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

Chapter Description (1/2)

Duration: 22 Hours.

Purpose:

- **Introduction to the NumPy library.**
- Introduction to linear algebra: vector and matrix operations.
- **Introduction to the Pandas library.**
- **Introduction to the Matplotlib and Seaborn libraries: visualization techniques.**

Target Audience:

- This course is intended for those who are willing to go beyond the basic Python programming.
- This course is a prerequisite for machine learning and deep learning.

Prerequisite:

- A student taking this course should be able to **program in Python.**
- A student taking this course should be somewhat familiar with vectors and matrices.
- A student taking this course should be familiar with Excel spreadsheets.

Chapter Description (2/2)

Objectives:

- **Create, manipulate and operate with NumPy arrays.**
- Understand the basics of linear algebra.
- Create, manipulate and operate with Series objects.
- **Create, manipulate and operate with DataFrame objects.**
- **Utilize data summarization, aggregation and transformation techniques.**
- **Create basic as well as advanced visualization types.**
- Carry out exploratory data analysis (EDA).

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

DataFrame Manipulation (12/13)

Sorting:

- It is possible to sort the rows of a DataFrame using one or more columns.

```
In[1] : df.sort_values(by='bloodtype')                          # Sort in ascending order.
In[2] : df.sort_values(by='bloodtype', ascending=False)         # Sort in descending order.
In[3] : df.sort_values(by=['bloodtype','gender’])               # Sort using two columns.
```

DataFrame Manipulation (13/13)

Hierarchical indexing with MultiIndex:

```
In[1] : my_header = ['a','b','c']                                          # Column names.
In[2] : my_index_out = ['G1']*3 + ['G2']*3                                 # Labels for the outer layer.
In[3] : my_index_in = [1,2,3]*2                                            # Labels for the inner layer.
In[4] : my_index_zipped = list(zip(my_index_out, my_index_in))     # Create a list of tuples with the labels.
In[5] : my_index = pd.MultiIndex.from_tuples(my_index_zipped)      # Create the MultiIndex.
In[6] : df = pd.DataFrame(data=np.random.randn(6,3),index=my_index,columns=my_header)    # Apply the MultiIndex.
In[7] : df
```

Out[7]:

		a	b	c
G1	**1**	0.708643	1.526325	-0.522276
	2	-0.112115	0.366355	-0.127317
	3	-0.020839	0.023037	0.167214
G2	**1**	-1.300622	-0.310416	-0.840097
	2	0.088279	-1.596302	1.367721
	3	-0.998479	-0.476471	-2.038437

DataFrame Summarization (1/7)

Grouping and summarizing:

```
In[1] : df.groupby('gender').mean()                          # Average of all the numeric variables by gender types.
In[2] : df.groupby('gender')['height'].mean()                # Average height by gender types.
In[3] : df.groupby('gender')[['height','weight']].mean()     # Average height and weight by gender types.
In[4] : df.groupby('gender')['height'].describe()            # Statistical summary of height by gender types.
In[5] : df.groupby(['gender','bloodtype'])['height'].mean()  # Average height by gender and bloodtype types.
```

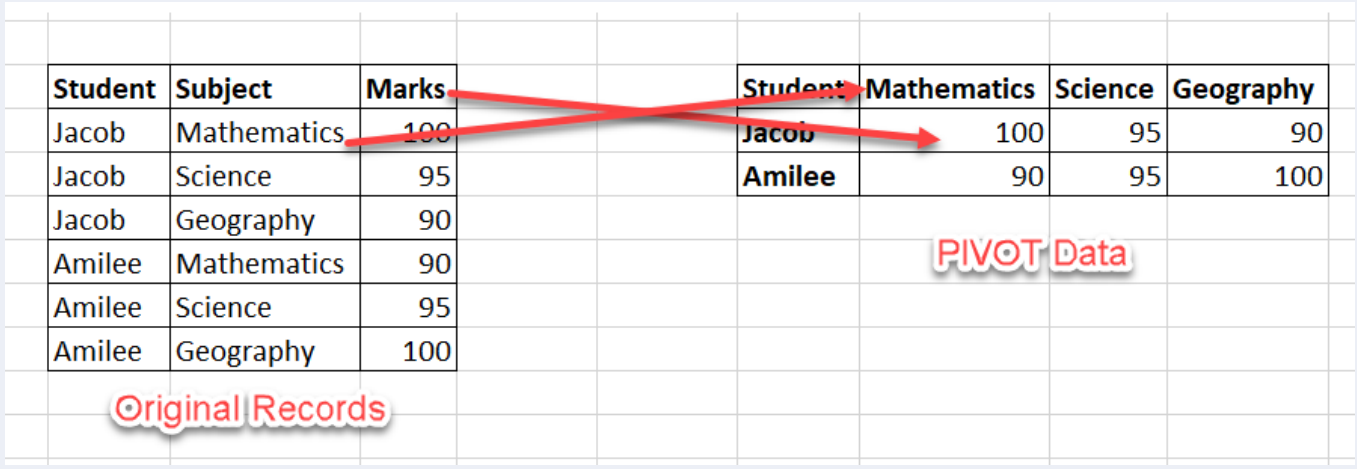
Out[5]:

```
gender  bloodtype
F       A             172.450000
        AB            170.100000
        B             158.200000
        O             164.433333
M       A             165.700000
        AB            181.050000
        B             174.550000
        O             166.200000
Name: height, dtype: float64
```

DataFrame Summarization

Pivoting:

- A **pivot table** is a table of grouped values that aggregates the individual items of a more extensive table within one or more discrete categories.



DataFrame Summarization (2/7)

Pivoting:

- Manipulate the indices and the columns and then summarize.

```
In[1] : # Use the columns 'A' and 'B' as indices, and leave the column 'C' as such.
In[2] : # Then summarize by calculating the average of the column 'E'. numpy.mean() is the default.
In[3] : pd.pivot_table(df, index=['A','B'], columns='C', values='E')
```

Out[3]:

	C	large	small
A	B		
bar	one	6.0	8.0
	two	9.0	9.0
foo	one	4.5	2.0
	two	NaN	5.5

DataFrame Summarization (3/7)

Pivoting:

- Manipulate the indices and the columns and then summarize.

```
In[1] : # Use the columns 'A' and 'B' as indices, and leave the column 'C' as such.
In[2] : # Then summarize by calculating the median of the column 'E'. Also, fill the missing values with 0.
In[3] : pd.pivot_table(df, index=['A','B'], columns='C', values='E', aggfunc=np.median, fill_value=0)
```

Out[3]:

	C	large	small
A	B		
bar	one	6.0	8.0
	two	9.0	9.0
foo	one	4.5	2.0
	two	0.0	5.5

DataFrame Summarization (4/7)

Pivoting:

- Manipulate the indices and the columns and then summarize.

```
In[1] : # Use the columns 'A' and 'B' as indices.
In[2] : # Then summarize by calculating the averages of the columns 'D' and 'E'.
In[3] : pd.pivot_table(df, index=['A','B'], values=['D','E'], aggfunc=np.mean)
```

Out[3]:

		D	E
A	**B**		
bar	**one**	4.500000	7.000000
	two	6.500000	9.000000
foo	**one**	1.666667	3.666667
	two	3.000000	5.500000

DataFrame Summarization (5/7)

Pivoting:

- Manipulate the indices and the columns and then summarize.

```
In[1] : # Use the columns 'A' and 'B' as indices.
In[2] : # Then summarize the columns 'D' and 'E' using different statistical functions.
In[3] : pd.pivot_table(df, index=['A','B'], values=['D','E'], aggfunc={'D':np.mean,'E':np.median})
```

Out[3]:

		D	E
A	B		
bar	one	4.500000	7.0
	two	6.500000	9.0
foo	one	1.666667	4.0
	two	3.000000	5.5

DataFrame Summarization (6/7)

Statistics:

```
In[1] : df.sum(axis=0)                    # Column sums.
In[2] : df.sum(axis=1)                    # Row sums.
In[3] : df.mean(axis=0, skipna=False      # Column averages without skipping the missing values.
In[4] : df.describe()                     # Descriptive statistics of the columns (variables).
In[5] : df.count(axis=0)                  # Non-missing values along the columns.
In[6] : df.A.corr(df.B)                   # Correlation between the column 'A' and the column 'B'.
In[7] : df.corr()                         # Correlation matrix taking the numeric variables pair-wise.
In[8] : df.corrwith(df.A)                 # Correlations between 'A' and the other numeric variables.
```

DataFrame Summarization (7/7)

Missing value detection and processing:

```
In[1] : df.isnull()                          # A DataFrame with True where missing values are found.
In[2] : (df.isnull()).sum(axis=0)            # Count the missing values for each column.
In[3] : (df.isnull()).mean(axis=0)           # Proportions of the missing values for each column.
In[4] : df.dropna(axis = 0)          # Drop rows where one or more missing values are found.
In[5] : df.dropna(axis = 1)          # Drop columns where one or more missing values are found
In[6] : df.dropna(axis=0, thresh = 3)        # Drop the rows with less than 3 normal values.
In[7] : df.fillna(value=0)                   # Fill the missing values with 0.
```

Coding Exercise #0207

Follow practice steps on 'ex_0207.ipynb' file

Coding Exercise #0208

Follow practice steps on 'ex_0208.ipynb' file