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. This is part of your "class activity". The **deadline** for submitting this assignment is **Friday, Oct 15**. Submit your solution on **Gradescope**.

Exercises

Numpy

Check out the Numpy-Basics file and read the documentation at 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 [1]:
```
# 1. All the summary statistics (mean, std, median, max, min, ...)
import numpy as np
from scipy import stats
import time

arr = np.random.randint(100, size=100)

print(arr)
```

```
[ 8 72 58 33  7 62 27 54 14 19 77 69 45 92 15  3  6 15 56 86 77 90 85 46
 79 30 87 40 21 75 49 42 27  6 42 86 99 16 81 25 96  6 13  8 46 48 26 78
 23 53 60 26 37 79 57 58 83 84  4 18 22 42  6 18 59 51 54 60 39 43 54 22
 10 89 58 51 87 89  8 59  1 62 12 26 70 16 32  5 10 37 86 45 31 59 25 31
 56 85 46 46]
```

In [2]:
```
print("Max : ", np.max(arr))
print("Min : ", np.min(arr))
print("Mean : ", np.mean(arr))
print("Median : ", np.median(arr))
print("Variance : ", np.var(arr))

print("Standard deviation : ", arr.std())
print("Percentile : ", np.percentile(arr, 50))
```

```
Max :  99
Min :  1
Mean :  45.26
Median :  45.5
Variance :  763.9924000000002
Standard deviation :  27.640412442653606
Percentile :  45.5
```

In [3]:
```
# Statistics using scipy
print("Statistic:  ", stats.describe(arr))
```

```
Statistic:   DescribeResult(nobs=100, minmax=(1, 99), mean=45.26, variance=771.70949494949
51, skewness=0.16256844147855076, kurtosis=-1.150104274504278)
```

In [4]:
```
# 2.  Comparing the time for finding the max using python built-in functions and NumPy co
start = time.time()

max_value = arr.max()

print("Took time to find max using python build in function: ", (time.time() - start) * 1(

start = time.time()

max_value = np.max(arr)
print("Took time to find max using numpy: ", (time.time()-start)*1000)
```

```
Took time to find max using python build in function:  0.14901161193847656
Took time to find max using numpy:  0.1862049102783203
```

In [5]:
```
#3. Create a new array that is the base-2 logarithm of your array
np.seterr(divide = 'ignore')
base_2_arr = np.log2(arr)
print(base_2_arr)
```

```
[3.         6.169925   5.857981   5.04439412 2.80735492 5.95419631
 4.7548875  5.7548875  3.80735492 4.24792751 6.26678654 6.10852446
 5.4918531  6.52356196 3.9068906  1.5849625  2.5849625  3.9068906
 5.80735492 6.42626475 6.26678654 6.4918531  6.40939094 5.52356196
 6.30378075 4.9068906  6.4429435  5.32192809 4.39231742 6.22881869
 5.61470984 5.39231742 4.7548875  2.5849625  5.39231742 6.42626475
 6.62935662 4.         6.33985    4.64385619 6.5849625  2.5849625
 3.70043972 3.         5.52356196 5.5849625  4.70043972 6.28540222
 4.52356196 5.72792045 5.9068906  4.70043972 5.20945337 6.30378075
 5.83289001 5.857981   6.37503943 6.39231742 2.         4.169925
 4.45943162 5.39231742 2.5849625  4.169925   5.88264305 5.67242534
 5.7548875  5.9068906  5.28540222 5.42626475 5.7548875  4.45943162
 3.32192809 6.47573343 5.857981   5.67242534 6.4429435  6.47573343
 3.         5.88264305 0.         5.95419631 3.5849625  4.70043972
 6.12928302 4.         5.         2.32192809 3.32192809 5.20945337
 6.42626475 5.4918531  4.95419631 5.88264305 4.64385619 4.95419631
 5.80735492 6.40939094 5.52356196 5.52356196]
```

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 [6]:
```
# Your code here
mat1 = np.random.randint(450, size=(3, 4))
mat1
```

Out[6]:
```
array([[399, 341, 342, 144],
       [347, 449,  27, 227],
       [136, 129, 366, 182]])
```

In [7]:
```
mat2 = np.sqrt(mat1)
mat2
```

Out[7]:
```
array([[19.97498436, 18.46618531, 18.49324201, 12.        ],
       [18.62793601, 21.1896201 ,  5.19615242, 15.06651917],
       [11.66190379, 11.35781669, 19.13112647, 13.49073756]])
```

In [8]:
```
greater = (mat2 > 20).sum()
greater
```

Out[8]:
```
1
```

In [9]:
```
mat3 = np.dot(mat1, mat2.T)
mat3
```

Out[9]:
```
array([[22319.67671647, 18604.86981183, 17011.62656567],
       [18445.95431094, 19538.42918838, 12725.27815109],
       [14051.26235293,  9910.7585666 , 12508.48379304]])
```

In [10]:
```
index = np.logical_or(mat3 < 10, mat3 > 30)
mat3[index]
```

Out[10]:
```
array([22319.67671647, 18604.86981183, 17011.62656567, 18445.95431094,
       19538.42918838, 12725.27815109, 14051.26235293,  9910.7585666 ,
       12508.48379304])
```

Pandas

Check out the pandas-basics file on Blackbaord. pandas documentation is very detailed and useful and contains a short tutorial. You can also check this Cheatsheet

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 [11]:
```
import pandas as pd
data = pd.read_csv("movies_by_year.csv")

data
```

Out[11]:

	Year	Total Gross	Number of Movies	#1 Movie
0	2015	11128.5	702	Star Wars: The Force Awakens
1	2014	10360.8	702	American Sniper
2	2013	10923.6	688	Catching Fire
3	2012	10837.4	667	The Avengers
4	2011	10174.3	602	Harry Potter / Deathly Hallows (P2)

	Year	Total Gross	Number of Movies	#1 Movie
5	2010	10565.6	536	Toy Story 3
6	2009	10595.5	521	Avatar
7	2008	9630.7	608	The Dark Knight
8	2007	9663.8	631	Spider-Man 3
9	2006	9209.5	608	Dead Man's Chest
10	2005	8840.5	547	Revenge of the Sith
11	2004	9380.5	551	Shrek 2
12	2003	9239.7	506	Return of the King
13	2002	9155.0	479	Spider-Man
14	2001	8412.5	482	Harry Potter / Sorcerer's Stone
15	2000	7661.0	478	The Grinch
16	1999	7448.0	461	The Phantom Menace
17	1998	6949.0	509	Saving Private Ryan
18	1997	6365.9	510	Titanic
19	1996	5911.5	471	Independence Day
20	1995	5493.5	411	Toy Story
21	1994	5396.2	453	Forrest Gump
22	1993	5154.2	462	Jurassic Park
23	1992	4871.0	480	Aladdin
24	1991	4803.2	458	Terminator 2
25	1990	5021.8	410	Home Alone
26	1989	5033.4	502	Batman
27	1988	4458.4	510	Rain Man
28	1987	4252.9	509	Three Men and a Baby
29	1986	3778.0	451	Top Gun
30	1985	3749.2	470	Back to the Future
31	1984	4031.0	536	Beverly Hills Cop
32	1983	3766.0	495	Return of the Jedi
33	1982	3453.0	428	E.T.
34	1981	2966.0	173	Raiders / Lost Ark
35	1980	2749.0	161	The Empire Strikes Back

In [12]:
```
# Get the summary statistics (mean,max,min,std) of Total_Gross for movies that were highes
movies_btween_1995_2005 = data.loc[data['Year'].between(1995, 2005)]
stats_1995_2005 = movies_btween_1995_2005['Total Gross'].describe()
stats_1995_2005
```

Out[12]:
```
count       11.000000
mean      7714.281818
std       1399.791323
min       5493.500000
```

```
25%       6657.450000
50%       7661.000000
75%       8997.750000
max       9380.500000
Name: Total Gross, dtype: float64
```

In [13]:

```
# Movies during 2005-2015 period
movies_btween_2005_2015 = data.loc[data['Year'].between(2005, 2015)]
stats_2005_2015 = movies_btween_2005_2015['Total Gross'].describe()
stats_2005_2015
```

Out[13]:

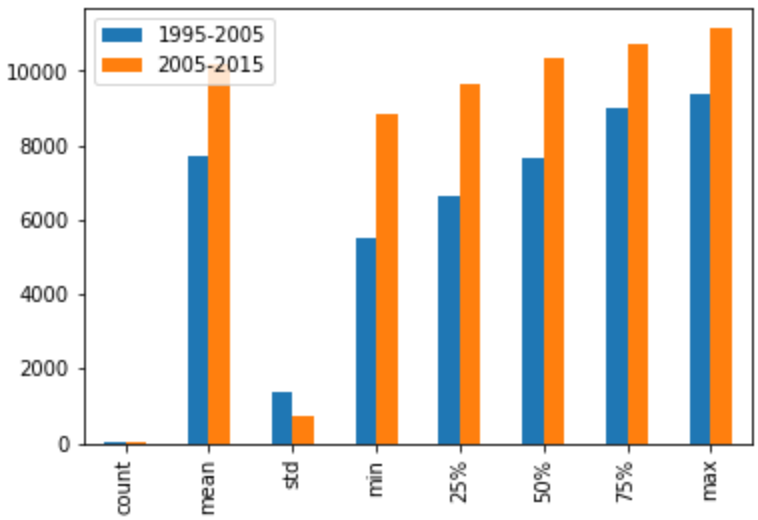
```
count       11.000000
mean     10175.472727
std        744.507534
min       8840.500000
25%       9647.250000
50%      10360.800000
75%      10716.450000
max      11128.500000
Name: Total Gross, dtype: float64
```

In [14]:

```
plotdata = pd.DataFrame({
    "1995-2005":stats_1995_2005,
    "2005-2015":stats_2005_2015
    },
    index=["count", "mean", "std", "min", "25%", "50%", "75%", "max"]
)
plotdata.plot(kind="bar")
```

Out[14]:

```
<AxesSubplot:>
```



As we can see in the comparision the mean increased but the standard deviation decreased, thus there was not much difference in the gross total

4. Pandas basics with risk dataset (2 pts)

Download the "Risk.csv" file from Blackboard *(Weekly content : Week 8)*

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 [15]:

```
df_risk = pd.read_csv('Risk.csv')
df_risk.head(10)
```

Out[15]:

	ID	AGE	INCOME	GENDER	MARITAL	NUMKIDS	NUMCARDS	HOWPAID	MORTGAGE	STORECAR	L
0	100756	44	59944	m	married	1	2	monthly	y	2	
1	100668	35	59692	m	married	1	1	monthly	y	1	
2	100418	34	59508	m	married	1	1	monthly	y	2	
3	100416	34	59463	m	married	0	2	monthly	y	1	
4	100590	39	59393	f	married	0	2	monthly	y	1	
5	100657	41	59276	m	married	1	2	monthly	y	1	
6	100702	42	59201	m	married	0	1	monthly	y	2	
7	100319	31	59193	f	married	1	2	monthly	y	1	
8	100666	28	59179	m	married	1	1	monthly	y	2	
9	100389	30	59036	m	married	1	1	monthly	y	2	

In [16]:

```
# 1. The minimum and maximum values of income

min_income = df_risk['INCOME'].min()
print("Minimum values of income: ", min_income)

max_income = df_risk['INCOME'].max()
print("Maximum values of income: ", max_income)
```

```
Minimum values of income:  15005
Maximum values of income:  59944
```

In [17]:

```
#2. The mean income based on gender

income_mean = df_risk.groupby('GENDER')['INCOME'].mean()

print("mean income based on gender : \n\n", income_mean)
```

```
mean income based on gender :

 GENDER
f    25370.143958
m    25794.089706
Name: INCOME, dtype: float64
```

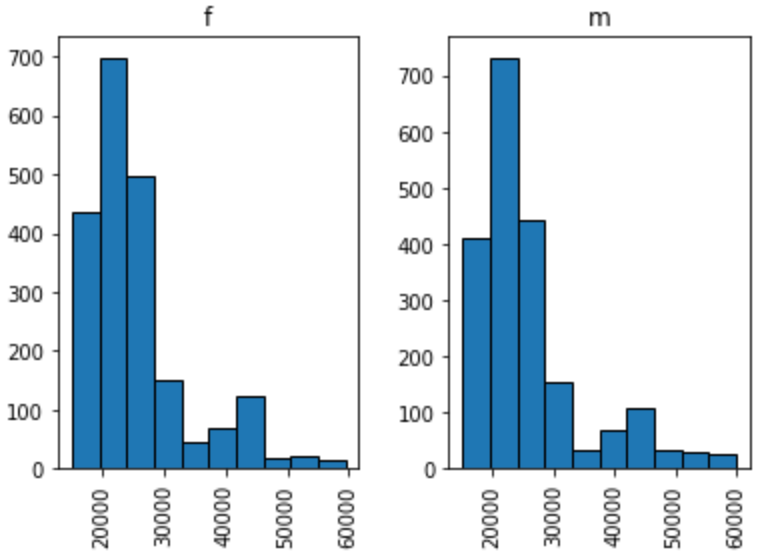
In [18]:

```
#3. Draw the histogram of income distribution based on gender

# Pandas histogram by group:
df_risk.hist(by='GENDER',
       column='INCOME', edgecolor ='black')
```

Out[18]:

```
array([<AxesSubplot:title={'center':'f'}>,
       <AxesSubplot:title={'center':'m'}>], dtype=object)
```



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] or use this box link 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 [25]:

```
# loading the data
import glob
files = glob.glob("namesbystate/*.TXT")

df = pd.DataFrame()

columns = ["State","Gender","Year","Name","Count"]

for f in files:
    csv = pd.read_csv(f, header=None, names = columns)
    df = df.append(csv)

df.head(10)
```

Out[25]:

	State	Gender	Year	Name	Count
0	IN	F	1910	Mary	619
1	IN	F	1910	Helen	324
2	IN	F	1910	Ruth	238
3	IN	F	1910	Dorothy	215
4	IN	F	1910	Mildred	200
5	IN	F	1910	Margaret	196
6	IN	F	1910	Thelma	137
7	IN	F	1910	Edna	113
8	IN	F	1910	Martha	112
9	IN	F	1910	Hazel	108

In [28]:
```
# Men and women count
gender_count = df.groupby("Gender")["Count"].sum()
print("Men count: ", gender_count[0])
print("Women count: ", gender_count[1])
```

```
Men count:  150868994
Women count:  163223872
```

In [30]:
```
# Count unique boy/girl names
unique_names = df.groupby("Gender")["Name"].unique()
print("Girl unique names: ",len(unique_names[0]))
print("Boy unique names: ",len(unique_names[1]))
```

```
Girl unique names:  21026
Boy unique names:  13926
```

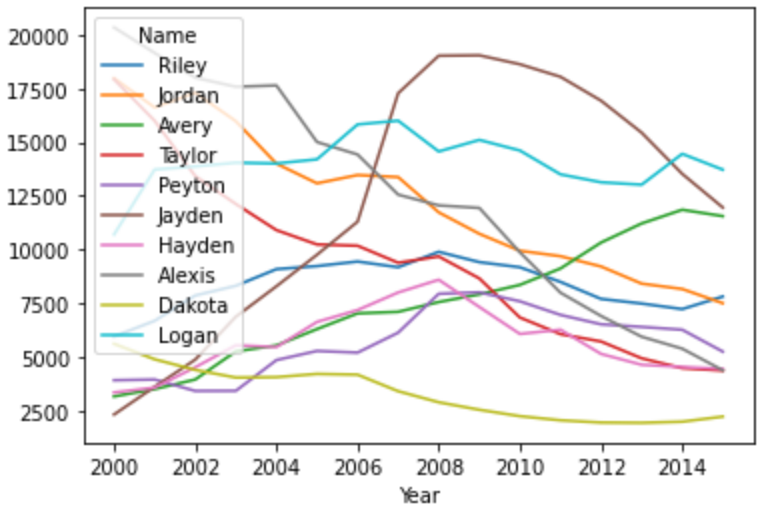
In [57]:
```
# Find top 10 popular names between 2000 - 2015 and plot their trend
df_2000_2015 = df[df["Year"].between(2000,2015)]

# n = 10
top_10_names = df_2000_2015['Name'].value_counts()[:10].index.tolist()
top_10_names
```

Out[57]:
```
['Riley',
 'Jordan',
 'Avery',
 'Taylor',
 'Peyton',
 'Jayden',
 'Hayden',
 'Alexis',
 'Dakota',
 'Logan']
```

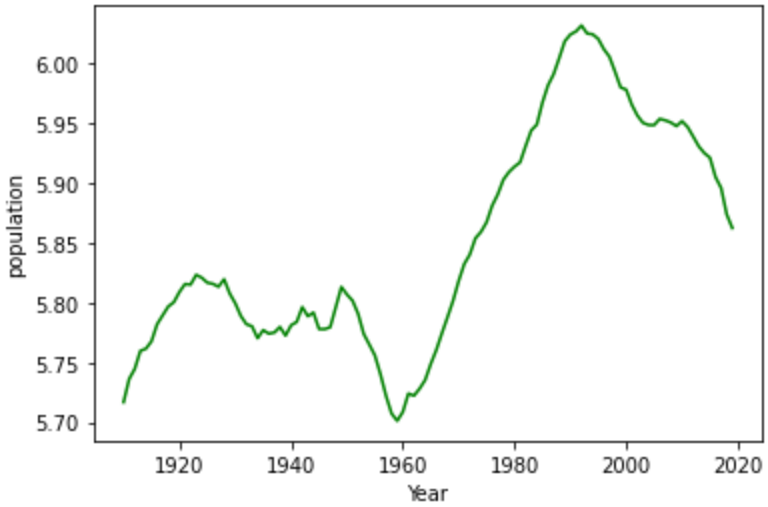
In [58]:
```
# plot their trend
import matplotlib.pyplot as plt
%matplotlib inline

pivoted_df = df_2000_2015.pivot_table(index="Name", columns="Year",values="Count", aggfunc
pivoted_df.loc[top_10_names, :].transpose().plot();
```



In [62]:
```
df["namelength"] = df["Name"].str.len()

df.groupby('Year')['namelength'].mean().plot(color = 'green' , linestyle ='solid');
plt.ylabel('population');
```


6.00
5.95
5.90
5.85
5.80
5.75
5.70
population
1920
1940
1960
1980
2000
2020
Year

In [ ]: