In [1]:
```
import pandas as pd
```

In [2]:
```
list1 = ["Jane", "John", "June", "Jim", "Jay"]
list2 = ["Ford", "BMW", "Mini", "Mercedes", "Toyota"]
list3 = ["Blue", "Grey", "Red", "White", "White"]
list4 = ["1.6l", "2.0l", "1.6l", "2.2l", "1.2l"]
df1 = pd.DataFrame({"Make":list2, "Color":list3, "Capacity":list4}, index = list1)
df1
```

Out[2]:

	Capacity	Color	Make
Jane	1.6l	Blue	Ford
John	2.0l	Grey	BMW
June	1.6l	Red	Mini
Jim	2.2l	White	Mercedes
Jay	1.2l	White	Toyota

Updating values via indexation

In [3]:
```
df1.loc["Jane", "Color"] = "Orange" # Jane gives us the rows to update,
                                    # Color tells us which column to update
df1
```

Out[3]:

	Capacity	Color	Make
Jane	1.6l	Orange	Ford
John	2.0l	Grey	BMW
June	1.6l	Red	Mini
Jim	2.2l	White	Mercedes
Jay	1.2l	White	Toyota

In [4]:
```
df1.loc[df1["Color"].isin(["White"]), "Color"] = "Off-White" # The first part gives us the rows to update,
                                                             # The second part tells us which column to updat
e
df1
```

Out[4]:

	Capacity	Color	Make
Jane	1.6l	Orange	Ford
John	2.0l	Grey	BMW
June	1.6l	Red	Mini
Jim	2.2l	Off-White	Mercedes
Jay	1.2l	Off-White	Toyota

Replacing values

An example across the entire dataframe, replacing entire cell values

In [5]:
```
df1.replace(["Off-White", "Grey"], ["White", "Silver"], inplace = True)
df1
```

Out[5]:

	Capacity	Color	Make
Jane	1.6l	Orange	Ford
John	2.0l	Silver	BMW
June	1.6l	Red	Mini
Jim	2.2l	White	Mercedes
Jay	1.2l	White	Toyota

An example for a single series, replacing partial cell values

In [6]:
```
df1["Capacity"].replace({'l':''}, inplace = True, regex=True)
df1
```

Out[6]:

	Capacity	Color	Make
Jane	1.6	Orange	Ford
John	2.0	Silver	BMW
June	1.6	Red	Mini
Jim	2.2	White	Mercedes
Jay	1.2	White	Toyota

Changing the type of a series

Using astype()

In [7]:
```
df1.info() # All 3 series currently classified as "object" aka "string"
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 5 entries, Jane to Jay
Data columns (total 3 columns):
Capacity    5 non-null object
Color       5 non-null object
Make        5 non-null object
dtypes: object(3)
memory usage: 320.0+ bytes
```

In [8]:
```
df1["Color"] = df1["Color"].astype("category")    # Convert color to a categorical variable
df1["Capacity"] = df1["Capacity"].astype("float") # Convert capacity to a float

df1.info() # Here we see the results of the updates - the data didn't change
           # but the format did, so that now we can e.g. perform calcs on Capacity
           # which we could not have done while it was classified as an Object
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 5 entries, Jane to Jay
Data columns (total 3 columns):
Capacity    5 non-null float64
Color       5 non-null category
Make        5 non-null object
dtypes: category(1), float64(1), object(1)
memory usage: 477.0+ bytes
```

Using pd.to_numeric()

In [9]:
```
df1["Capacity"] = df1["Capacity"].astype("object") # Now let's put Capacity back
df1.loc["Jane", "Capacity"] = "1.6l"               # And then introduce a non-numeric value in one field
df1
```

Out[9]:

	Capacity	Color	Make
Jane	1.6l	Orange	Ford
John	2	Silver	BMW
June	1.6	Red	Mini
Jim	2.2	White	Mercedes
Jay	1.2	White	Toyota

In [ ]:
```
df1["Capacity"] = df1["Capacity"].astype("float") # THIS script will now end in an error
                                                  # because the l in 1.6l can't be converted
```

In [10]:
```
df1["Capacity"] = pd.to_numeric(df1["Capacity"], errors = "coerce")
df1
# This is an alternative method to convert data types - the argument errors = "coerce"
# is pretty handy if your data has some exceptions in it, fills NaN where no conversion is possible
# pd.to_datetime() also has errors = "coerce" which can be useful
```

Out[10]:

	Capacity	Color	Make
Jane	NaN	Orange	Ford
John	2.0	Silver	BMW
June	1.6	Red	Mini
Jim	2.2	White	Mercedes
Jay	1.2	White	Toyota

Dealing with null values

Filling with another default value

In [11]:
```
df1["Capacity"].fillna(0, inplace = True) # Fills the NaN values in the series with the specified value
                                          # or indeed the entire dataframe (which would seldom make sense!)
df1
```

Out[11]:

	Capacity	Color	Make
Jane	0.0	Orange	Ford
John	2.0	Silver	BMW
June	1.6	Red	Mini
Jim	2.2	White	Mercedes
Jay	1.2	White	Toyota

Removing rows with NaN

In [12]:
```
# Here are some additional ways to deal with them:
# df.dropna()                           Removes ALL rows in the entire dataframe with
#                                       one or more null values
# df.dropna(how = 'all'                 Removes only rows where all columns contain null values
# df.dropna(subset = ["Column name"])   Removes rows only where there is a null value in the
#                                       specified column name
```

Altering the shape of the dataframe

Adding columns quickly

In [13]:
```
df1["Model"] = ""
df1
```

Out[13]:

	Capacity	Color	Make	Model
Jane	0.0	Orange	Ford	
John	2.0	Silver	BMW	
June	1.6	Red	Mini	
Jim	2.2	White	Mercedes	
Jay	1.2	White	Toyota	

Adding columns at a specified location

In [14]:
```
df1.insert(1, "Service Interval", "20000km")
df1
```

Out[14]:

	Capacity	Service Interval	Color	Make	Model
Jane	0.0	20000km	Orange	Ford	
John	2.0	20000km	Silver	BMW	
June	1.6	20000km	Red	Mini	
Jim	2.2	20000km	White	Mercedes	
Jay	1.2	20000km	White	Toyota	

Adding rows

In [15]:
```
extra_row = pd.Series({"Capacity": 1.6, "Color": "Blue", "Make": "Honda", "Model": "Jazz"})
extra_row.name = "James"
df1 = df1.append(extra_row)
df1
```

Out[15]:

	Capacity	Service Interval	Color	Make	Model
Jane	0.0	20000km	Orange	Ford	
John	2.0	20000km	Silver	BMW	
June	1.6	20000km	Red	Mini	
Jim	2.2	20000km	White	Mercedes	
Jay	1.2	20000km	White	Toyota	
James	1.6	NaN	Blue	Honda	Jazz

Removing columns

In [16]:
```
df1.drop("Service Interval", axis = 1, inplace = True) # axis = 1 says you're looking a columns
df1
```

Out[16]:

	Capacity	Color	Make	Model
Jane	0.0	Orange	Ford	
John	2.0	Silver	BMW	
June	1.6	Red	Mini	
Jim	2.2	White	Mercedes	
Jay	1.2	White	Toyota	
James	1.6	Blue	Honda	Jazz

Removing rows

In [17]:
```
df1.drop("John", axis = 0, inplace = True) # axis = 0 says you're looking a rows
df1
```

Out[17]:

	Capacity	Color	Make	Model
Jane	0.0	Orange	Ford	
June	1.6	Red	Mini	
Jim	2.2	White	Mercedes	
Jay	1.2	White	Toyota	
James	1.6	Blue	Honda	Jazz

In [18]:
```
df1 = df1[df1["Color"] != "White"] # a simple way to remove rows based on a condition
df1
```

Out[18]:

	Capacity	Color	Make	Model
Jane	0.0	Orange	Ford	
June	1.6	Red	Mini	
James	1.6	Blue	Honda	Jazz

Transposing the data

In [19]:
```
df1 = df1.transpose() # flips the data around if it's more convenient
df1
```

Out[19]:

	Jane	June	James
Capacity	0	1.6	1.6
Color	Orange	Red	Blue
Make	Ford	Mini	Honda
Model			Jazz