In [1]:

```
import numpy as np
from pandas import Series,DataFrame
import pandas as pd
```

In [2]:

```
#Now we'll learn how to deal with missing data, a very common task when analyzing

data = Series(['one','two', np.nan, 'four'])
```

In [3]:

```
#Show data
data
```

Out[3]:

```
0      one
1      two
2      NaN
3     four
dtype: object
```

In [5]:

```
#Find the missing values
data.isnull()
```

Out[5]:

```
0     False
1     False
2      True
3     False
dtype: bool
```

In [7]:

```
#We can simply drop the NAN
data.dropna()
```

Out[7]:

```
0      one
1      two
3     four
dtype: object
```

In [14]:

```
# In a DataFrame we need to be a little more careful!

dframe = DataFrame([[1,2,3],[np.nan,5,6],[7,np.nan,9],[np.nan,np.nan,np.nan]])
```

In [15]:

```
#Show
dframe
```

Out[15]:

	0	1	2
0	1	2	3
1	NaN	5	6
2	7	NaN	9
3	NaN	NaN	NaN

In [16]:

```
clean_dframe = dframe.dropna()
```

In [17]:

```
#Show
clean_dframe
```

Out[17]:

	0	1	2
0	1	2	3

In [12]:

```
#Note all rows where an NA occured was a drop of the entire row
```

In [18]:

```
#We can also specify to only drop rows that are complete missing all data
dframe.dropna(how='all')
```

Out[18]:

	0	1	2
0	1	2	3
1	NaN	5	6
2	7	NaN	9

In [22]:

```
#Or we can specify to drop columns with missing data
dframe.dropna(axis=1)

#This should drop all columns out since every column contains at least 1 NAN
```

Out[22]:

0
1
2
3

In [26]:

```
#We can also threshold teh missing data as well

#For example if we only want rows with at least 3 data points
dframe2 = DataFrame([[1,2,3,np.nan],[2,np.nan,5,6],[np.nan,7,np.nan,9],[1,np.nan,

#Show
dframe2
```

Out[26]:

	0	1	2	3
0	1	2	3	NaN
1	2	NaN	5	6
2	NaN	7	NaN	9
3	1	NaN	NaN	NaN

In [28]:

```
#Droppin any rows tht dont have at least 2 data points
dframe2.dropna(thresh=2)
```

Out[28]:

	0	1	2	3
0	1	2	3	NaN
1	2	NaN	5	6

In [29]:

```
#Dropiing rows without at least 3 data points
dframe2.dropna(thresh=3)
```

Out[29]:

	0	1	2	3
0	1	2	3	NaN
1	2	NaN	5	6

In [30]:

```
#We can also fill any NAN
dframe2.fillna(1)
```

Out[30]:

	0	1	2	3
0	1	2	3	1
1	2	1	5	6
2	1	7	1	9
3	1	1	1	1

In [33]:

```
#Can also fill in diff values for diff columns
dframe2.fillna({0:0,1:1,2:2,3:3})
```

Out[33]:

	0	1	2	3
0	1	2	3	3
1	2	1	5	6
2	0	7	2	9
3	1	1	2	3

In [34]:

```
#Note that we still have access to the original dframe
dframe2
```

Out[34]:

	0	1	2	3
0	1	2	3	NaN
1	2	NaN	5	6
2	NaN	7	NaN	9
3	1	NaN	NaN	NaN

In [35]:

```
#If we want to modify the exsisting object, use inplace
dframe2.fillna(0,inplace=True)
```

In [36]:

```
#Now let's see the dframe
dframe2
```

Out[36]:

	0	1	2	3
0	1	2	3	0
1	2	0	5	6
2	0	7	0	9
3	1	0	0	0

In [ ]:

```
#Awesome! Next we'll learn about Index Hierarchy
```