Introduction to Pandas: A Short Tutorial

- Pandas is an open source, BSD-licensed library
- High-performance, easy-to-use data structures and data analysis tools
- Built on top of NumPy, and provides an efficient implementation of a DataFrame
- Makes data analysis fast and easy in Python

DataFrame : A multidimensional array with attached row and column labels

Pandas API Reference: https://pandas.pydata.org/pandas-docs/stable/reference/index.html (https://pandas.pydata.org/pandas-docs/stable/reference/index.html)

This is an introduction to Pandas. You can try to complete the tutorial yourself and check the model answers for help, when needed

In [1]:

```
# Import Pandas
import pandas as pd
```

In [2]:

```
# Create two lists with information from Baby names in England and Wales: 2018
# https://www.ons.gov.uk/peoplepopulationandcommunity/birthsdeathsandmarriages/livel
names  = ['Adam', 'Sophie', 'Charlie', 'Anna', 'Bobby', 'Florence', 'George', 'Mia']
births = [1508,   1929,      3336,     409,    652,    1974,        4949,    2418]

# Merge these two lists together using the zip function
# https://docs.python.org/3.3/library/functions.html
babiesDataSet = list(zip(names, births))
```

In [3]:

```
# Print the combined list
babiesDataSet
```

Out[3]:

```
[('Adam', 1508),
 ('Sophie', 1929),
 ('Charlie', 3336),
 ('Anna', 409),
 ('Bobby', 652),
 ('Florence', 1974),
 ('George', 4949),
 ('Mia', 2418)]
```

In [4]:

```
# Use Pandas to create a dataframe
df = pd.DataFrame(data=babiesDataSet, columns=['Name', "Births"])
```

In [5]:

```
# Display the dataframe
df
```

Out[5]:

	Name	Births
0	Adam	1508
1	Sophie	1929
2	Charlie	3336
3	Anna	409
4	Bobby	652
5	Florence	1974
6	George	4949
7	Mia	2418

In [6]:

```
# Export the dataframe to csv
df.to_csv("birthsUK2018.csv", index=False, header=False)
```

In [7]:

```
# Import data to dataframe
file = "birthsUK2018.csv" #location is relative
births =  pd.read_csv(file, header=None, names=['Name', 'Births'])
```

In [8]:

```
# Show the dataframe
# The numbers [0,1,2,3,4] in the first column are part of the index of the datafram
births
```

Out[8]:

	Name	Births
0	Adam	1508
1	Sophie	1929
2	Charlie	3336
3	Anna	409
4	Bobby	652
5	Florence	1974
6	George	4949
7	Mia	2418

In [9]:

```
# Check the data types of columns
births.dtypes
```

Out[9]:

```
Name      object
Births     int64
dtype: object
```

In [10]:

```
# Get general info about the dataframe
# - There are 8 records in the data set
# - There is a column named "Name" of type object (non numeric) with 8 values
# - There is a column named "Births" of type numeric with 8 values
births.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8 entries, 0 to 7
Data columns (total 2 columns):
Name      8 non-null object
Births    8 non-null int64
dtypes: int64(1), object(1)
memory usage: 208.0+ bytes
```

In [11]:

```
# Check the data types of Births column
births.Births.dtype
```

Out[11]:

```
dtype('int64')
```

In [12]:

```
# Print the top 5 rows
births.head()
```

Out[12]:

	Name	Births
0	Adam	1508
1	Sophie	1929
2	Charlie	3336
3	Anna	409
4	Bobby	652

In [13]:

```
# Print the last 5 rows
births.tail()
```

Out[13]:

	Name	Births
3	Anna	409
4	Bobby	652
5	Florence	1974
6	George	4949
7	Mia	2418

In [14]:

```
# Print the name of columns
births.columns
```

Out[14]:

```
Index(['Name', 'Births'], dtype='object')
```

In [15]:

```
# Get the dataframe as an array
df.values
```

Out[15]:

```
array([['Adam', 1508],
       ['Sophie', 1929],
       ['Charlie', 3336],
       ['Anna', 409],
       ['Bobby', 652],
       ['Florence', 1974],
       ['George', 4949],
       ['Mia', 2418]], dtype=object)
```

In [16]:

```
# Get the index of the dataframe
births.index
```

Out[16]:

```
RangeIndex(start=0, stop=8, step=1)
```

In [17]:

```
# Access the names column
births['Births']
```

Out[17]:

```
0    1508
1    1929
2    3336
3     409
4     652
5    1974
6    4949
7    2418
Name: Births, dtype: int64
```

In [18]:

```
# Access the Births column
births.Name
```

Out[18]:

```
0        Adam
1      Sophie
2     Charlie
3        Anna
4       Bobby
5    Florence
6      George
7         Mia
Name: Name, dtype: object
```

In [19]:

```
# Find the maximum number of births
births['Births'].max()
```

Out[19]:

```
4949
```

In [20]:

```
# Get the name associated with the max births
births['Name'][df['Births']==df['Births'].max()].values
```

Out[20]:

```
array(['George'], dtype=object)
```

In [21]:

```
# Find the unique names
births['Name'].unique()
```

Out[21]:

```
array(['Adam', 'Sophie', 'Charlie', 'Anna', 'Bobby', 'Florence', 'Geor
ge',
       'Mia'], dtype=object)
```

In [22]:

```
# Get some descriptive statistics for the number of births
births['Births'].describe()
```

Out[22]:

```
count       8.000000
mean     2146.875000
std      1467.739218
min       409.000000
25%      1294.000000
50%      1951.500000
75%      2647.500000
max      4949.000000
Name: Births, dtype: float64
```

In [23]:

```
# Get the names with births more than 2000
births[births['Births'] > 2000]
```

Out[23]:

	Name	Births
2	Charlie	3336
6	George	4949
7	Mia	2418

In [24]:

```
# Get the names startomg with "A"
births[births['Name'].str.contains("A")]
```

Out[24]:

	Name	Births
0	Adam	1508
3	Anna	409

In [25]:

```
# Add another column with the country set for all rows as UK
import numpy as np
births['Country'] = np.repeat('UK', len(births))
births
```

Out[25]:

	Name	Births	Country
0	Adam	1508	UK
1	Sophie	1929	UK
2	Charlie	3336	UK
3	Anna	409	UK
4	Bobby	652	UK
5	Florence	1974	UK
6	George	4949	UK
7	Mia	2418	UK

In [26]:

```
# Add a column with the gender of the babies
gender = np.tile(('Male', "Female"), int(len(births)/2))
births['Gender'] = gender
births
```

Out[26]:

	Name	Births	Country	Gender
0	Adam	1508	UK	Male
1	Sophie	1929	UK	Female
2	Charlie	3336	UK	Male
3	Anna	409	UK	Female
4	Bobby	652	UK	Male
5	Florence	1974	UK	Female
6	George	4949	UK	Male
7	Mia	2418	UK	Female

In [27]:

```
# Add a column the indicates for each name its percentage over the total births
births['Percentage'] = births['Births']/births['Births'].sum()*100
births
```

Out[27]:

	Name	Births	Country	Gender	Percentage
0	Adam	1508	UK	Male	8.780204
1	Sophie	1929	UK	Female	11.231441
2	Charlie	3336	UK	Male	19.423581
3	Anna	409	UK	Female	2.381368
4	Bobby	652	UK	Male	3.796215
5	Florence	1974	UK	Female	11.493450
6	George	4949	UK	Male	28.815138
7	Mia	2418	UK	Female	14.078603

In [28]:

```
# Delete the country column
births.drop('Country', axis=1)
```

Out[28]:

	Name	Births	Gender	Percentage
0	Adam	1508	Male	8.780204
1	Sophie	1929	Female	11.231441
2	Charlie	3336	Male	19.423581
3	Anna	409	Female	2.381368
4	Bobby	652	Male	3.796215
5	Florence	1974	Female	11.493450
6	George	4949	Male	28.815138
7	Mia	2418	Female	14.078603

In [29]:

```
# Access multiple columns
births[['Name', 'Births', 'Percentage']]
```

Out[29]:

	Name	Births	Percentage
0	Adam	1508	8.780204
1	Sophie	1929	11.231441
2	Charlie	3336	19.423581
3	Anna	409	2.381368
4	Bobby	652	3.796215
5	Florence	1974	11.493450
6	George	4949	28.815138
7	Mia	2418	14.078603

In [30]:

```
# Subset the data based on index location
births.iloc[:3]
```

Out[30]:

	Name	Births	Country	Gender	Percentage
0	Adam	1508	UK	Male	8.780204
1	Sophie	1929	UK	Female	11.231441
2	Charlie	3336	UK	Male	19.423581

In [31]:

```
births.loc[2:3]
```

Out[31]:

	Name	Births	Country	Gender	Percentage
2	Charlie	3336	UK	Male	19.423581
3	Anna	409	UK	Female	2.381368

In [32]:

```
# Find based on index value
births.at[0, 'Births']
```

Out[32]:

```
1508
```

In [33]:

```
# Query the data
births.query ('Gender == "Female"')
```

Out[33]:

	Name	Births	Country	Gender	Percentage
1	Sophie	1929	UK	Female	11.231441
3	Anna	409	UK	Female	2.381368
5	Florence	1974	UK	Female	11.493450
7	Mia	2418	UK	Female	14.078603

In [34]:

```
# Query the data
births.query ('Births < 1000 and Gender == "Male"')
```

Out[34]:

	Name	Births	Country	Gender	Percentage
4	Bobby	652	UK	Male	3.796215

In [35]:

```
# Get the births by gender
genderBirths = births.groupby('Gender').sum()
genderBirths
```

Out[35]:

Gender	Births	Percentage
Female	6730	39.184862
Male	10445	60.815138

In [36]:

```
# Sort the dataframe by name
births.sort_values(by='Name')
```

Out[36]:

	Name	Births	Country	Gender	Percentage
0	Adam	1508	UK	Male	8.780204
3	Anna	409	UK	Female	2.381368
4	Bobby	652	UK	Male	3.796215
2	Charlie	3336	UK	Male	19.423581
5	Florence	1974	UK	Female	11.493450
6	George	4949	UK	Male	28.815138
7	Mia	2418	UK	Female	14.078603
1	Sophie	1929	UK	Female	11.231441

In [37]:

```
# Sort the dataframe by the number of births in descending order
births.sort_values(by='Births', ascending=False)
```

Out[37]:

	Name	Births	Country	Gender	Percentage
6	George	4949	UK	Male	28.815138
2	Charlie	3336	UK	Male	19.423581
7	Mia	2418	UK	Female	14.078603
5	Florence	1974	UK	Female	11.493450
1	Sophie	1929	UK	Female	11.231441
0	Adam	1508	UK	Male	8.780204
4	Bobby	652	UK	Male	3.796215
3	Anna	409	UK	Female	2.381368

In [38]:

```
# Add a column with the county each child has been born in
births['County'] = ['Yorkshire', 'Essex', 'Yorkshire', 'Yorkshire', 'Kent', 'Kent',
births
```

Out[38]:

	Name	Births	Country	Gender	Percentage	County
0	Adam	1508	UK	Male	8.780204	Yorkshire
1	Sophie	1929	UK	Female	11.231441	Essex
2	Charlie	3336	UK	Male	19.423581	Yorkshire
3	Anna	409	UK	Female	2.381368	Yorkshire
4	Bobby	652	UK	Male	3.796215	Kent
5	Florence	1974	UK	Female	11.493450	Kent
6	George	4949	UK	Male	28.815138	Yorkshire
7	Mia	2418	UK	Female	14.078603	Essex

In [39]:

```
# Group the data based on Gender and then by County
births.groupby(['Gender', 'County']).sum()
```

Out[39]:

		Births	Percentage
Gender	**County**		
Female	**Essex**	4347	25.310044
	Kent	1974	11.493450
	Yorkshire	409	2.381368
Male	**Kent**	652	3.796215
	Yorkshire	9793	57.018923

In [ ]:

```

```