The following exercises are designed to help you get familiar with the basics of numpy, pandas, and matplotlib packages. These exercises are extension of material we discuss during lecture. You can access the lecture .ipynb files to get familiar with the basics.

In order to complete this assignment, first you need to import proper packages and read data files into proper data structures.

The **deadline** for submitting this assignment is **Friday, Feb 19**. Submit your solution on **Gradescope**.

Exercises

Numpy

Check out the Numpy-Basics file and read the documentation at http://www.numpy.org (http://www.numpy.org) Now it's your turn to do some practice with NumPy:

1. Create a 1-dimensional NumPy array of 100 random integers (1 pt)

1. Find and print all the summary statistics (mean, std, median, max, min, ...)
2. Compare the time for finding the max using python built-in functions and NumPy corresponding function
3. Create a new array that is the base-2 logarithm of your array

In [204]:
```
# 1.  Find and print all the summary statistics (mean, std, median, max, mi
import numpy as np
randint = np.random.randint(0,11,size=100)
print("Array", randint)
print("Sum",np.sum(randint))
print("Mean",np.mean(randint))
print("Standard Deviation",np.std(randint))
print("Median",np.median(randint))
print("Maximum",np.max(randint))
print("Minimum",np.min(randint))
```

```
Array [ 8  4 10  0  9  9  6  1  9  2  8  4  5  4  7  2  0  7  6 10  0  8
3  8
  6  1  6  4  4  7  0  0  7 10  0  6  1  0  8  7  7  9  4  9  9  6  1  8
  9  5  9  7  1 10  5  0  1  6  4  0  2  9 10  0  4  9  5  7  0  3  6  2
  8  8  3  9  9  8  2  2  8  5  3  0  3  9  7  6  7  9  7  7  9  5  5  2
  8  6  5  9]
Sum 533
Mean 5.33
Standard Deviation 3.184509381364734
Median 6.0
Maximum 10
Minimum 0
```

In [205]:
```
# 2.  Compare the time for finding the max using python built-in functions
print("Time Compared:")
%timeit max(randint)
%timeit np.max(randint)
```

```
Time Compared:
8.39 µs ± 54.5 ns per loop (mean ± std. dev. of 7 runs, 100000 loops eac
h)
2.73 µs ± 26.7 ns per loop (mean ± std. dev. of 7 runs, 100000 loops eac
h)
```

In [206]:
```
# 3. Create a new array that is the base-2 logarithm of your array
log = np.log2(randint)
print("Log Array is:",log)
```

```
Log Array is: [3.         2.         3.32192809       -inf 3.169925   3.1
69925
 2.5849625  0.         3.169925   1.         3.         2.
 2.32192809 2.         2.80735492 1.               -inf 2.80735492
 2.5849625  3.32192809       -inf 3.         1.5849625  3.
 2.5849625  0.         2.5849625  2.         2.         2.80735492
       -inf       -inf 2.80735492 3.32192809       -inf 2.5849625
 0.               -inf 3.         2.80735492 2.80735492 3.169925
 2.         3.169925   3.169925   2.5849625  0.         3.
 3.169925   2.32192809 3.169925   2.80735492 0.         3.32192809
 2.32192809       -inf 0.         2.5849625  2.               -inf
 1.         3.169925   3.32192809       -inf 2.         3.169925
 2.32192809 2.80735492       -inf 1.5849625  2.5849625  1.
 3.         3.         1.5849625  3.169925   3.169925   3.
 1.         1.         3.         2.32192809 1.5849625        -inf
 1.5849625  3.169925   2.80735492 2.5849625  2.80735492 3.169925
 2.80735492 2.80735492 3.169925   2.32192809 2.32192809 1.
 3.         2.5849625  2.32192809 3.169925  ]


<ipython-input-206-fa4393d3e0b9>:2: RuntimeWarning: divide by zero encoun
tered in log2
  log = np.log2(randint)
```

2. Create a 2-dimensional NumPy array of (3,4) random integers (mat1) (2 pts)

1. Create another 2-D array that is the square root of your original array (mat2)
2. Find how many values are greater than 20 using np.count_nonzero( ) function (you can also use np.sum( )
3. Perform a dot product between two 2-D arrays (mat3)
4. Find all the values that are less than 10 and greater than 30 in mat3

In [209]:
```
# 1. Create another 2-D array that is the square root of your original arra
import numpy as np
mat1 = np.random.randint(100,size=(3,4))
print("Normal Array", mat1)
mat2 = np.sqrt(mat1)
print("Square Root Array", mat2)
```

```
Normal Array [[74 43  5 99]
 [38 61 10  7]
 [35  3 19 13]]
Square Root Array [[8.60232527 6.55743852 2.23606798 9.94987437]
 [6.164414   7.81024968 3.16227766 2.64575131]
 [5.91607978 1.73205081 4.35889894 3.60555128]]
```

In [210]:
```
# 2. Find how many values are greater than 20 using np.count_nonzero(  ) fu
count = np.count_nonzero(mat1 > 20)
print("vales greater than 20 In mat1:",count)
```

```
vales greater than 20 In mat1: 6
```

In [211]:
```
# 3. Perform a dot product between two 2-D arrays (mat3)
mat2 = np.random.randint(100,size=(4,2))
mat3 = np.dot(mat1,mat2)
print("dot product is:",mat3)
```

```
dot product is: [[7902 9100]
 [5952 5975]
 [1478 5033]]
```

In [213]:
```
# 4. Find all the values that are less than 10 and greater than 30 in mat3
print("Values less than 10 and greater than 30 are as follows:")
print(mat3[mat3<10])
print(mat3[mat3>30])
```

```
Values less than 10 and greater than 30 are as follows:
[]
[7902 9100 5952 5975 1478 5033]
```

Pandas

Check out the pandas-basics file on Blackbaord. pandas documentation is very detailed and useful and contains a short tutorial (https://pandas.pydata.org/pandas-docs/stable/user_guide/10min.html). You can also check this Cheatsheet (https://github.com/pandas-dev/pandas/blob/master/doc/cheatsheet/Pandas_Cheat_Sheet.pdf)

Now it's your turn to do some exercises and get familiar with Python. First, you are going to answer some questions about the movies dataset we already explored:

3. Pandas basics with movies_by_year data (2 pts)

Download the "movies_by_year.csv" file from Blackboard *(Weekly content : Week 6)*

1. Get a quick statistical profile of Total_Gross column
2. Get the summary statistics (mean,max,min,std) of Total_Gross for movies that were highest grossing movie of the year during 1995 - 2005
3. Do the same thing for movies during 2005-2015 period
4. What are some insights you gain from this comparison?

In [182]:
```
# 1. Get a quick statistical profile of Total_Gross column
import pandas as pd
df = pd.read_csv("movies_by_year.csv")
df
print("Total Gross Column:",df['Total Gross'])
```

```
Total Gross Column: 0     11128.5
1     10360.8
2     10923.6
3     10837.4
4     10174.3
5     10565.6
6     10595.5
7      9630.7
8      9663.8
9      9209.5
10     8840.5
11     9380.5
12     9239.7
13     9155.0
14     8412.5
15     7661.0
16     7448.0
17     6949.0
18     6365.9
19     5911.5
20     5493.5
21     5396.2
22     5154.2
23     4871.0
24     4803.2
25     5021.8
26     5033.4
27     4458.4
28     4252.9
29     3778.0
30     3749.2
31     4031.0
32     3766.0
33     3453.0
34     2966.0
35     2749.0
Name: Total Gross, dtype: float64
```

In [188]:
```
# 2. Get the summary statistics (mean,max,min,std) of Total_Gross for movie

df_1995=df.loc[df['Year']>=1995]
df_2005=df.loc[df['Year']<=2005]
op = pd.merge(df_1995,df_2005)
print(op)
print("Mean:", op['Total Gross'].mean())
print("Max:", op['Total Gross'].max())
print("Min", op['Total Gross'].min())
print("Standard Deviation:", op['Total Gross'].std())
```

```
    Year  Total Gross  Number of Movies                        #1 Movie
0   2005       8840.5               547              Revenge of the Sith
1   2004       9380.5               551                        Shrek 2
2   2003       9239.7               506               Return of the King
3   2002       9155.0               479                        Spider-Man
4   2001       8412.5               482  Harry Potter / Sorcerer's Stone
5   2000       7661.0               478                       The Grinch
6   1999       7448.0               461               The Phantom Menace
7   1998       6949.0               509              Saving Private Ryan
8   1997       6365.9               510                          Titanic
9   1996       5911.5               471                Independence Day
10  1995       5493.5               411                        Toy Story
Mean: 7714.2818181818175
Max: 9380.5
Min 5493.5
Standard Deviation: 1399.791322889367
```

In [189]:
```
# 3. Do the same thing for movies during 2005-2015 period

df2005=df.loc[df['Year']>=2005]
df2015=df.loc[df['Year']<=2015]
op1 = pd.merge(df2005,df2015)
print(op1)
print("Mean:", op1['Total Gross'].mean())
print("Max:", op1['Total Gross'].max())
print("Min", op1['Total Gross'].min())
print("Standard Deviation:", op1['Total Gross'].std())
```

```
    Year  Total Gross  Number of Movies                            #1 Mo
vie
0   2015      11128.5               702         Star Wars: The Force Awak
ens
1   2014      10360.8               702                      American Sni
per
2   2013      10923.6               688                        Catching F
ire
3   2012      10837.4               667                         The Aveng
ers
4   2011      10174.3               602  Harry Potter / Deathly Hallows
(P2)
5   2010      10565.6               536                         Toy Stor
y 3
6   2009      10595.5               521                               Ava
tar
7   2008       9630.7               608                      The Dark Kni
ght
8   2007       9663.8               631                         Spider-Ma
n 3
9   2006       9209.5               608                     Dead Man's Ch
est
10  2005       8840.5               547                   Revenge of the S
ith
Mean: 10175.472727272729
Max: 11128.5
Min 8840.5
Standard Deviation: 744.5075339993667
```

In [190]:
```
# 4. What are some insights you gain from this comparison?

print("Insights")
print("1. we can infer that the mean for period 1995-2005 was less than tha
print("2. from standard deviation we can say that the period of 1995-2005 h
print("3. min for 1995-2005 vs 2005-2015 shows that there is a huge differe
```

```
Insights
1. we can infer that the mean for period 1995-2005 was less than that of
2005-2015 and that the later earned more as compred to the former
2. from standard deviation we can say that the period of 1995-2005 had a
greater SD as comapred to the 2005-2015 which indicates that the gross in
come was close by and that there was less scattering
3. min for 1995-2005 vs 2005-2015 shows that there is a huge difference s
imilarly for max also if seen the min of 2005-2015 is almost close to the
max of 1995-2005 which itself sows the difference
```

4. Pandas basics with risk dataset (2 pts)

Download the "Risk.csv" file from Blackboard *(Weekly content : Week 6)*

1. Find the minimum and maximum values of income
2. Find the mean income based on gender
3. Draw the histogram of income distribution based on gender

In [194]:
```
# 1. Find the minimum and maximum values of income
import pandas as pd
df = pd.read_csv("risk.csv")
print("Income:")
print(df['INCOME'])
print("\nIncome Minimum:",df['INCOME'].min())
print("Income Maximum:",df['INCOME'].max())
```

```
Income:
0       59944
1       59692
2       59508
3       59463
4       59393
        ...
4112    15035
4113    15032
4114    15020
4115    15018
4116    15005
Name: INCOME, Length: 4117, dtype: int64

Income Minimum: 15005
Income Maximum: 59944
```

In [87]:
```
# 2. Find the mean income based on gender
df_male = df['GENDER']=='m'
print("Male Mean Income",df[df_male]['INCOME'].mean())


df_female = df['GENDER']=='f'
print("Female Mean Income",df[df_female]['INCOME'].mean())
```

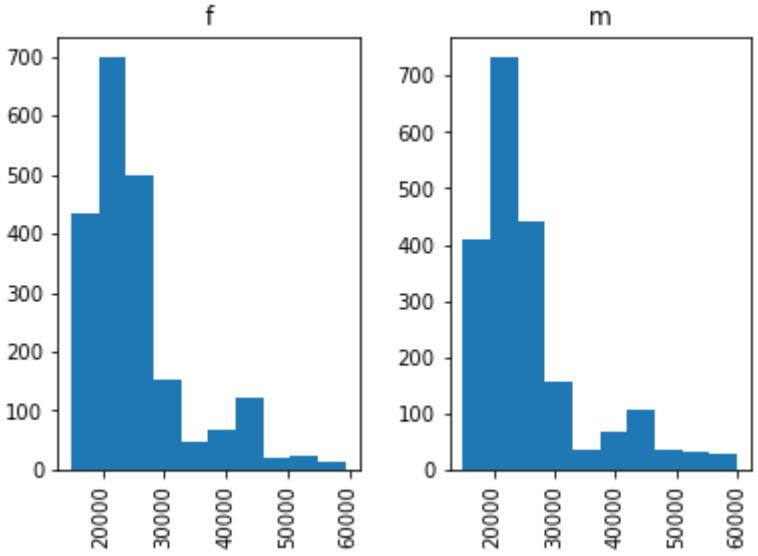
```
Male Mean Income 25794.089705882354
Female Mean Income 25370.1439576312
```

In [91]:
```
# 3. Draw the histogram of income distribution based on gender
df.hist(column ="INCOME", by ="GENDER")

print("x-axis:  INCOME")

print("y-axis:  FREQUENCY")
```

```
x-axis:  INCOME
y-axis:  FREQUENCY
```



5. EDA with famous baby name dataset (3 pts)

For downloading the data set go to [https://www.ssa.gov/oact/babynames/state/namesbystate.zip](https://www.ssa.gov/oact/babynames/state/namesbystate.zip%5D) or use this box link (https://app.box.com/s/i52qu3mgmpwvikiggjmwwydgycw4d4bt) to access all the text files.

1. How many men and women were counted?
2. Count unique boy/girl names.
3. Find top 10 popular names between 2000 - 2015 and plot their trend
4. Create a new column titled namelength (you can use str.len()). Then plot the averge length of names

In [196]:
```
# combining all files
filenames = ['AK.TXT','HI.TXT','MI.TXT','NV.TXT','TX.TXT','AL.TXT','IA.TXT'
with open('/Users/aayushmakharia/Downloads/result.txt', 'w') as outfile:
    for fname in filenames:
        with open(fname) as infile:
            for line in infile:
                outfile.write(line)

df = pd.read_csv("result.txt")
df.columns=['State','Gender','Birth Year','Name','Count']
print(df)
```

```
        State Gender  Birth Year      Name  Count
0          AK      F        1910     Annie     12
1          AK      F        1910      Anna     10
2          AK      F        1910  Margaret      8
3          AK      F        1910     Helen      7
4          AK      F        1910     Elsie      6
...       ...    ...         ...       ...    ...
6122884    TN      M        2019      Yael      5
6122885    TN      M        2019   Zachery      5
6122886    TN      M        2019    Zaidyn      5
6122887    TN      M        2019    Zavier      5
6122888    TN      M        2019      Zayd      5

[6122889 rows x 5 columns]
```

In [197]:
```
# 1. How many men and women were counted?
import matplotlib as plt
df.groupby('Gender')['Count'].sum()
```

Out[197]:
```
Gender
F    150868980
M    163223872
Name: Count, dtype: int64
```

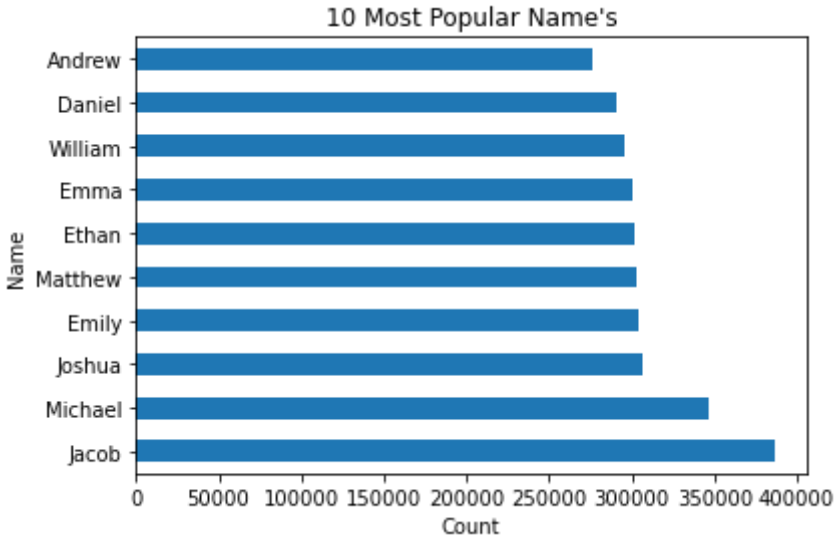
In [198]:
```
# 2. Count unique boy/girl names.
print("Unique Male and Female name are:\n",df.groupby('Gender')['Name'].nun
```

```
Unique Male and Female name are:
 Gender
F    21026
M    13926
Name: Name, dtype: int64
```

In [146]:
```
# 3. Find top 10 popular names between 2000 - 2015 and plot their trend
from matplotlib import pyplot as plt
n1 = df.loc[df['Birth Year'] >= 2000]
n2 = df.loc[df['Birth Year'] <= 2015]
name = pd.merge(n1,n2)
pop_name = name.groupby('Name')['Count'].sum().nlargest(10)
print("10 Popular Name's are:\n",pop_name)
plt.xlabel("Count")
plt.ylabel("Name")
plt.title("10 Most Popular Name's")
pop_name.plot.barh()
```

```
10 Popular Name's are:
 Name
Jacob      386717
Michael    346741
Joshua     306586
Emily      303612
Matthew    302991
Ethan      301361
Emma       300853
William    295115
Daniel     291198
Andrew     275715
Name: Count, dtype: int64
```

Out[146]:
```
<AxesSubplot:title={'center':"10 Most Popular Name's"}, xlabel='Count', y
label='Name'>
```



In [199]:
```
# 4. Create a new column titled namelength ( you can use str.len()). Then p
df['NameLength'] = df['Name'].str.len()
df
```

Out[199]:

	State	Gender	Birth Year	Name	Count	NameLength
0	AK	F	1910	Annie	12	5
1	AK	F	1910	Anna	10	4
2	AK	F	1910	Margaret	8	8
3	AK	F	1910	Helen	7	5
4	AK	F	1910	Elsie	6	5
...	...	...	...	...	...	...
6122884	TN	M	2019	Yael	5	4
6122885	TN	M	2019	Zachery	5	7
6122886	TN	M	2019	Zaidyn	5	6
6122887	TN	M	2019	Zavier	5	6
6122888	TN	M	2019	Zayd	5	4

6122889 rows × 6 columns

In [163]:
```
# plotting avergae name of length
df_Avg_Year = df.groupby('Birth Year')['NameLength'].mean()
plt.title('Average Name Length')
plt.xlabel('Year')
plt.ylabel('Average')
df_Avg_Year.plot()
```

Out[163]:
```
<AxesSubplot:title={'center':'Average Name Length'}, xlabel='Birth Year',
ylabel='Average'>
```