Python for Data Science

Agenda

1. Pop Quiz
2. Common Python Libraries for Data Science
3. NumPy and Pandas
4. Common NumPy functions
5. Common Pandas functions
6. Merge vs Join in Pandas
7. Example of Join

Pop Quiz

1. What are the data types in Python?
2. What are some of the common Python libraries for Data Science?
3. Can you list some of the common functions in Pandas?
4. What are the applications of the functions like group by, merge, join etc?

Common Python Libraries for Data Science

Library	Use
NumPy	Handling multi-dimensional arrays
Scipy	Scientific computation package
Matplotlib, Seaborn	Data visualisation
Pandas	Handling tabular data
Scikit-learn	Machine learning

NumPy

- Stands for Numerical Python
- It is one of the fundamental packages for mathematical, logical, and statistical operations with Python
- It contains
    - Powerful N-dimensional array object, called ndarray
    - Large set of functions for creating, manipulating, and transforming ndarrays
- ndarrays can only contain data of a single datatype
- Useful in linear algebra, vector calculus, random number capabilities, etc

Pandas

- Pandas is one of the fundamental packages for analysis and manipulation of tabular data
- Offers two major data structures - series & dataframe
- We can think of a pandas dataframe like an excel spreadsheet that is storing some data in rows and columns.
- A pandas dataframe is made up of several pandas series
    - Each column of a dataframe is a series.
- Pandas dataframes can contain data of multiple datatypes

Common NumPy Functions

Function	Description
np.array()	To create an array
np.arange()	Return evenly spaced values within a given interval
np.linspace()	Return evenly spaced numbers over a specified interval
np.zeros()	To create an array of zeros
np.ones()	To create an array of ones
np.transpose()	Permute array dimensions

Common NumPy Functions

Function	Description
np.random.rand()	To create an array of specified shape filled with random values
np.random.randint()	Return random integers from low (inclusive) to high (exclusive)
np.random.randn()	Return a sample (or samples) from the “standard normal” distribution.
np.concatenate()	Concatenate two arrays
np.save()	Save an array to a binary file in .npy format.
np.savez()	Save several arrays into a single file in uncompressed .npz format.

Common Pandas Functions

Function	Description
pd.read_csv()	Read a comma-separated values (csv) file into DataFrame
df.loc[]	Access a group of rows and columns by label(s)
df.iloc[]	Purely integer-location based indexing for selection by position
df.drop()	Drop specified labels from rows or columns
pd.concat()	To concatenate two pandas objects
pd.merge()	To merge the pandas dataframes
df.groupby()	To split, apply or combine the data structures

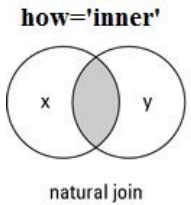
Common Pandas Functions

Function	Description
df.value_counts()	To get count of some attributes
df.unique()	To get unique values
df.dtype	To get the data types
df.shape	To get the shape (number or rows and columns)
df.head()	To get the top rows
df.tail()	To get the last rows
df.describe()	To get the quick statistic summary

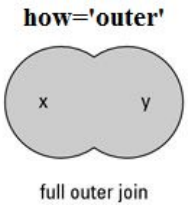
Merge vs Join

- **Join:** The **join** method works best when we are joining dataframes on their indexes (though you can specify another column to join on for the left dataframe).
- **Merge:** The **merge** method is more versatile and allows us to specify columns besides the index to join on for both dataframes.

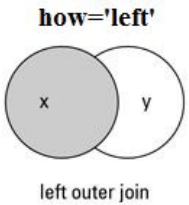
Natural join - Intersection	Full outer join - Union	Left outer join	Right outer join
To keep only rows that match from the data frames how=**'inner'.**	To keep all rows from both data frames, how=**'outer'.**	To include all the rows of your data frame x and only those from y that match how =**'left'.**	To include all the rows of your data frame y and only those from x that match, how=**'right'.**



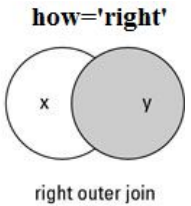
natural join



full outer join



left outer join



right outer join

Example of Join

Left Join

Index	Customer_id	Product
0	1	Oven
1	2	Oven
2	3	Oven
3	4	Television
4	5	Television
5	6	Television

Left Table

Index	Customer_id	State
0	2	California
1	4	California
2	6	Texas

Right Table

Index	Customer_id	Product	State
0	1	Oven	nan
1	2	Oven	California
2	3	Oven	nan
3	4	Television	California
4	5	Television	nan
5	6	Television	Texas

After Left Join datasets on Customer_id

Syntax :
```
merged = pd.merge(left, right, on = 'Customer_id', how = 'left')
```

Right Join

Index	Customer_id	Product
0	1	Oven
1	2	Oven
2	3	Oven
3	4	Television
4	5	Television
5	6	Television

Left Table

Index	Customer_id	State
0	2	California
1	4	California
2	6	Texas

Right Table

Index	Customer_id	State	State
0	2	Oven	California
1	4	Television	California
2	6	Television	Texas

After right Join datasets on Customer_id

Syntax :
```
merged = pd.merge(left, right, on = 'Customer_id', how = 'right')
```

Example of Join

Inner Join

Index	Id	Name	Age
0	1	Alex	25
1	2	Amy	23
2	3	Allen	22
3	4	Alice	21
4	5	Ayoung	24

Left Table

Index	Id	Subject
0	1	sub2
1	2	sub4
2	3	sub3
3	4	sub6
4	5	sub5

Right Table

Index	Id	Name	Age	Subject
0	1	Alex	25	sub2
1	2	Amy	23	sub4
2	3	Allen	22	sub3
3	4	Alice	21	sub6
4	5	Ayoung	24	sub5

After Merging datasets on Id

Syntax :
```
merged = pd.merge(left, right, on = 'id')
```

Outer/Full Join

Index	Customer_id	Product
0	1	Oven
1	2	Oven
2	3	Oven
3	4	Television
4	5	Television
5	6	Television

Left Table

Index	Customer_id	State
0	2	California
1	4	California
2	6	Texas

Right Table

Index	Customer_id	Product	State
0	1	Oven	nan
1	2	Oven	California
2	3	Oven	nan
3	4	Television	California
4	5	Television	nan
5	6	Television	Texas

After Outer Join datasets on Customer_id

Syntax :
```
merged = pd.merge(left, right, on = 'Customer_id', how = 'outer')
```

Great Learning

Happy Learning !