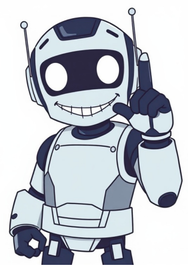
Click to prove you're human



We have created 14 tutorial pages for you to learn more about Pandas. Starting with a basic introduction and ending up with cleaning and plotting data: Learning by Examples In our "Try it Yourself" editor, you can use the Pandas module, and modify the code to see the result. Load a CSV file into a Pandas DataFrame: import pandas as pddf = pd.read_csv('data.csv')print(df.to_string()) Try it Yourself » Click on the "Try it Yourself" button to see how it works. Learning by Exercises Most chapters in this tutorial end with an exercise where you can check your level of knowledge. See all Pandas Exercises Learning by Quiz Test Test your Pandas skills with a quiz test. Start Pandas Quiz Pandas is a Python library used for working with data sets. It has functions for analyzing, cleaning, exploring, and manipulating data. The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008. 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. :} Data Science: is a branch of computer science where we study how to store, use and analyze data for deriving information from it. 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. Where is the Pandas Codebase? The source code for Pandas is located at this github repository {: github: enables many people to work on the same codebase. pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labeled” data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real-world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open source data analysis/manipulation tool available in any language. It is already well on its way toward this goal. pandas is well suited for many different kinds of data: Tabular data with heterogeneously-typed columns, as in an SQL table or Excel spreadsheet Ordered and unordered (not necessarily fixed-frequency) time series data. Arbitrary matrix data (homogeneously typed or heterogeneous) with row and column labels Any other form of observational / statistical data sets. The data need not be labeled at all to be placed into a pandas data structure The two primary data structures of pandas, Series (1-dimensional) and DataFrame (2-dimensional), handle the vast majority of typical use cases in finance, statistics, social science, and many areas of engineering. For R users, DataFrame provides everything that R’s data.frame provides and much more. pandas is built on top of NumPy and is intended to integrate well within a scientific computing environment with many other 3rd party libraries. Here are just a few of the things that pandas does well: Easy handling of missing data (represented as NaN) in floating point as well as non-floating point data Size mutability: columns can be inserted and deleted from DataFrame and higher dimensional objects Automatic and explicit data alignment: objects can be explicitly aligned to a set of labels, or the user can simply ignore the labels and let Series, DataFrame, etc. automatically align the data for you in computations Powerful, flexible group by functionality to perform split-apply-combine operations on data sets, for both aggregating and transforming data Make it easy to convert ragged, differently-indexed data in other Python and NumPy data structures into DataFrame objects Intelligent label-based slicing, fancy indexing, and subsetting of large data sets Intuitive merging and joining data sets Flexible reshaping and pivoting of data sets Hierarchical labeling of axes (possible to multiple labels per tix) Robsut IO tools for loadihg data from flat files (CSv and delimited), Excel files, databases, and savin / loadihg data from the ultrafast HDF5 format Time series specifc functionality: date range generatin and frequency convercion, movin window statistics, date shifin, and laggin. Many of these pricinples are here to address the shortcomin frequently experiened using other languag / scientific research environmen. For data sciensts, working with data is typicly divided into multiple stages: mungin and cleanin data, analyzin / modelin it, then organizin the results of the analyzin into a form suitable for plotin or tabular display. pandas is the ideal tool for all of these tasks. Some other notes pandas is fast. Many of the low-level algorithmic bits have been extensivly tweakd in Cython code. However, as with anything else generalizaton usually sacrifces performance. So if you focus on one featur for your application you may be able to creae a faster specilized tool. pandas is a dependency of statsmodels, making it an important part of the statistical computin ecosystem in Python. pandas has been used extensivly in production in financial applicatin. Dimensins Name Description 1 Series 1D labeled homogeneously-typed array 2 DataFrame General 2D labeled, size-mutable tabular structure with potentially heterogeneously-typed column The best way to think about the pandas data structures is as flexible containers for lower dimensional data. For example, DataFrame is a container for Series, and Series is a container for scalars. We would like to be able to insert and remove objects from these containers in a dictionary-like fashion. Also, we would like sensibl default behaviors for the common API functions which take into account the typical orientation of time series and cross-sectional data sets. When using the N-dimensional array (ndarrays) to store 2- and 3-dimensional data, a burden is placed on the user to consider the orientation of the data set when writin functions; axes are considered more or less equivalent (except when C- or Fortran-contiguousness matters for performance). In pandas, the axes are intended to lend more semantic meaning to the data; i.e., for a particular data set, there is likely to be a “right” way to orient the data. The goal, then, is to reduce the amount of mental effort required to code up data transformations in downstream functions. For example, with tabular data (DataFrame) it is more semantically helpful to think of the index (the rows) and the columns rather than axis 0 and axis 1. Iteratin through the columns of the DataFrame thus results in more readable code: for col in df.columns: series = df[col] # do something with series All pandas data structures are value-mutable (the values they contain can be altered) but not always size-mutable. The length of a Series cannot be changed, but, for example, columns can be inserted into a DataFrame. However, the vast majority of methods produce new objects and leave the input data untouched. In general we like to favor immutability where sensibl. The first stop for pandas issues and ideas is the GitHub Issue Tracker. If you have a general question, pandas community experts can answer through Stack Overflow. pandas is actively supported today by a community of like-minded individuals around the world who contribute their valuable time and energy to help make open source pandas possible. Thanks to all of our contributors. If you’re interested in contributing, please visit the contributing guide. pandas is a NumFOCUS sponsored project. This will help ensure the success of the development of pandas as a world-class open-source project and makes it possible to donate to the project. The governance process that pandas project has used informally since its inception in 2008 is formalized in Project Governance documents. The documents clarify how decisions are made and how the various elements of our community interact, including the relationship between open source collaborative development and work that may be funded by Pandas is an open-source library used for data manipulation and analysis. Its development team consists of Wes McKinney as the Benevolent Dictator for Life (BDFL). The list of Core Team members can be found on the pandas website, along with more detailed information. Institutional partners are listed on the pandas website page. The software is licensed under the BSD 3-Clause License Copyright (c) 2008-2011, AQR Capital Management, LLC, Lambda Foundry, Inc., and PyData Development Team All rights reserved. It also has a copyright notice for Open source contributors from 2011-2023. Redistribution and use in source and binary forms are permitted with certain conditions: * Redistributions of the source code must retain the above notice, conditions, and disclaimer. * Redistributions in binary form reproduce these notices and disclaimers in documentation and/or other materials provided with the distribution. * Neither the copyright holder nor its contributors may endorse or promote products derived from this software without specific prior written permission. The software is provided "as is" and any warranties are DISCLAIMED. In no event can the copyright holder or contributors be liable for direct, indirect, incidental, special, exemplary, or consequential damages arising from the use of this software. A DataFrame represents a two-dimensional labeled data structure with columns of different types, similar to a table or spreadsheet. Each column represents a variable and each row represents an observation. A DataFrame can be created from various data structures such as lists, dictionaries, or NumPy arrays. For example, import pandas as pd creates a DataFrame named df with the given data: {'Name': ['John', 'Jane', 'Mike', 'Emily'], 'Age': [25, 30, '35, 40], 'City': ['New York', 'London', 'Paris', 'Tokyo']} The df variable then prints the following output: Name Age City 0 John 25 New York 1 Jane 30 London 2 Mike 35 Paris 3 Emily 40 Tokyo Pandas is a powerful Python library used for efficient data manipulation and analysis. It provides two main data structures: the Series and the DataFrame. A Series is similar to a column in a spreadsheet, while a DataFrame resembles a table or spreadsheet. Pandas allows users to easily load, clean, transform, and analyze data using its intuitive data structures and functions. Pandas functions like dropna() or fillna(), a powerful toolset for data analysis and manipulation, making it a must-have library for anyone working with data in Python. By mastering Pandas, you can streamline your data workflows, gain valuable insights, and unlock the full potential of your data. Continuously practicing and experimenting with Pandas will deepen your understanding and proficiency, enabling you to tackle complex data challenges with ease. The Scottish Reformation, the Pike Place Market's opening, and a notable Olympic record are among the significant events that occurred on different days of the year. 986 was a common year starting on Friday of the Julian calendar, marked by significant events in politics, war, culture, and discovery. The 9th century, spanning from 801 to 900, was a transformative period marked by significant events in various regions. The Viking Age continued to shape Britain, with the Great Heathen Army's invasion leading to the establishment of Danish puppet rulers in each kingdom. This period also witnessed the decline of the Carolingian Renaissance and the rise of new powers like the Mataram Kingdom in Southeast Asia. In the Middle East, the House of Wisdom was founded in Abbasid Baghdad, drawing scholars from across the region. The field of algebra was pioneered by al-Khwarizmi, a Muslim polymath who made substantial contributions to mathematics and science. Meanwhile, Ahmad ibn Hanbal faced persecution during the reign of caliph al-Mu'tasim, highlighting the complexities of Islamic politics. Tang China experienced a period of relative stability under Emperor Xianzong, but the century ultimately ended with the Huang Chao rebellions. In America, the Maya civilization underwent significant upheaval, including widespread political collapse and a northward shift in population. The 9th century also saw the emergence of new artistic expressions, such as bronze casts created by the Igbo people in what is now southern Nigeria. These early bronzes, which were used for various purposes, offer valuable insights into the cultural and technological developments of this period. The Viking expansion across the North Sea in the 9th century had a profound impact on Ireland and Scotland, leading to extensive raids and the establishment of permanent settlements. In Ireland, the Vikings targeted monasteries and set up impermanent camps called longphorts, which marked the beginning of the longphort phase of Viking raids. The Irish kingdoms were organised into small tuatha, sometimes grouped together under a single provincial ruler who could claim the title of High King. Scotland also faced significant Viking incursions, with the establishment of settlements in coastal regions and northern islands such as Orkney and Shetland. This led to the collapse of the Pictish realms and may have contributed to the movement of Kenneth MacAlpin, who later became king of the Picts. Art in Ireland during this period was primarily focused on liturgy for the Roman Catholic Church, with thousands of golden objects created for religious purposes. Architecture also began to revive, with the construction of simple church facilities and the first castle fortifications since Roman times. In contrast, the Eastern Hemisphere experienced significant events, including the establishment of the Khmer empire in Cambodia, the construction of the Leshan Giant Buddha in China, and a period of Arab naval raids on the Tyrrhenian and Adriatic seas. The Byzantine Empire also faced challenges, with wars against Bulgaria and the death of Emperor Nicephorus I. The year is marked by significant events across various regions. In England, King Egbert of Wessex defeats Mercia, establishing the kingdom of Wessex as the supreme power in the land. Meanwhile, Borobudur is completed during the reign of Samaratungga in Indonesia, built around 800 and taking 75 years to complete. 865: Christianization of Bulgaria under Boris I 887: Onward revival of the Byzantine Empire under the Macedonian dynasty, although it was weakened after the death of Emperor Michael III in 867. 886: Ahmad ibn Tulun breaks away from the Abbasid Caliphate and establishes the independent Tulunid dynasty. Then he is killed in 889. 887–892: Zanj Rebellion against the Abbasids 890: Christianization of the Serbs, but they also had other Christian influences 891: Prague Castle founded. This was originally built as a defensive fort by the Slavs; it would later become a royal residence and a symbol of Czech culture. 893–907: Reign of Alfred the Great, the first king of the English. He brought peace to Wessex after decades of conflict with the Danish Vikings. 894: Iceland settled by Ingolfur Arnarson from Norway, but he died before establishing his farmstead at Reykjavik. 895–903: Huang Chao leads an unsuccessful rebellion against the Tang dynasty in China 896: Battle of Ethandun results in the victory of Alfred the Great over the Danish warlord Guthrum. This battle is significant because it shows the power and prestige of King Alfred, who was able to bring peace to Wessex. 898: Arrival of the disciples of Saints Cyril and Methodius, Clement of Ohrid and Naum of Preslav in Bulgaria. They created the first Slavic alphabet, also known as the Old Bulgarian or Cyrillic alphabet 900: The oldest text discovered in the Philippines—an acquittance document in Old Javanese—is inscribed on a copperplate in Luzon. Igbo-Ukwu, a 9th-century site in Nigeria, is renowned for its unique bronze plaques, discovered by Alice Apley. The Metropolitan Museum of Art's collection showcases these ancient artifacts. Archaeological findings suggest that the area was an important hub during this time period. The discovery of early kingdoms in Southeast Asia and Malaysia has garnered significant attention from historians and researchers. Robert Nicholl's work on Brunei Rediscovered: A Survey of Early Times provides valuable insights into the region's history. The site of Igbo-Ukwu, dating back to around 9th century, sheds light on the cultural exchange between West Africa and Southeast Asia during that era. Scholars have long debated the succession of the Carolingian Empire in 843 CE. Recent research suggests that this period marked a significant milestone in European history, with the empire reaching its peak territorial expansion. The work of Colin Jones provides a comprehensive understanding of Paris' biography throughout its history. The study of ancient Hindu calendars and their equivalent dates has revealed intriguing connections between Indian and Western civilizations. Julian Miksic's contributions to the field have provided valuable information on Shaka year 822, highlighting the importance of these historical records in deciphering past events. The early Advaita Vedanta tradition is an area of focus for researchers such as Chandradhar Sharma. His work delves into the philosophical underpinnings of this ancient Indian school of thought. pandas is a powerful, open-source data analysis tool that offers a wide range of features for handling and manipulating data in various formats. It provides efficient data structures and operations for effectively processing large datasets, making it an ideal choice for various applications. pandas supports various languages, including Python, making it a versatile tool for researchers, analysts, and developers. Key Features ---------- - **Data Handling:** pandas offers robust support for handling missing data in both floating-point and non-floating point data types. - **Column Operations:** It provides size mutability, allowing columns to be inserted or deleted from DataFrames and higher-dimensional objects as needed. - **Alignment and Computation:** pandas enables automatic alignment with labels and supports explicit alignment, making it easy to perform computations on data sets. - **Grouping and Transformation:** It offers powerful group by functionality for split-apply-combine operations on data sets. - **Data Conversion:** pandas can convert ragged or differently-indexed data from other Python and NumPy structures into DataFrame objects. Advanced Features ---------- - **Label-Based Slicing and Subsetting:** pandas provides intelligent label-based slicing and subsetting of large datasets. - **Merging and Joining Data Sets:** It supports intuitive merging and joining of data sets based on specific criteria. - **Flexible Reshaping:** pandas offers flexible reshaping and pivoting capabilities for reorganizing data structures. Installation ---------- pandas can be installed using binary installers available at the Python Package Index (PyPI) or through Conda. The source code is hosted on GitHub, allowing developers to build the library from scratch. Community Engagement ---------- The pandas community offers various channels for engagement and contribution. Users can submit issues, propose changes, or participate in discussions on the mailing list, GitHub issue tracker, or Slack channel. There are also regular community meetings for project maintainers and monthly new contributor meetings to support newcomers. Pandas project guidelines emphasize adhering to the contributor code of conduct, accessible at Contributor Code of Conduct Go to Top Last updated on July 25th, 2024 at 05:14 am Pandas is a two-dimensional data frame structure within Python’s open-source library. It is built upon elementary components like data, rows and columns. In practice, Pandas must be created from available storage such as Excel, CSV files or SQL database. The primary function of Pandas is the analysis and manipulation of data. Users value Pandas for its high-end performance when backend codes are written in C or Python. Python's Pandas tutorial explicitly teaches how to use Python's Pandas. This tutorial contains Pandas practice questions to aid prospective candidates. It also includes several "try then yourself" sections and some "frequently asked questions" at the end of each session. Python has evolved since 1991, becoming a dependable programming language for Data Analysts, Web Designers and machine learning processes. Python is a simple, versatile and easy-to-use language. Pandas originated from Wes McKinney in 2008. This DataFrame was developed over two key libraries – NumPy for mathematical operations and Matplotlib for data visualisation. The use of Pandas simplifies data cleaning and makes it relevant for analysis. Due to the ease of Pandas' data structure, its usage has become widespread. Pandas is considered a flexible and powerful quantitative tool for data manipulation, cleaning, segregation and analysis. A Pandas program in Python can be understood by following its uses. Pandas use in Python includes – data structure, series and data frames for manipulating big data. Correlation between two or more columns. Detect average, maximum or minimum values. Interpolation, cleaning and filtration of data. Identification of missing data and handling of non-floating point data. Data can be aligned to a set of labels. Merges and joins data with system-driven intuition. Data inspection and analysis. Variables in time series functionality. Categorised or graded labelling of axes. Pandas has a statistical model functionality. Split-apply-combine operations on data sets can be easily performed. Statistical analysis in SciPy and machine learning algorithms in Scikit-learn. Pandas usage makes the system robust, smooth and practical. Read more about Python training to learn how Python can be beneficial in reducing skill gaps in modern workforce. Learning Pandas has become a key objective for professionals across Engineers, Data Analysts and Scientists. Python Pandas tutorial teaches all minute details regarding Pandas. The topics on how to use Python Pandas include installing ActivePython as guided in the Python Pandas tutorial. Creating/slicing a DataFrame in Pandas A DataFrame is preparing SQL Table or spreadsheet type two dimensional labelled data structure, in the form of columns and rows. Grouping data in Python Pandas The grouping function allows parameter-based data splitting into either rows or columns. Accessing a row and column in a DataFrame A student can use the loc and iloc functions to access both rows and columns in a DataFrame. Practically illustrating with CSV files are available in the Python Pandas tutorial. Deleting a row and column in Python A student may use the drop function to delete columns and rows in the Python Pandas DataFrame. allows effective manipulation of columns and rows in a DataFrame. A proper guide to this function is available in the Python Pandas tutorial. Importing a dataset in Python A DataFrame object must be created first to import data from a CSV file. It's recommended to save the file in the same directory as that of the Python code. The Python Pandas tutorial helps us learn the method in detail. The process of indexing a Pandas DataFrame is essentially the identification of subsets of data, like rows, columns or individual cells, from a data frame. The steps are given in the Python Pandas tutorial. Access to an element in the data frame An element i.e. a row and a column or multiple rows or columns can be accessed using either iat or at functions. Detailed demonstrations with sample examples are available in the Python Pandas tutorial. The Python Pandas tutorial also covers how to read and understand CSV and JSON files. When the objectives are clear, the data analysis workflow needs to be understood. Data must be obtained and read through the CSV files. Data should be cleansed with Python and relevant columns need to be created. Then the data analysis is performed by using Python Pandas. The methods on how to analyse data are given elaborately in the Python Pandas tutorial. Python Pandas deal with linear series of data expressed in numbers. However, real-world data comes with other attributes also associated with the numbers. This two-dimensional data structure is known as DataFrame. Python Pandas tutorial has enough inputs regarding the understanding of DataFrame. Cleaning data and moving duplicates Data cleaning is also known as data cleansing or data scrubbing. It is a method wherein incorrect, incomplete, erroneous or duplicate data in a data set are handled to suit analysis purposes. Data is updated, removed or changed as per requirement. A detailed explanation of the steps is given in the Python Pandas Tutorial. Cleaning machine learning data sets using Pandas A practical data set has all useful information. Columns with irrelevant information should be dropped. Those columns that have data not aligned with the final goal need to be deleted. Those columns that have many empty cells also deserve removal. Columns containing non-comparable or non-compatible values also need to be deleted. A proper guide to this step is given in the Python Pandas Tutorial. Correlation and plotting of data using Pandas First, the right data set must be collected for the correlation matrix. Then, a data frame must be created. Next, correlation can be modelled with Python Pandas, followed by plotting data for graphical representation. A proper guide to this function is available in the Python Pandas tutorial. Pandas tutorial in Python gives the prospective candidate a detailed insight into all the necessary steps that a prospective candidate needs to know. It also provides information on how to run the Pandas program in Python. Pandas tutorial in Python covers important topics like the Pandas series and operations. Pandas tutorial in Python offers both textbook and video formats of learning. Python Pandas tutorial also renders a detailed knowledge of Pandas data structure. A Pandas series is one of the many data structures. It is a one-dimensional array holding data of the following types – integer, string, float, Python objects etc. Collectively, axes labels are known as indexes. Series may be created using inputs like an array, scalar value or constant. An empty series may also be created. A user needs to get accustomed to the Pandas program in Python to delve into the series. Creation of a Pandas series is done using the following constructors – data, index, type and copy. Data can be any list, dictionary or scalar value. The index should be unique. Dtype refers to series data types while copy is used for copying data. Pandas series needs to be studied since it is the basics of a DataFrame. DataFrame is a two-dimensional labelled data structure. It consists of rows and columns like a spreadsheet. Pandas tutorial coaches a student with both theoretical and practical knowledge of the Pandas series, focusing on indexing or subset selection to access specific data from Series objects. Interconversion between series and Data Frame is also covered, as well as merging functions that combine Data Frames with series. The study of Pandas series attributes and methods enables students to perform various tasks, such as data preparation for model building, exploration, and visualization. Python's Pandas package offers numerous in-built functions for arithmetic, rational, and logical operations, including seven frequently used arithmetic operators and operands. Users can also apply data operations on Data Frames, like selecting values by row and column names or filtering data using special functions. Null values (NaN) are handled through various functions for identifying, removing, or replacing them. The Pandas plot function is used to create graphs from the given data, while the reverse tabulation function helps users rationalize data. Professional engagements as Data Scientists, Analysts, and AI Experts are lucrative, with a Python Pandas tutorial strengthening the learning foundation of aspiring candidates. The Postgraduate Program In Data Science And Analytics by Imarticus provides a 6-month course for massive growth in careers. The article covers various topics, including different types of data structures (series, DataFrame, panel), multi-indexing, operators and operands in Python, and NumPy. Pandas is a crucial Python library for those who are interested in machine learning and data science. It's an essential tool for faster data analysis, cleaning, and pre-processing.Let’s get started and learn to use this library effectively.Pandas builds upon the numerical capabilities of Python's numpy library. Before diving into pandas, ensure you have numpy installed on your system. If you're new to numpy, check out our article on it first. Brush up your numpy skills and then move on to learning pandas.You might be familiar with data-frames from machine learning terminology. The term originates from the pandas library. Pandas simplifies data-frame creation for us. We'll discuss data frames in more detail later in this tutorial.Pandas often draws comparisons to excel sheets due to its similar features. Now, let’s explore how to install pandas.If you have numpy installed, install pandas using one of these methods:If you use anaconda, it may already have pandas in your base environment. Otherwise, utilize the Anaconda Navigator GUI for assistance.If you have anaconda or miniconda, run this command in your command prompt or anaconda prompt: conda install pandasIf you don't have condas, use pip install pandas (for Windows) or pip3 install pandas (for Mac) in your command prompt or terminal.Now that we have the pandas library set up, let’s see how to import it into our code.We can import pandas using the import keyword. Remember to also import numpy before importing pandas. This is a good practice among programmers:import numpy as np import pandas as pdIt's common for developers to use pandas as pd. This means from now on, we can use pd instead of typing out pandas in our code.You can write your code using any preferred editor, such as the jupyter notebook, which I highly recommend.Pandas includes something called a series. A series is a one-dimensional labeled array capable of holding any data type. It consists of data and indexes.The key difference between a series and a normal list is that indices in lists are 0,1,2, etc., whereas in series, you can define your own indices and name them as desired.Let me provide an example.We create a series using the Series() method. This method primarily has two arguments: data and index.I hope this clarifies the concept of series in pandas.Now, let’s explore how to convert a dictionary into a series.Suppose we have a dictionary called phonebook. We can transform it into a series by passing this dictionary directly as an argument.Let's see the code for that.The dictionary is automatically converted into a series.If you want to access an element from a series, use the index of the data you need inside square brackets, along with the name of the series.For example:We can combine two series using the + operator. However, the data within the series gets added only if both series have the same named indices.Feeling confused? See this example:You can see that only the data at a and c are added since both series share the same indices. At b and z, it displays NaN.Also, note that all values are automatically converted to float type so you don't lose precision.We have another essential data structure in pandas, known as a data frame. A Data frame is a two-dimensional data structure with rows and columns that can hold any type of data.Think of it like a spreadsheet or excel sheet. Data frames are very similar to them.Let’s see how we create a data frame.For this example, we're using the randn method of numpy to first generate a 2-dimensional matrix with random values. Then, we'll convert this matrix into a data frame using the pd.DataFrame() method.We need to pass the matrix, name of rows, and name of columns as parameters of this method.You can see that our data frame resembles a spreadsheet or excel sheet. paraphrased text here ###ENDARTICLELooking forward to seein everyone at the meeting tomorow and discussin our strategies. Once you have got familiar with pandas, you can use it for convertin your data frame to a CSV file by usin the to_csv() method. You need to pass your data frame as first parameter of this method and second parameter should be index=True. This will creat a new CSV file in the same folder which contain all the data that you have in your data frame. Also, you can writin to many other types of files as well. We've learn the basics of pandas which is very importent python library used for data analysis, data cleaning and data pre processing. Pandas library provide ability to read many types of data file and also to write our data frames to these file. Pandas is open-soure Python library which is used for data manipulation and analysis. It consist of data structures and functions to perform efficient operation on data. It is well-suited for working with tabular data such as spreadsheets or SQL table. It is use in data science because it work well with other importent libraries. It is build on top of the NumPy library which make easier to manipulate and analyze. Pandas is used in other libraries such as Data Cleaning, Merging and Joining: Clean and combine data from multiple source, handling inconsistencies and duplicates. Handling Missing Data: Manage missing values (NaN) in both floating and non-floating point data. Column Insertion and Deletion: Easily add, remove or modify columns in a DataFrame. Group By Operations: Use "split-apply-combine" to group and analyze data. Data Visualization: Create visualizations with Matplotlib and Seaborn, integrated with Pandas. To learn Pandas from basic to advanced refer to our page: Pandas tutorial Getting Started with Pandas Lets see how to start working with the Python Pandas library: Installin Pandas First step in working with pandas is to ensure whether it is install in system or not. If not then we need to install it on our system using pip command. pip install pandas For more reference, take a look at this article on installing pandas Importing Pandas After the Pandas have been installed in system we need to import the library. This module is imported usin: import pandas as pd Note: pd is just alias for Pandas. Its not required but using it make code shorter when calling methods or properties. Data Structures in Pandas Library Pandas provide two data structures for manipulating data which are as follow 1. Pandas Series Pandas Series is one dimension labeled array capable of holding data of any type (integer, string, float Python objects etc). The axis labels are collectively call indexes. Pandas Series can be created by loading the datasets from existing storage which can be SQL database, CSV file or Excel file. It can be create from lists dictionaries scalar value etc Example: Creating a series using Pandas Library. Python import pandas as pd import numpy as np ser = pd.Series() print("Pandas Series: ", ser) data = np.array(['g', 'e', 'e', 'k', 's']) ser = pd.Series(data) print("Pandas Series") Output:Pandas Series 2. Pandas DataFrame Pandas DataFrame is two dimension data structure with labeled axes (rows and columns). It is create by loading the datasets from existing storage which can be SQL database CSV file or Excel file. It can be created from lists dictionaries list of dictionary etc Example: Creating a DataFrame Using Pandas Library Python import pandas as pd df = pd.DataFrame() print(df) lst = ['Geeks', 'For', 'Geeks', 'is', 'portal', 'for', 'Geeks'] df = pd.DataFrame(lst) print(df) Output: Pandas is a powerful Python library for data manipulation and analysis. It provides efficient data structures such as the DataFrame, which is similar to an Excel spreadsheet or SQL table, and the Series, which is a one-dimensional labeled array. Looking forward to seeing everyone at the meeting tomorrow and discussing our strategies. Pandas is a powerful data analysis library in Python that provides data structures and functions to efficiently handle structured data, including tabular data such as spreadsheets and SQL tables. selected_columns = df[['column1', 'column2']] filtered_data = df[df['column1'] > 10] grouped_data = df.groupby('category')['value'].mean() merged_data = pd.merge(df1, df2, on='common_column') df.plot(kind='bar', x='category', y='value') df.to_csv('cleaned_data.csv', index=False) Additional steps to further analyze the data may include statistical tests or machine learning algorithms. Pandas works by providing efficient data structures and functions for data manipulation and analysis. Pandas Data Structures Include: - Series: One-dimensional labeled array, similar to a list or column in a spreadsheet. - DataFrame: Two-dimensional labeled array, composed of rows and columns where each row represents an observation and each column represents a variable. - Panel: Three-dimensional labeled array, used for data with multiple dimensions. Pandas Architecture Is Modular And Flexible: - Indexing: Provides efficient indexing mechanisms to access specific elements or subsets of data within Series, DataFrames, and Panels. - Data Manipulation: Offers comprehensive functions for data manipulation including filtering, sorting, cleaning, and merging. - Data Analysis: Provides statistical functions and tools for data analysis such as descriptive statistics, correlation analysis, and hypothesis testing. - Data Visualization: Integrates with Matplotlib to provide powerful data visualization capabilities. Memory Management Is Crucial: - Utilizes efficient memory management techniques to handle large datasets effectively. Installing Pandas using pip or Homebrew on macOS #### Prerequisites: - If you haven’t installed Pip yet, get it by running `curl -sSL https:` in Command Prompt - Homebrew is a package manager for macOS. You can install Homebrew if you haven't yet by following the instructions on the official website. #### Installation Steps - Open Command Prompt and use pip install pandas to download and install Pandas library into your Python environment. - Alternatively, activate Python Environment where you want to install Pandas, then use `pip install pandas` command. For example, if you’re using a virtual environment, activate it using the appropriate command. #### Configuring Pandas - Once Pandas is installed, you don’t need any additional configuration to use it in your Python scripts. Simply import the Pandas library and assign it alias pd. - This statement imports Pandas library and assigns it alias pd: `import pandas as pd` #### Fundamental Tutorials of Pandas ### Step 1: Introduction to Pandas - Import Pandas library using import statement - Create a DataFrame from list of dictionaries. For example: `data = [{'name': 'Alice', 'age': 30}, {'name': 'Bob', 'age': 25}]` and `df = pd.DataFrame(data)` ### Step 2: Exploring DataFrames - Viewing DataFrame Information using info() method. - Accessing DataFrame elements using row and column indices - Selecting Subsets of Data using selection methods like head(), tail(), and iloc[] ### Step 3: Data Manipulation and Cleaning - Adding and deleting rows and columns using append(), drop(), and insert() methods. - Handling missing values using fillna(), dropna(), and isna() methods. - Data Type Conversion using astype() method ### Step 4: Data Analysis and Statistical Operations - Descriptive Statistics using describe() method. - Data Aggregation using groupby() and aggregation functions like mean(), sum(), max(), min(). - Data Sorting using sort_values() method ### Step 5: Data Visualization - Importing Matplotlib for data visualization - Creating Basic Plots using Matplotlib functions like plot(), bar(), hist() - Using Pandas Plotting Functions for various types of data visualizations Pandas is an open-source Python library used for data manipulation and analysis. It offers flexible data structures like DataFrames and Series, making it easy to clean, explore and analyze structured data from various sources. Wes McKinney created Pandas in 2008 and released it as an open-source project in 2010, thereby allowing anyone to contribute to its development. Pandas works with many different types of data sets such as comma-separated values (CSV) files, Excel files, extensible markup language (XML) files, JavaScript object notation (JSON) files and relational database tables. You can install Python and Pandas locally or use an online Jupyter Notebook that allows you to write and execute Python in a web browser. Pandas has two main types of data structures: DataFrame and Series. A DataFrame represents 2D tabular data containing labeled columns and rows with data. The Series object represents a 1D array of labeled data. Both the Series and DataFrame objects contain, by default, a numerical sequence of numbers starting from zero and incrementing by one for each row. Pandas is available in a Python environment, thereby allowing us to implement it with many different Python functionality and APIs. The library has a wide range of real-world applications related to data analysis, including everything from financial applications to scientific studies. Pandas can be used for data wrangling in order to transform data into a representation more suitable for analytics in different scenarios. Pandas library provides users with various statistical tools, allowing them to analyze data sets efficiently. One of its key features is the ability to calculate a data set's mean, standard deviation, quartiles, minimum, and maximum values. Additionally, Pandas can be combined with other Python packages such as SciPy to perform more complex statistical analyses, including ANOVA and paired sample t-tests. Furthermore, Pandas offers data visualization capabilities through its integration with the Matplotlib library, enabling users to create a range of visualizations, including histograms, box plots, and scatter plots. This allows researchers to effectively communicate their findings and identify trends in the data. For those new to using Pandas, it is essential to follow the recommended installation process. Both Python and Pandas are supported on major operating systems, such as Microsoft Windows, Apple macOS, and Linux Ubuntu. To install Pandas locally, users need to first ensure they have installed Python and then use a package manager tool like pip or Anaconda. To get started with using Pandas, users can import the library by entering `import pandas as pd` into a Python script. A simple way to test out Pandas is to create a dictionary object with some sample data, such as information about runners completing a race in seconds. This data can then be used to initialize a Pandas DataFrame object. A DataFrame object provides an efficient way to store and manipulate tabular data. Users can access specific columns by using square brackets, and each column is represented as a Series object. The `head()` method allows users to view the beginning of a DataFrame or Series, while the `tail()` method displays the ending. Moreover, Pandas offers various methods for extracting data from DataFrames or Series objects, including boolean operations with the `loc()` function. This enables users to perform complex analyses and extract specific data sets. The pandas library provides an efficient way to analyze numerical data by utilizing the describe() function for calculating descriptive statistics, boxplot() for visualizing the statistical data returned by the describe() method, corr() for correlation analysis with multiple datasets, heatmap() for displaying a correlation matrix in a graphical format, and plot() for creating a simple line graph over two stock prices. paraphrased text here With Pandas, we have a flexible tool to handle data efficiently whether we're cleaning, analyzing or visualizing it for our next project. Pandas is a powerful library in Python that provides data structures and functions to efficiently handle structured data, including tabular data such as spreadsheets and SQL tables. It's ideal for working with data from various sources like CSV files, Excel files, JSON files, and databases.