Home Resources Data Science & Business Analytics Pandas Cheat Sheet for Data Science in Python

Pandas Cheat Sheet for Data Science in Python

By Simplilearn

Last updated on Sep 19, 2025

8963

Share This Article:

The pandas package will almost probably be used if you're keen on collaborating with data in Python. However, it's simple to forget the precise syntax for performing anything even after learning pandas - even in our virtual pandas course. For your convenience, we have developed a Pandas Cheat Sheet that lists the most typical pandas jobs.

You should just not rely on only this, it's important to note before we go into the cheat sheet. We strongly advise taking our inclusive Python course if you have not learned any pandas yet. This cheat sheet is not intended to teach you all there is to know about pandas; rather, it will help you locate and recall information you already know.

Although the quick, adaptable, and creative Pandas data objects are intended to considerably simplify real-world data analysis, this may not be the case right away for individuals who are j starting started. The possibilities are daunting precisely because this software has so many capabilities built in. These Pandas cheat sheets may be useful in that situation. It's a brief introduction to the fundamentals of Pandas that you'll require to get started using Python to organize your data.

As a result, if you are just starting your data science journey with Pandas, you may use it as a handy reference. Alternatively, for those of you who haven't begun yet, you can simply employ it as a guide to simplify the process to learn about or even utilize it.

Guide: Pandas Cheat Sheet

It is recommended to save our page for further help.

We'll utilize the abbreviations listed below in this cheat sheet:

Each pandas Series object|df

Every pandas DataFrame object |s

You'll notice as you go down that we've arranged related commands utilizing subheadings so you can quickly search for and discover the right syntax according to the task you're seeking to do.

Also, a quick reminder – to make use of the commands described here, you'll first import the required libraries like so:

Data Importing

Obtaining some data is the first step in any type of data analysis. You have a wide range of choices using Pandas for adding data to your Python workbook:

- pd.read_csv(filename) # From a CSV file
- pd.read_table(filename) # From a delimited text file (like TSV)
- pd.read_excel(filename) # From an Excel file
- pd.read_sql(query, connection_object) # Reads from a SQL table/database
- pd.read_json(json_string) # Reads from a JSON formatted string, URL or file.
- pd.read_html(URL) # Parses an HTML URL, string, or file and takes out tables to a set of data frames
- pd.read_clipboard() # Takes the idea of your clipboard and pastes it to read table()
- pd.DataFrame(dict) # From a dict, items for columns names, values for data as set

Exploring Data

Following the import of your information into a Pandas data frame, you can visualize the data using the following techniques:

- df.shape() # Prints the number of rows as well as columns in a data frame
- df.head(n) # Prints first n rows of the DataFrame
- df.tail(n) # Prints last n rows of the DataFrame
- df.info() # Index, Datatype, and Memory details
- df.describe() # Summary statistics for numerical columns
- s.value_counts(dropna=False) # Views unique values and counts
- df.apply(pd.Series.value_counts) # Unique values and counts for every columns
- df.describe() # brief statistics for numerical columns
- df.mean() # Returns the mean of every columns

- df.corr() # Returns the correlation between columns in a DataFrame
- df.count() # Returns the number of non-null values in each DataFrame column
- df.max() # Returns the biggest value in every column
- df.min() # Returns the lowest value in every column
- df.median() # Returns the median of every column
- df.std() # Returns the standard deviation of every column

Selecting

In order to review or conduct further analysis on the data, you may frequently need to choose only one piece or a specific subset of the data. These techniques will come in very handy:

- df[col] # Returns column with label col as Series
- df[[col1, col2]] # Returns Columns as a new DataFrame
- s.iloc[0] # Selection by position (selects first element)
- s.loc[0] # Selection by index (selects element at index 0)
- df.iloc[0,:] # First row
- df.iloc[0,0] # First element of first column

Cleaning Data

It is likely that you will need to clean up the data if you are using real-world examples. These are a few beneficial techniques:

- df.columns = ['a','b','c'] # Renames columns
- pd.isnull() # Checks for null Values, Returns Boolean Array
- pd.notnull() # Opposite of s is null()
- df.dropna() # Drops all rows that contain null values
- df.dropna(axis=1) # Drops all columns that contain null values
- df.dropna(axis=1,thresh=n) # Drops all rows have have less than n non null values
- df.fillna(x) # Replaces all null values with x
- s.fillna(s.mean()) # Replaces all null values with the mean (mean can be replaced with almost any function from the statistics section)
- s.astype(float) # Converts the datatype of the series to float
- s.replace(1,'one') # Replaces all values equal to 1 with 'one'
- s.replace([1,3],['one','three']) # Replaces all 1 with 'one' and 3 with 'three'
- df.rename(columns=lambda x: x + 1) # Mass renaming of columns
- df.rename(columns={'old_name': 'new_ name'}) # Selective renaming
- df.set_index('column_one') # Changes the index
- df.rename(index=lambda x: x + 1) # Mass renaming of index

Sort, Filter, and Group By

Techniques for grouping, classifying, and limiting your data include:

- df[df[col] > 0.5] # Rows where the col column is greater than 0.5
- df[(df[col] > 0.5) & (df[col] < 0.7)] # Rows where 0.5 < col < 0.7

- df[(df[col] > 0.5) & (df[col] < 0.7)] # Rows where 0.5 < col < 0.7
- df.sort_values(col1) # Sorts values by col1 in ascending order
- df.sort_values(col2,ascending=False) # Sorts values by col2 in descending order
- df.sort_values([col1,col2], ascending=[True,False]) # Sorts values by col1 in ascending order then col2 in descending order
- df.groupby(col) # Returns a groupby object for values from one column
- df.groupby([col1,col2]) # Returns a groupby object values from multiple columns
- df.groupby(col1)[col2].mean() # Returns the mean of the values in col2, grouped by the values in col1 (mean can be replaced with almost any function from the statistics section)
- df.pivot_table(index=col1, values= col2,col3], aggfunc=mean) # Creates a pivot table that groups by col1 and calculates the mean of col2 and col3
- df.groupby(col1).agg(np.mean) # Finds the average across all columns for every unique column 1 group
- df.apply(np.mean) # Applies a function across each column
- df.apply(np.max, axis=1) # Applies a function across each row

Together and Separately

Techniques for fusing two data frames together:

- df1.append(df2) # Adds the rows in df1 to the end of df2 (columns should be identical)
- pd.concat([df1, df2],axis=1) # Adds the columns in df1 to the end of df2 (rows should be identical)
- df1.join(df2,on=col1,how='inner') # SQL-style joins the columns in df1 with the columns on df2 where the rows

Writing Data

Finally, there are numerous methods you can transfer your data once your analysis has provided results:

- df.to_csv(filename) # Writes to a CSV file
- df.to_excel(filename) # Writes to an Excel file
- df.to_sql(table_name, connection_object) # Writes to a SQL table
- df.to_json(filename) # Writes to a file in JSON format
- df.to_html(filename) # Saves as an HTML table
- df.to_clipboard() # Writes to the clipboard

Want to Learn More?

We haven't even begun to touch the surface of what Python and data science can achieve for you, but we hope that our cheat sheet on Python for data science has provided you with a glimpse of what is possible. If you want to learn more about data science, check out this course.

The Professional Certificate in Data Science and Generative AI, developed in partnership with IBM, jumpstarts your career in this area and gives you the top-notch instruction and abilities needed to succeed. The course provides in-depth instruction in the most sought-after Data Science and Machine Learning abilities as well as practical experience with important technologies and tools such as Python, R, Tableau, and machine learning principles. To advance your career in data science, get to be a data scientist by delving deeply into the complexities of interpretation of data, ma g methods like machine learning, and developing strong programming abilities.

Data Science & Business Analytics Courses Duration and Fees

Data Science & Business Analytics programs typically range from a few weeks to several months, with fees varying based on program and institution.

Professional Certificate in Data Analytics and Generative AI

Cohort Starts: **25 Sep, 2025**

8 months **$2,790**

Professional Certificate Program in Data Analytics and Generative AI

Cohort Starts: **25 Sep, 2025**

11 months **$2,290**

Professional Certificate in Data Science and Generative AI

Cohort Starts: **29 Sep, 2025**

6 months **$2,790**

Data Science Course

11 months **$1,099**

Data Analyst Course

11 months **$1,099**

Recommended Reads

R Cheat Sheet: A Guide of R Glossary

SOFTWARE DEVELOPMENT

The Best Python Pandas Tutorial

Data Science Interview Gu

3511

9 Jun, 2025

197608

4 Sep, 2024

9 May, 2023

Acknowledgement