Cognitive Computing UCS420
Pandas

Pandas

- Pandas is a Python library used for working with data sets.
- It has functions for analyzing, cleaning, exploring, and manipulating data.

Why Use Pandas?

- Pandas allows us to analyze big data and make conclusions based on statistical theories.
- Pandas can clean messy data sets, and make them readable and relevant.
- Relevant data is very important in data science.

What Can Pandas Do?

- Pandas gives you answers about the data. Like:
  - Is there a correlation between two or more columns?
  - What is average value?
  - Max value?
  - Min value?
- Pandas are also able to delete rows that are not relevant, or contains wrong values, like empty or NULL values. This is called *cleaning* the data.

Pandas DataFrames

- A Pandas DataFrame is a 2-dimensional data structure, like a 2-dimensional array, or a table with rows and columns.
- Pandas DataFrame is a two-dimensional, size-mutable, and heterogeneous data structure (similar to a table in a relational database or an Excel spreadsheet).

Pandas DataFrames

- Example:
- Create a simple Pandas DataFrame:

-
  ```
  import pandas as pd

  data = {
    ’calories’: [420, 380, 390],
    ’duration’: [50, 40, 45]
  }

  #load data into a DataFrame object:
  df = pd.DataFrame(data)

  print(df)
  ```

Pandas DataFrames

- Locate Row:
- As you can see from the result (previous slide), the DataFrame is like a table with rows and columns.
- Pandas use the **loc** attribute to return one or more specified row(s).

Pandas DataFrames

- Example-: Return row 0:
- #refer to the row index:
  print(df.loc[0])

Example

-
  ```
  import pandas as pd

  mydataset = {
    'cars': ["BMW", "Volvo", "Ford"],
    'passings': [3, 7, 2]
  }

  myvar = pd.DataFrame(mydataset)

  print(myvar)
  ```

Pandas Series

- A Pandas Series is like a column in a table.
- It is a one-dimensional array holding data of any type.
- Example:
- Create a simple Pandas Series from a list:
- import pandas as pd

  ```
  a = [1, 7, 2]

  data1 = pd.Series(a)

  print(data1)
  ```

Labels

- If nothing else is specified, the values are labeled with their index number. First value has index 0, second value has index 1 etc.
- This label can be used to access a specified value.
- With the index argument, you can name your own labels.

Labels

- Example
- Create your own labels:
- import pandas as pd

  ```
  a = [1, 7, 2]

  myvar = pd.Series(a, index = ["x", "y", "z"])

  print(myvar)
  ```

Navigating Data Frame

- **iloc** exclusively uses integer positions for accessing data.
- As a result, it makes it particularly useful when dealing with data where labels might be unknown or irrelevant.

- df.iloc[row number/slice]
- df.iloc[4], df.iloc[1:4], df.iloc[:], df.iloc[1:4, 5:8]

Navigating Data Frame

- df.iloc[4]-This command selects the 5th row (index 4) from the DataFrame df. It returns a single row as a Series.
- df.iloc[1:4]:This command selects a slice of rows from index 1 to 3 (excluding index 4) from df. It returns multiple rows as a DataFrame.

Navigating Data Frame

- df.iloc[:]
- -This command selects all rows and columns from df. It’s essentially the same as df, returning the entire DataFrame.
- df.iloc[1:4, 5:8]:This command selects rows from index 1 to 3 (excluding 4) and columns from index 5 to 7 (excluding 8). It returns the specified subset as a DataFrame.

Navigating Data Frame

- df.iloc[:,2]-This will select all rows (:) for the specified column index (3rd column), effectively giving you the entire column without specifically extracting any single row.
- This is the closest way to extract a column with .iloc without targeting individual rows.

Pandas Read CSV

- A simple way to store big data sets is to use CSV files (comma separated files).
- CSV files contains plain text and is a well know format that can be read by everyone including Pandas.
- In our examples we will be using a CSV file (Download from Kaggle).

Pandas Read CSV

- Example:
- Load the CSV into a DataFrame:
- import pandas as pd

  df = pd.read_csv('data.csv’)
- #show only first 5 rows
- df.head()
- #show all the rows
- print(df.to_string())
- #show last 5 rows
- print("\nLast 5 rows:")
- print(df.tail(5))

Pandas Read CSV

- The pd.read_csv() function is used to read the data from the data.csv file.
- df.to_string() converts the entire DataFrame df into a string representation, showing all rows and columns.
- If you have a large DataFrame with many rows, Pandas will only return the first 5 rows, and the last 5 rows:

Delete a column from Dataset

- You can delete a column or feature from a dataset-
  - df.drop(df.columns[1], axis=1, inplace=True)

•**Column Selection**: df.columns[1] is used to select the second column.
•**Axis Parameter**: axis=1 specifies you are dropping a column. For rows, use axis=0.
•**Inplace=True -** If you want to modify the DataFrame in place.
•**Inplace=False-** If you do not want to modify the DataFrame in place

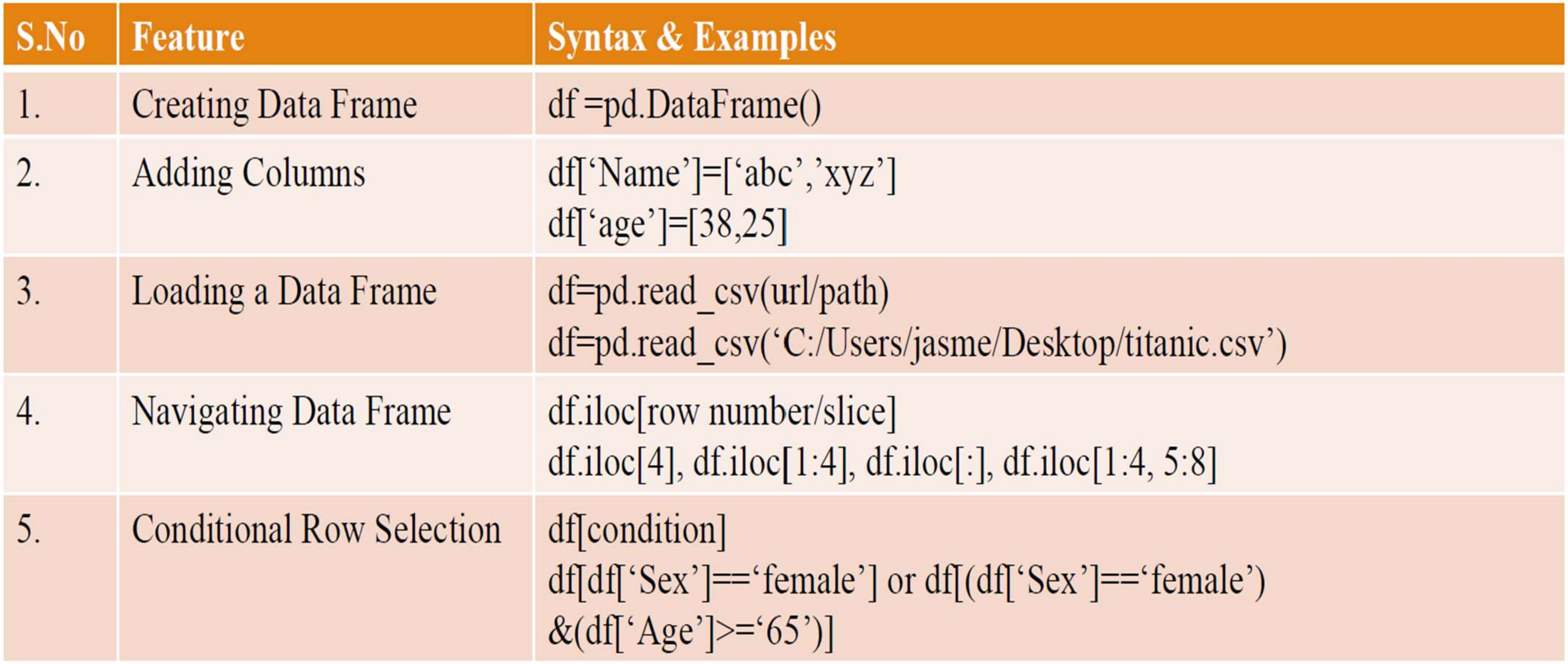
Delete a row from Dataset

- You can delete a row or feature from a dataset-
  - df.drop(1, axis=0, inplace=True)

- **Row Selection**: The first parameter ‘1’ is used to select the first row.
- **Axis Parameter**: axis=0 specifies you are dropping a row.
- **Inplace=True -** If you want to modify the DataFrame in place.
- **Inplace=False-** If you do not want to modify the DataFrame in place

Pandas-Some other useful commands

Import pandas as pd



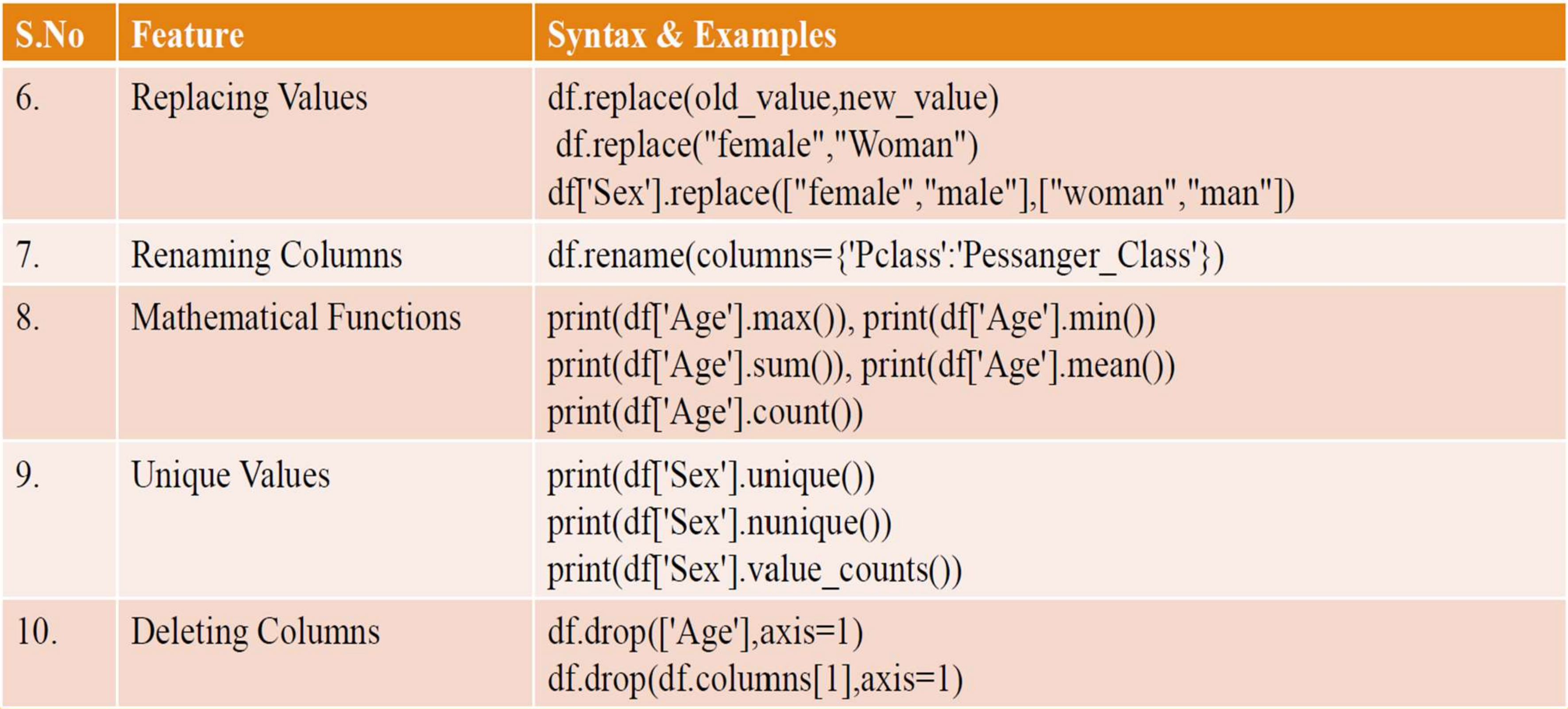
S.No	Feature	Syntax & Examples
1.	Creating Data Frame	df =pd.DataFrame()
2.	Adding Columns	df[‘Name’]=[‘abc’,’xyz’] df[‘age’]=[38,25]

3.	Loading a Data Frame	df=pd.read_csv(url/path)
		df=pd.read_csv(‘C:/Users/jasme/Desktop/titanic.csv’)

4.	Navigating Data Frame	df.iloc[row number/slice] df.iloc[4], df.iloc[1:4], df.iloc[:], df.iloc[1:4, 5:8]
5.	Conditional Row Selection	df[condition] df[df[‘Sex’]==‘female’] or df[(df[‘Sex’]==‘female’) &(df[‘Age’]>=‘65’)]

Pandas-Some other useful commands

Import pandas as pd



S.No	Feature	Syntax & Examples

6.	Replacing Values	df.replace(old_value,new_value)
		df.replace("female","Woman")
		df['Sex'].replace(["female","male"],["woman","man"])

7.	Renaming Columns	df.rename(columns={'Pclass':'Pessanger_Class'})

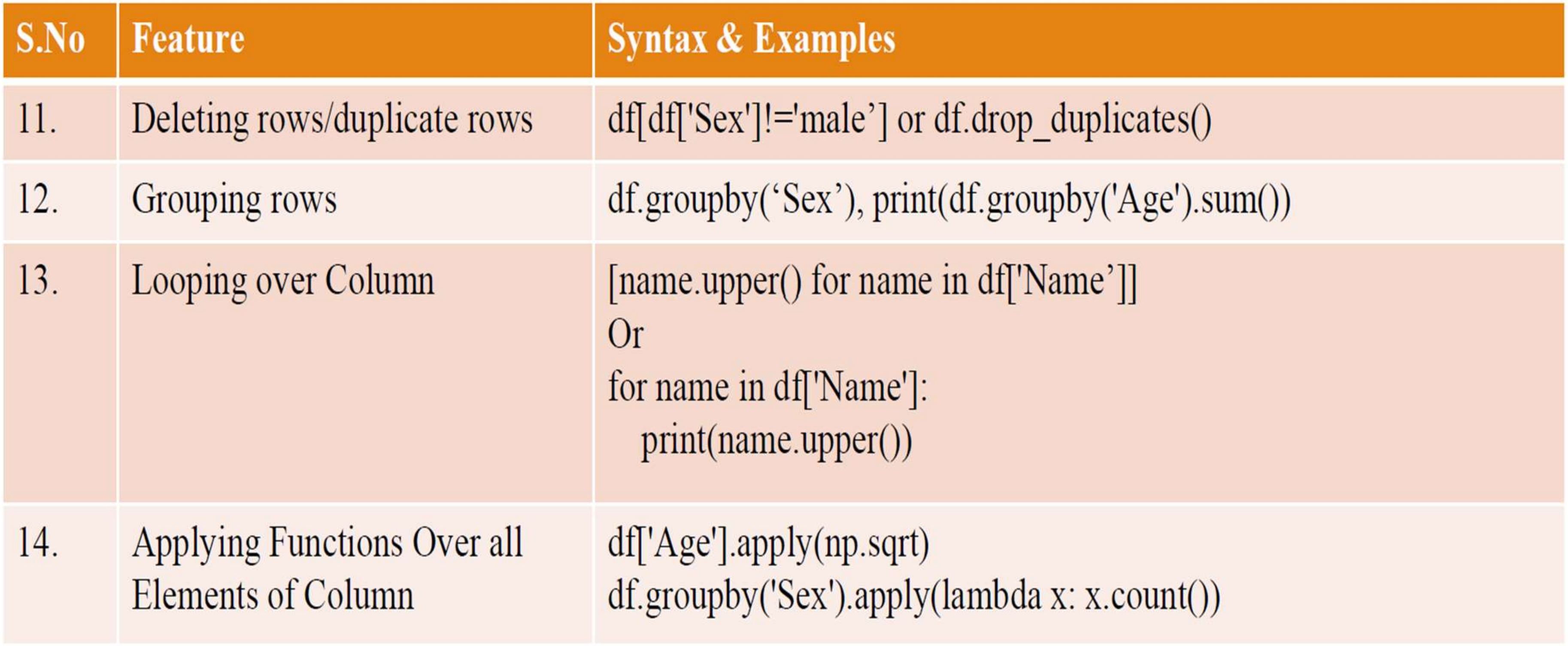
8.	Mathematical Functions	print(df['Age'].max()), print(df['Age'].min())
		print(df['Age'].sum()), print(df['Age'].mean())
		print(df['Age'].count())

9.	Unique Values	print(df['Sex'].unique())
		print(df['Sex'].nunique())
		print(df['Sex'].value_counts())

10.	Deleting Columns	df.drop(['Age'],axis=1)
		df.drop(df.columns[1],axis=1)

Pandas-Some other useful commands

Import pandas as pd



S.No	Feature	Syntax & Examples
11.	Deleting rows/duplicate rows	df[df['Sex']!='male’] or df.drop_duplicates()
12.	Grouping rows	df.groupby(‘Sex’), print(df.groupby('Age').sum())
13.	Looping over Column	[name.upper() for name in df['Name’]] Or for name in df['Name']: print(name.upper())
14.	Applying Functions Over all Elements of Column	df['Age'].apply(np.sqrt) df.groupby('Sex').apply(lambda x: x.count())