Deshpande_Charudatta_Python Tutorial

January 4, 2018

In [28]:
```
import numpy as np
import pandas as pd
from collections import Counter
import matplotlib.pyplot as plt
%matplotlib inline
```

0.1 Class Activity 1: Getting Started

Write down the grade calculation code in the following cell.

In [29]:
```
#TODO Add Grade Calculation Code here
grade = 'F'
marks = int(input('Enter your marks '))
if marks >90:
    grade = 'A'
elif marks>80 and marks <91:
    grade= 'B'
else:
    grade='F'
print ('Marks = %d, Grade = %s' %(marks, grade))
```

```
Enter your marks 98
Marks = 98, Grade = A
```

0.2 Class Activity 2: List

In [30]:
```
lst=[1,2,2,4,5,6]
```

In [31]:
```
def mean_lst(lst):
    return np.mean(lst)
def mode_lst(lst):
    return Counter(lst).most_common()[0][0]
```

In [32]:
```
def custom_mean(lst1):
# Add Your Code here
    sum1=0
    for k in lst1:
```

```
            sum1+=k
        return sum1/len(lst1)
```

In [33]:
```
a=mean_lst(lst)
b=custom_mean(lst)
print (a,b)
mean_lst(lst)==custom_mean(lst)
```

```
3.33333333333 3.3333333333333335
```

Out[33]: True

0.3 Class Activity 3 : Mode of a List

In [34]:
```
def custom_mode(lst1):
    #TODO Add Your Code here
    counts_dict={}
    for k in lst1:
        if k in counts_dict:
            counts_dict[k]+=1
        else:
            counts_dict[k]=1
    max_count=0
    max_key=-1
    for k in counts_dict:
        if counts_dict[k]>=max_count:
            max_key=k
            max_count=counts_dict[k]
    return max_key
```

In [35]:
```
custom_mode(lst)
```

Out[35]: 2

In [36]:
```
mode_lst(lst)
```

Out[36]: 2

In [37]:
```
mode_lst(lst)==custom_mode(lst)
```

Out[37]: True

In [38]:
```
s=pd.Series([1,2,3])
s
```

Out[38]:
```
0    1
1    2
2    3
dtype: int64
```

1 Class Activity 4: Classes

In [39]:
```
class CustomClass:
    #Add Code here
        def __init__(self):
            print ('Constructor Called')

        def factorial(self, n):
            result=1
            for i in range(1,n+1):
    #             print i
                result=result*i
            return result
        def permutations(self,n,r):
            return self.factorial(n)/float(self.factorial(n-r))
        def combinations(self,n,r):
            return self.factorial(n)/float(self.factorial(n-r)*self.factorial(r))
    c=CustomClass()
    c.factorial(5)
```

```
Constructor Called
```

Out[39]: 120

In [40]:
```
c=CustomClass()
```

```
Constructor Called
```

In [41]:
```
c.factorial(5)==120
```

Out[41]: True

In [42]:
```
c.permutations(5,3)==60
```

Out[42]: True

In [43]:
```
c.combinations(5,3)==10
```

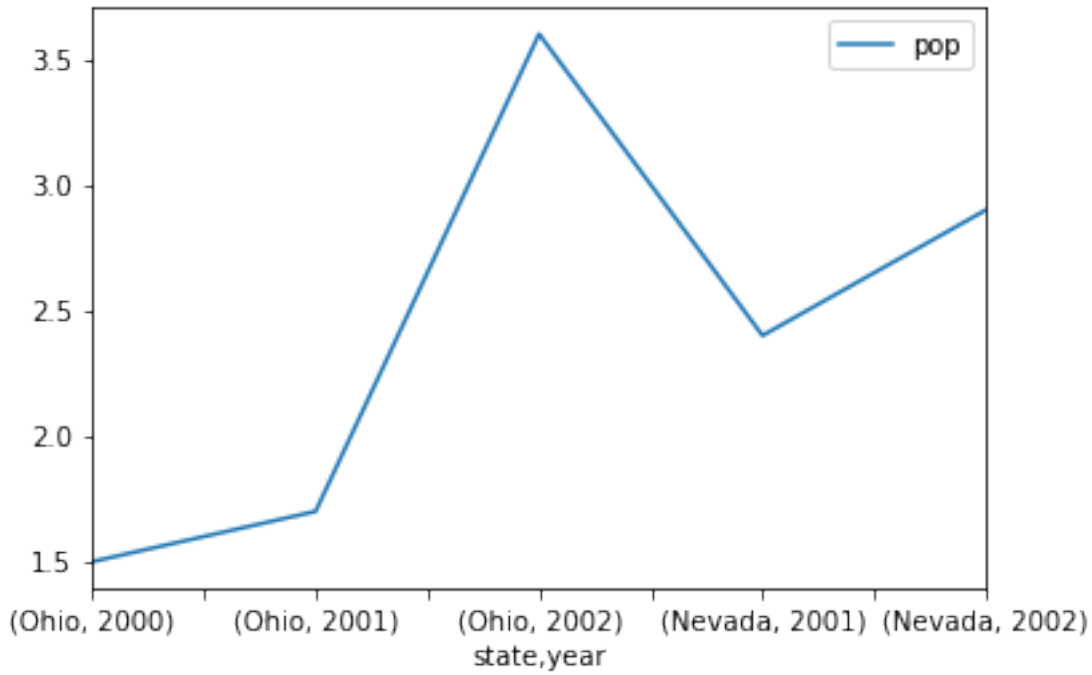
Out[43]: True

1.1 Class Activity 5: Plotting Pop

In [44]:
```
#TODO : Add your code here
s=pd.Series([1,2,3])
s
s[s<3]
s.apply(lambda x:x+1)
dict1={'A':3.5,'B':3,'C':2.5}
```

```
s1=pd.Series(dict1)
s1
data={'state':['Ohio','Ohio','Ohio','Nevada','Nevada'],
      'year':[2000,2001,2002,2001,2002],
      'pop':[1.5,1.7,3.6,2.4,2.9]}
df=pd.DataFrame(data)
# df['pop'].plot()
df2=df.set_index(['state','year'])
df2[['pop']].plot()
```

Out[44]: <matplotlib.axes._subplots.AxesSubplot at 0x235e15a79e8>



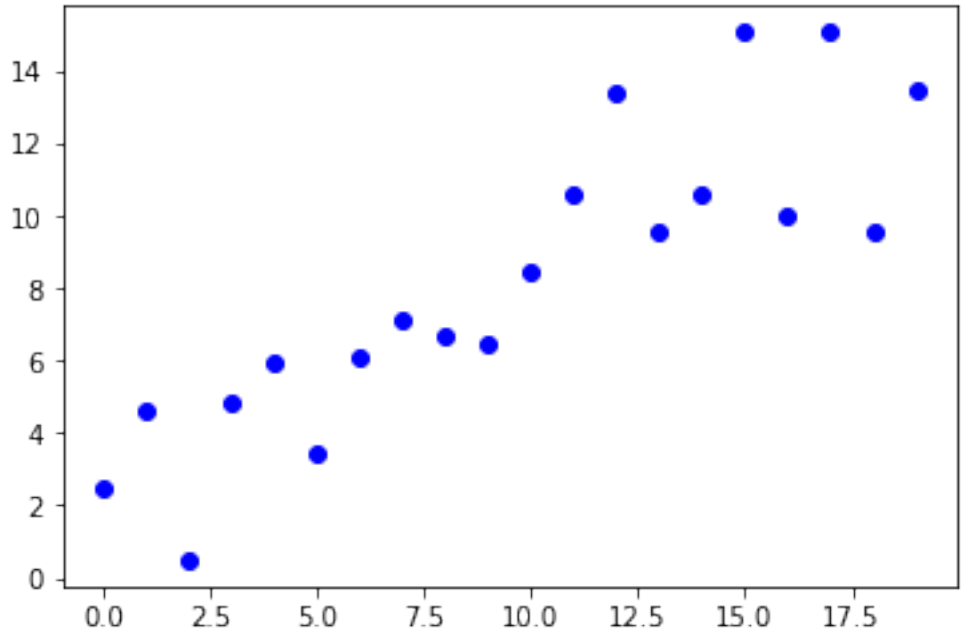
1.2 Regression Code

In [45]:
```
import scipy.stats as stats
import statsmodels.api as sm
import numpy as np
import matplotlib.pyplot as plt


x = np.array(range(20))
y = 3 + 0.5 * x + 2 * np.random.randn(20)
#plot the data
plt.plot(x, y, 'bo')
plt.show()
```

```
results = sm.OLS(y, x).fit()
print (results.summary())

slope= results.params[0]
plt.plot(x, y, 'bo')
plt.hold(True)
# Plot a line
y =  slope * x
plt.plot(x, y, 'r-')
plt.show()
```



OLS Regression Results

Dep. Variable:	y	R-squared:	0.937
Model:	OLS	Adj. R-squared:	0.933
Method:	Least Squares	F-statistic:	281.6
Date:	Thu, 04 Jan 2018	Prob (F-statistic):	7.52e-13
Time:	08:16:20	Log-Likelihood:	-45.027
No. Observations:	20	AIC:	92.05
Df Residuals:	19	BIC:	93.05
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]

```
x1             0.7964      0.047     16.782      0.000       0.697       0.896
==============================================================================
Omnibus:                        1.599   Durbin-Watson:                   1.899
Prob(Omnibus):                  0.450   Jarque-Bera (JB):                1.038
Skew:                          -0.551   Prob(JB):                        0.595
Kurtosis:                       2.826   Cond. No.                         1.00
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.


C:\Users\deshc\Anaconda3\lib\site-packages\ipykernel_launcher.py:18: MatplotlibDeprecationWarni
    Future behavior will be consistent with the long-time default:
    plot commands add elements without first clearing the
    Axes and/or Figure.
C:\Users\deshc\Anaconda3\lib\site-packages\matplotlib\__init__.py:805: MatplotlibDeprecationWa
  mplDeprecation)
C:\Users\deshc\Anaconda3\lib\site-packages\matplotlib\rcsetup.py:155: MatplotlibDeprecationWarn
  mplDeprecation)
```