Next project Ideas:

```
Started on 7/14/23; Completed 7/16/23
1. Data Analysis of measuring sea ice changes :  https://www.ncei.noaa.gov/access/monitoring/snow-

Started on 7/14/23; Completed 7/15/23
2. Predicting the occurrences of wildfires based on rainfall - finished; wrote data into a tuple.

Started on 7/15/23; Completed - 7/16/23
3. Mapping the distance between stars - Next up: https://medium.com/analytics-vidhya/the-distance-
https://machinelearningmastery.com/distance-measures-for-machine-learning/; https://towardsdatasci

Started on 7/16/23; Completed same day.
4. Predicting AZ water levels in five years;
 https://www.phoenix.gov/waterservices/resourcesconservation/yourwater/historicaluse;
https://news.asu.edu/20221115-arizona-impact-future-water-arizona;https://new.azwater.gov/aaws/map
```

Sent from my iPhone

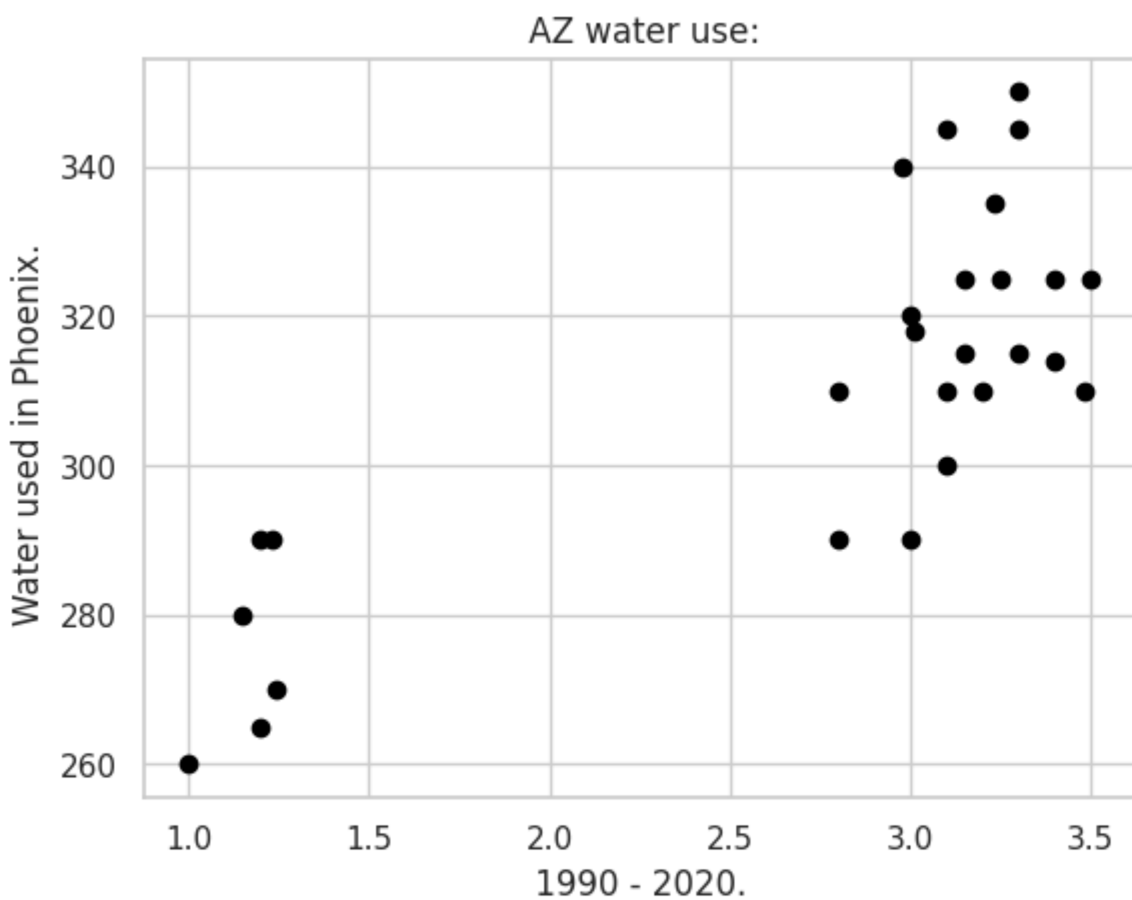
```
# Calulating Phx, AZ water levels over the last five years and future predictions:

# Import relevant libraries:

import matplotlib
import matplotlib.pyplot as plt

import numpy as np
from datetime import date, timedelta
from datetime import datetime
import pandas as pd
from sklearn.linear_model import LinearRegression
from scipy import stats

from sklearn import preprocessing

plt.rc("font", size=14)
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
import seaborn as sns
sns.set(style="white")
sns.set(style="whitegrid", color_codes=True)
```

✓ 0s completed at 10:39 PM

```
water_used = [280, 290, 260,265, 290, 270, 325, 345,310, 325, 310, 325, 350,
              315, 314, 325, 335, 340, 345, 315, 318, 290, 310, 320, 310, 300,
              290, 300, 310, 310]
water_used1 = np.array([280, 290, 260,265, 290, 270, 325, 345,310, 325, 310, 325, 350,
              315, 314, 325, 335, 340, 345, 315, 318, 290, 310, 320, 310, 300,
              290, 300, 310, 310])
population_of_Phoenix_overtime =[1.15, 1.2, 1, 1.2, 1.23, 1.24, 3.15, 3.30,
                                 3.10, 3.4,3.48, 3.50, 3.3,3.3,3.4, 3.25, 3.23,
                                 2.98, 3.10, 3.15, 3.01, 3, 2.8, 3, 3.2, 3.1, 2.8,
                                 3.3]
length_of_water_used = len(water_used)
print(length_of_water_used, "years")
start_date = datetime.strptime("1990", "%Y")
end_date = datetime.strptime("2020", "%Y")

# Difference of water usages per annum:
differences_seen = np.triu(water_used1.reshape(len(water_used1), -1)- water_used1)
averages_seen = np.mean(differences_seen[differences_seen != 0])
Y = 'Y'
date_list = pd.date_range(start_date, end_date, freq=Y)
print(len(date_list)); print("length of date list")
print(date_list)
print("These are the differences seen:", differences_seen,"." )
print("These are the average differences seen:", averages_seen, ".")
# In much of Arizona projects are required to prove they have 100 years of h20.

# Graphing time:



water_used2 = [280, 290, 260,265, 290, 270, 325, 345,310, 325, 310, 325, 350,
              315, 314, 325, 335, 340, 345, 315, 318, 290, 310, 320, 310, 300,
              290]
population_of_Phoenix_overtime =[1.15, 1.2, 1, 1.2, 1.23, 1.24, 3.15, 3.30,
                                 3.10, 3.4,3.48, 3.50, 3.3,3.3,3.4, 3.25, 3.23,
                                 2.98, 3.10, 3.15, 3.01, 3, 2.8, 3, 3.2, 3.1,
                                 2.8]

plt.xlabel(" 1990 - 2020. ")
plt.ylabel("Water used in Phoenix. ")
plt.title(" AZ water use:")
plt.scatter(population_of_Phoenix_overtime, water_used2, c = "black")
plt.figure()



answer_to_Phoenix = (len(population_of_Phoenix_overtime))

print(answer_to_Phoenix, "phoenix population")
```

```
30 years
30
length of date list
DatetimeIndex(['1990-12-31', '1991-12-31', '1992-12-31', '1993-12-31',
               '1994-12-31', '1995-12-31', '1996-12-31', '1997-12-31',
               '1998-12-31', '1999-12-31', '2000-12-31', '2001-12-31',
               '2002-12-31', '2003-12-31', '2004-12-31', '2005-12-31',
               '2006-12-31', '2007-12-31', '2008-12-31', '2009-12-31',
               '2010-12-31', '2011-12-31', '2012-12-31', '2013-12-31',
               '2014-12-31', '2015-12-31', '2016-12-31', '2017-12-31',
               '2018-12-31', '2019-12-31'],
              dtype='datetime64[ns]', freq='A-DEC')
These are the differences seen: [[  0 -10  20  15 -10  10 -45 -65 -30 -45 -30 -45 -70
  -65 -35 -38 -10 -30 -40 -30 -20 -10 -20 -30 -30]
 [  0   0  30  25   0  20 -35 -55 -20 -35 -20 -35 -60 -25 -24 -35 -45 -50
  -55 -25 -28   0 -20 -30 -20 -10   0 -10 -20 -20]
 [  0   0   0  -5 -30 -10 -65 -85 -50 -65 -50 -65 -90 -55 -54 -65 -75 -80
  -85 -55 -58 -30 -50 -60 -50 -40 -30 -40 -50 -50]
 [  0   0   0   0 -25  -5 -60 -80 -45 -60 -45 -60 -85 -50 -49 -60 -70 -75
  -80 -50 -53 -25 -45 -55 -45 -35 -25 -35 -45 -45]
 [  0   0   0   0   0  20 -35 -55 -20 -35 -20 -35 -60 -25 -24 -35 -45 -50
  -55 -25 -28   0 -20 -30 -20 -10   0 -10 -20 -20]
 [  0   0   0   0   0   0 -55 -75 -40 -55 -40 -55 -80 -45 -44 -55 -65 -70
  -75 -45 -48 -20 -40 -50 -40 -30 -20 -30 -40 -40]
 [  0   0   0   0   0   0   0 -20  15   0  15   0 -25  10  11   0 -10 -15
  -20  10   7  35  15   5  15  25  35  25  15  15]
 [  0   0   0   0   0   0   0   0  35  20  35  20  -5  30  31  20  10   5
    0  30  27  55  35  25  35  45  55  45  35  35]
 [  0   0   0   0   0   0   0   0   0 -15   0 -15 -40  -5  -4 -15 -25 -30
  -35  -5  -8  20   0 -10   0  10  20  10   0   0]
 [  0   0   0   0   0   0   0   0   0   0  15   0 -25  10  11   0 -10 -15
  -20  10   7  35  15   5  15  25  35  25  15  15]
 [  0   0   0   0   0   0   0   0   0   0   0 -15 -40  -5  -4 -15 -25 -30
  -35  -5  -8  20   0 -10   0  10  20  10   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0 -25  10  11   0 -10 -15
  -20  10   7  35  15   5  15  25  35  25  15  15]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0  35  36  25  15  10
    5  35  32  60  40  30  40  50  60  50  40  40]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   1 -10 -20 -25
  -30   0  -3  25   5  -5   5  15  25  15   5   5]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 -11 -21 -26
  -31  -1  -4  24   4  -6   4  14  24  14   4   4]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 -10 -15
  -20  10   7  35  15   5  15  25  35  25  15  15]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  -5
  -10  20  17  45  25  15  25  35  45  35  25  25]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   -5  25  22  50  30  20  30  40  50  40  30  30]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0  30  27  55  35  25  35  45  55  45  35  35]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0  -3  25   5  -5   5  15  25  15   5   5]
```

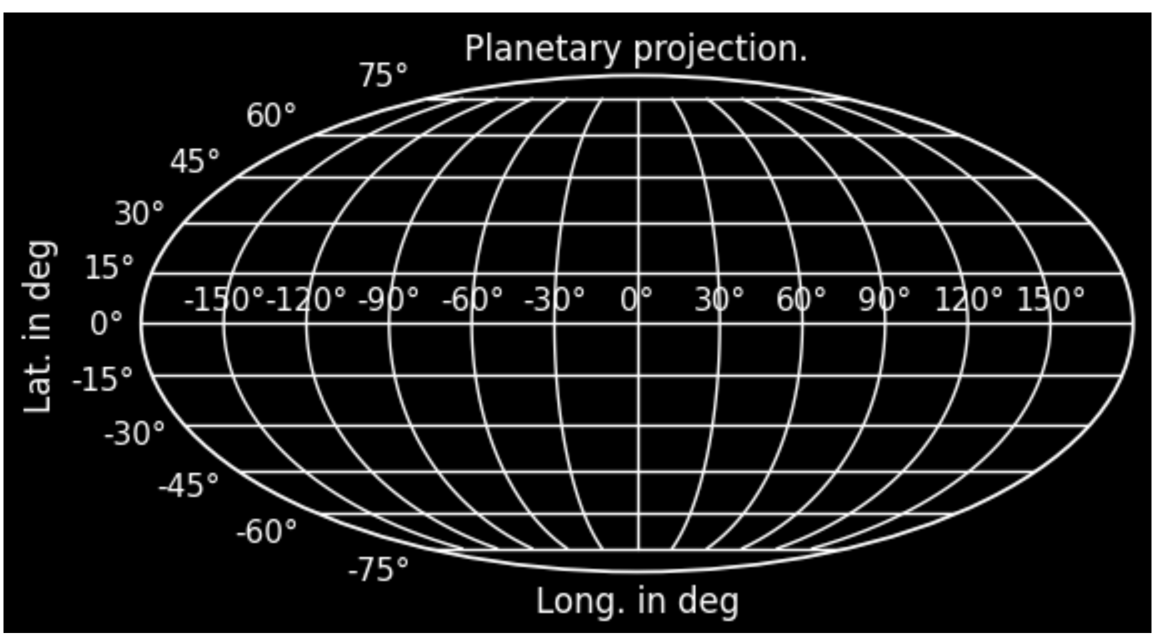
```
   0   0  -5  25   5  -5   5  15  25  15   5   5]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0  28   8  -2   8  18  28  18   8   8]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0 -20 -30 -20 -10   0 -10 -20 -20]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0 -10   0  10  20  10   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0  10  20  30  20  10  10]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0  10  20  10   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0  10   0 -10 -10]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0 -10 -20 -20]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0 -10 -10]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0]] .
These are the average differences seen: -7.935802469135803 .
27 phoenix population
```



```
#  Mapping the distance between stars - Next up

# import relevant libraries

import matplotlib.pyplot as plt
import os, sys
import pandas as pd
import shapely.geometry as sgeom
import numpy as np
from scipy.spatial import minkowski_distance
from math import sqrt
from scipy.spatial.distance import cityblock

# Algorithms needed for the project:
# minkowski_distance; # distance formula; # k-means

# Focus on two stars


# Manhattan Distance; Minkowski Distance(import from scipy);

# The location of the two stars seen below is expressed as a vector of two numbers
# North Star
one = 430 # light years
North_star = [39.761223,-75.719101] # longitude, latitude
# Southern Cross
two = 88.6 # light years
Southern_cross = [37.8184, 144.9525] # longitude, latitude

# Find the distance between these two areas:
distance_between_stars = [] # Euclidean distance
distance1_setup = np.sqrt((37.8184 - 39.761223)**2 + ((144.9525 - (-75.719101)))**2 )
print("The distance between the two stars is", distance1_setup,"light years away.", "\n")

# Defined the vectors earlier as North Star and Southern Cross
cityblock_formula_concept = cityblock(North_star, Southern_cross)
manhattan_distance = [cityblock_formula_concept]
print(cityblock_formula_concept, manhattan_distance)
if cityblock_formula_concept == manhattan_distance:
  print(True,"..." "  It's all good;" " the math checks out. ")
print(" The two stars are", manhattan_distance, "light years away. ")
# Use and implement the above two algorithms and report their results below.


# Difference between formulas results:

total_results = 222.61442399999999 - 220.68015327417308
print("""The difference between using
```

```
the euclidean vs manhatten formulas is a total of"
""",total_results, """light years""")
# Chart the stars on the graph.
"""
North_star = ( 39.761223,-75.719101) # longitude, latitude
# Southern Cross
two = 88.6 # light years
Southern_cross = (37.8184, 144.9525) # longitude, latitude

"""
print("\n")
# Fix the background in order to improve contrast for readability.
plt.style.use('dark_background')
plt.figure()
plt.subplot(projection = "mollweide")
plt.title(" Planetary projection. ")
plt.grid("True")
# Add labels for longitude and lattitude.
plt.xlabel('Long. in deg')
plt.ylabel('Lat. in deg')
plt.savefig('empty mollweide')
plt.show()
```

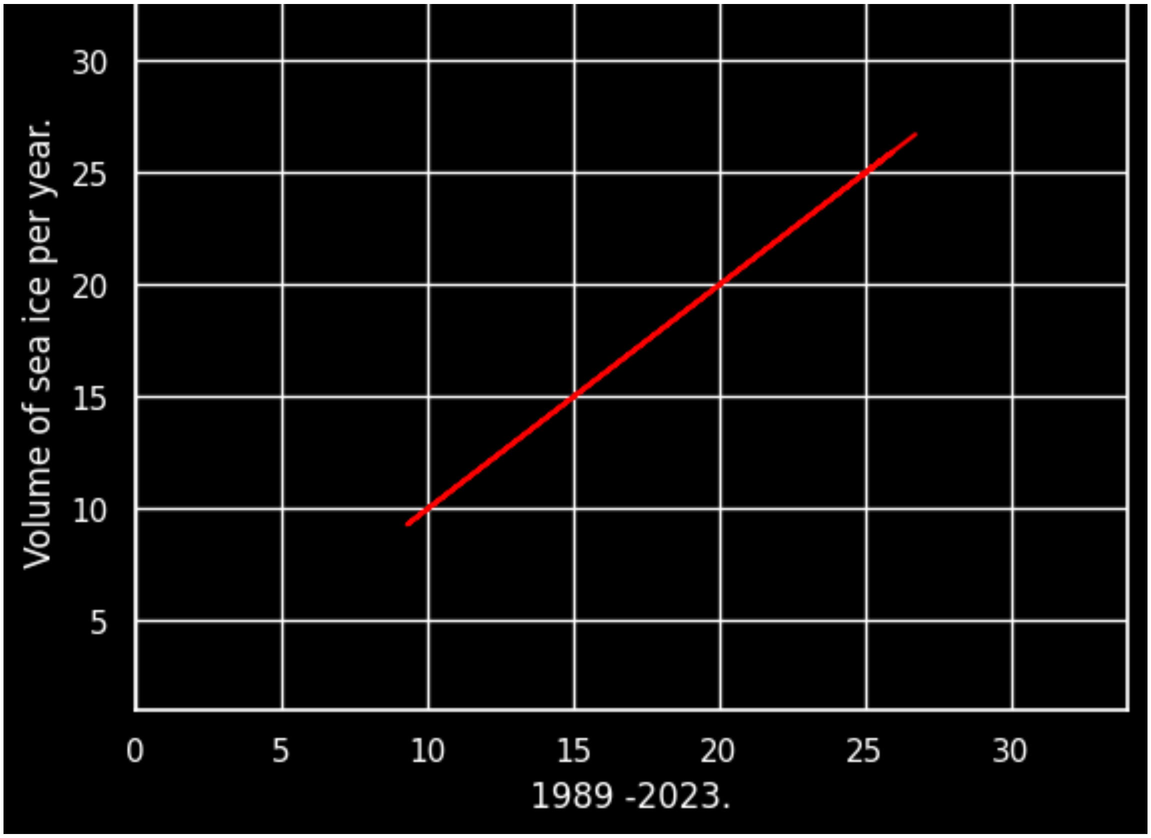
```
The distance between the two stars is 220.68015327417308 light years away.

222.61442399999999 [222.61442399999999]
True ...  It's all good; the math checks out.
 The two stars are [222.61442399999999] light years away.
The difference between using
the euclidean vs manhatten formulas is a total of"
 1.9342707258269058 light years
```



```
# 1st Project: Data Analysis of Sea Ice Changes over time:
"""
Calculating sea ice goes back to 700 AD which was documented by the Vikings.
Air temperature records go  back to the 1880s which was done recorded in
11 locations. This goes back to 1933. Other notable research includes the 1909
expedition with Peary.(earth observatory Nasa)

Ocean water looks different than sea ice due to different microwaves emitted.
This can be seen on satellites as well. Artic ice is susceptible to
oscillation. This is due to the lack of landmass.
"""


# Import relevant libraries:

import pandas as pd
import matplotlib as pyplot
import numpy as np
from datetime import date, timedelta
from datetime import datetime
# Actual code:

# Compare sea ice changes over a 30 year period; graph the results

# Make custom data set as a list.

# Take the average of difference between all the years, make a formula to calculate future

# Global sea ice data set

Global_sea_ice = [26.72, 24.69, 25.54, 25.90, 24.94, 25.38, 25.65, 24.67,
                  25.17, 25.03, 25.82, 24.72, 25.18, 24.96, 25.02, 25.58, 24.73,
                  25.95, 24.86, 25.02, 25.43, 25.56, 24.88, 23.96, 25.59, 9.74,
                  9.44, 9.37, 24.51, 25.27, 25.23, 25.00, 9.30, 24.22, 25.51,
                  25.72, 25.35, 23.65, 23.17, 23.67, 22.84, 23.86, 24.31,23.10,
                  21.98] # Covers 1989 to 20231989
Global_sea_ice2 = np.array([26.72, 24.69, 25.54, 25.90, 24.94, 25.38, 25.65, 24.67,
                  25.17, 25.03, 25.82, 24.72, 25.18, 24.96, 25.02, 25.58, 24.73,
                  25.95, 24.86, 25.02, 25.43, 25.56, 24.88, 23.96, 25.59, 9.74,
                  9.44, 9.37, 24.51, 25.27, 25.23, 25.00, 9.30, 24.22])
Length_global_sea_ice_2 = len(Global_sea_ice2); print(Length_global_sea_ice_2)
dates = 45
print(Global_sea_ice[35])
start_date = datetime.strptime("1989", "%Y")
end_date = datetime.strptime("2023", "%Y")
Y = 'Y'
date_list = pd.date_range(start_date, end_date, freq=Y)
print(len(date_list)); print("length of date list")
print(date_list)
```

```
print("\n")
plt.title(" Sea Ice Volumes from 1989 to 2023.")
plt.xlim(0, 34 )
plt.ylim(1, 34)
plt.plot(Global_sea_ice2, Global_sea_ice2, c = "red")
plt.xlabel(" 1989 -2023. ")
plt.ylabel(" Volume of sea ice per year. ")
plt.show()
print(len(Global_sea_ice)) ; print("Global sea ice length is 45")
number_of_years_covered_in_data_set = (len(Global_sea_ice))
print(number_of_years_covered_in_data_set)

# Average difference between the years down below:
Global_sea_ice1 = np.array([26.72, 24.69, 25.54, 25.90, 24.94, 25.38, 25.65, 24.67,
                  25.17, 25.03, 25.82, 24.72, 25.18, 24.96, 25.02, 25.58, 24.73,
                  25.95, 24.86, 25.02, 25.43, 25.56, 24.88, 23.96, 25.59, 9.74,
                  9.44, 9.37, 24.51, 25.27, 25.23, 25.00, 9.30, 24.22, 25.51,
                  25.72, 25.35, 23.65, 23.17, 23.67, 22.84, 23.86, 24.31,23.10,
                  21.98]) # Covers 1989 to 2023

differences_seen = np.triu(Global_sea_ice1.reshape(len(Global_sea_ice1), -1)- Global_sea_ic
averages_seen = np.mean(differences_seen[differences_seen != 0])


print( "and the averages seen are", averages_seen)
# Multiply by 1.3600808897876642

This_year_2023 = 21.98
Next_year = 1.3600808897876642 * 21.98
print(Next_year)
```

```
34
25.72
34
length of date list
DatetimeIndex(['1989-12-31', '1990-12-31', '1991-12-31', '1992-12-31',
               '1993-12-31', '1994-12-31', '1995-12-31', '1996-12-31',
               '1997-12-31', '1998-12-31', '1999-12-31', '2000-12-31',
               '2001-12-31', '2002-12-31', '2003-12-31', '2004-12-31',
               '2005-12-31', '2006-12-31', '2007-12-31', '2008-12-31',
               '2009-12-31', '2010-12-31', '2011-12-31', '2012-12-31',
               '2013-12-31', '2014-12-31', '2015-12-31', '2016-12-31',
               '2017-12-31', '2018-12-31', '2019-12-31', '2020-12-31',
               '2021-12-31', '2022-12-31'],
              dtype='datetime64[ns]', freq='A-DEC')
```




```
45
Global sea ice length is 45
45
and the averages seen are 1.3600808897876642
29.89457795753286
```

```
""" Project number 2(redid it later on);
project is meant to practice and demonstrate for fulltime roles:

# Focus on Los Angeles, California for efficiencies sake:




# Record rainfall over the last 11 years

# Covers years 2012 - 2022

# import relevant libraries:

import matplotlib.pyplot as plt
import numpy as np



print("Years covers the amount of rainfall in each year. 2012 - 2022")

y1 = 5.85
y2 = 6.08
```

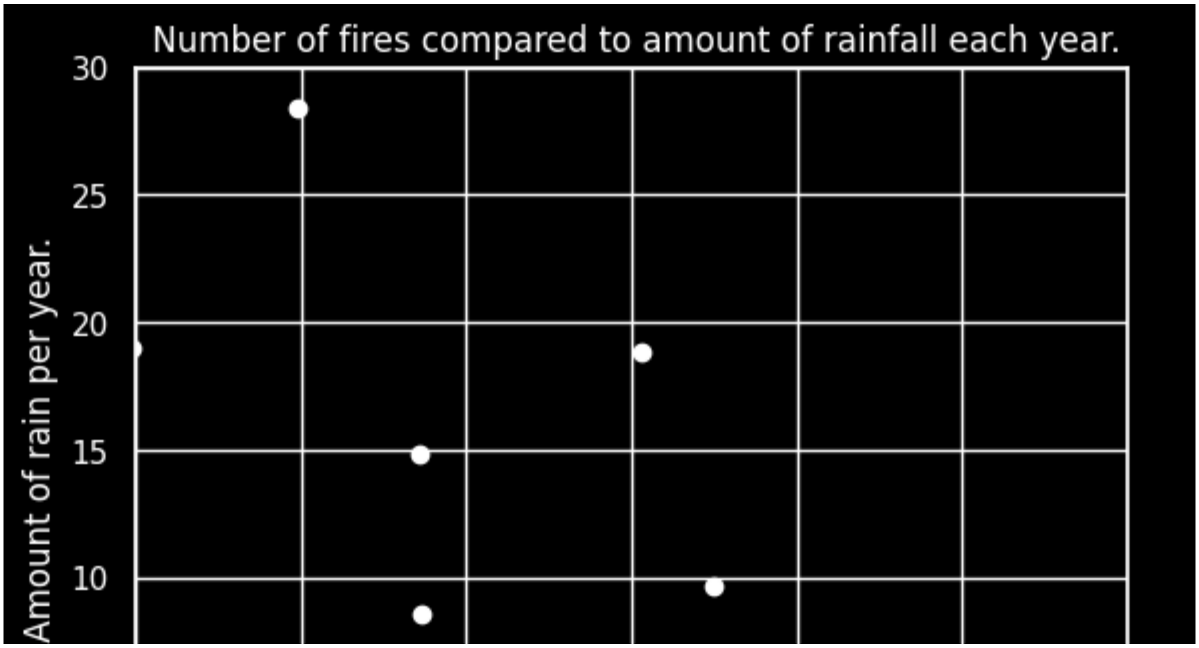
```
y3 = 8.52
y4 = 9.65
y5 = 19.00
y6 = 4.79
y7 = 18.82
y8 = 14.86
y9 = 5.82
y10 = 5.82
y11 = 28.40

print("Seasons refers to the part about wildfires; stands for wildfire season")

# d stands for distance burned.

s1 = [7950]
s1d = [869599]
s2 = [9907]
s2d = [601635]
s3 = [7865]
s3d = [625540]
s4 = [8745]
s4d = [893362]
s5 = [6986]
s5d = [669534]
s6 = [9560]
s6d = [1548429]
s7 = [8527]
s7d = [1975086]
s8 = [7860]
s8d = [259823]
s9 = [9639]
s9d = [4397809]
s10 = [8835]
s10d = [2568948]
s11 = [7490]
s11d = [362455]

# Needed data:

seasons = s1, s2, s3, s4, s5, s6, s7, s8, s9, s10, s11
years = y1, y2, y3, y4, y5,y6, y7, y8, y9, y10, y11
data = years

# Record the number of fires graph it using matplot.
# Make a chart and label the data.
# x = number of fires; y = rain, and z = squarefootage burned.

x = (7950, 9907, 7865, 8745, 6986, 9560, 8527, 7860, 9639, 8835,
7490) # number of fires
"""
```

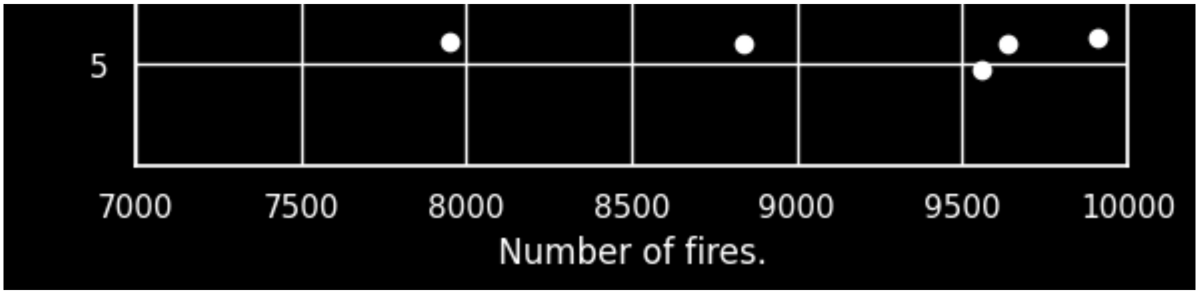
```
"""r1 = x[0]; r2 = x[1]; r3 = x[2]; r4 = x[3]; r5 = x[4]; r6 = x[5]; r7 = x[6]
r8 = x[7]; r9 = x[8]; r10 = x[9]; r11 = x[10
print(r1, r2, r3, r4, r5, r6, r7, r8, r9, r10)
years_needed = (r1, r2, r3, r4, r5, r6, r7, r8, r9, r10) ]
5.82, 5.82, 28.40) # rainfall
"""
"""plt.plot(years_needed,y);
plt.xlabel(" Number of fires: ")
plt.ylabel(" Amount of rainfall per year: ")
#plt.show()

plt.plot(x,y, label = " Number of fires compared to amount of rainfall each year. ")


# Measure the correlation between rainfall and the number of fires.
# Measure the correlation between rainfall and squarefootage burned.


"""
```

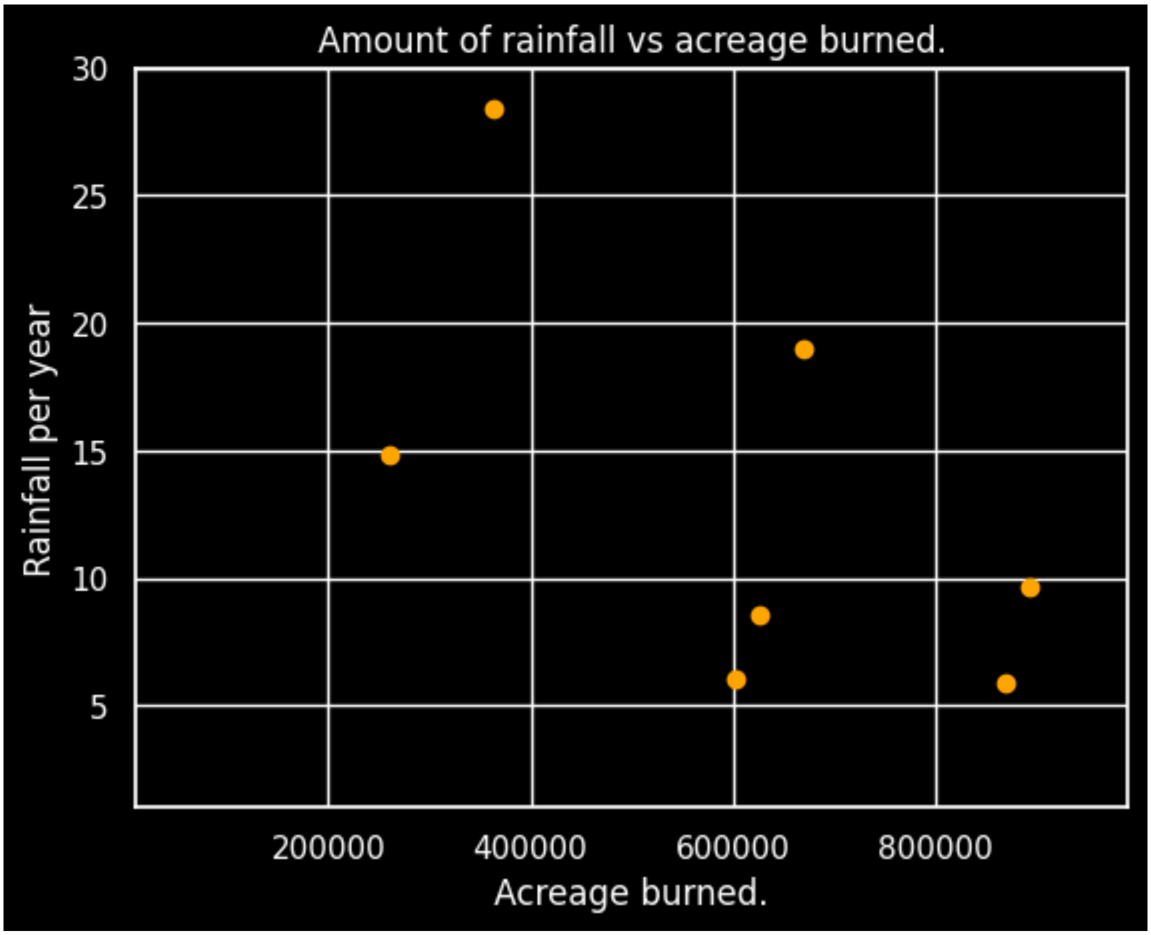
```
'plt.plot(years_needed,y);\nplt.xlabel(" Number of fires: ")\nplt.ylabel(" Amount of
rainfall per year: ")\n#plt.show()\n\nplt.plot(x,y, label = " Number of fires compar
```

```
# Project number 2 for fulltime roles: Predicting the occurrences of wildfires based on ra

# Focus on Los Angeles, California for efficiencies sake:




# Record rainfall over the last 11 years

# Covers years 2012 - 2022

# import relevant libraries:

import matplotlib.pyplot as plt
import numpy as np




# d stands for distance burned.




# Needed data:
```

```
# Record the number of fires graph it using matplot.
# Make a chart and label the data.
# x = number of fires; y = rain, and z = squarefootage burned.

x = (7950, 9907, 7865, 8745, 6986, 9560, 8527, 7860, 9639, 8835,
7490) # number of fires
y = (5.85, 6.08, 8.52, 9.65, 19.00, 4.79, 18.82, 14.86,
5.82, 5.82, 28.40) # rainfall


plt.title(" Number of fires compared to amount of rainfall each year.")
plt.xlim(7000,10000)
plt.ylim(1, 30)
plt.scatter(x,y,c ="white")
plt.xlabel(" Number of fires. ")
plt.ylabel(" Amount of rain per year. ")
# Measure the correlation between rainfall and the number of fires.
# Measure the correlation between rainfall and squarefootage burned.
plt.show()

print("\n")
a = (869599, 601635, 625540, 893362, 669534,1548429,1975086,259823,4397809,2568948,362455)
b = (5.85, 6.08, 8.52, 9.65, 19.00, 4.79, 18.82, 14.86,
5.82, 5.82, 28.40)
plt.title(" Amount of rainfall vs acreage burned. ")
plt.xlim(10000, 990000)
plt.ylim(1,30)
plt.scatter(a,b, c = "orange")
plt.xlabel(" Acreage burned. "); "\n"
plt.ylabel(" Rainfall per year ")
plt.show()
```




5
7000
7500
8000
8500
9000
9500
10000
Number of fires.


Amount of rainfall vs acreage burned.
30
25
20
15
10
5
Rainfall per year
200000
400000
600000
800000
Acreage burned.