SAMSUNG

Together for Tomorrow!
Enabling People
Education for Future Generations

Samsung Innovation Campus

Artificial Intelligence Course

Together for Tomorrow!
Enabling People
Education for Future Generations

Chapter 3.

Python Libraries

AI Course

Samsung Innovation Campus

Together for Tomorrow!
Enabling People
Education for Future Generations

Chapter 3.

Python Libraries

UNIT 2.

Pandas Package

Samsung Innovation Campus

Unit 2.
Pandas Package

What this unit is about:

- This unit is an introduction to the **Pandas library**.
- You will learn how to create **Series** and **DataFrame** objects.
- You will learn how to **manipulate and operate** on the Series and DataFrame objects.
- You will learn **how to prepare** and wrangle over **data**.

Expected outcome:

- Ability to utilize Series and DataFrame objects.
- Ability to **transform data** as part of the **data preparation and pre-processing**.
- Ability to carry out exploratory data analysis (EDA).

How to check your progress:

- Coding Exercises.
- Quiz.

Chapter 3.

Python Libraries

Pandas Library (1/2)

What is Pandas?

- An open source Python library that provides **data structures** and **data analysis tools**.
- Provides **data wrangling, processing and modeling capabilities** for Python comparable to R language.
- **Highly optimized for performance** when dealing with data.

Pandas Library (2/2)

Structured vs Unstructured data:

- **Unstructured data**: such as **raw data obtained by web-scraping, log files**, etc.
- **Structured data**: such as data contained in **CSV files, Excel spreadsheets, SQL tables**, etc.

Pandas Series (1/6)

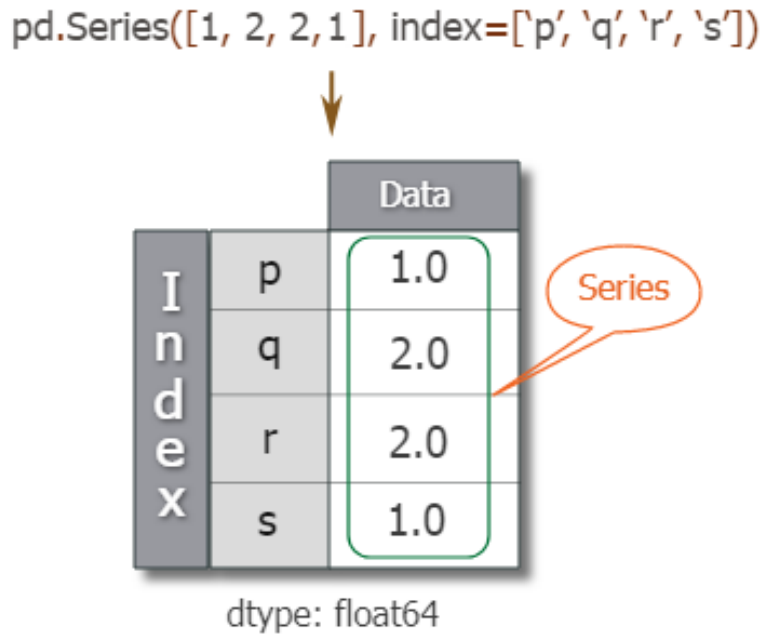
About **Pandas Series**:

- Similar to **one dimensional NumPy array**.
- Has **attributes such as index, name**, etc.
- Supports **vectorized operations**.

Pandas Series (1/6)

About **Pandas Series**:

- Series has **attributes such as index, name**, etc.
- Series is like a one dimentional array.



Pandas Series (2/6)

Creating a Series object:

```
In[1] : type(df)                                                    # 'df' is a DataFrame object.
Out[1]: pandas.core.frame.dataFrame
In[2] : type(df.a)                                                  # A column of 'df' is a Series object.
Out[2]: pandas.core.series.Series
In[3] : my_data = np.array([220, 215, 93,64])
In[4] : eye = pd.Series(data=my_data, index=['Brown','Blue','Hazel','Green'])   # Create a new Series object.
In[5] : eye
Out[5]:
Brown 220
Blue 215
Hazel 93
Green 64
dtype: int32
```

Pandas Series (3/6)

Series attributes and methods:

```
In[1] : ser.name                    # Name attribute of the Series object 'ser'.
In[2] : ser.index                   # Index attribute of the Series object 'ser'.
In[3] : ser.values                  # Values of the Series given as a NumPy array.
```

```
In[1] : ser.sort_values()           # Sort the Series values from the smallest to the largest.
In[2] : ser.sort_index()            # Sort the Series indices from the smallest to the largest.
In[3] : ser.nunique()               # Number of unique values in the Series.
In[4] : ser.unique()                # Unique values of the Series given as a NumPy array
In[5] : ser.value_counts()          # Returns a frequency table.
```

More information can be found at
https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Series.html

Pandas Series (4/6)

Indexing and slicing Series:

```
In[1] : ser = pd.Series([0,10,20,30,40], index = ['a','b','c','d','e'])      # Create a Series object.
In[2] : ser[1]                                                               # Element at the position 1.
Out[2]:
10
In[2] : ser['b']                                                  # Element at the position 'b'.
Out[3]:
10
In[4] : ser[1:3]                                                             # Get the elements at the positions 1 and 2.
Out[4]:
b   10
c   20
dtype: int64
```

Pandas Series (5/6)

Operations with Series:

```
In[1] : ser1 = pd.Series([0,1,2,3,4], index = [0,1,2,3,4])  # Create a Series.
In[2] : ser1 = pd.Series([0,1,2,3,4], index = [4,3,2,1,0])  # Create a Series.
In[3] : ser1 + ser2                                          # Element-wise addition matched by the index.
Out[3]:
0   4
1   4
2   4
3   4
4   4
dtype: int64
In[4] : ser3 = ser1 * ser2                                   # Element-wise multiplication matched by the index.
In[5] : ser4 = ser1/2                                        # Divide a Series by a scalar value.
```

Pandas Series (6/6)

Operations with Series:

In[1] : ser_height.apply(lambda x: x/100) # Apply element-wise a **lambda function**.

Out[1]:

```
0     1.653
1     1.701
2     1.750
3     1.821
4     1.680
5     1.620
6     1.552
7     1.769
8     1.785
9     1.761
10    1.671
11    1.800
12    1.622
13    1.761
14    1.582
15    1.686
16    1.692
Name: height, dtype: float64
```

Coding Exercise #0204

Follow practice steps on 'ex_0204.ipynb' file

Pandas DataFrame (1/5)

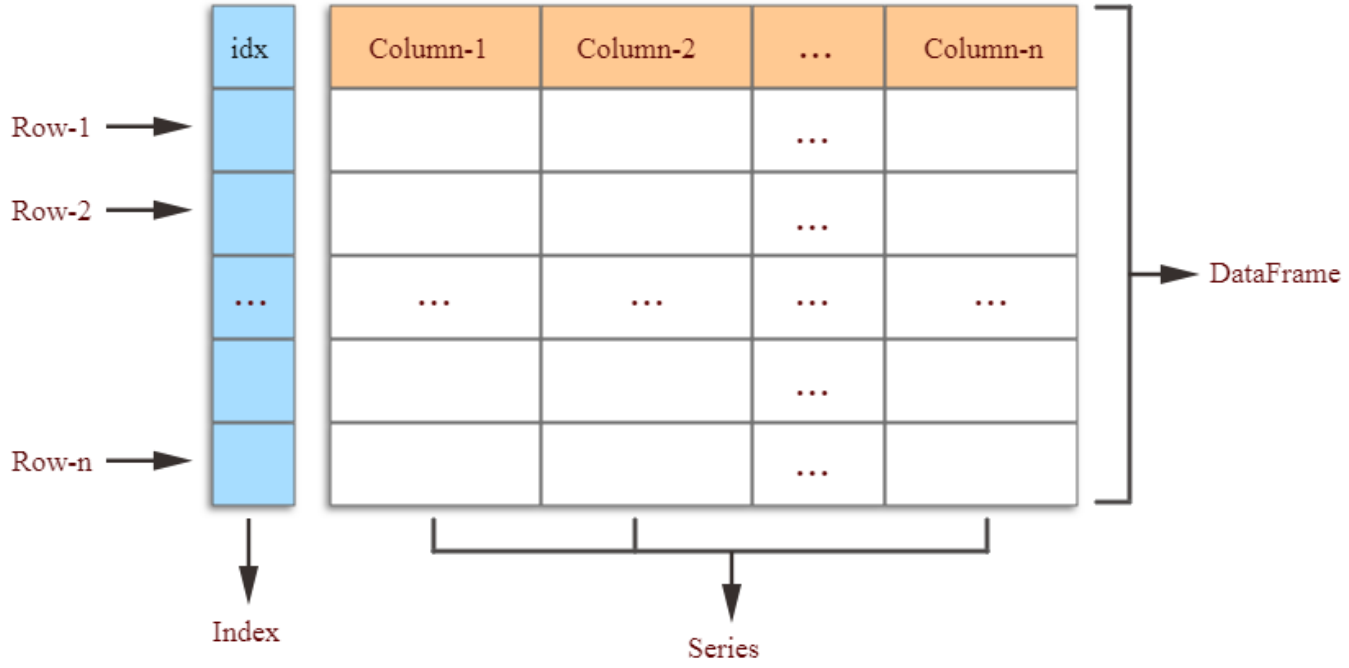
About **Pandas DataFrame**:

- Similar to **two-dimensional NumPy array**.
- The columns may have **different data types**.
- Ideal for holding data that was previously in **CSV files, Excel spreadsheets, SQL tables**, etc.
- Has **attributes such as columns, index**, etc.
- One can think of a DataFrame as a **collection of Series objects**.

Pandas DataFrame (1/5)

About **Pandas DataFrame**:

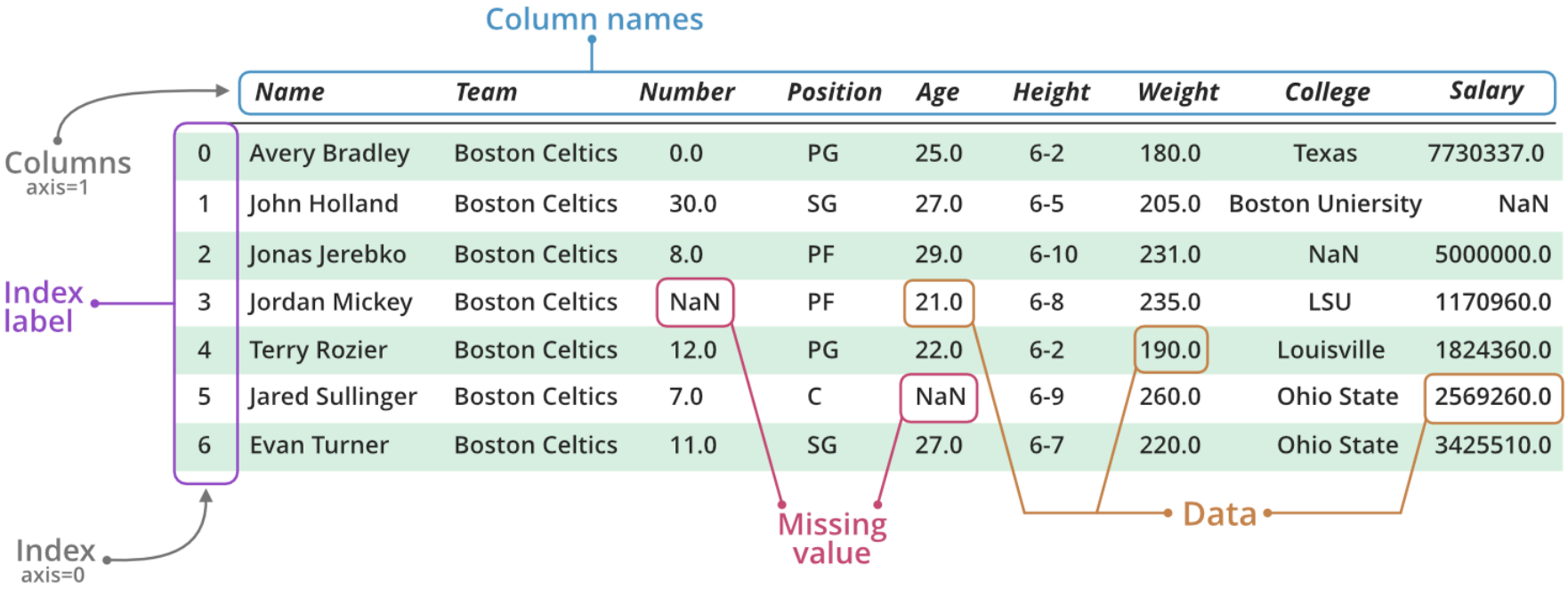
- Similar to **a 2D arraymensional array**.



Pandas DataFrame (1/5)

About **Pandas DataFrame**:

- Similar to **a 2D arraymensional array**.



	Name	Team	Number	Position	Age	Height	Weight	College	Salary
0	Avery Bradley	Boston Celtics	0.0	PG	25.0	6-2	180.0	Texas	7730337.0
1	John Holland	Boston Celtics	30.0	SG	27.0	6-5	205.0	Boston Uniersity	NaN
2	Jonas Jerebko	Boston Celtics	8.0	PF	29.0	6-10	231.0	NaN	5000000.0
3	Jordan Mickey	Boston Celtics	NaN	PF	21.0	6-8	235.0	LSU	1170960.0
4	Terry Rozier	Boston Celtics	12.0	PG	22.0	6-2	190.0	Louisville	1824360.0
5	Jared Sullinger	Boston Celtics	7.0	C	NaN	6-9	260.0	Ohio State	2569260.0
6	Evan Turner	Boston Celtics	11.0	SG	27.0	6-7	220.0	Ohio State	3425510.0

Pandas DataFrame (2/5)

Creating a DataFrame object:

```
In[1] : import pandas as pd                                          # Import Pandas, alias 'pd'.
In[2] : df = pd.read_csv('my_file.csv', header='infer', encoding='ISO-8859-1')    # Read in from a CSV file.
In[3] : type(df)
Out[3]: pandas.core.frame.dataFrame
```

```
In[1] : data = { 'NAME' : ['Jake', 'Jennifer', 'Paul', 'Andrew'], 'AGE': [24,21,25,19], 'GENDER':['M','F','M','M']}
In[2] : df = pd.DataFrame(data)                                      # Create a DataFrame from a dictionary object.
In[3] : df
Out[3]:
    AGE      GENDER      NAME
0   24          M         Jake
1   21          F       Jennifer
2   25          M         Paul
3   19          M       Andrew
```

Pandas DataFrame (3/5)

DataFrame **attributes and methods**:

```
In[1] : df.info()                          # Show the information such as non-null values, data types, etc.
In[2] : df.head(n)                         # Return the topmost n rows.
In[3] : df.tail(n)                         # Return the bottom n rows.
In[4] : df.columns                         # Show the DataFrame's column names.
In[5] : df.columns = ['A', 'B' , 'C',...]  # Replace the DataFrame's column names.
In[4] : df.index                           # Show the DataFrame's indices.
```

More information can be found at
https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Series.html

Pandas DataFrame (4/5)

Indexing and slicing DataFrame:

```
In[1] : df.A                                    # Get the column 'A' as a Series.
In[2] : df.B                                    # Get the column 'B' as a Series.
In[3] : df['A']                                 # Get the column 'A' as a Series.
In[4] : df['A','B']]                            # Get the columns 'A' and 'B'. Display as a DataFrame.
In[5] : df.loc[:, ['A','B']]                    # Get the columns 'A' and 'B'. Display as a DataFrame.
In[6] : df.loc[:, (header == 'A' ) | (header == 'B' )]  # Get the columns 'A' and 'B'. Display as a DataFrame.
In[7] : df.iloc[:, [0,1]]                       # Get the columns 0 and 1. Display as a DataFrame.
In[8] : df.loc[n]                               # Get the row n.
In[9] : df.iloc[n]                              # Get the row n.
In[10]: df.loc[n:m]                             # Display the rows n through m.
In[11]: df.iloc[n:m]                            # Display the rows n through m-1.
In[12]: df.drop(columns=['B','C'])              # All the columns excluding 'B' and 'C'.
In[13]: df.loc[:, (header != 'B' ) & (header != 'C' )]  # All the columns excluding 'B' and 'C'.
```

Pandas DataFrame (5/5)

Conditional selection of DataFrame rows:

```
In[1] : df[ df.GENDER == 'M' ]                                    # Display the rows where GENDER is 'M'.
In[2] : df[ -(df.GENDER == 'M') ]                                 # Display the rows where GENDER is not 'M'.
In[3] : df[ df.HEIGHT > 170 ]                                     # Display the rows where HEIGHT > 170.
In[4] : df[ (df.HEIGHT > 170) & (df.HEIGHT < 180) ]               # Combine the conditions with AND.
In[5] : df[ (df.GENDER == 'M' ) & (df.HEIGHT < 180) ]   # Combine the conditions with AND.
In[6] : df[ (df.HEIGHT < 160) | (df.HEIGHT > 180) ]               # Combine the conditions with OR.
In[7] : df[ (df.GRADE == 1 ) | (df.GRADE ==4 ) ]                  # Combine the conditions with OR.
In[8] : df[ (df.GENDER == 'M' ) & ((df.HEIGHT < 160) | (df.HEIGHT > 180) ) ]          # Combine the conditions.
```

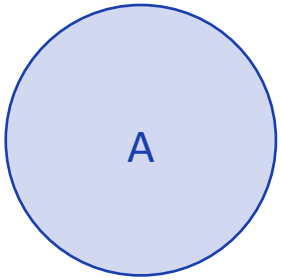
Coding Exercise #0205

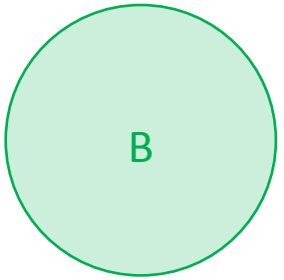
Follow practice steps on 'ex_0205.ipynb' file

Python Libraries

DataFrame Manipulation (1/13)

Merging DataFrames:



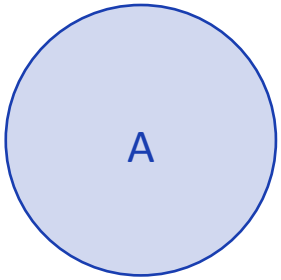


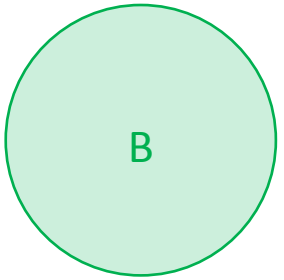
Name	Gender	Age
Harry Potter	Male	23
David Baker	Male	31
John Smith	Male	22
Juan Martinez	Male	36
Jane Connor	Female	30

Name	Position	Wage
John Smith	Intern	25000
Alex Du Bois	Team Lead	75000
Joanne Rowling	Manager	90000
Jane Connor	Manager	70000

DataFrame Manipulation (2/13)

Merging DataFrames: A.Name and B.Name used as key.



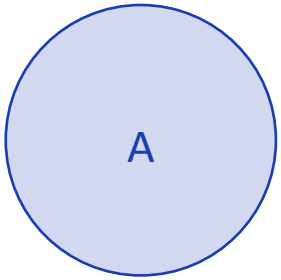


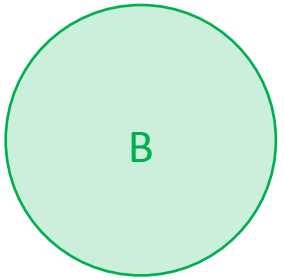
Name	Gender	Age
Harry Potter	Male	23
David Baker	Male	31
John Smith	Male	22
Juan Martinez	Male	36
Jane Connor	Female	30

Name	Position	Wage
John Smith	Intern	25000
Alex Du Bois	Team Lead	75000
Joanne Rowling	Manager	90000
Jane Connor	Manager	70000

DataFrame Manipulation (3/13)

Merging DataFrames: **Inner join**.



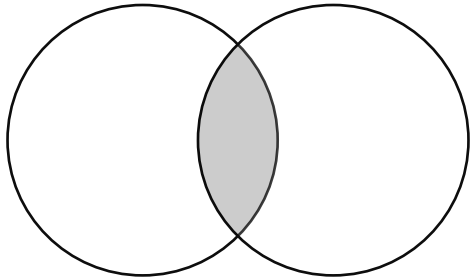


Name	Gender	Age
~~Harry Potter~~	~~Male~~	~~23~~
~~David Baker~~	~~Male~~	~~31~~
John Smith	Male	22
~~Juan Martinez~~	~~Male~~	~~36~~
Jane Connor	Female	30

Name	Position	Wage
John Smith	Intern	25000
~~Alex Du Bois~~	~~Team Lead~~	~~75000~~
~~Joanne Rowling~~	~~Manager~~	~~90000~~
Jane Connor	Manager	70000

DataFrame Manipulation (4/13)

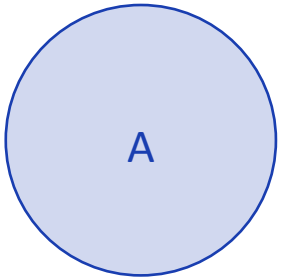
Merging DataFrames: Inner join.

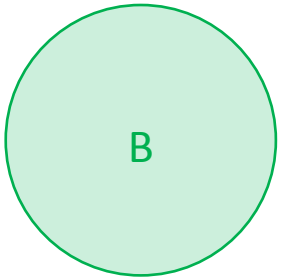


A.Name	Gender	Age	B.Name	Position	Wage
John Smith	Male	22	John Smith	Intern	25000
Jane Connor	Female	30	Jane Connor	Manager	70000

DataFrame Manipulation (5/13)

Merging DataFrames: **Left join.**



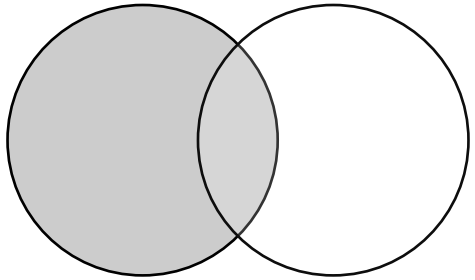


Name	Gender	Age
Harry Potter	Male	23
David Baker	Male	31
John Smith	Male	22
Juan Martinez	Male	36
Jane Connor	Female	30

Name	Position	Wage
John Smith	Intern	25000
~~Alex Du Bois~~	~~Team Lead~~	~~75000~~
~~Joanne Rowling~~	~~Manager~~	~~90000~~
Jane Connor	Manager	70000

DataFrame Manipulation (6/13)

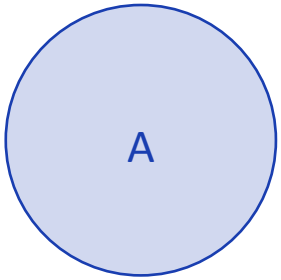
Merging DataFrames: Left join.

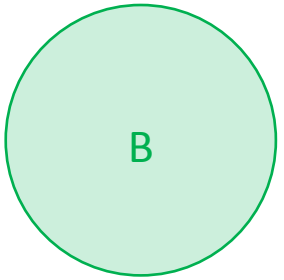


A.Name	Gender	Age	B.Name	Position	Wage
Harry Potter	Male	23	NA	NA	NA
David Baker	Male	31	NA	NA	NA
John Smith	Male	22	John Smith	Intern	25000
Juan Martinez	Male	36	NA	NA	NA
Jane Connor	Female	30	Jane Connor	Manager	70000

DataFrame Manipulation (7/13)

Merging DataFrames: **Right join**.



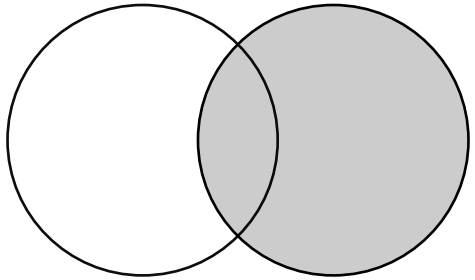


Name	Gender	Age
~~Harry Potter~~	~~Male~~	~~23~~
~~David Baker~~	~~Male~~	~~31~~
John Smith	Male	22
~~Juan Martinez~~	~~Male~~	~~36~~
Jane Connor	Female	30

Name	Position	Wage
John Smith	Intern	25000
Alex Du Bois	Team Lead	75000
Joanne Rowling	Manager	90000
Jane Connor	Manager	70000

DataFrame Manipulation (8/13)

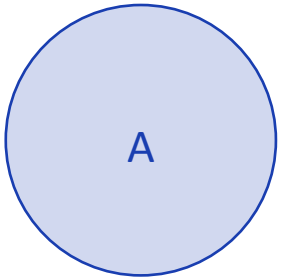
Merging DataFrames: Right join.

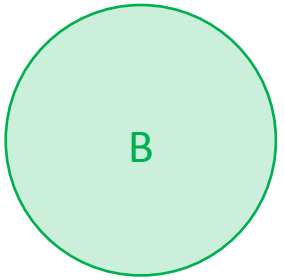


A.Name	Gender	Age	B.Name	Position	Wage
John Smith	Male	22	John Smith	Intern	25000
NA	NA	NA	Alex Du Bois	Team Lead	75000
NA	NA	NA	Joanne Rowling	Manager	90000
Jane Connor	Female	30	Jane Connor	Manager	70000

DataFrame Manipulation (9/13)

Merging DataFrames: **Full outer join**.



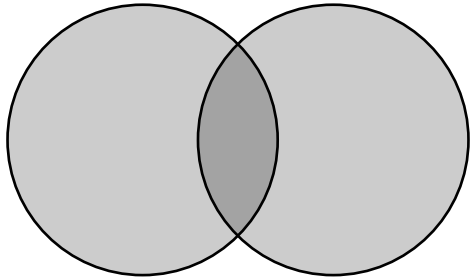


Name	Gender	Age
Harry Potter	Male	23
David Baker	Male	31
John Smith	Male	22
Juan Martinez	Male	36
Jane Connor	Female	30

Name	Position	Wage
John Smith	Intern	25000
Alex Du Bois	Team Lead	75000
Joanne Rowling	Manager	90000
Jane Connor	Manager	70000

DataFrame Manipulation (10/13)

Merging DataFrames: Full outer join.



A.Name	Gender	Age	B.Name	Position	Wage
Harry Potter	Male	23	NA	NA	NA
David Baker	Male	31	NA	NA	NA
John Smith	Male	22	John Smith	Intern	25000
Juan Martinez	Male	36	NA	NA	NA
Jane Connor	Female	30	Jane Connor	Manager	70000
NA	NA	NA	Alex Du Bois	Team Lead	75000
NA	NA	NA	Joanne Rowling	Manager	90000

DataFrame Manipulation (11/13)

Merging and binding DataFrames:

```
In[1] : pd.merge(df_left, df_right, on='NAME’)                                          # Inner join.
In[2] : pd.merge(df_left, df_right, left_on='NAME', right_on = 'NAME', how='inner')     # Inner join.
In[3] : pd.merge(df_left, df_right, left_on='NAME', right_on = 'NAME', how='left')# Left join.
In[4] : pd.merge(df_left, df_right, left_on='NAME', right_on = 'NAME', how='right')     # Right join.
In[5] : pd.merge(df_left, df_right, left_on='NAME', right_on = 'NAME', how='outer')     # Full outer join.
```

```
In[1] : pd.concat([df_A, df_B], sort=True)              # Bind vertically by matching the column names.
In[2] : pd.concat([df_A, df_B], axis=1, sort=True)      # Bind horizontally by matching the indices.
```

Coding Exercise #0206

Follow practice steps on 'ex_0206.ipynb' file