HW 9

SDS348 Spring 2021

Name: Santhosh Saravanan

EID : sks3648

This homework is due on April 26, 2021 at 8am. Submit a pdf file on Gradescope.

For all questions, include the Python commands/functions that you used to find your answer. Answers without supporting code will not receive credit. Write full sentences to describe your findings.

Question 1: (14 pts)

1.1 (2 pts) The dataset `faithful` contains information about eruptions of the Old Faithful geyser in Yellowstone National Park. Run the code below to download the dataset from GitHub. What type of object is it? Take a peek at the first few rows using `.head()`, looks familiar?

In [1]:
```
# Import package pandas
import pandas as pd
```

In [2]:
```
# Import dataset
faithful = pd.read_csv("https://raw.githubusercontent.com/laylaguyot/datasets/main//faithful.csv")
```

In [4]:
```
faithful.head()
type(faithful)
```

Out[4]:
```
pandas.core.frame.DataFrame
```

This dataset is familiar and it's a pandas DataFrame object.

1.2 (2 pts) What are the minimum and maximum values of the variables `eruptions` and `waiting` (both measured in minutes)? You can access individual variables in a dataframe using the `.` operator (e.g., `faithful.eruptions`).

In [9]:
```
print("The minimum eruption time: " + str(min(faithful.eruptions)) + " minutes.")
print("The maximum eruption time: " + str(max(faithful.eruptions)) + " minutes.")
print("The minimum waiting time: " +  str(min(faithful.waiting)) + " minutes.")
print("The maximum waiting time: " + str(max(faithful.waiting)) + " minutes.")
```

```
The minimum eruption time: 1.6 minutes.
The maximum eruption time: 5.1 minutes.
The minimum waiting time: 43 minutes.
The maximum waiting time: 96 minutes.
```

Minimum eruption time is 1.6 minutes. Maximum eruption time is 5.1 minutes. Minimum waiting time is 43 minutes. Maximum waiting time is 96 minutes.

1.3 (2 pts) Using the package `numpy` , what is the standard deviation of the variable `eruptions` ?

In [19]:
```
# Import package numpy
import numpy as np
```

In [ ]:
```

```

In [12]:
```
np.std(faithful.eruptions)
```

Out[12]:
```
1.1392712102257678
```

The standard deviation of the variable eruptions is 1.13927 minutes.

1.4 (2 pts) Recall how logical indexing of a dataframe works in Python. In the example code below, I ask Python for the number of rows and columns in the dataset where the variable `waiting` takes on values greater than 60. Then I ask for the average of the variable `eruptions` when the variable `waiting` is above 60. What is the mean of the variable `eruptions` when waiting is *less than* 1 hour?

In [13]:
```
# rows and columns
faithful[faithful.waiting>60].shape
```

Out[13]:
```
(189, 3)
```

In [20]:
```
# mean of eruptions for waiting values greater than 60
np.mean(faithful[faithful.waiting > 60].eruptions)
```

Out[20]:
```
4.138587301587303
```

In [21]:
```
np.mean(faithful[faithful.waiting < 60].eruptions)
```

Out[21]:
```
1.9982727272727274
```

The mean of the variable eruptions when waiting is less than 1 hour is 1.9983 minutes.

1.5 (2 pts) What is the standard deviation of the variable `eruptions` when `waiting` is *greater than* the median?

In [22]:
```
np.std(faithful[faithful.waiting > np.median(faithful.waiting)].eruptions)
```

Out[22]:
```
0.37160308361435546
```

The standard deviation of the variable eruption when waiting is greater than the median is 0.3716.

1.6 (4 pts) Both variables are measured in minutes. Create two new variables named `eruptions_h` and `waiting_h` that give each variable in hours rather than minutes and add them to the dataset `faithful`. To help get you started, I have given you code that creates a new variable called `eruptions_minus_one`. Instead, computes the requested transformation. Take a peek at the first few rows using `.head()`. What is the mean `waiting` time in minutes? in hours?

In [55]:
```
faithful['eruptions_h'] = (faithful['eruptions']/60)
faithful['waiting_h'] = (faithful['waiting']/60)
faithful.head()
print("The mean waiting time: " + str(np.mean(faithful.waiting)) + " minute
s.")
print("The mean waiting time: " + str(np.mean(faithful.waiting_h)) + " hour
s.")
```

```
The mean waiting time: 70.8970588235294 minutes.
The mean waiting time: 1.1816176470588227 hours.
```

The mean waiting time in minutes is 70.89705 minutes. The mean waiting time in hours is 1.18161 hours.

Question 2: (11 pts)

2.1 (3 pts) Create a list `food` containing the names of your favorite foods. Your list should contain at least 5 different kinds of food. Sort the list so that the names appear in alphabetical order. How many items are in the list?

In [59]:
```
food = ["Chicken Curry", "Ramray Wong pasta", "Alfredo Pasta ", "Choccy Choco
late Cake", "Classic American CheeseBurger",  "Chicken Crispers", "Chow Mein"
, "Sweetfire Chicken Breast", "Banana Pudding", "Bean Rice", "Lobster Biscuit
s", "Caesar Salad"]
print(sorted(food))
print(len(food))
```

```
['Alfredo Pasta ', 'Banana Pudding', 'Bean Rice', 'Caesar Salad', 'Chicken Cr
ispers', 'Chicken Curry', 'Choccy Chocolate Cake', 'Chow Mein', 'Classic Amer
ican CheeseBurger', 'Lobster Biscuits', 'Ramray Wong pasta', 'Sweetfire Chick
en Breast']
12
```

There are 12 items in the list.

2.2 (2 pts) Using the function `.sort()`, sort the list in alphabetical order. What is your first favorite food in alphabetical order?

In [60]:
```
food = sorted(food)
print(food[0])
```

```
Alfredo Pasta
```

My first favorite food in alphabetical order is Alfredo Pasta.

2.3 (3 pts) Imagine that you have spent a week eating only your favorite foods. Create a dictionary `food_dict` that contains the names of your favorite foods as `keys` & counts for each time you ate that food as `values`. How many times did you eat that week?

In [61]:
```
counts = [2,3,2,1,3,3,2,3,3,2,6,2]
food_dict = dict(zip(food, counts))
print(food_dict)
print(sum(food_dict.values()))
```

```
{'Alfredo Pasta ': 2, 'Banana Pudding': 3, 'Bean Rice': 2, 'Caesar Salad': 1,
'Chicken Crispers': 3, 'Chicken Curry': 3, 'Choccy Chocolate Cake': 2, 'Chow
Mein': 3, 'Classic American CheeseBurger': 3, 'Lobster Biscuits': 2, 'Ramray
Wong pasta': 6, 'Sweetfire Chicken Breast': 2}
32
```

I ate 32 times that week. I tend to eat a lot because I have a fast metabolism; some say it's a gift and I think it's a curse.

2.4 (1 pt) Which of your favorite food did you eat the most often?

In [62]:
```
import operator

max(food_dict.items(), key=operator.itemgetter(1))[0]
```

Out[62]:
```
'Ramray Wong pasta'
```

I ate the Ramray Wong pasta the most often.

2.5 (2 pts) Using a `for` loop, multiply the values in your dictionary by 4 so that you can estimate how many times you would eat your favorite food per month. Has this ever happened in real life?!

In [63]:
```
for key in food_dict:
  food_dict[key]*=4
print(food_dict)
print(sum(food_dict.values()))
```

```
{'Alfredo Pasta ': 8, 'Banana Pudding': 12, 'Bean Rice': 8, 'Caesar Salad':
4, 'Chicken Crispers': 12, 'Chicken Curry': 12, 'Choccy Chocolate Cake': 8,
'Chow Mein': 12, 'Classic American CheeseBurger': 12, 'Lobster Biscuits': 8,
'Ramray Wong pasta': 24, 'Sweetfire Chicken Breast': 8}
128
```

This has happened in real life unforunately. I was saved due to my high metabolism, but I feel that a normal person would get a serious health condition and be as dead as a doornail before reaching the hospital :).

/