**Working with Data in Python**

This module explains the basics of working with data in Python and begins the path with learning how to read and write files. Continue the module and uncover the best Python libraries that will aid in data manipulation and mathematical operations.

**Learning Objectives**

* Explain how Pandas use data frames.
* Use Pandas library for data analysis.
* Read text files using Python libraries including "open" and "with".
* Utilize NumPy to create one-dimensional and two-dimensional arrays.
* Write and save files in Python.

# **Reading and Writing Files with Open**

Simple Explanation:

* **Opening a File**: You can use Python's open function to access a file. You need to provide the file's name and the mode (like 'r' for reading).
* **File Object**: When you open a file, you get a file object that allows you to read or write data.
* **Reading Data**: You can read the entire file at once or line by line.
  + read() reads the whole file.
  + readline() reads one line at a time.
  + readlines() reads all lines and gives you a list of them.
* **Using with Statement**: It's better to use the with statement when opening files because it automatically closes the file after you're done, which is a good practice.
* **Closing the File**: Always close the file when you're done to free up resources.

Summary:

* The video likely covers how to use Python's open function to create a file object, read data from a text file, and the importance of closing files. It emphasizes using the with statement for better file handling practices.

# **Writing files with open**

Simple Explanation:

* **Writing to Files**: You can create and write text to files in Python using the open function.
* **Creating a File**: To create a file, you use open with the file name and set the mode to 'w' (write). This will create a new file or overwrite an existing one.
* **Using 'with' Statement**: The with statement is used to ensure that the file is properly closed after writing.
* **Writing Data**: You can write data to the file using the write method. You can write multiple lines by calling write multiple times.
* **Appending to Files**: If you want to add data to an existing file without overwriting it, you can open the file in 'a' (append) mode.
* **Copying Files**: You can read from one file and write its content to another file using loops.

Summary:

* The video explains how to use Python's open function to create and write to text files.
* It covers creating a new file, writing data, appending to existing files, and copying content from one file to another.
* The use of the with statement is emphasized for managing file operations safely.

# **Pandas**

## **Pandas: Loading Data**

Simplified Explanation:

* **Pandas** is a tool that helps you work with data easily.
* You can **import** pandas into your code using the command import pandas as pd. This means you can use its features by typing pd instead of pandas every time.
* To read data from a **CSV file** (a common file format for data), you use the function pd.read\_csv().
* The data you read is stored in a **dataframe**, which is like a table with rows and columns.
* You can look at the first few rows of the dataframe using the method head().
* You can also create a dataframe from a **dictionary**, where the keys are column names and the values are lists of data.
* You can access specific data in the dataframe using methods like iloc (for index-based access) and loc (for label-based access).

Summary:

* The video covers how to use the pandas library for data analysis in Python.
* It explains how to import pandas, read data from CSV files, and create dataframes.
* It also shows how to access and manipulate data within dataframes using various methods.

## **Working with and Saving Data**

Simplified Explanation:

* **Data Frames**: Think of a data frame as a table with rows and columns, similar to a spreadsheet.
* **Unique Elements**: If you want to find out how many different items (like colors or years) are in a column, you can use the unique method in Pandas. This helps you see all the distinct values in that column.
* **Filtering Data**: If you want to create a new table with only certain data (like songs released after 1979), you can use conditions. You check each row to see if it meets your criteria (true or false) and then select those rows to create a new data frame.
* **Saving Data**: Finally, you can save this new data frame to a file (like a CSV) using the to\_csv method.

Summary:

* The video explains how to use Pandas to find unique elements in a data frame, filter data based on conditions, and save the results in different formats. It emphasizes the ease of manipulating large datasets with just a few lines of code.

# **Numpy in Python**

## **One Dimensional Numpy**

Simplified Explanation:

* **Numpy Arrays**: Think of a Numpy array like a list in Python, but it's more powerful. All elements in a Numpy array are of the same type (like all numbers).
* **Creating Arrays**: You can create a Numpy array from a regular list by importing Numpy and using a simple command.
* **Accessing Elements**: Just like lists, you can access elements in a Numpy array using their index (position).
* **Attributes**: Numpy arrays have special properties:
  + **Size**: Total number of elements.
  + **ndim**: Number of dimensions (1D, 2D, etc.).
  + **Shape**: The size of the array in each dimension.
* **Basic Operations**: You can perform mathematical operations like addition, subtraction, and multiplication on Numpy arrays easily and quickly.
* **Vector Operations**: The video explains how to add and multiply vectors (arrays) using Numpy, which is faster than using regular Python lists.
* **Universal Functions**: Numpy can apply functions (like calculating the mean) to all elements in an array at once.

Summary:

* The video introduces **Numpy** and its advantages for scientific computing.
* It explains how to create and manipulate Numpy arrays, including accessing elements and using attributes.
* Basic operations like vector addition, subtraction, and multiplication are demonstrated.
* The video highlights the efficiency of Numpy compared to regular Python lists, especially for large datasets.

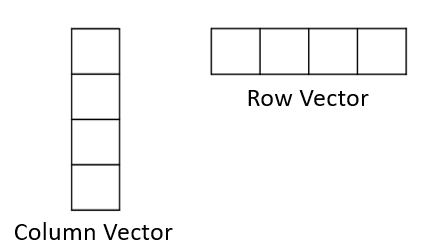
# **Matrix Mathematics**

**Reading: Matrix Mathematics**

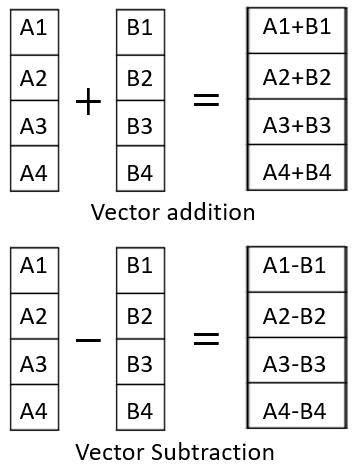
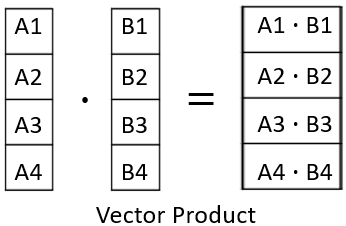
**Estimated effort: 5 mins**

You have seen that you can use Numpy package functions to perform different types of operations on arrays and matrices. In this reading, you will learn how these operations work mathematically.

**1D Arrays : Vectors**

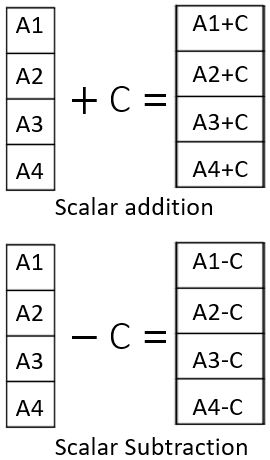
A 1D array is often termed as a vector. Depending upon the orientation of the data, the vector can be classified as a row vector or a column vector. This is illustrated in the image below.  


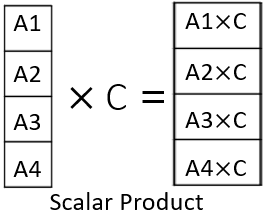
Mathematically, we can add, subtract, and take the product of two vectors, provided they are the same shape. The images below highlight the mathematical operations conducted on a pair of vectors.

All three of these operations are conducted on corresponding elements of individual vectors. The resulting array always has the same size as that of the two original vectors.

To a single vector, we can also add a constant (scalar addition), subtract a constant (scalar subtraction) and multiply a constant (scalar multiplication) to any vector. The images below illustrate these operations.



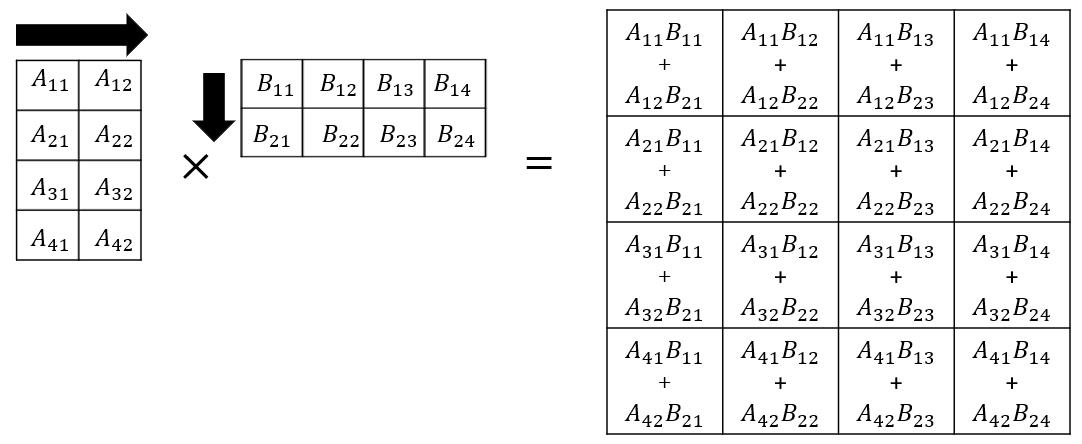


**2D Arrays : Matrices**

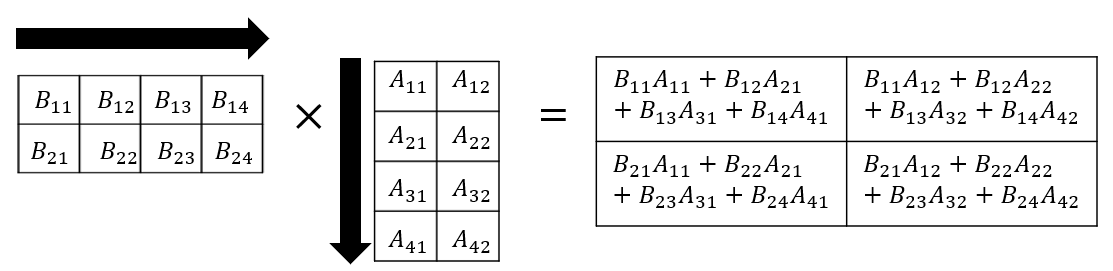
A 2D array is also called a Matrix. These are typically rectangular arrays with data stored in different rows. All of the operations mentioned above are also applicable to the 2D arrays. However, the Dot product of 2D matrices follows a different rule.

As illlustrated in the images below, the dot product is carried out by multiplying and adding corresponding elements of rows of the first matrix with the elements of columns of the second matrix. As a result, the output matrix from the multiplication will have a modified shape.

The general rule is that the dot product of an m X n matrix can be done only with an n X p matrix, and the resultant matrix will have the shape m X p. In the example shown below, the 4 X 2 matrix is multiplied with the 2 X 4 matrix to generate a 4 X 4 matrix.



In the reverse example, when 2 X 4 matrix is multiplied with the 4 X 2 one, the resultant will be a 2 X 2 matrix.



*Note: Dot product of a row vector with a column vector, with the same number of elements, would return a single scalar value. Dot product of a column vector with a row vector, will return a 2D matrix.*

# **2-D Numpy Arrays**

Simple Explanation:

* **Numpy Arrays**: Think of a Numpy array as a table with rows and columns. In this video, we specifically look at 2D arrays, which are like a grid.
* **Creating Arrays**: You can create a 2D array from a list of lists. Each inner list represents a row in the table.
* **Dimensions and Shape**: The number of dimensions (or axes) of the array is called **ndim**. For a 2D array, it has 2 dimensions (rows and columns). The **shape** tells you how many rows and columns there are.
* **Indexing and Slicing**: You can access elements in the array using indices. The first index refers to the row, and the second index refers to the column. You can also slice the array to get a subset of it.
* **Basic Operations**: You can perform operations like addition and multiplication on arrays, similar to how you would with matrices. For example, adding two arrays means adding their corresponding elements.

Summary:

* The video explains how to create and manipulate 2D Numpy arrays.
* It covers how to access elements using indexing and slicing.
* Basic operations like addition and multiplication of arrays are demonstrated.
* The video emphasizes understanding dimensions, shape, and how to visualize arrays as grids.

# **Some Context on API’s**

**Some Context on APIs**

**Estimated Effort: 5 mins**

**What are APIs?**

APIs, or Application Programming Interfaces, are a crucial part of software development. They allow developers to create new applications by leveraging existing functionality from other systems. APIs define how software components should interact and facilitate communication between various products and services without requiring direct implementation.

**Importance of APIs**

APIs are essential for any engineer because they provide a way to access data and functionality from other systems, which can save time and resources. For instance, APIs can be used to integrate applications into the existing architecture of a server or application, allowing developers to communicate between various products and services without requiring direct implementation.

APIs are also important because they enable developers to create new applications by leveraging existing functionality from other systems. This can help developers throughout the engineering and development process of apps.

APIs are used in a wide range of applications, from social media platforms to e-commerce websites. They are also used in mobile applications, web applications, and desktop applications.

**Applications of APIs**

APIs have a wide range of applications, some of which are:

1. **Social media platforms:** Social media platforms like Facebook, Twitter, and Instagram use APIs to allow developers to access their data and functionality. This allows developers to create applications that can interact with these platforms and provide additional functionality to users.
2. **E-commerce websites:** E-commerce websites like Amazon and eBay use APIs to allow developers to access their product catalogs and other data. This allows developers to create applications that can interact with these platforms and provide additional functionality to users.
3. **Weather applications:** Weather applications like AccuWeather and The Weather Channel use APIs to access weather data from various sources. This allows developers to create applications that can provide users with up-to-date weather information.
4. **Maps and navigation applications:** Maps and navigation applications like Google Maps and Waze use APIs to access location data and other information. This allows developers to create applications that can provide users with directions, traffic updates, and other location-based information.
5. **Payment gateways:** Payment gateways like PayPal and Stripe use APIs to allow developers to access their payment processing functionality. This allows developers to create applications that can process payments securely and efficiently.
6. **Messaging applications:** Messaging applications like WhatsApp and Facebook Messenger use APIs to allow developers to access their messaging functionality. This allows developers to create applications that can interact with these platforms and provide additional functionality to users.

**Conclusion**

In summary, APIs are an essential part of software development, and they provide a way to access data and functionality from other systems. They are used in a wide range of applications and can help developers save time and resources while creating new applications.

# **Summary**

**Module 4 Summary: Working with Data in Python**

Congratulations! You have completed this module. At this point, you know that:

* Python uses the open() function and allows you to read and write files, providing access to the content within the file for reading. It also allows overwriting it for writing and specifies the file mode (for example, r for reading, w for writing, a for appending).
  + To read a file, Python uses an open function along with *r.*
  + Python uses the**open with** function to read and process a file attribute, that is, from open to close.
  + In Python, you use the **open** method to edit or overwrite a file.
  + To write a file, Python uses the **open** function along with *w.*
  + In Python, "a" indicates that the program has appended to the file.
  + In Python, “\n” signifies that the code should start on a new line.
  + Python uses various methods to print lines from attributes.
* Pandas is a powerful Python library for data manipulation and analysis, providing data structures and functions to work with structured data like data frames and series.
  + You import the file (panda) by using the import command followed by the file name.
  + In Python, you use the **as** command to provide a shorter name for the file.
  + In Pandas, you use a data frame (df) to specify the files to read.
  + DataFrames consist of rows and columns.
  + You can create new DataFrames by using the column or columns of a specific DataFrame.
  + We can work with data in a DataFrames and save the results in different formats.
  + In Python, you use the **Unique** method to determine unique elements in a column of the DataFrames.
  + You use the inequality operator along with df to assign a Boolean value to the selected column in DataFrames.
  + You save a new DataFrame as a different DataFrame, which may contain values from an earlier DataFrame.
* NumPy is a Python library for numerical and matrix operations, offering multidimensional array objects and a variety of mathematical functions to work with data efficiently.
  + NumPy is a basis for Pandas.
  + A NumPy array or ND array is similar to a list, usually of a fixed size with the same kind of element.
* A one-dimensional NumPy array is a linear sequence of elements with a single axis, like a traditional list, but optimized for numerical computations and array operations.
  + You can access elements in a NumPy using an index.
  + You use the attribute **dtype** to get the data type of the array elements.
  + You use **nsize** and **ndim** to get the size and dimension of the array, respectively.
  + You can use indexing and slicing methods in NumPy.
  + Vector additions are widely used operations in Python.
  + Representing vector addition with line segments or arrows is useful.
  + NumPy codes work much faster, which is helpful with lots of data.
  + You perform vector subtraction by replacing the addition sign with a negative sign.
  + Multiplying an array by a scalar in Python entails multiplying each element of the array by the scalar value, leading to a new array in which each element scales by the scalar.
  + Hadamard product refers to the element-wise multiplication of two arrays of the same shape, resulting in a new array where each element is the product of the corresponding elements in the input arrays.
  + The dot product in Python is the sum of the element-wise products of two arrays, often used for vector and matrix operations to find the scalar result of multiplying corresponding elements and summing them.
  + When working with NumPy, it is common to utilize libraries like Matplotlib to create graphs and visualizations from numerical data stored in NumPy arrays.
* A two-dimensional NumPy array is a grid-like structure with rows and columns suitable for representing data as a matrix or a table for numerical computations.
  + In NumPy, "shape" refers to an array's dimensions (number of rows and columns), indicating its size and structure.
  + You use the attribute "size" to obtain the size of an array.
  + You use rectangular attributes to access the various elements in an array.
  + You use a scalar to multiply elements in NumPy.

# **Glossary: Working with Data in Python**

Welcome! This alphabetized glossary contains many of the terms you'll find within this course. This comprehensive glossary also includes additional industry-recognized terms not used in course videos. These terms are important for you to recognize when working in the industry, participating in user groups, and participating in other certificate programs.

| **Term** | **Definition** |
| --- | --- |
| .csv file | A .csv (Comma-Separated Values) file is a plain text file format for storing tabular data, where each line represents a row and uses commas to separate values in different columns. |
| .txt file | A .txt (Text) file is a common file format that contains plain text without specific formatting, making it suitable for storing and editing textual data. |
| Append | To "append" means to add or attach something to the end of an existing object, typically used in the context of adding data to a file or elements to a data structure like a list in Python. |
| Attribute | An "attribute" in Python refers to a property or characteristic associated with an object, which can be accessed using dot notation. |
| Broadcasting in NumPy | Broadcasting in NumPy allows arrays with different shapes to be combined in element-wise operations by automatically extending smaller arrays to match the shape of larger ones, making operations more flexible. |
| Component | In NumPy, a "component" typically refers to a specific element or value within a multi-dimensional array, which can be accessed using indexing. |
| Computation | Computation in NumPy involves performing numerical operations on arrays and matrices, making it a powerful library for mathematical and scientific computing in Python. |
| Data analysis | Data analysis is the process of inspecting, cleaning, transforming, and interpreting data to discover useful information, draw conclusions, and support decision-making. |
| DataFrames | A DataFrames in Pandas is a two-dimensional, tabular data structure for storing and analyzing data, consisting of rows and columns. |
| Dependencies | Dependencies in Pandas are external libraries or modules, such as NumPy, that Pandas rely on for fundamental data manipulation and analysis functionality. |
| File attribute | File attributes generally refer to properties or metadata associated with files, which are managed at the operating system level. |
| File object | A "file object" in Python represents an open file, allowing reading from or writing to the file. |
| Grid | In Python, a "grid" typically refers to a two-dimensional structure composed of rows and columns, often used to represent data in a tabular format or for organizing objects in a coordinate system. |
| Hadamard Product | The Hadamard product is a mathematical operation that involves element-wise multiplication of two matrices or arrays of the same shape, producing a new matrix with each element being the product of the corresponding elements in the input matrices. |
| Importing pandas | To import Pandas in Python, you use the statement: import pandas as pd, which allows you to access Pandas functions and data structures using the abbreviation "pd." |
| Index | In Python, an "index" typically refers to a position or identifier used to access elements within a sequence or data structure, such as a list or string. |
| Libraries | Libraries in Python are collections of pre-written code modules that provide reusable functions and classes to simplify and enhance software development. |
| Linespace | In Python, "linespace" refers to a NumPy function that generates an array of evenly spaced values within a specified range. |
| NumPy | NumPy in Python is a fundamental library for numerical computing that provides support for large, multi-dimensional arrays and matrices, as well as a variety of high-level mathematical functions to operate on these arrays. |
| One dimensional NumPy | A one-dimensional NumPy array is a linear data structure that stores elements in a single sequence, often used for numerical computations and data manipulation. |
| Open function | In Python, the "open" function is used to access and manipulate files, allowing you to read from or write to a specified file. |
| Pandas | Pandas is a popular Python library for data manipulation and analysis, offering data structures and tools for working with structured data like tables and time series. |
| Pandas library | Pandas library in Python refer to the various modules and functions within the Pandas library, which provides powerful data structures and data analysis tools for working with structured data. |
| Plotting Mathematical Functions | Plotting mathematical functions in Python involves using libraries like Matplotlib to create graphical representations of mathematical equations, aiding visualization, and analysis. |
| Shape | In NumPy, "shape" refers to an array's dimensions (number of rows and columns), describing its size and structure. |
| Slicing | Slicing in NumPy entails extracting specific portions of an array by specifying a range of indices, enabling you to work with subsets of the data. |
| Two dimensional NumPy | A two-dimensional NumPy array is a structured data representation with rows and columns, resembling a matrix or table, ideal for various data manipulation and analysis tasks. |
| Universal Functions | Universal functions (ufuncs) in NumPy are functions that operate element-wise on arrays, providing efficient and vectorized operations for a wide range of mathematical and logical operations. |
| Vector addition | Vector addition in Python involves adding corresponding elements of two or more vectors, producing a new vector with the sum of their components. |
| Visualizations | Visualizations in Python refer to the creation of graphical representations, such as charts, plots, and graphs, to illustrate and communicate data and trends visually. |