```
## Getting started
```

In [2]:
```
import numpy as np
import pandas as pd
```

~~~
## DataFrame basics

### A few of the fundamental routines for selecting, sorting, adding and aggregating data in DataFrames

Difficulty: *easy*

Consider the following Python dictionary `data` and Python list `labels`:

``` python
data = {'animal': ['cat', 'cat', 'snake', 'dog', 'dog', 'cat', 'snake', 'cat', 'dog', 'dog'],
        'age': [2.5, 3, 0.5, np.nan, 5, 2, 4.5, np.nan, 7, 3],
        'visits': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
        'priority': ['yes', 'yes', 'no', 'yes', 'no', 'no', 'no', 'yes', 'no', 'no']}

labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
~~~

1. Create a DataFrame `df` from this dictionary `data` which has the index `labels`. (2 points)
```

In [3]:
```

data = {'animal': ['cat', 'cat', 'snake', 'dog', 'dog', 'cat', 'snake', 'ca
        'age': [2.5, 3, 0.5, np.nan, 5, 2, 4.5, np.nan, 7, 3],
        'visits': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
        'priority': ['yes', 'yes', 'no', 'yes', 'no', 'no', 'no', 'yes', 'n

labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
```

2. Display a summary of the basic information about this DataFrame and its data. (2 points)

In [4]:
```
df = pd.DataFrame(data)
```

In [5]:
```
df.index = labels
```
```

In [6]:
```
#display dataframe data
df
```

Out[6]:

| | animal | age | visits | priority |
|---|---|---|---|---|
| **a** | cat | 2.5 | 1 | yes |
| **b** | cat | 3.0 | 3 | yes |
| **c** | snake | 0.5 | 2 | no |
| **d** | dog | NaN | 3 | yes |
| **e** | dog | 5.0 | 2 | no |
| **f** | cat | 2.0 | 3 | no |
| **g** | snake | 4.5 | 1 | no |
| **h** | cat | NaN | 1 | yes |
| **i** | dog | 7.0 | 2 | no |
| **j** | dog | 3.0 | 1 | no |

In [7]:
```
#display info about dataframe
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 10 entries, a to j
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   animal    10 non-null     object 
 1   age       8 non-null      float64
 2   visits    10 non-null     int64  
 3   priority  10 non-null     object 
dtypes: float64(1), int64(1), object(2)
memory usage: 400.0+ bytes
```

3. Return the first 3 rows of the DataFrame `df`. (2 points)

In [8]:
```
df.head(3)
```

Out[8]:

	animal	age	visits	priority
a	cat	2.5	1	yes
b	cat	3.0	3	yes
c	snake	0.5	2	no

4. Select just the 'animal' and 'age' columns from the DataFrame `df`. (2 points)

In [9]:
```
df.iloc[:,0:2]
```

Out[9]:

	animal	age
a	cat	2.5
b	cat	3.0
c	snake	0.5
d	dog	NaN
e	dog	5.0
f	cat	2.0
g	snake	4.5
h	cat	NaN
i	dog	7.0
j	dog	3.0

5. Select only the rows where the number of visits is greater than 2. (2 points)

In [10]:
```
df[df['visits'] > 2]
```

Out[10]:

	animal	age	visits	priority
b	cat	3.0	3	yes
d	dog	NaN	3	yes
f	cat	2.0	3	no

6. Select the rows where the age is missing, i.e. is `NaN`. (2 points)

In [11]:
```
df[df['age'].isnull()]
```

Out[11]:

	animal	age	visits	priority
d	dog	NaN	3	yes
h	cat	NaN	1	yes

7. Select the rows where the animal is a cat *and* the age is less than 3. (2 points)

In [12]:
```
df[(df['animal'] =='cat') & (df['age'] < 3)]
```

Out[12]:

	animal	age	visits	priority
a	cat	2.5	1	yes
f	cat	2.0	3	no

8. Calculate the sum of all visits (the total number of visits). (2 points)

In [13]:
```
df.iloc[:,2].sum()
```

Out[13]:
```
19
```

9. Calculate the mean age for each different animal in `df`. (2 points)

In [14]:
```

df.groupby("animal").age.mean()
```

Out[14]:
```
animal
cat   2.5
dog   5.0
snake   2.5
Name: age, dtype: float64
```

**10.** Count the number of each type of animal in `df`. (2 points)

In [15]:
```
df.groupby("animal").animal.count()
```

Out[15]:
```
animal
cat   4
dog   4
snake   2
Name: animal, dtype: int64
```

## Pandas: Charts and Graphs (10 poins)

Pandas offers great graphing functions that work in conjuncture with MatPlotlib.
Run the import below

In [16]:
```
Be sure we import MapPlotLib as plt and NumPy as np to be used with Panda
import matplotlib.pyplot as plt
```

**16.** Basic plotting: plot

### (2 points) Create a basic Bar Plot with the following data frame 'df'

In [17]:
```
df = pd.DataFrame(np.random.rand(10,4),columns=['a','b','c','d'])
```

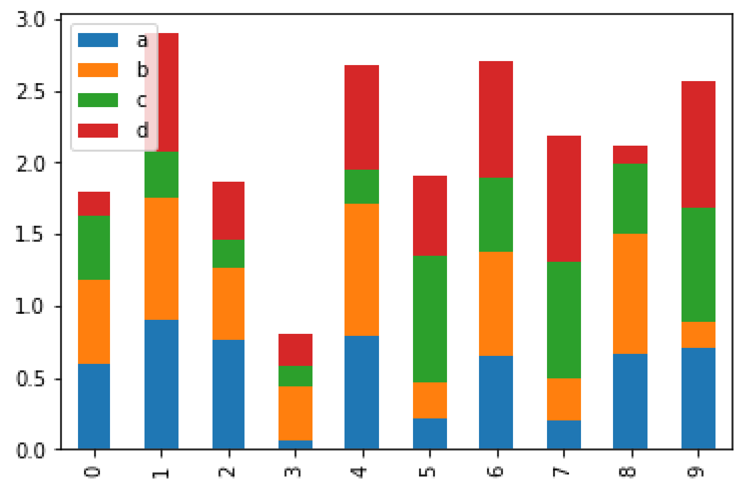
**17.** Basic Plotting: Stacked Plots

**(2 points) produce a stacked bar plot using the data frame 'df'**

In [18]:
```
df = pd.DataFrame(np.random.rand(10,4),columns=['a','b','c','d'])
```

In [19]:
```
stackBar = df.plot.bar(stacked = True)
stackBar
```

Out[19]:
```
<AxesSubplot:>
```

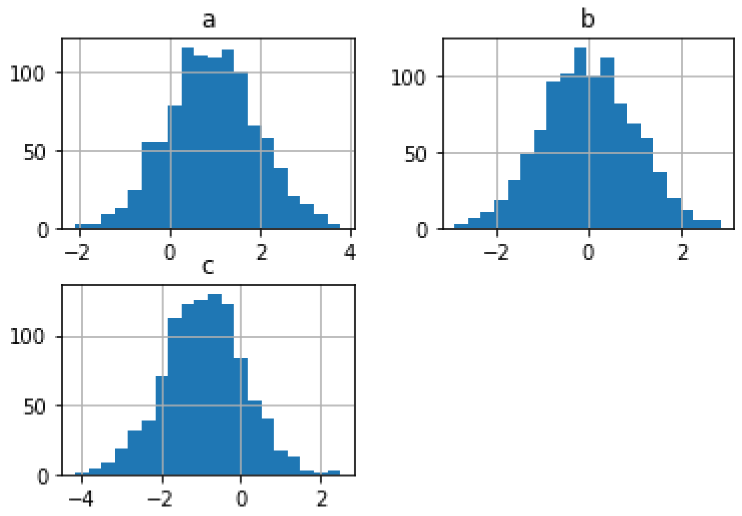


**18.** Plotting: Histograms

**(2 points) Create a histogram with 20 bins using the data frame ‘df’**

In [20]:
```
df = pd.DataFrame({'a':np.random.randn(1000)+1,'b':np.random.randn(1000),'c
np.random.randn(1000) - 1}, columns=['a', 'b', 'c'])
```

```
In [21]: pandasHist = df.hist(bins=20)
```



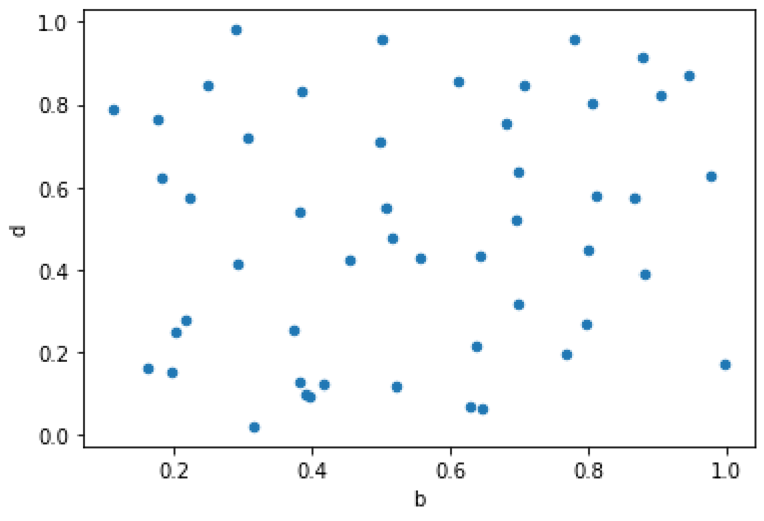
19. Plotting: Scatter plots

(2 points) Create a scatter plot of columns `b` and `d` using the data frame 'df'

```
In [22]:
df = pd.DataFrame(np.random.rand(50, 4), columns=['a', 'b', 'c', 'd'])


scplot = df.plot.scatter(x ='b',y = 'd')
scplot
```

```
Out[22]: <AxesSubplot:xlabel='b', ylabel='d'>
```



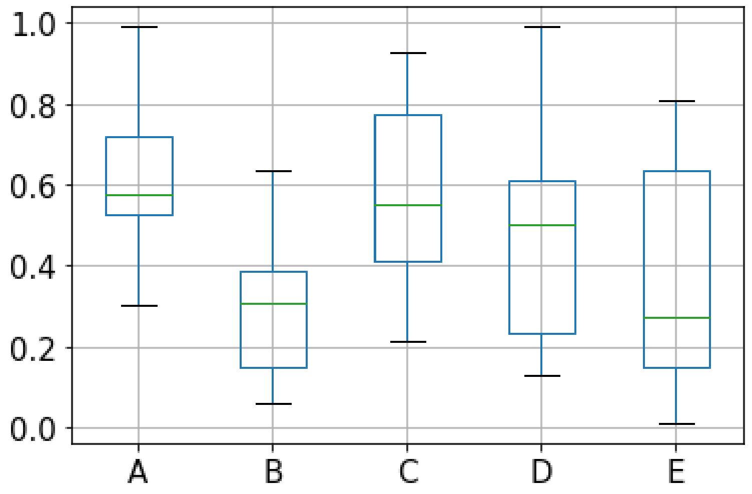
20. Plotting: Box Plots

(2 points) Create a Box Plot using the data frame ‘df’

In [23]:

```
df = pd.DataFrame(np.random.rand(10, 5), columns=['A', 'B', 'C', 'D', 'E'])


boxPlot = df.boxplot(grid=True, rot=0, fontsize=15)
```


1.0
0.8
0.6
0.4
0.2
0.0
A
B
C
D
E