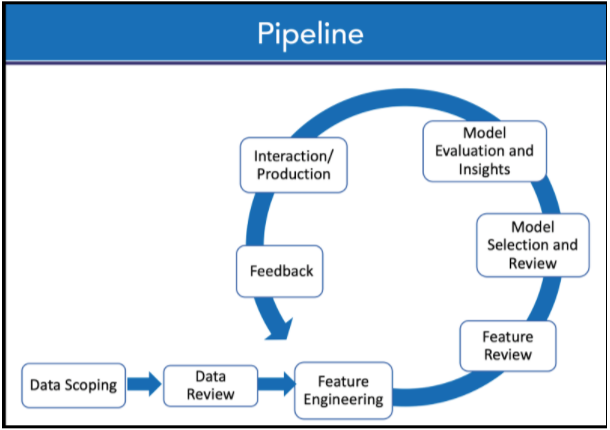
#### Getting Started With Python & Data Analysis

## The Complete Data Science Pipeline

The data science pipeline can be described as an end-to-end process in which each step contributes to producing the final insights. Every data science project begins with defining a clear problem it aims to solve or business/technical questions to provide answers to.  Data is the core of data science, hence, scoping and collecting the right data for a project is very crucial to achieving the required results. To collect data, the source it will be collected from has to be identified. Downloading or crawling from the internet, questionnaires and surveys are some common methods used to obtain data. 

Figure 1.2 - Complete Data Science Pipeline

The next step in the pipeline involves wrangling, reviewing and transforming the data from a messy/raw form to a more appropriate state for ease of use. Although this can be time-consuming, it is very essential to clean the data extensively since machine learning models are only as good as the data provided - garbage in garbage out. Conducting Exploratory Data Analysis (EDA) on the cleaned data using visualisations and statistical methods gives a quick insight into the various patterns and relationships between features in the dataset. Modelling involves using statistical and machine learning methods for classifying and clustering the processed data to create predictive models. Several evaluation methods are employed to compare the performance of these models and continuously improve before a final model is selected. Finally, all the work done in the pipeline is irrelevant if the results cannot be interpreted and communicated properly to the appropriate audience. It is imperative to present findings from the analysis done through visualisations and clear reporting. For the most part, the data science pipeline is not a linear process; it’s instead an iterative process.

## Python for Data Analysis

**-Why Python is important for data analysis**

Python is a programming language widely used by developers and data scientists. It is particularly popular because it is easy to use, has a simple syntax that helps readability and also quick to learn and adapt to. Data can be presented in different forms such as CSV, JSON, Excel files, database etc. Python is very efficient in processing and wrangling most data types. Its massive community makes resources readily available including packages, tools and libraries used in data science some of which are: Pandas, Numpy, Matplotlib, Scikit-Learn and TensorFlow.

## Getting Started With Jupyter Notebook and Google Colab

**-Setting up an Integrated Development Environment with Jupyter Notebook and Python 3 through Anaconda installations**

Jupyter notebook is an interactive web environment that supports many programming languages including Python and R, allowing for explanatory text, images and visualisation. It is a preferred environment for data scientists that runs on a web browser without requiring access to the internet and can be easily set up with Anaconda - an open-source software that has a distribution of data science and machine learning packages for scientific computing with an environment manager that removes the complexities of package management and deployment. Download the latest version of Anaconda for Python 3 from [the official website](https://www.anaconda.com/products/individual) and follow the instructions to install. To use Jupyter notebooks, open the anaconda navigator and launch Jupyter notebook. Google Colaboratory, known as Colab, is a free cloud-based Jupyter notebook with TPU and GPU.  It is easily accessible and existing libraries can be used and new libraries installed. To get started with Colab, create a new notebook [here](https://colab.research.google.com/).

## Libraries for Python Data Analysis

**-Introduction to essential Python libraries for data analysis: Pandas, NumPy, Matplotlib, Seaborn, and SciPy**

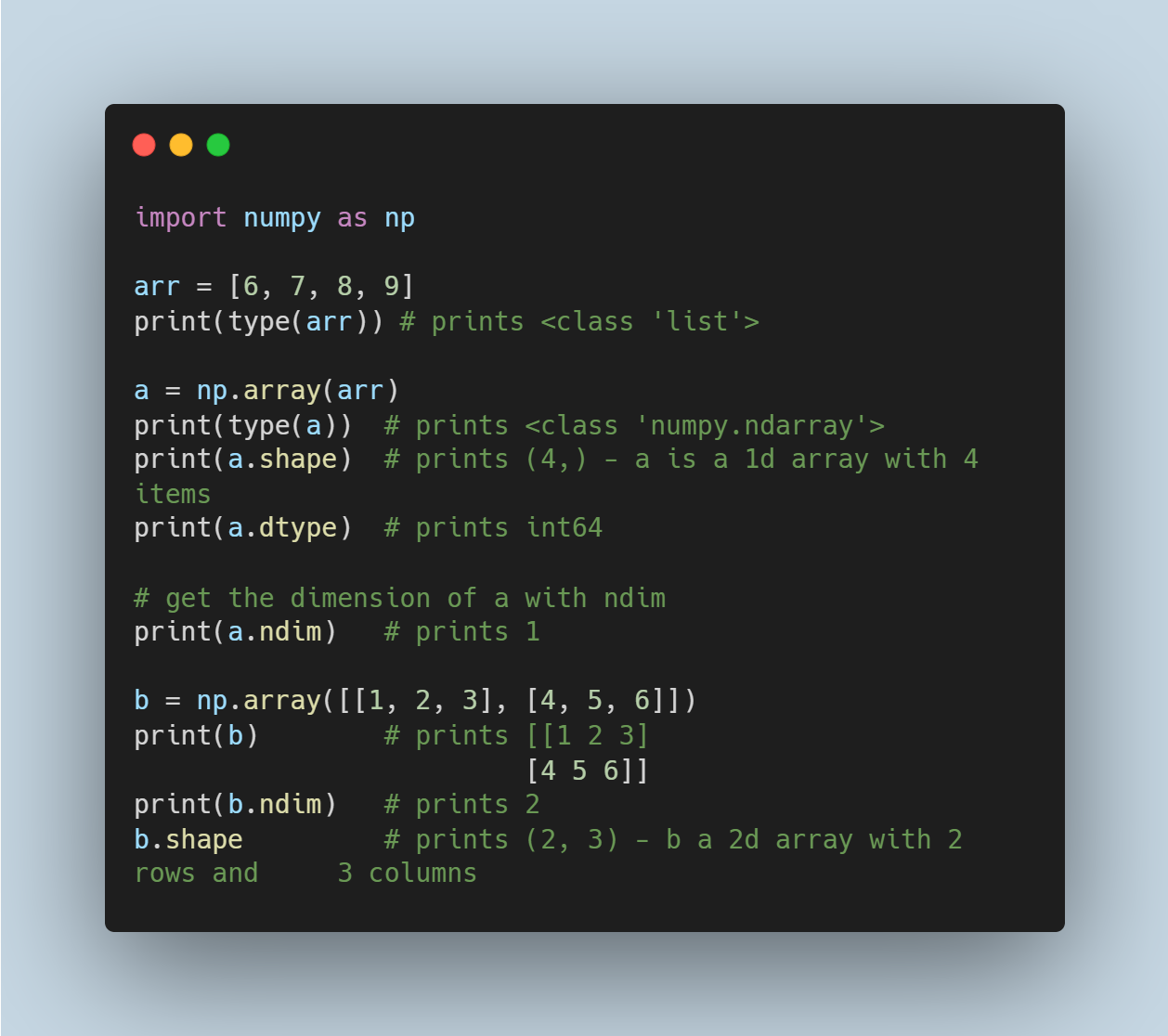
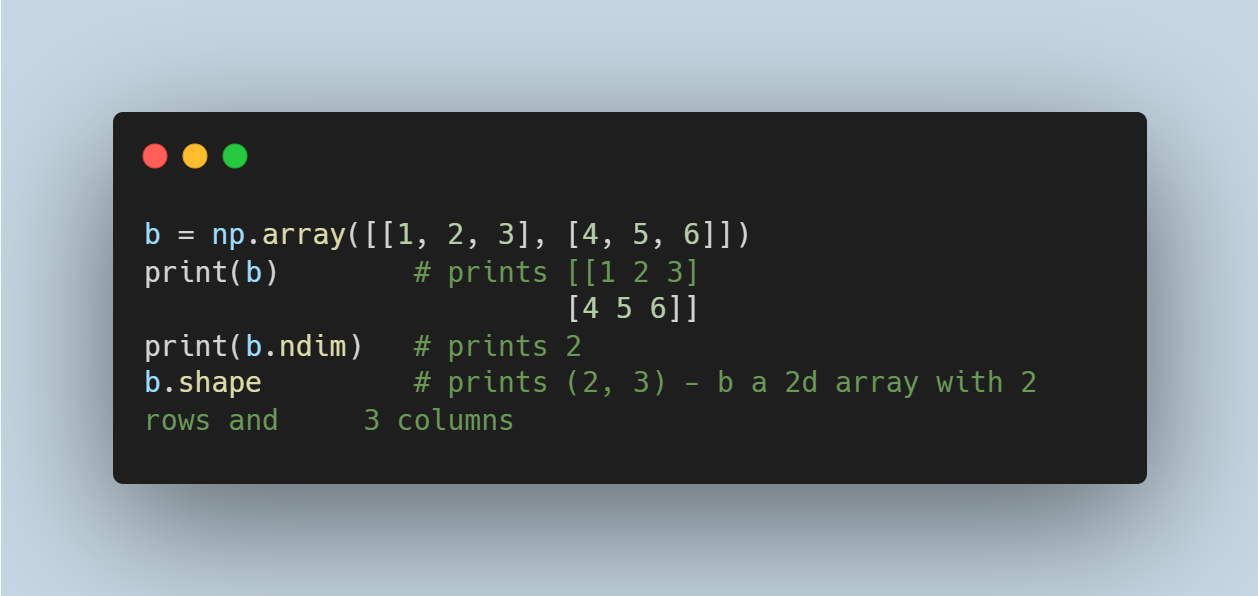
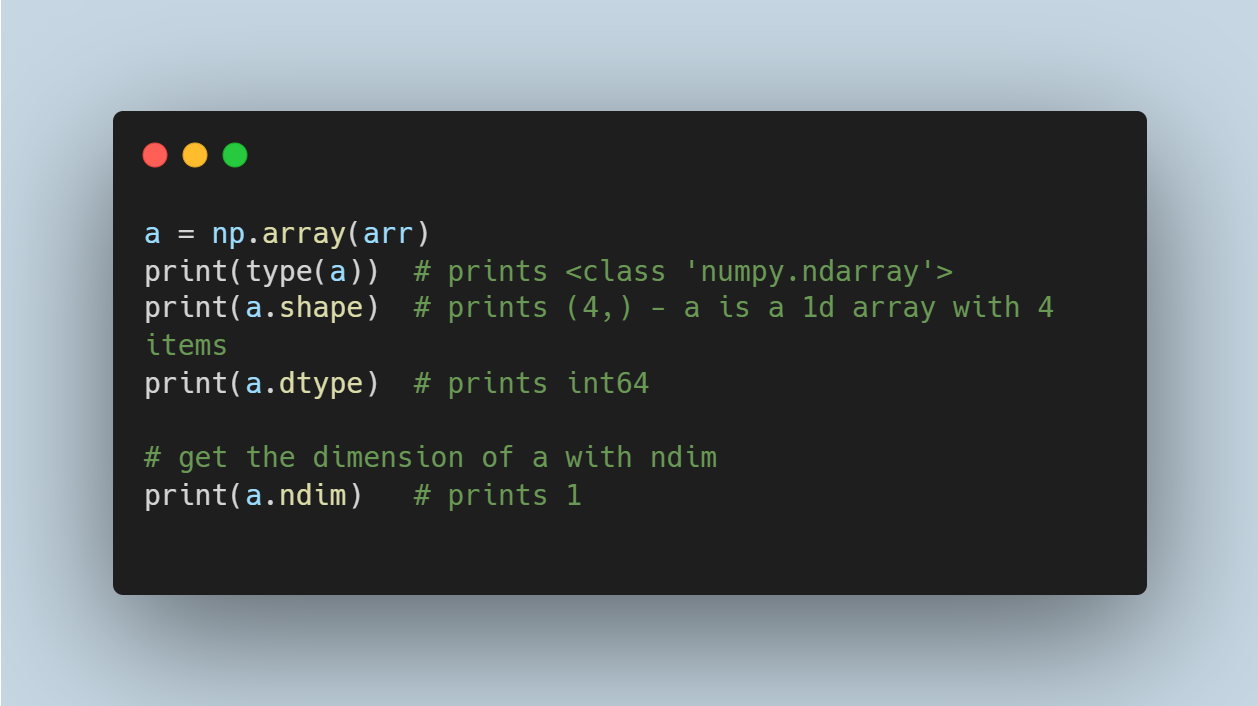
Pandas, NumPy, SciPy, Matplotlib, and Seaborn are essential Python libraries used for data analysis. Numerical computations for arrays and multidimensional matrices in data analysis are often done with the Numeric Python library - NumPy. Pandas is a toolkit built on NumPy with data structures called dataframes; used on numerical and time-series data for quick and easy data manipulation, cleaning and analysis.  SciPy can be described as a scientific package that uses NumPy arrays as its basic data structure. Matplotlib and Seaborn are plotting libraries capable of handling large datasets and producing both interactive and statistical graphics.

#### NumPy Array and Vectorization

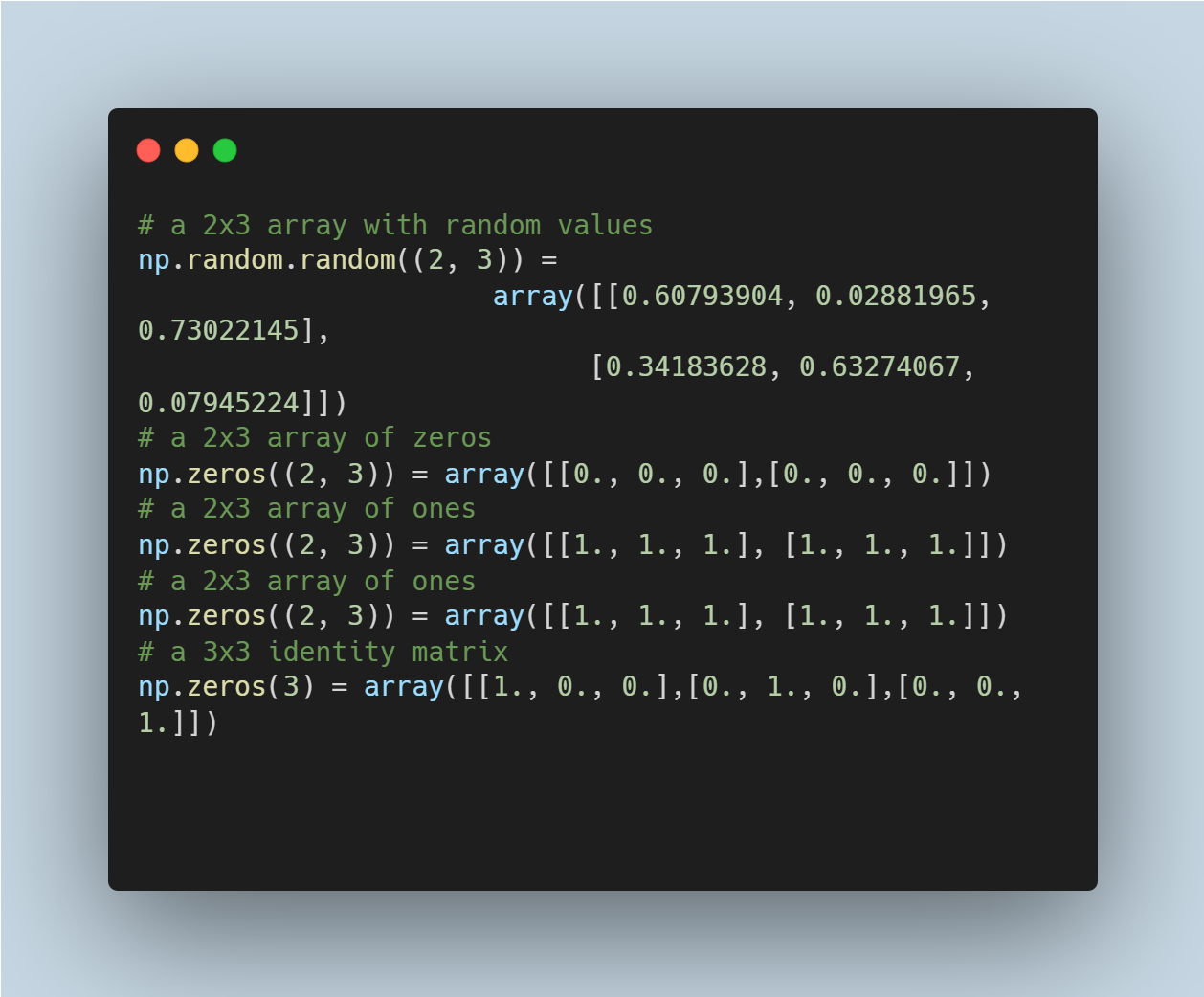
## Introduction to NumPy & Creating Arrays.

As mentioned previously, NumPy is a library that has ***ndarray*** as its basic data structure used to handle arrays and matrices. A NumPy array has a grid of values all of which are of the same data type, mostly integers and floats. These arrays can also be created from Python lists.

Below are some examples:

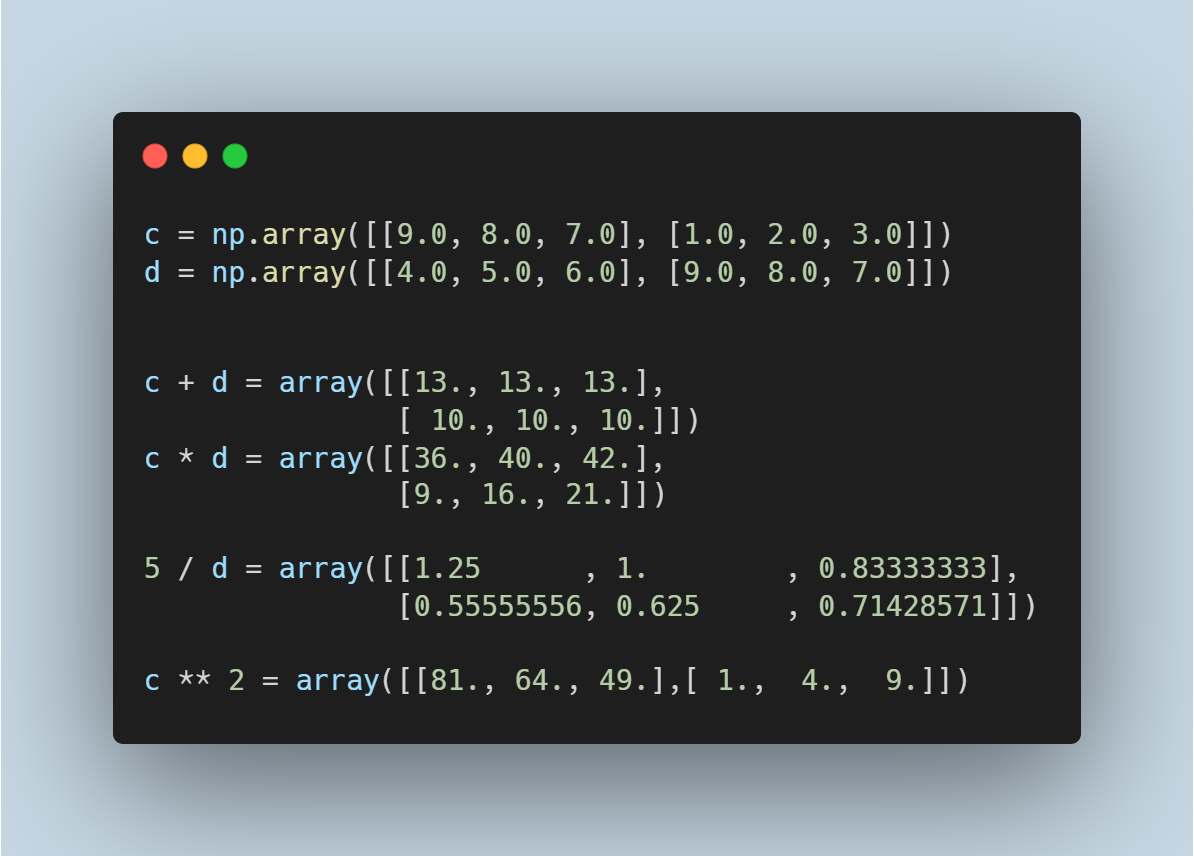


There are also some inbuilt functions that can be used to initialize numpy which include**empty(), zeros(), ones(), full(), random.random().**



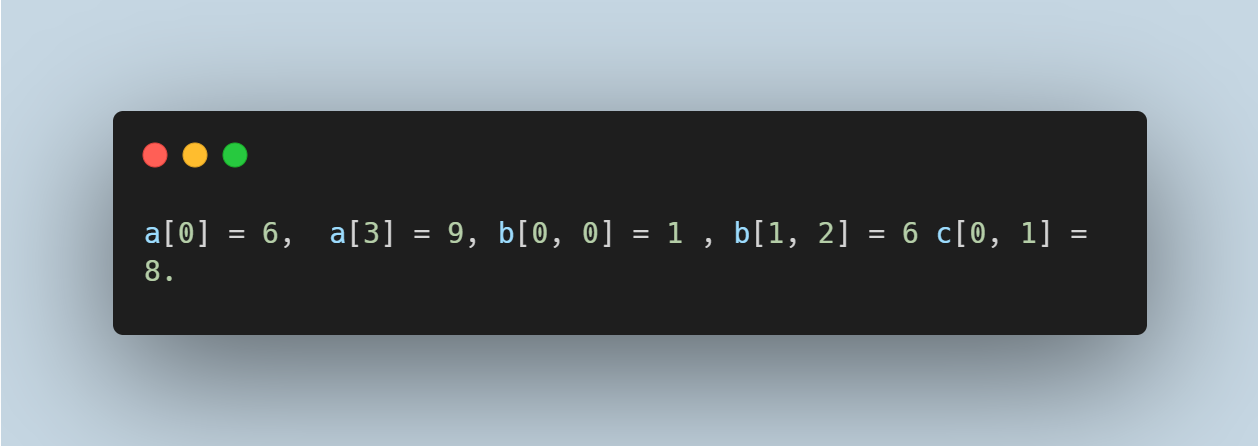
## Intra-operability of Arrays and Scalars.

Vectorisation in numpy arrays allows for faster processing by eliminating for loops when dealing with arrays of equal shape. This allows for batch arithmetic operations on the arrays by applying the operator elementwise. Similarly, scalars are also propagated element-wise across an array. For arrays with different sizes, it is impossible to perform element-wise operations instead; numpy handles this by broadcasting provided the dimensions of the arrays are the same or, one of the dimensions of the array is 1.

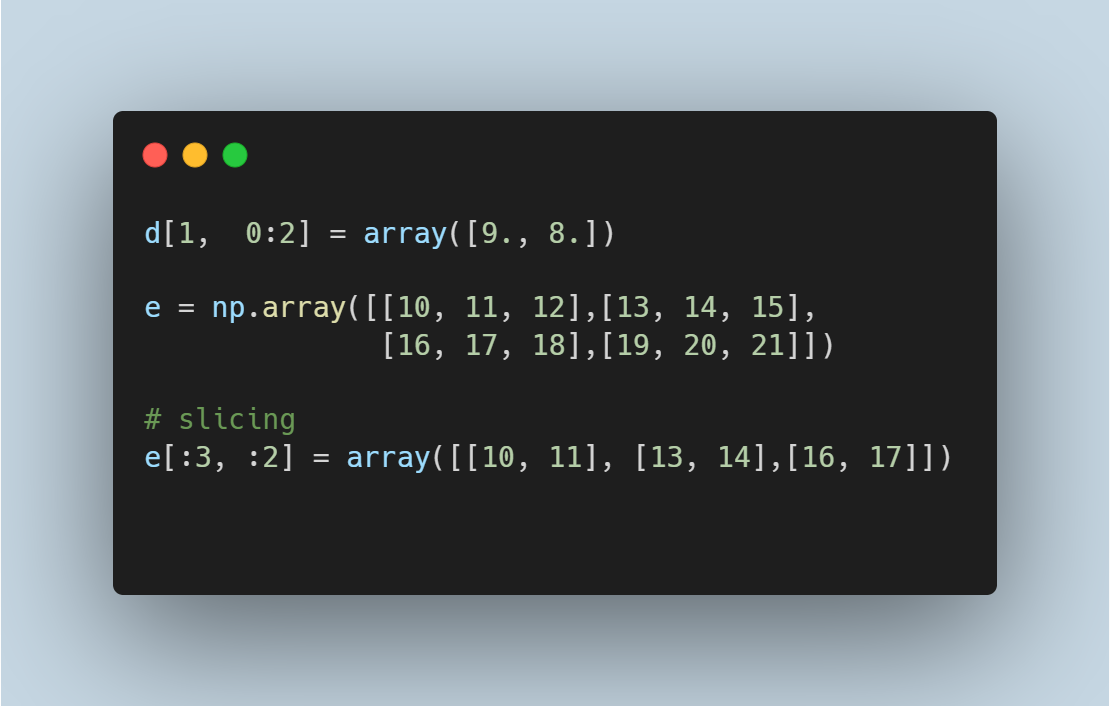


## Indexing With Arrays & Using Arrays for Data Processing

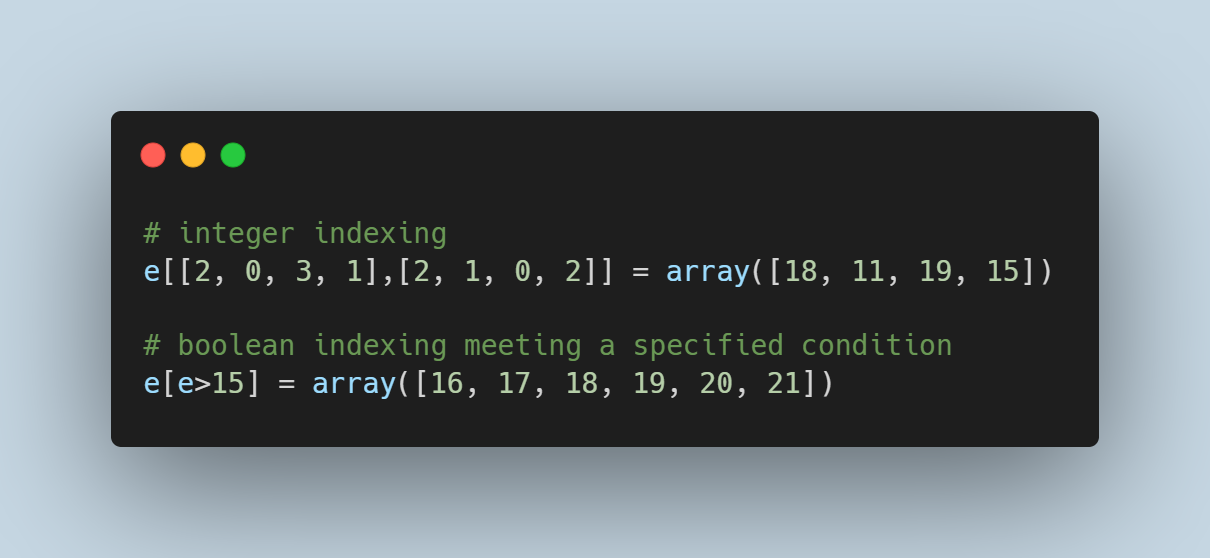
The elements in the example arrays above can be accessed by indexing like lists in Python such that:



Elements in arrays  can also be retrieved by slicing rows and columns or a combination of indexing and slicing.



There are other advanced methods of indexing which are shown below.



Numpy also has inbuilt mathematical functions like **sum(), mean(), std(), corrcoef(), min()** and others. It interestingly allows for comparing arrays using == to check if two arrays have the same elements,  elements in the first array are greater than or less than those of the second array using  > and  <.

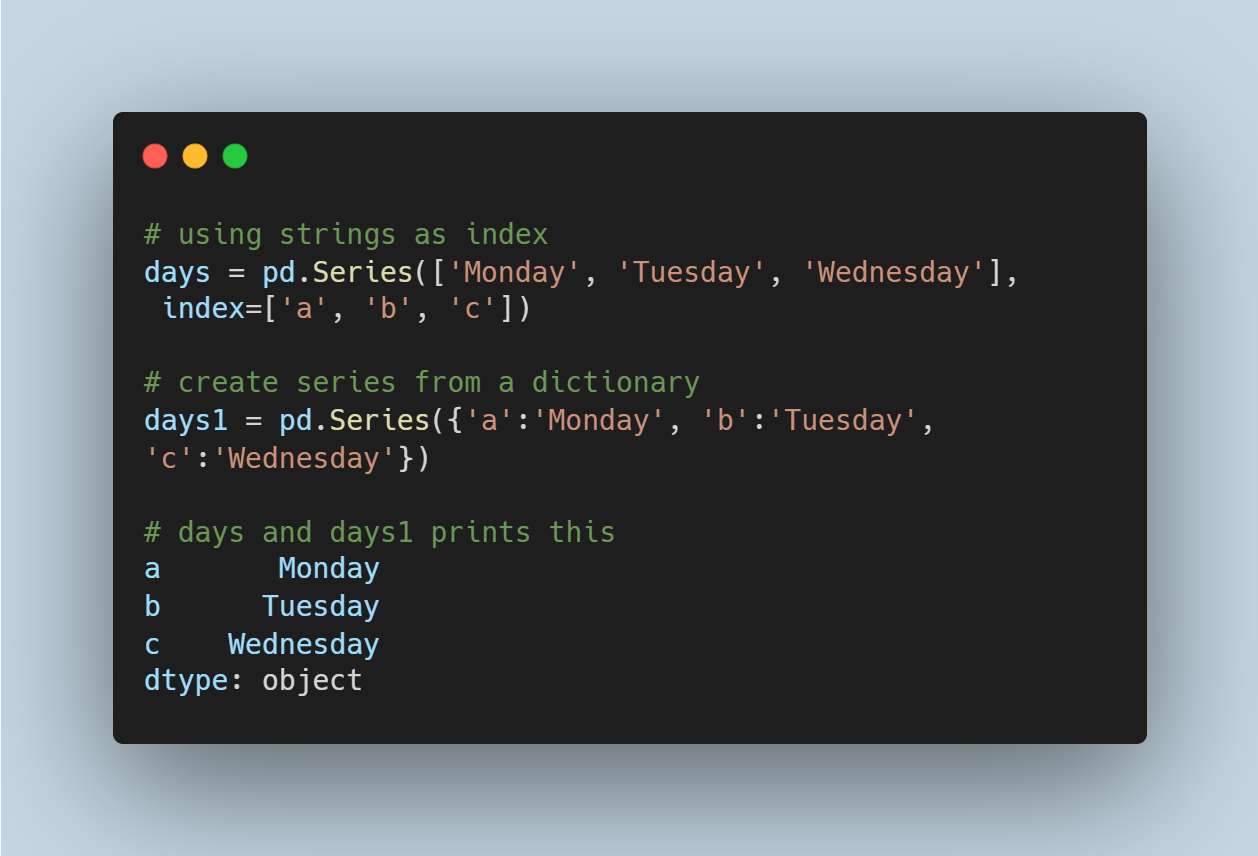
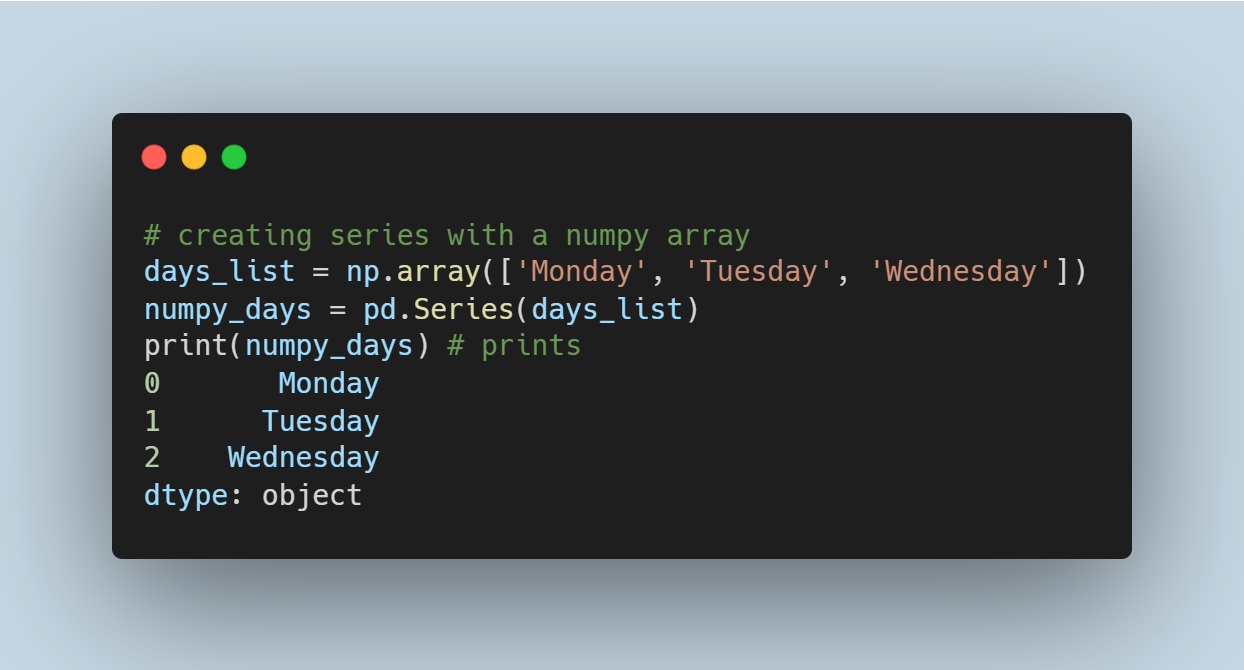
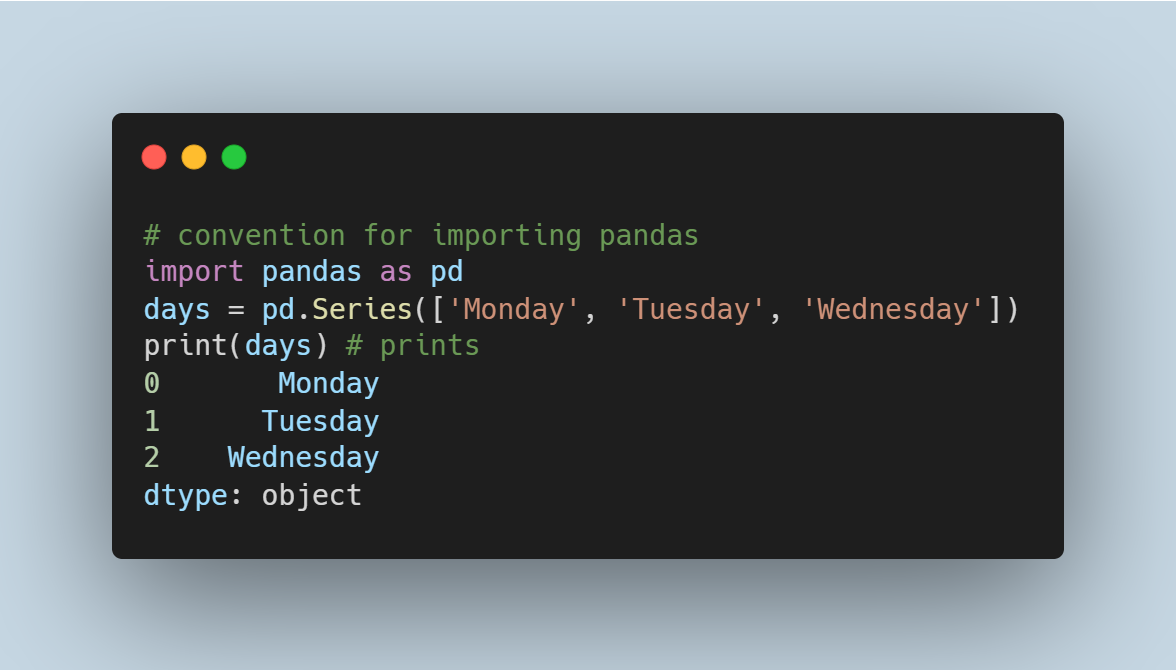
## File Input and Output With Arrays

Numpy arrays can be loaded from and saved to binary files with ***.npy*** as the extension using **load()** and **save()** respectively. This can also be done with text files with text files using **loadtxt()** and**savetxt().**

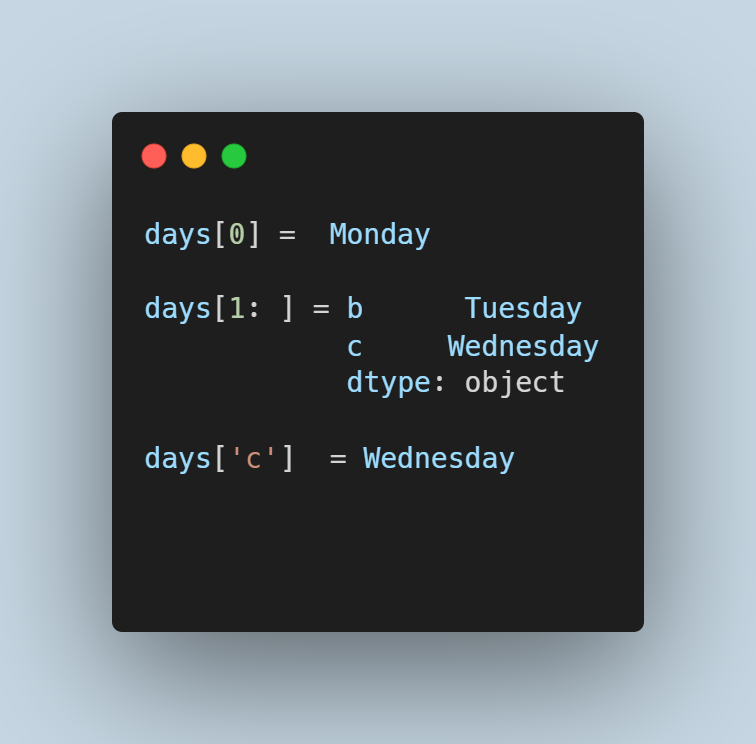
**Pandas - So Much More Than A Cute Animal**

Introducing Pandas data structures: Series, DataFrames and Index objects.

Pandas is a library  built on Numpy which is used for data manipulation, with other ways of indexing other than integers. Series, DataFrame, and index are the basic data structures in this library.  Series in pandas can be referred to as a one dimensional array with homogenous elements of different types somewhat similar to numpy arrays; however, it can be indexed differently with specified descriptive labels or integers.



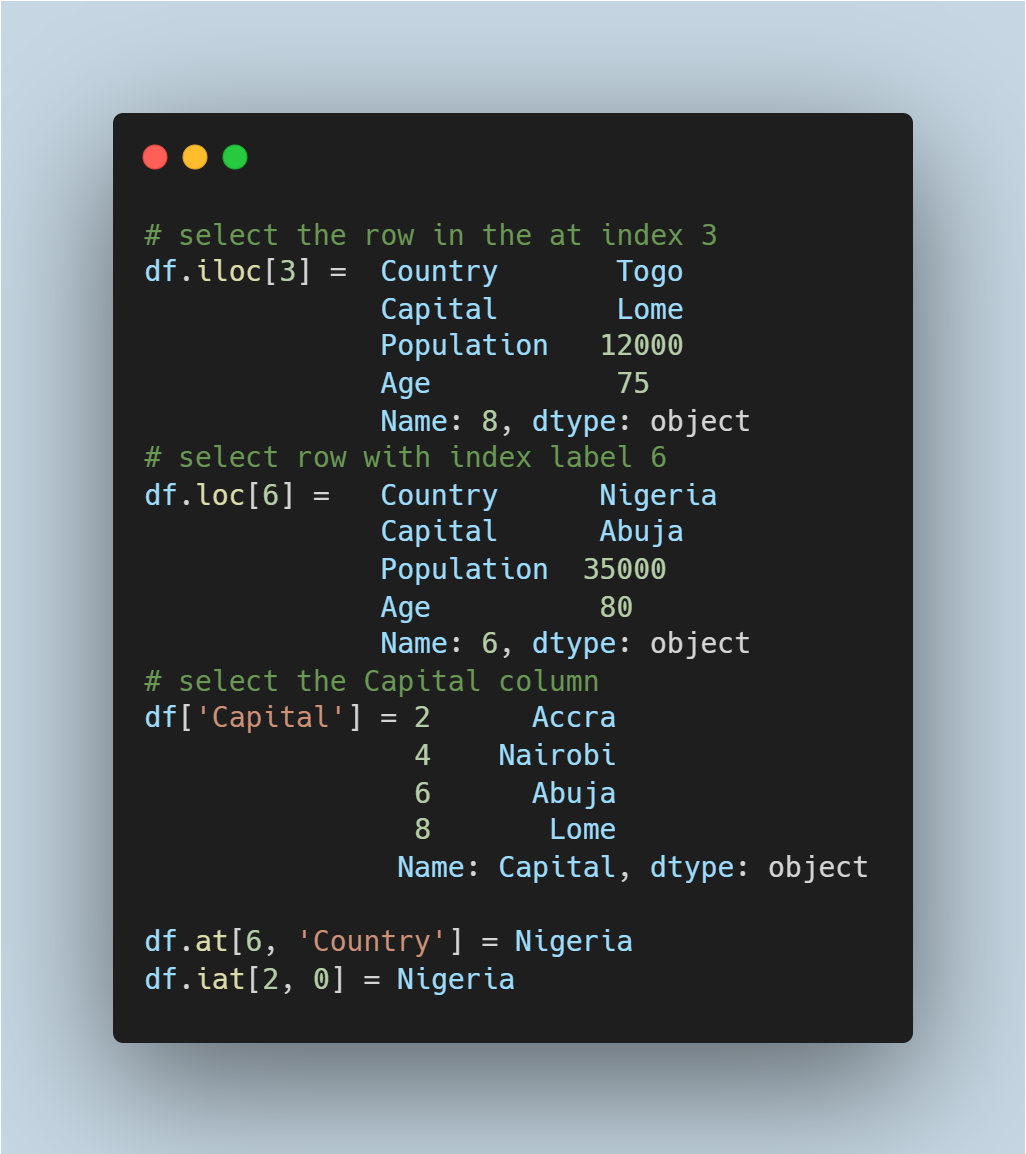
Series can be accessed using the specified index as shown below



A DataFrame can be described as a table (2 dimensions) made up of many series with the same index. It holds data in rows and columns just like a spreadsheet. Series, dictionaries, lists, other dataframes, and numpy arrays can be used to create new ones.



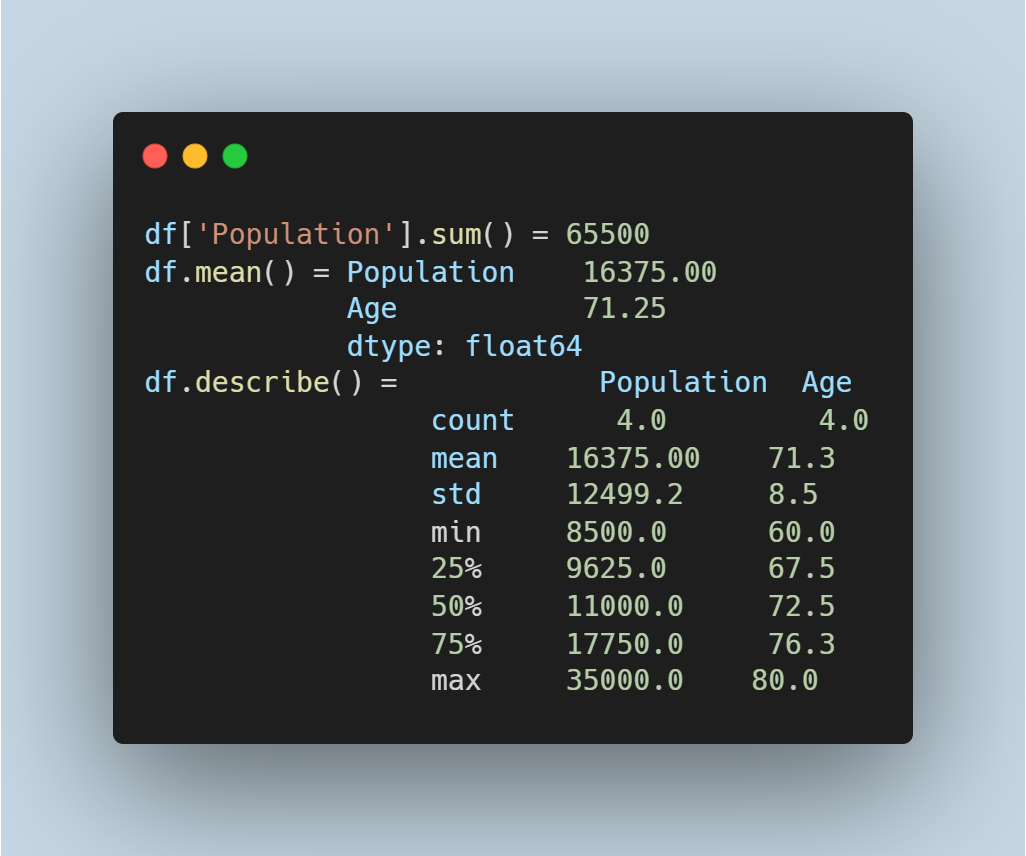
*at*, *iat*, *iloc* and *loc* are accessors used to retrieve data in dataframes. iloc selects values from the rows and columns by using integer index to locate positions, while loc selects rows or columns using labels. at and iat are used to retrieve single values such that at uses the column and row labels and iat uses indices.



Finally, Indexes in pandas are immutable arrays with unique elements. They can also be described as ordered sets for retrieving data in a dataframe and collaborating with multiple dataframes.

* The important Pandas functionalities: indexing, reindexing, selection, group, drop entities, ranking, sorting, duplicates and indexing by hierarchy.
* Summary and descriptive statistics: measure of central tendency, measure of dispersion, skewness and kurtosis, correlation and multicollinearity.

Similar to Numpy, Pandas has some functions that provide descriptive statistics such as the measures of central tendency, dispersion, skewness and kurtosis, correlation and multicollinearity. Some functions are **mode(), median(), mean(), sum(), std(), var(), skew(), kurt()**and**min().** The describe function gives the summary  of the numeric columns in a dataframe displaying count, mean, standard deviation, interquartile range, minimum and maximum values.



* The missing data enigma: Importance, types and handling missing data.

Often, data used for analysis in real life scenarios is incomplete as a result of omission, faulty devices, and many other factors. Pandas represent missing values as NA or NaN which can be filled, removed, and detected with functions like **fillna(), dropna(), isnull(), notnull(), replace().**

