Importing the Pandas library

Relevel
by Unacademy

Importing the Pandas library

Syntax: import pandas

Once pandas are imported, we can use different functions in pandas libraries to manipulate dataframes and SeriesSeries.

Generally, it's imported under pd alias.

example: import pandas as pd

After this, pandas can be referred to as pd.

Introduction to SeriesSeries

Series is a one-dimensional labelled array that can hold data of any type (integer, string, float, etc.)

A series can be created using various inputs like –

-
- Array
- Dict
- Scalar value or constant

We use the panda's library in pandas to handle and create Series. A Series in pandas is like a column in a table.

Move to jupyter notebook to section "Creating Pandas Series."

Introduction to Dataframe

Dataframe is a 2-dimensional data structure, like a 2-D array. We can see it as a potentially heterogeneous tabular data structure with rows and columns.

Dataframe consists of three main components: data, rows, and columns.

Name	Team	Number	Position	Age	Height	Weight	College	Salary
Jae Crowder	Boston Celtics	99.0	SF	25.0	6-6	235.0	Marquette	6796117.0
R.J. Hunter	Boston Celtics	28.0	SG	22.0	6-5	185.0	Georgia State	1148640.0
Jonas Jerebko	Boston Celtics	8.0	PF	29.0	6-10	231.0	NaN	5000000.0
Amir Johnson	Boston Celtics	90.0		29.0	6-9	240.0	NaN	12000000.0
Jordan Mickey	Boston Celtics	55.0	F	21.0	6-8	235.0	LSU	1170960.0

We use the panda's library to crea

Move to jupyter notebook to section "Creating Pandas DataFrame."

Introduction to Dataframe

Pandas dtype	Python type	Usage
object	str or mixed	Text or mixed numeric and non-numeric values
int64	int	Integer numbers
float64	float	Floating point numbers
bool	bool	True/False values
datetime64	NA	Date and time values
timedelta[ns]	NA	Differences between two datetimes
category	NA	Finite list of text values

Pandas is a high-level data manipulation tool built on the NumPy package. The fundamental data structure in Pandas is called the DataFrame. DataFrames are incredibly powerful as they allow you to store and manipulate tabular data in rows of observations and columns.

Numpy vs. Pandas

Common features of Numpy

- Using an easy and fast framework, NumPy enables to work on a homogenous dataset
- It helps to build data objects with multiple dimensions
- Provides robust matrix manipulation methods
- Helps to broadcast the applied operations

Common features of Pandas

- Joining and merging datasets can be done through pandas.
- Missing data and data alignment can be handled through pandas.
- It includes tools to read and write data in-memory data structures.
- It facilitates the transformation of high-dimensional data into lower-dimensional data through hierarchical axis indexin

Comparison between Numpy and Pandas

Pandas	Numpy
For tabular data, pandas is preferred.	For numerical data, numpy is preferred.
A powerful tool of pandas is Dataframe, and a series	Powerful tool of numpy is arrays.
Pandas consume a lot of memory	Numpy is efficient in terms of memory
Works with 2D tabular objects.	Works with multidimensional arrays
Can work with heterogeneous datatype	Requires homogenous datatype
Better performance when the number of rows is 50K or less	Better performance when the number of rows is 500k or more

Reading CSV file in Pandas

read_csv() method is used to read csv files using pandas. We can store csv data into data frames and then use it further.

**Move to jupyter notebook for example of reading csv file in pandas

https://colab.research.google.com/drive/13ZfEXewwFjY7J5bOswfDGTg64e6DghWu?usp=sharing

CSV Dataset

https://docs.google.com/spreadsheets/d/1l7wq20vVdK7Y2W1ZnJgorGXzds6PsnDy/edit?usp=sharing&ouid=105724640260392096470&rtpof=true&sd=true

Writing files in Pandas

Saving a pandas dataframe to csv or Excel

Pandas allow users to write the created dataframe in CSV or Excel format. We use to_csv() and to_excel() method to perform this task.

** **Move to jupyter notebook for examples related to Saving a dataframe to CSV or Excel**

Viewing basic details

Pandas describe() method is used to view basic statistical details of dataframe or SeriesSeries like mean, percentile, min, max values etc.

Syntax: DataFrame.describe(include=None, exclude=None)

Params:

Include: includes only mentioned datatype in describing data frame.

Exclude: excludes only mentioned datatype in describing data frame.

Return type: Statistical data frame summary

** **Move to jupyter notebook for example of viewing basic details of data**

Dealing with Columns in Pandas DataFrame

We can do many basic operations on a column like addition, insertion, deletion, renaming etc.

- **Selecting columns**

We can select the column(s) from DataFrame by calling the column name(s).

Move to jupyter notebook for example on selecting columns

- **Adding column**

We can declare a new list and add it to the existing dataframe to add a new column.

** **Move to jupyter notebook for example on adding a column to existing dataframe**

Dealing with Columns in Pandas DataFrame

- **Create a new column based on the existing column**

While working with pandas, there are several times when we will need to create a new column with the help of an existing column for creating a new variable.

- **Create a new column using the assign function**

Dataframe.assign() function create and assign new column(s) to a DataFrame. It returns a new (a copy) object with the new columns added to the original ones.

Move to jupyter notebook, for example, on creating a new column based on an existing column

- **Deleting Column**

We can use the drop() method to delete a column from dataframe

** **Move to jupyter notebook for example on deleting column in existing dataframe**

Dealing with Rows in Pandas DataFrame

- **Selecting Rows**

Row selection can be made using the .iloc[] method that we will study in the next section in this class

- **Adding Rows**

we use the concat() and append() functions to concatenate multiple data frames into a single dataframe

- **Deleting Rows**

We can use the drop method to delete the rows.

Move to jupyter notebook for example on adding rows and deleting rows in dataframe

Indexing

Indexing in pandas means selecting particular rows and columns of data from a DataFrame. It can also be known as Subset Selection.

Pandas support three types of indexing:

Dataframe.[ ]; This [ ] operator is also known as the indexing operator

Dataframe.loc[ ] : This method is used for labels

Dataframe.iloc[ ] : This method is used for positions or integer-based indexing

Indexing using indexing operator [] :

The indexing operator can be used for indexing by passing in the number of rows or mentioning column names.

Move to jupyter notebook for indexing operator example.

Indexing

Indexing using .loc[ ] :

This method helps to select the data by the row or column label. It can select subsets of rows or columns.

Move to jupyter notebook for .loc[] example

Extracting rows using Pandas .iloc[]

.iloc[] method is used when either the index is unknown or not numeric SeriesSeries. Rows can be extracted using an imaginary index if the position isn't visible in the data frame.

Move to jupyter notebook for .iloc[] exampl

Some other methods for indexing in DataFrame

Function	Description
Dataframe.head()	Return top n rows of a data frame.
Dataframe.tail()	Return bottom n rows of a data frame.
Dataframe.at[]	Access a single value for a row/column label pair.
Dataframe.iat[]	Access a single value for a row/ column pair by integer position.
Dataframe.where[]	returns same shape object and whose corresponding entries are true based on given condition

Move to jupyter notebook for examples related to other methods for indexing in Dataframe.

QUIZ

Q.1 Series can be created from

a) Array

b) Dictionary

c) Scaler value

d) All of them

Solution:

d) All of them

Explanation:

All of the above options can be used to create a series