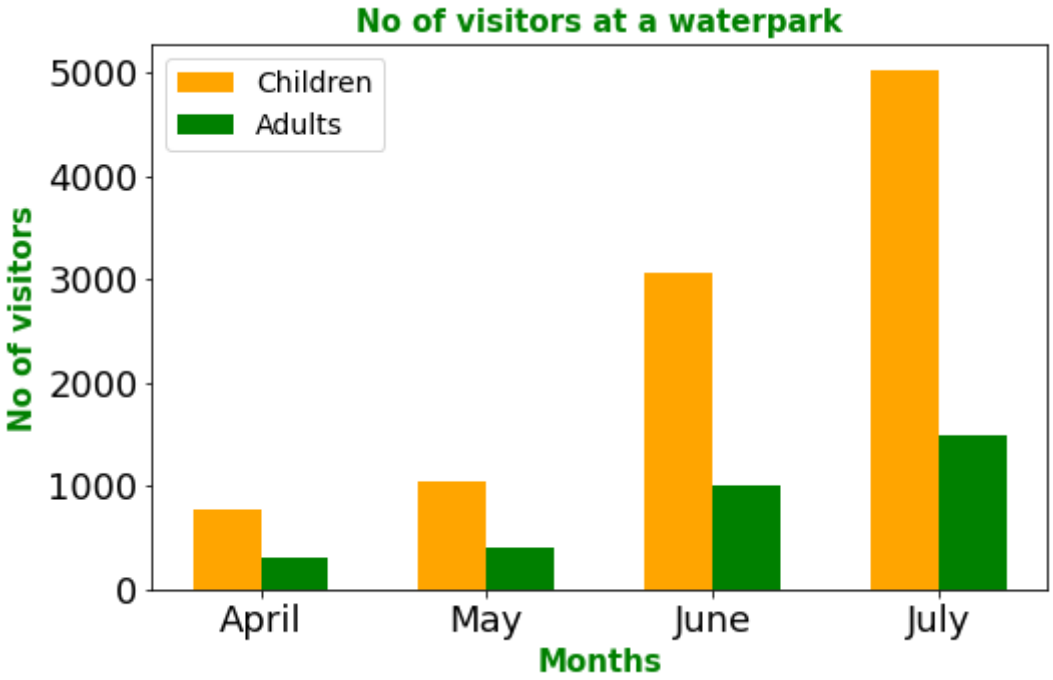
Part 1: Data Visualizations

1. Grouped Bar Plots:

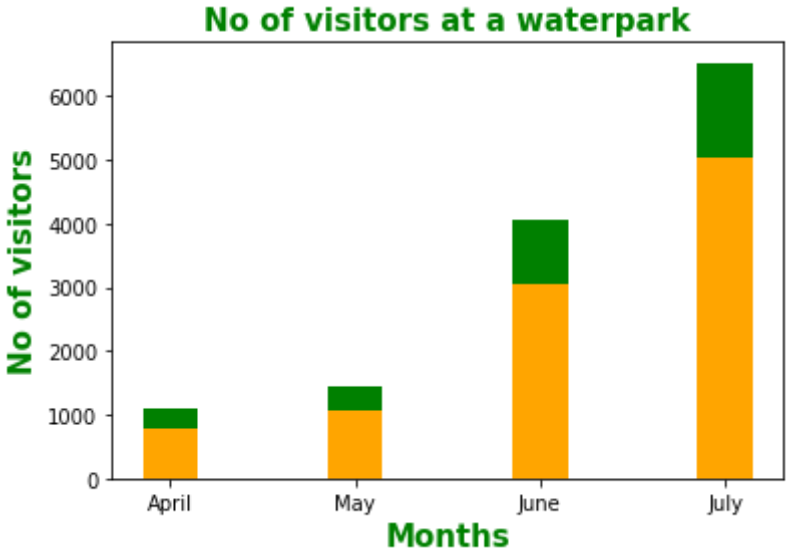
Make both a side-by-side bar plot and a stacked bar plot that displays the number of child visitors and the number of adult visitors at a waterpark in the months of April, May, June and July. Be sure to include titles, legends and appropriate labels sufficiently sized for readability. April Children: 780 Adults: 315 May Children: 1050 Adults: 400 June Children: 3056 Adults: 1000 July Children: 5025 Adults: 1500

In [243…

```
import matplotlib.pyplot as plt
import numpy as np
import math

months = ['April', 'May', 'June', 'July']
children = [780, 1050, 3056, 5025]
adults = [315, 400, 1000, 1500]

w=0.3

fig, ax = plt.subplots(figsize=(8,5))

bar1_position = np.arange(len(months))
bar2_position = [i+w for i in bar1_position]

plt.bar(bar1_position, children, w, label='Children', color='orange')
plt.bar(bar2_position, adults, w, label='Adults', color='green')

plt.title('No of visitors at a waterpark', fontweight='bold', fontsize=15, colo
plt.xlabel('Months', fontsize=15, fontweight='bold', color='green')
plt.ylabel('No of visitors', fontsize=15, fontweight='bold', color='green')
plt.xticks(bar1_position+w/2, months, fontsize=18)
plt.yticks(fontsize=18)
plt.legend(fontsize=14)
plt.show()
```



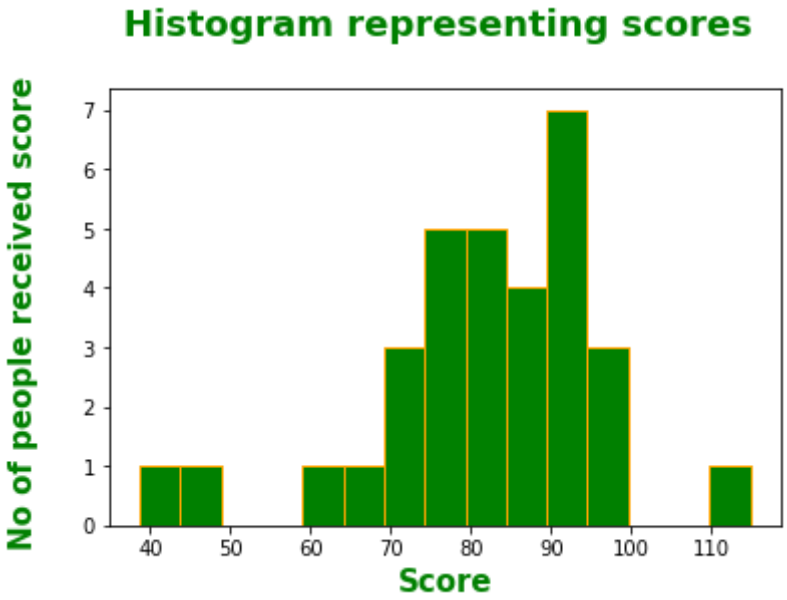
In [244…
```
plt.bar(months, children, w, label='Children', color='orange')
plt.bar(months, adults, w, bottom=children, label='Adults', color='green')


plt.title('No of visitors at a waterpark', fontweight='bold', fontsize=15, colo
plt.xlabel('Months', fontsize=15, color='green', fontweight='bold')
plt.ylabel('No of visitors', fontsize=15, color='green', fontweight='bold')
plt.show()
```



2. Histogram:

Make a histogram of the following scores from the Fall 2017 Data Structures course at Loyola University Chicago. Feel free to experiment on the best number of histogram bins for visualization. 114.8, 98.8, 97.3, 96, 94.1, 93.1, 93.1, 91.6, 91.5, 91.3, 90.3, 89.2, 87.5, 87.4, 85.2, 81.7, 81.6, 81.5, 80, 79.3, 78.2, 77.6, 77.1, 76.7, 75.1, 73.9, 72, 71, 64.6, 63.3, 47.2, 38.7

In [245…
```
scores = [114.8, 98.8, 97.3, 96, 94.1, 93.1, 93.1, 91.6, 91.5, 91.3, 90.3, 89.2

plt.hist(scores, bins=15, ec='orange', color='green')
plt.title('Histogram representing scores \n', fontsize=18, fontweight='bold', c
plt.xlabel('Score', fontsize=15, color='green', fontweight='bold')
plt.ylabel('No of people received score \n', fontsize=15, color='green', fontwe

plt.show()
```



3. Line Plot:

Create a line plot of sin(x) and cos(x + π/2) for -2π < x < 2π where x increases at intervals of π/4. Make the sin(x) graph red and make the cos(x+π/2) graph green Put both lines onto the same plot Using the same info as above, make a subplot with 2 different graphs- one graph for sin(x) and one graph for cos(x+π/2)

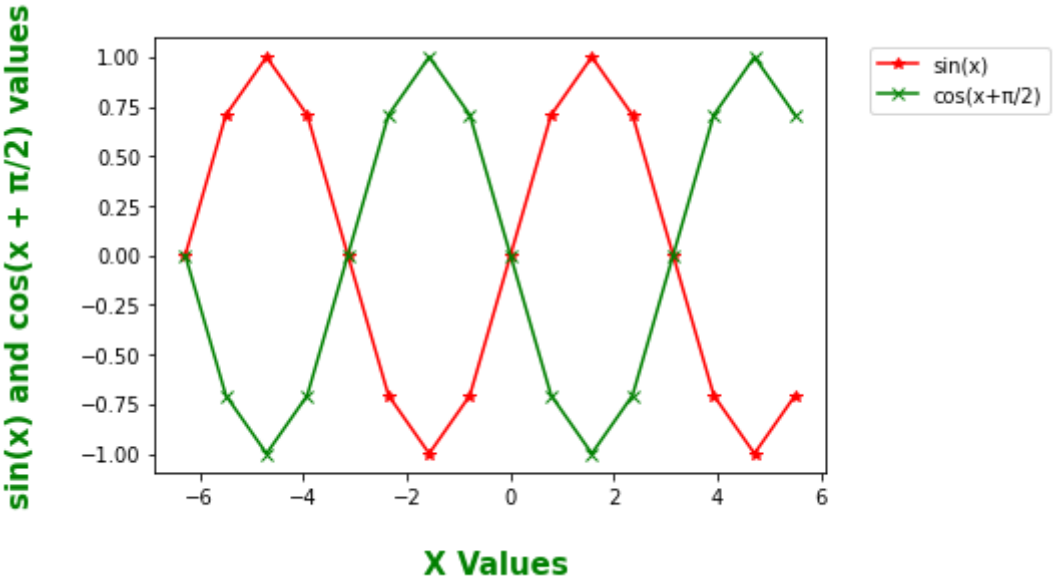
x = -6.283, -5.498, -4.712, -3.927, -3.142, -2.356, -1.571, -.7854, 0, .7854, 1.571, 2.356, 3.142, 3.927, 4.712, 5.498, 6.283

sin(x) = 0, .70711, 1, .70711, 0, -.70711, -1, -.70711, 0, .70711, 1, .70711, 0, -.70711, -1, -.70711, 0

cos(x + π/2) = 0, -.70711, -1, -.70711, 0, .70711, 1, .70711, 0, -.70711, -1, -.70711, 0, .70711, 1, .70711, 0

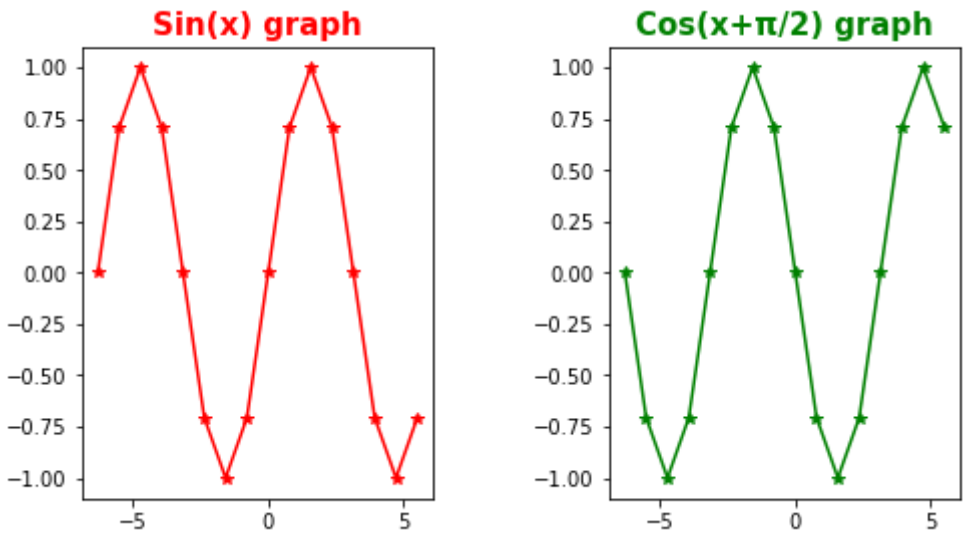
In [246…
```
x = np.arange(-2*math.pi, 2*math.pi, math.pi/4)

plt.plot(x, np.sin(x), color='red', marker='*', label='sin(x)')
plt.plot(x, np.cos(x+math.pi/2), color='green', marker='x', label='cos(x+π/2)')

plt.xlabel("\n X Values", fontsize=15, color='green', fontweight='bold')
plt.ylabel("sin(x) and cos(x + π/2) values\n", fontsize=15, color='green', font
plt.legend(bbox_to_anchor=(1.05, 1.0), loc="upper left")
```

```
plt.show()
```



In [247…

```
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(8, 5))

ax1.set_title('Sin(x) graph', fontsize=15, fontweight='bold', color='red')
ax1.plot(x, np.sin(x), color='red', marker='*')
ax2.set_title('Cos(x+π/2) graph', fontsize=15, fontweight='bold', color='green'
ax2.plot(x, np.cos(x+math.pi/2), color='green', marker='*')
fig.tight_layout(pad=5.0)
```



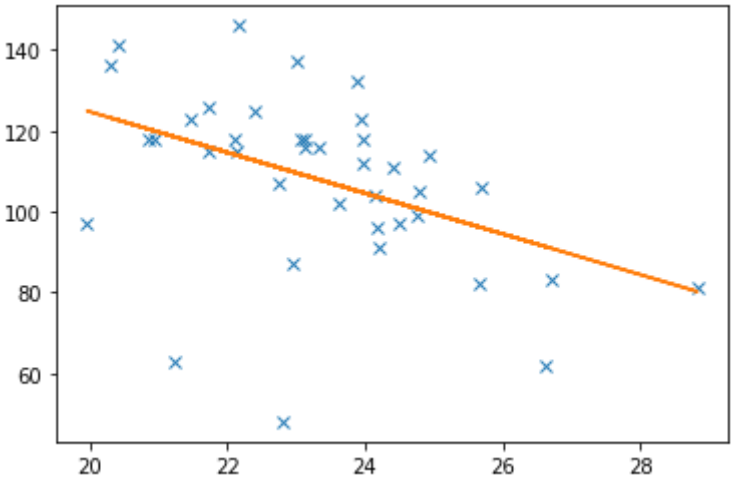
4. Scatter Plot:

Using the following data about winter temperatures affecting the number of days for lake ice at Lake Superior, construct a scatter plot to display the data. Include a line of best fit.

Mean Temperature (in Fahrenheit): 22.94, 23.02, 25.68, 19.96, 24.80, 23.98, 22.10, 20.30, 24.20, 22.74, 24.16, 24.94, 22.40, 22.14, 20.84, 25.66, 21.73, 24.49, 24.13, 22.17, 21.73, 20.41, 24.41, 23.95, 20.95, 26.71, 22.81, 23.11, 23.33, 28.83, 23.11, 21.47, 23.97, 24.75, 23.61,

23.08, 21.24, 26.63, 23.88 Days of Ice: 87, 137, 106, 97, 105, 118, 118, 136, 91, 107, 96, 114, 125, 115, 118, 82, 115, 97, 104, 146, 126, 141, 111, 123, 118, 83, 48, 118, 116, 81, 116, 123, 112, 99, 102, 118, 63, 62, 132

In [248…
```
meanTemperature = np.array([22.94, 23.02, 25.68, 19.96, 24.80, 23.98, 22.10, 20

days_of_ice = np.array([87, 137, 106, 97, 105, 118, 118, 136, 91, 107, 96, 114,

slope, intercept = np.polyfit(meanTemperature, days_of_ice, 1)
plt.plot(meanTemperature, days_of_ice, 'x')
plt.plot(meanTemperature, slope*meanTemperature+intercept)

plt.show()
```



Part 2: Basic Data Structure

1: Lists 1) Make a list with the spelled-out number strings ‘one’, ‘two’, ‘three’, ‘four’, and ‘five’ in that order and call it myList.

In [249…
```
myList = ['one', 'two', 'three', 'four', 'five']
myList
```

Out[249]:
```
['one', 'two', 'three', 'four', 'five']
```

2) Remove ‘three’ from the list using positional indexing.

In [250…
```
myList.pop(2)
myList
```

Out[250]:
```
['one', 'two', 'four', 'five']
```

3) Check if ‘four’ is in the list.

In [251…
```
'four' in numberStings
```

Out[251]:
```
True
```

4) Append 'six' to the end of the list, then print the length of the list.

In [252…
```
myList.append('six')
len(myList)
```

Out[252]:
```
5
```

5) Print the contents of the list, but also next to each item print the length of the string (e.g. one is 3, four is 4 using a for loop.

In [253…
```
for item in myList:
    print(item, 'is', len(item))
```

```
one is 3
two is 3
four is 4
five is 4
six is 3
```

6) Create a list only of the lengths of the strings and show your result. You can use the loop before to fill the list.

In [254…
```
lengthsOfList = []
for item in myList:
    lengthsOfList.append(len(item))

lengthsOfList
```

Out[254]:
```
[3, 3, 4, 4, 3]
```

2: Dictionaries

1) Make a dictionary with the keys be English words as below, and the values be the translation. You can use this language example (German) or choose your own. Note: you need to make sure all of these words are represented as strings, in quotes. apple - Apfel apples - Äpfel I - Ich and - und like - mag strawberries - Erdbeeren

In [255…
```
my_dictionary = {'apple':'Apfel', 'apples':'Äpfel', "I":'Ich', 'and':'und', 'li
my_dictionary
```

Out[255]:
```
{'apple': 'Apfel',
 'apples': 'Äpfel',
 'I': 'Ich',
 'and': 'und',
 'like': 'mag',
 'strawberries': 'Erdbeeren'}
```

2) Use the dictionary to look up the translation for 'apple' and 'like'.

In [256…
```
print('The translation for "apple" is "', my_dictionary['apple'],'"')
print('The translation for "like" is "', my_dictionary['like'], '"')
```

```
The translation for "apple" is " Apfel "
The translation for "like" is " mag "
```

3) Make a variable var with the string “I like apples and strawberries”.

In [257…
```
var = 'I like apples and strawberries'
```

4) Now create a list from var with each word a separate item (this is a string split operation).

In [258…
```
items = list(var.split())
items
```

Out[258]:
```
['I', 'like', 'apples', 'and', 'strawberries']
```

5) Iterate through the list you’ve created and replace any word in your dictionary with the translation.

In [259…
```
for i, item in enumerate(items):
    if item in my_dictionary:
        items[i] = my_dictionary[item]

items
```

Out[259]:
```
['Ich', 'mag', 'Äpfel', 'und', 'Erdbeeren']
```

6) Now take your new list and turn it into a string with spaces between the words.

In [260…
```
newList = ' '.join(items)
newList
```

Out[260]:
```
'Ich mag Äpfel und Erdbeeren'
```

3: Arrays

1) Create an array of zeros of size 8 x 8 and print the data type of the array.

In [261…
```
zerosArray = np.zeros((8, 8), int)
print(zerosArray.dtype)
```

```
int64
```

2) Fill the array with the numbers 1 to 64 first by row, then by column. You may want to use a for loop inside a for loop to do this.

In [262…
```
n = 0
for i, x in enumerate(zerosArray):
    for j, y in enumerate(zerosArray):
        zerosArray[i, j] = n + 1
        n = n + 1

zerosArray
```

Out[262]:
```
array([[ 1,  2,  3,  4,  5,  6,  7,  8],
       [ 9, 10, 11, 12, 13, 14, 15, 16],
       [17, 18, 19, 20, 21, 22, 23, 24],
       [25, 26, 27, 28, 29, 30, 31, 32],
       [33, 34, 35, 36, 37, 38, 39, 40],
       [41, 42, 43, 44, 45, 46, 47, 48],
       [49, 50, 51, 52, 53, 54, 55, 56],
       [57, 58, 59, 60, 61, 62, 63, 64]])
```

3) Transpose the array.

In [263…
```
zerosArray = np.transpose(zerosArray)
zerosArray
```

Out[263]:
```
array([[ 1,  9, 17, 25, 33, 41, 49, 57],
       [ 2, 10, 18, 26, 34, 42, 50, 58],
       [ 3, 11, 19, 27, 35, 43, 51, 59],
       [ 4, 12, 20, 28, 36, 44, 52, 60],
       [ 5, 13, 21, 29, 37, 45, 53, 61],
       [ 6, 14, 22, 30, 38, 46, 54, 62],
       [ 7, 15, 23, 31, 39, 47, 55, 63],
       [ 8, 16, 24, 32, 40, 48, 56, 64]])
```

4) Print only the top 4 rows and columns.

In [264…
```
print(zerosArray[:4, :4])
```

```
[[ 1  9 17 25]
 [ 2 10 18 26]
 [ 3 11 19 27]
 [ 4 12 20 28]]
```

5) Make a 1D array out of your 2D array with the numbers 1 to 64 in order (note the column vs row issue, you may need transposes.)

In [265…
```
zerosArray = np.transpose(zerosArray)
zerosArray = zerosArray.flatten()
zerosArray
```

Out[265]:
```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17,
       18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34,
       35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51,
       52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64])
```

6) Now take that 1D array you made from before and reshape it back to the original 2D array.

In [266…
```
print(np.reshape(zerosArray, (8, 8)))
```

```
[[ 1  2  3  4  5  6  7  8]
 [ 9 10 11 12 13 14 15 16]
 [17 18 19 20 21 22 23 24]
 [25 26 27 28 29 30 31 32]
 [33 34 35 36 37 38 39 40]
 [41 42 43 44 45 46 47 48]
 [49 50 51 52 53 54 55 56]
 [57 58 59 60 61 62 63 64]]
```

4: Application - Word Counts

Word counts are often used in text processing to automatically classify documents by topic. They are also used to automatically measure the “sentiment” by counting, for example, the number of positive or negative words used in the comment or essay. Write code to count the number of unique words in a very large string using the following steps. First convert the string to a list with each word a separate item in the list. Hint: use a string split function for your language, and make sure it separates by “ ”. Then use a dictionary to associate each word with a count. Note, the dictionary won’t be able to increment a key unless you add it first, so you may have to check to see if it exists before setting the original count of a word to 1. Print each word and its count afterwards, and test with an interesting block of text that will have multiple words counted multiple times. (Note, the words don’t have to be in any particular order.) For example: “how much wood would a woodchuck chuck if a woodchuck could chuck wood” Expected output: how - 1 much -1 wood - 2 would - 1 a - 2 woodchuck - 2 chuck - 2 if - 1 could - 1

In [269…
```
givenText = "how much wood would a woodchuck chuck if a woodchuck could chuck w
convertedList = givenText.split()
words_dictionary = {item: convertedList.count(item) for item in convertedList}
words_dictionary
```

Out[269]:
```
{'how': 1,
 'much': 1,
 'wood': 2,
 'would': 1,
 'a': 2,
 'woodchuck': 2,
 'chuck': 2,
 'if': 1,
 'could': 1}
```

Part 3: Data Frames

In this part, we will study a classic data set - the survivors in the sinking of the Titanic. As there were limited lifeboats, decisions were made prioritizing who would and would not survive. We will observe how different factors such as age, sex, and class affected a person’s chance of survival using data frames.

Steps:

1. Input the following data into a data frame called titanic, and display the entire data frame: Sex, Class, Survived, Died Children, First, 6, 0 Children, Second, 24, 0 Children, Third, 27, 52 Men, First, 57, 118 Men, Second, 14, 154 Men, Third, 75, 387 Men, Crew, 192, 693 Women, First, 140, 4 Women, Second, 80, 13 Women, Third, 76, 89 Women, Crew, 20, 3

In [293…
```
import pandas as pd
```

```
data = [['Children', 'First', 6, 0],
 ['Children','Second', 24, 0],
 ['Children', 'Third', 27, 52],
 ['Men', 'First', 57, 118],
 ['Men', 'Second', 14, 154],
 ['Men', 'Third', 75, 387],
 ['Men', 'Crew', 192, 693],
 ['Women', 'First', 140, 4],
 ['Women', 'Second', 80, 13],
 ['Women', 'Third', 76, 89],
 ['Women', 'Crew', 20, 3]]

titanic = pd.DataFrame(data, columns=['Sex', 'Class', 'Survived', 'Died'])
titanic
```

Out[293]:

	Sex	Class	Survived	Died
0	Children	First	6	0
1	Children	Second	24	0
2	Children	Third	27	52
3	Men	First	57	118
4	Men	Second	14	154
5	Men	Third	75	387
6	Men	Crew	192	693
7	Women	First	140	4
8	Women	Second	80	13
9	Women	Third	76	89
10	Women	Crew	20	3

1. Now only show the data of the people in first class.

In [294…

```
df_firstClass = titanic.loc[df['Class'] == 'First']
df_firstClass
```

Out[294]:

	Sex	Class	Survived	Died
0	Children	First	6	0
3	Men	First	57	118
7	Women	First	140	4

1. Delete the crew members from the data.

In [295…

```
titanic = titanic[titanic['Class'] != 'Crew']
titanic
```

Out[295]:

	Sex	Class	Survived	Died
0	Children	First	6	0
1	Children	Second	24	0
2	Children	Third	27	52
3	Men	First	57	118
4	Men	Second	14	154
5	Men	Third	75	387
7	Women	First	140	4
8	Women	Second	80	13
9	Women	Third	76	89

1. Create a new column that is the total number of people for that group (those who survived + died).

In [311…
```
titanic = titanic.eval('Total_Number_of_People = Survived + Died')
titanic
```

Out[311]:

	Sex	Class	Survived	Died	Percentage of people survived	Total_Number_of_People
0	Children	First	6	0	100.00	6
1	Children	Second	24	0	100.00	24
2	Children	Third	27	52	34.18	79
3	Men	First	57	118	32.57	175
4	Men	Second	14	154	8.33	168
5	Men	Third	75	387	16.23	462
7	Women	First	140	4	97.22	144
8	Women	Second	80	13	86.02	93
9	Women	Third	76	89	46.06	165

1. Create a new column with the percentage of people who survived.

In [314…
```
titanic['Percentage of people survived'] = round((titanic['Survived']/titanic['
titanic
```

Out[314]:

	Sex	Class	Survived	Died	Percentage of people survived	Total_Number_of_People
0	Children	First	6	0	100.00	6
1	Children	Second	24	0	100.00	24
2	Children	Third	27	52	34.18	79
3	Men	First	57	118	32.57	175
4	Men	Second	14	154	8.33	168
5	Men	Third	75	387	16.23	462
7	Women	First	140	4	97.22	144
8	Women	Second	80	13	86.02	93
9	Women	Third	76	89	46.06	165

1. Delete the column indicating the total number of people in that group.

In [298…
```
titanic = titanic.drop(columns=['Total_Number_of_People'])
titanic
```

Out[298]:

	Sex	Class	Survived	Died	Percentage of people survived
0	Children	First	6	0	100.00
1	Children	Second	24	0	100.00
2	Children	Third	27	52	34.18
3	Men	First	57	118	32.57
4	Men	Second	14	154	8.33
5	Men	Third	75	387	16.23
7	Women	First	140	4	97.22
8	Women	Second	80	13	86.02
9	Women	Third	76	89	46.06

1. Only show the rows where more than 80% of the people survived.

In [299…
```
titanic[titanic['Percentage of people survived'] >= 80]
```

Out[299]:

	Sex	Class	Survived	Died	Percentage of people survived
0	Children	First	6	0	100.00
1	Children	Second	24	0	100.00
7	Women	First	140	4	97.22
8	Women	Second	80	13	86.02

1. Then only show the rows where less than 40% of the people survived.

In [300…
```
titanic[titanic['Percentage of people survived'] <= 40]
```

Out[300]:

	Sex	Class	Survived	Died	Percentage of people survived
2	Children	Third	27	52	34.18
3	Men	First	57	118	32.57
4	Men	Second	14	154	8.33
5	Men	Third	75	387	16.23

1. Calculate the total number of people that survived and died for each class, then report the percentages. (Hint: Use a grouped calculation.)

In [301…
```
titanic.groupby('Class').sum()
```

Out[301]:

Class	Survived	Died	Percentage of people survived
First	203	122	229.79
Second	118	167	194.35
Third	178	528	96.47

1. Save your table in CSV format (as e.g. titanic_data.csv) with the first line as headers for the columns.

In [302…
```
titanic.to_csv('titanic_data.csv')
```

1. Duplicate the CSV file on your computer since you will be editing the copied version (e.g. titanic_data2.csv). Open the new CSV file in a text editor. Note the way the data is organized. Now, in the text editor, add new lines including the data for the crew that was removed earlier. (Help: the percentage of male crew and female crew that survived was 21.69% and 86.96%.)

1. Now read that updated CSV file into a new data frame called titanic2 and display the data.

In [304…
```
pd.read_csv('titanic_data2.csv')
```

Out[304]:

	Unnamed: 0	Sex	Class	Survived	Died	Percentage of people survived
0	0	Children	First	6	0	100%
1	1	Children	Second	24	0	100%
2	2	Children	Third	27	52	34.18%
3	3	Men	First	57	118	32.57%
4	4	Men	Second	14	154	8.33%
5	5	Men	Third	75	387	16.23%
6	7	Women	First	140	4	97.22%
7	8	Women	Second	80	13	86.02%
8	9	Women	Third	76	89	46.06%
9	10	Men	Crew	192	693	21.69%
10	11	Women	Crew	20	3	86.95%

In [ ]: