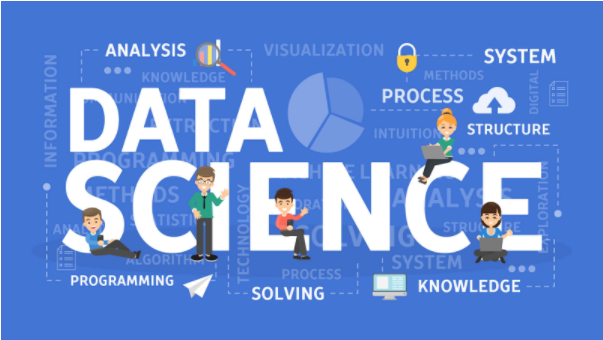
**Introduction to Python for Machine Learning - Understanding Electric Utility Data**

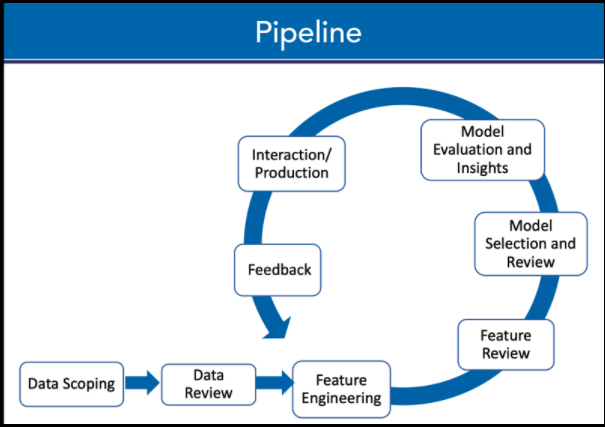
1.1: Introduction

  Figure 1.1 — What is Data Science? (source: Data Quest)

Hello there! Welcome to the first collection of the 'Learning Data Science' Series. In this series, we will explore how data science and [machine learning](https://hamoye.com/app/search/details/Q2539/machine%20learning) can be an invaluable tool for solving one of the grand challenges posed to humanity - [climate change](https://hamoye.com/app/search/details/Q125928/climate%20change).

This lesson will introduce the basics of manipulating, pre-processing, cleaning and wrangling data in Python. We’ll introduce the learner to the Jupyter environment for Python programming and how to use it to solve data analysis and machine learning problems. The project for this section is aimed at cleaning tabular datasets on public utility data containing millions of rows and tens of files in a structured format. This data is provided by the United States Energy Information Administration. **We’ll work to clean the data, wrangle the data, explore it, provide**[**summary statistics**](https://hamoye.com/app/course-details/118a97a3bf81f000)**and interesting visualization. We’ll explore the operating costs of individual power plants, see how fuel costs impact the viability of the different generation sources, highlight the competitiveness of renewable electricity and show how the generation mix of different utilities evolved overtime.**

1.1.1: Complete Data Science Pipeline

Figure 1.2 — 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.**  **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 give 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 most part, the data science pipeline is not a linear process.  Instead, it's an iterative process.**

1.1.2: 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**](https://hamoye.com/app/search/details/Q38347624/numpy)**, Matplotlib, Scikit-Learn and TensorFlow.**

1.1.3: 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/).

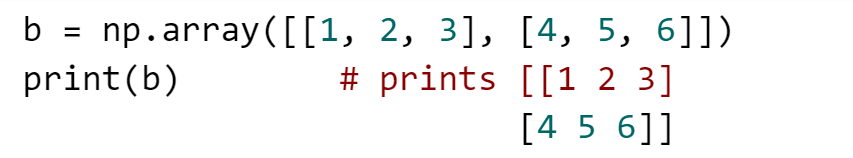
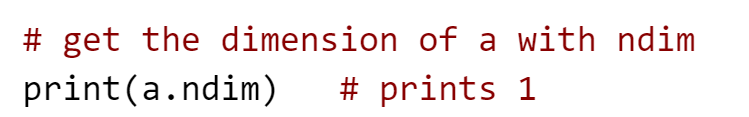
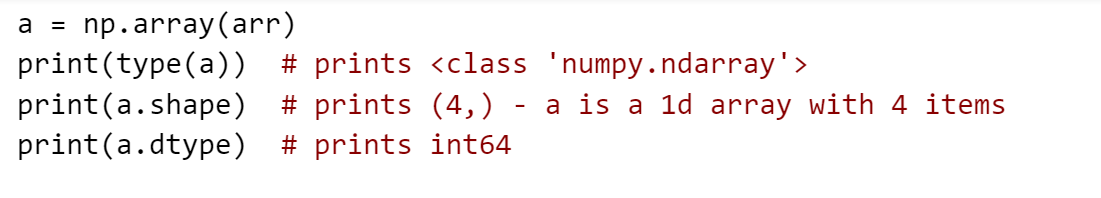
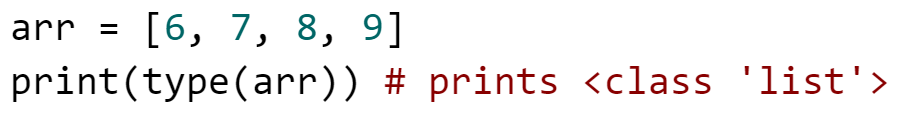
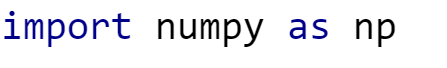
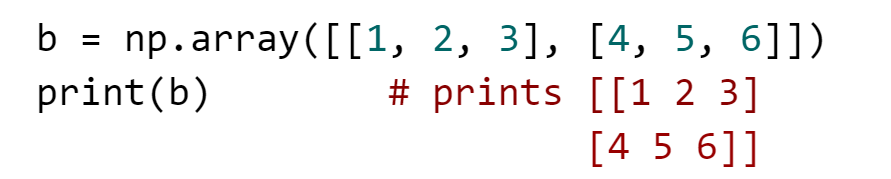
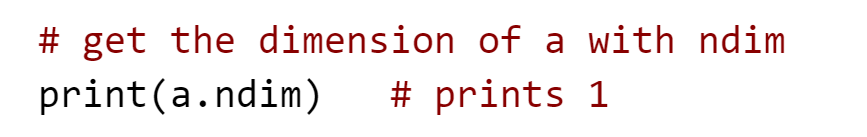
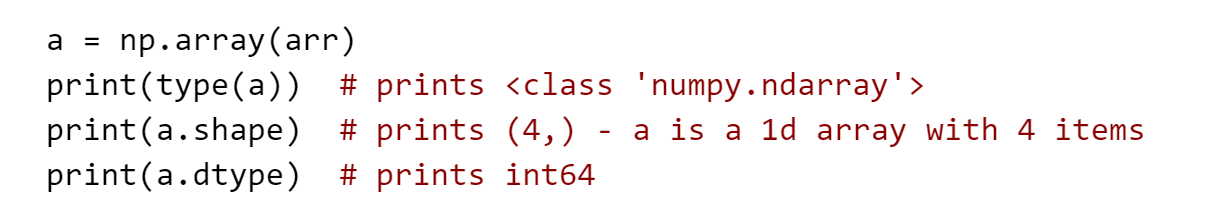
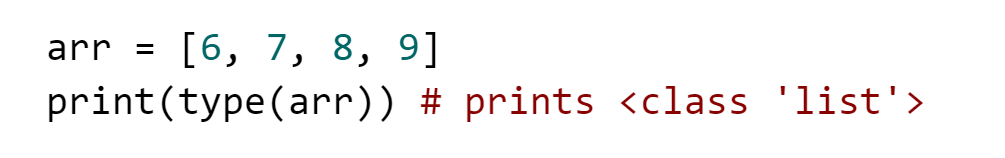
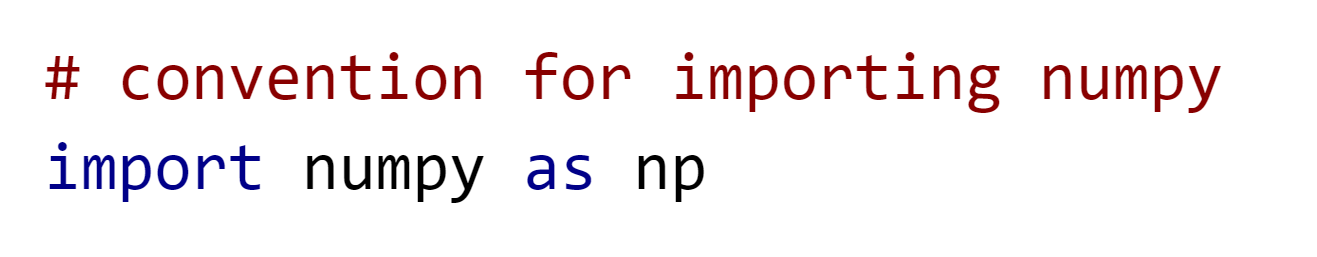
1.1.4: Libraries for Python Data Analysis

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

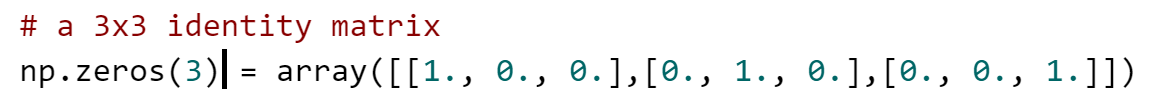
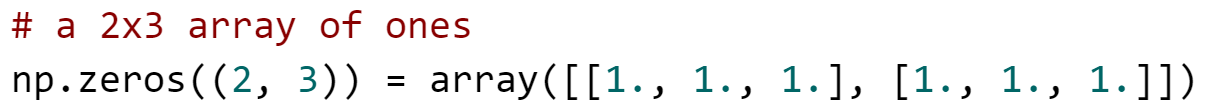
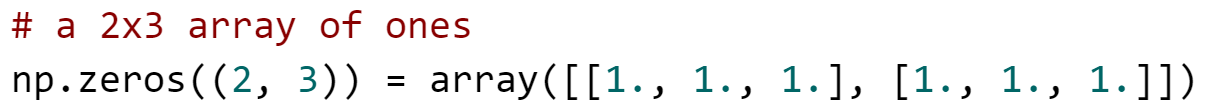
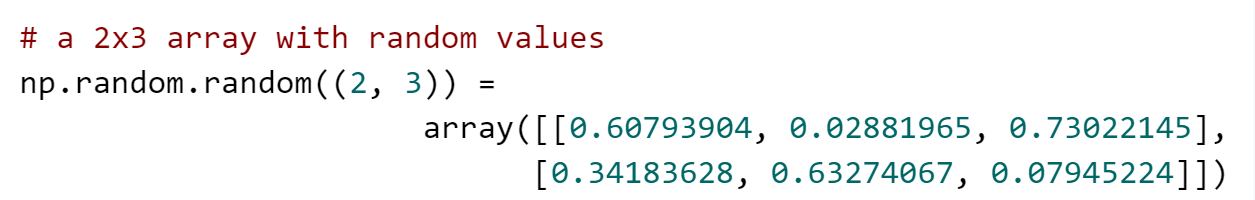
**Pandas, NumPy, SciPy, Matplotlib and Seaborn are essential Python libraries used for data analysis**. ***Numerical computations for arrays and multidimensional matrices in data analysis are often done with the Numeric Python library - NumPy***. ***Pandas is a toolkit built on NumPy with data structures 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**

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:

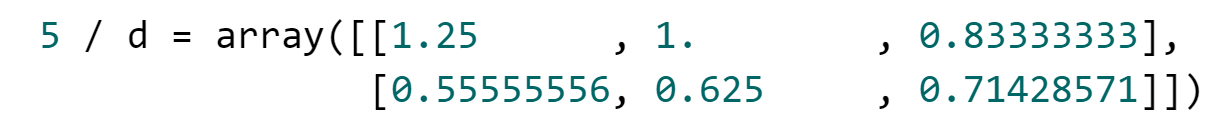
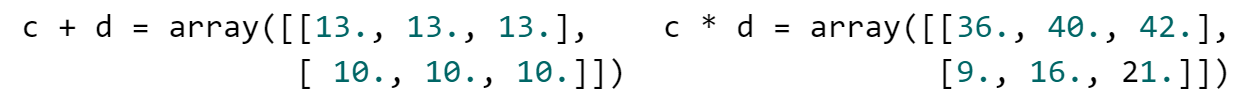
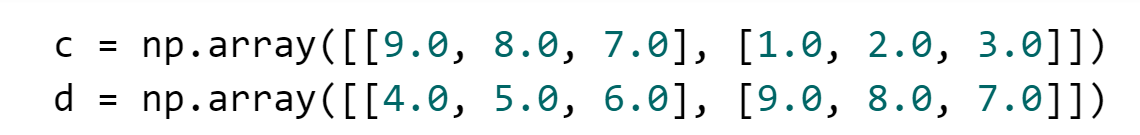


There are also some inbuilt functions that can be used to initialize numpy which include empty(), zeros(), ones(), full(), random.random().



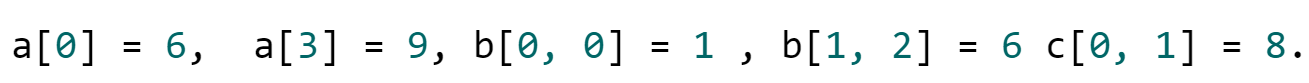
Intra-operability of arrays and scalars.

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](https://hamoye.com/app/search/details/Q11205/arithmetic) 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.

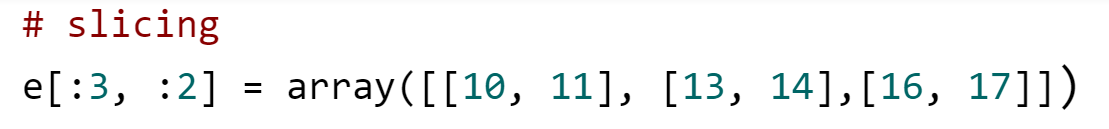
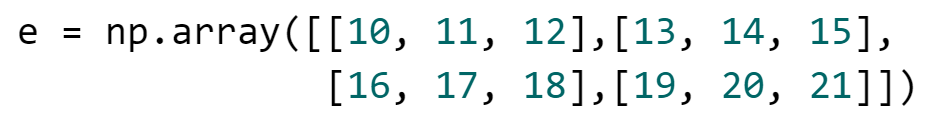


Indexing with arrays & using arrays for data processing

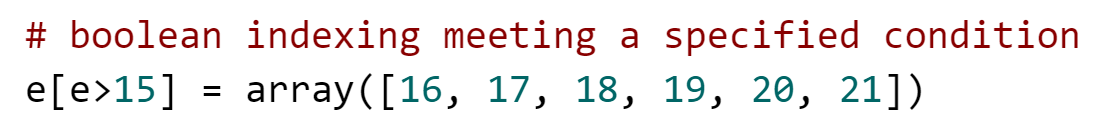
The elements in the example arrays above can be accessed by indexing like lists in Python such that:



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.



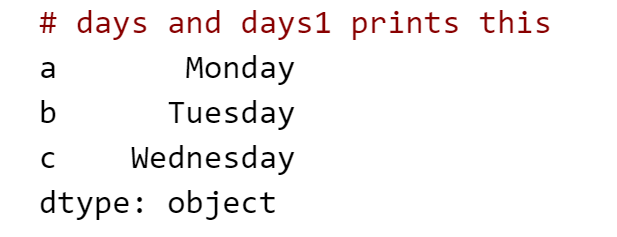
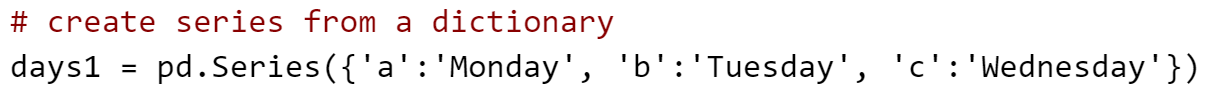
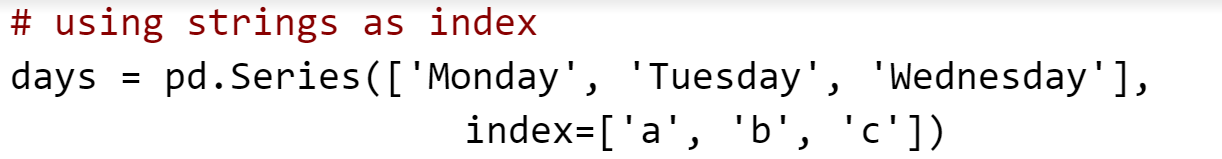
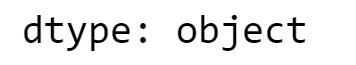
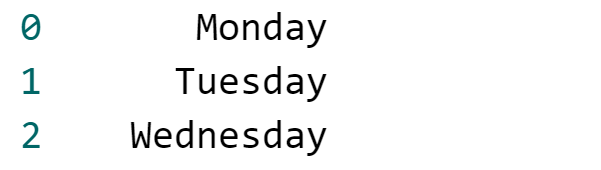
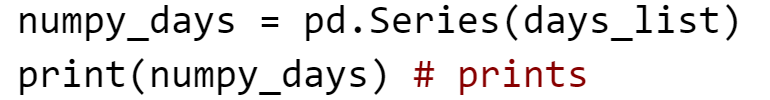
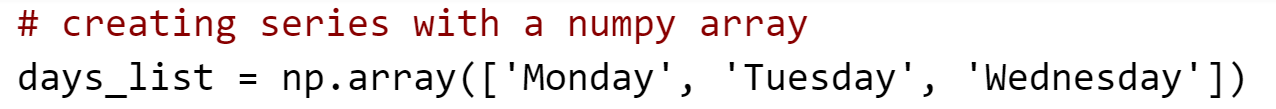
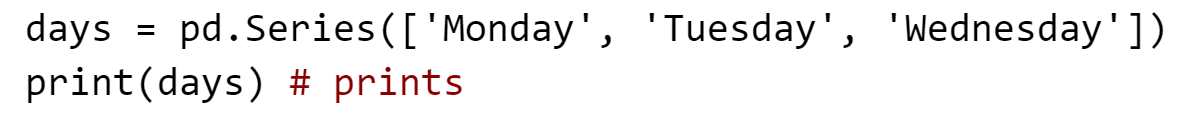
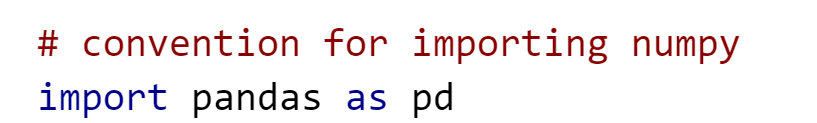
Numpy also has inbuilt mathematical functions like sum(), mean(), std(), corrcoef(), min() and others. It interestingly allows for comparing [arrays](https://hamoye.com/app/search/details/Q186152/array%20data%20structure) using == to check if two arrays have the same elements,  elements in the first array are greater than or less than those of the second array using  > and  <.

File input and output with arrays

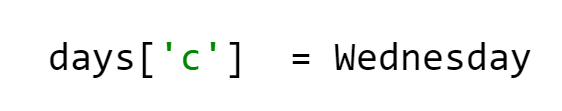
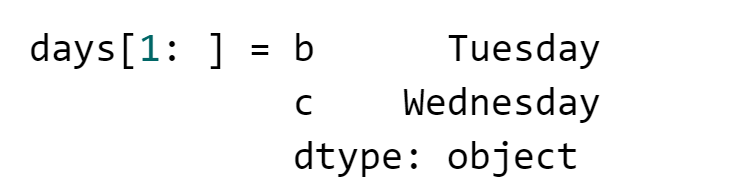
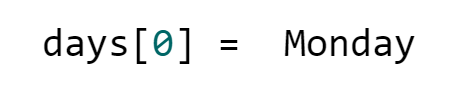
Numpy arrays can be loaded from and saved to [binary files](https://hamoye.com/app/search/details/Q3502441/Binary%20Interchange%20File%20Format) with .npy as the extension using load() and save() respectively. This can also be done with text files with text files using loadtxt() and savetxt().

**Pandas - So much more than a cute animal**

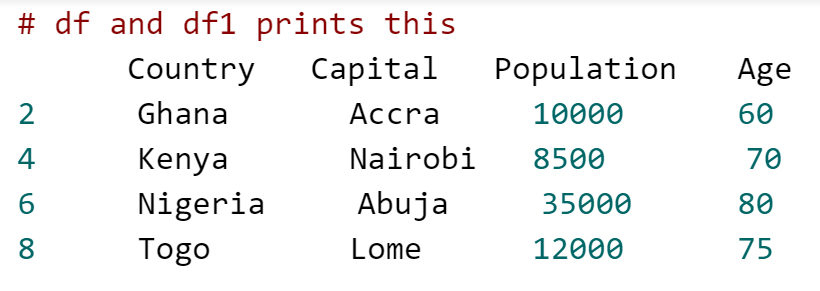
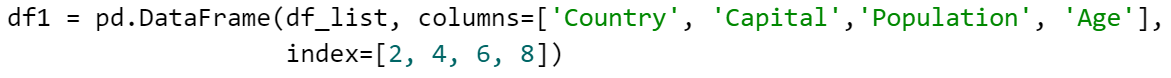
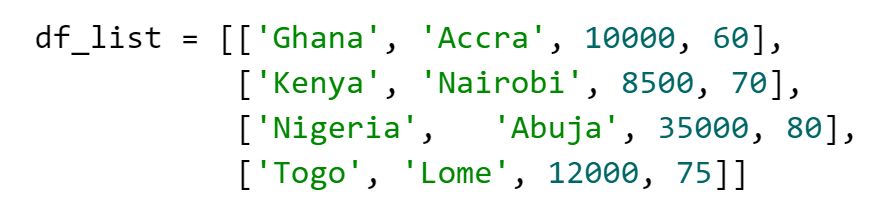
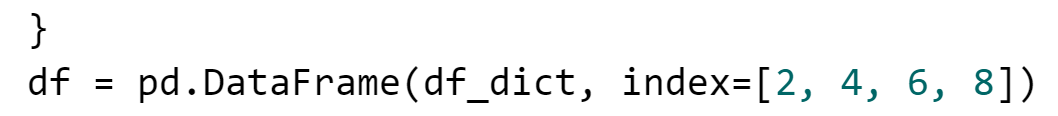
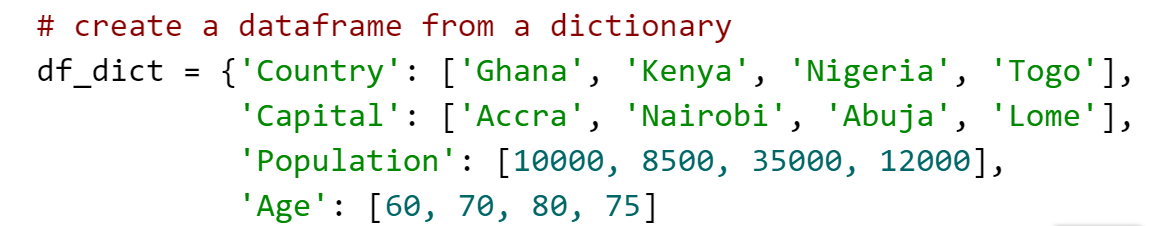
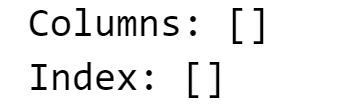
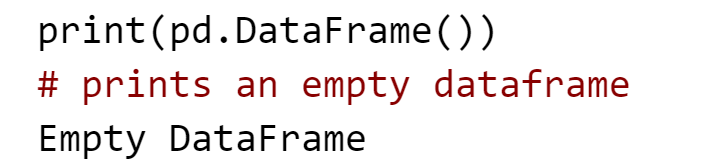
Pandas is a library used for data manipulation and built on Numpy with other ways of indexing other than using integers. Series, DataFrame and index are the basic data structures in this library.  Series in pandas can be referred to as a one dimensional array with homogenous elements of different types somewhat similar to numpy arrays however, it can be indexed differently with specified descriptive labels or [integers](https://hamoye.com/app/search/details/Q729138/integer%20data%20type).



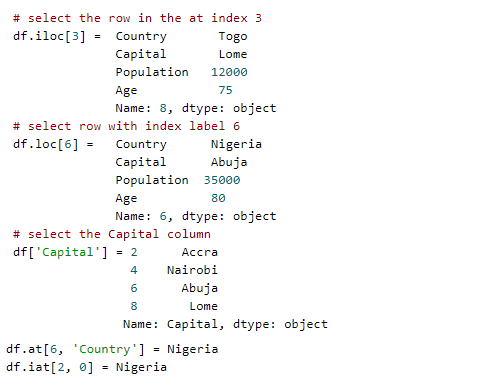
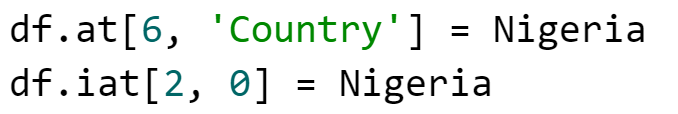
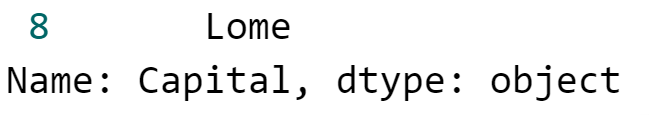
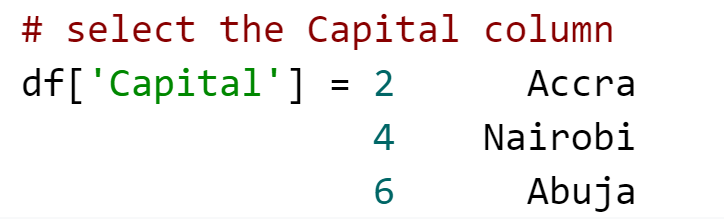
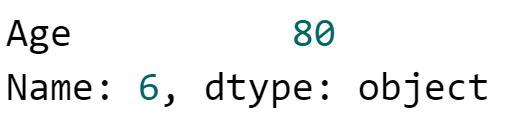
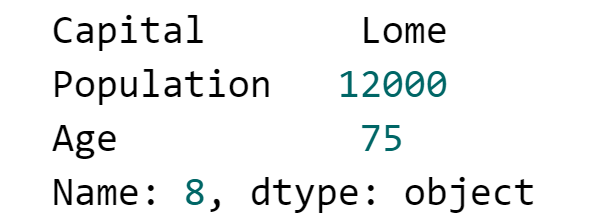
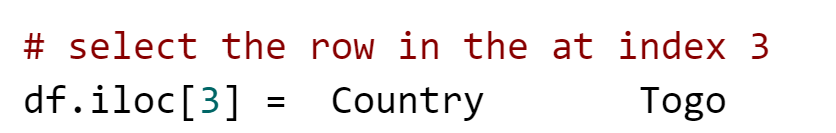
Series can be accessed using the specified index as shown below



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 row 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.

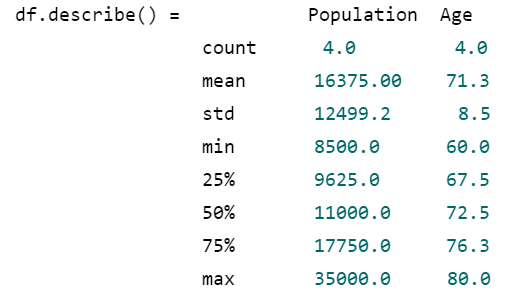
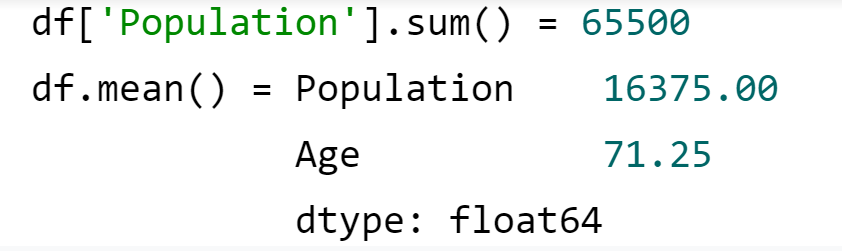


Finally, Indexes in pandas are immutable arrays with unique elements or can be described as ordered sets for retrieving data in a dataframe and collaborating with multiple dataframes.

The important Pandas functionalities: indexing, reindexing, selection, group, drop entities, ranking, sorting, duplicates and indexing by hierarchy.

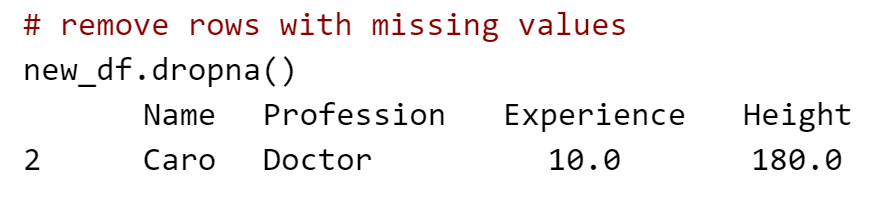
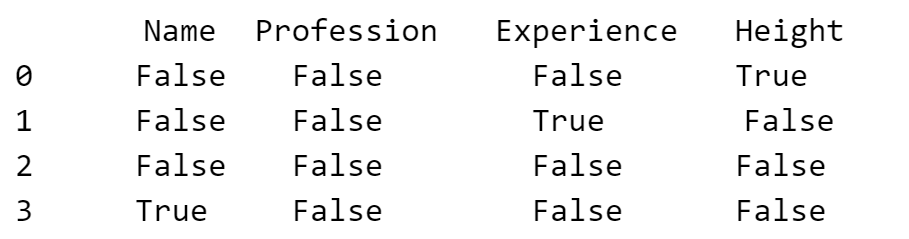
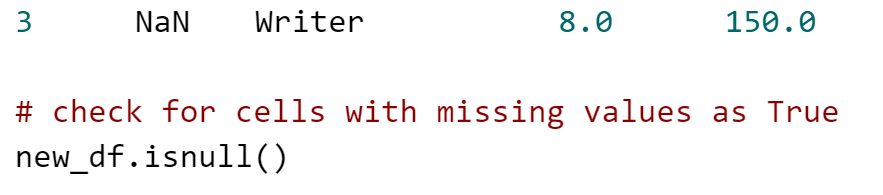
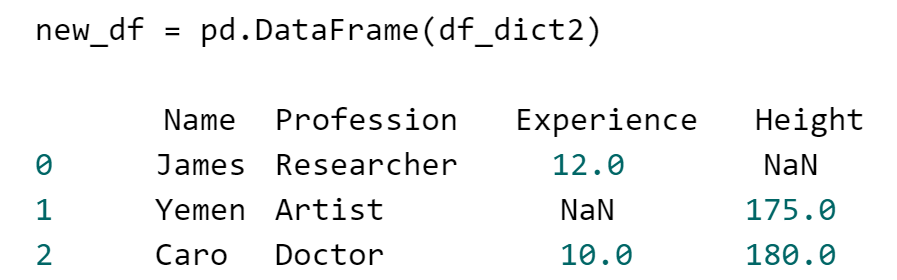
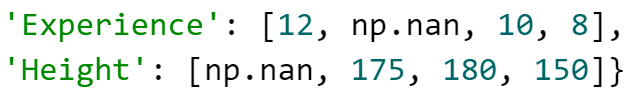
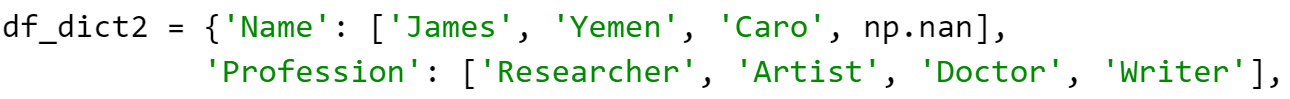
Summary and descriptive statistics: measure of central tendency, measure of dispersion, skewness and kurtosis, correlation and multicollinearity

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.

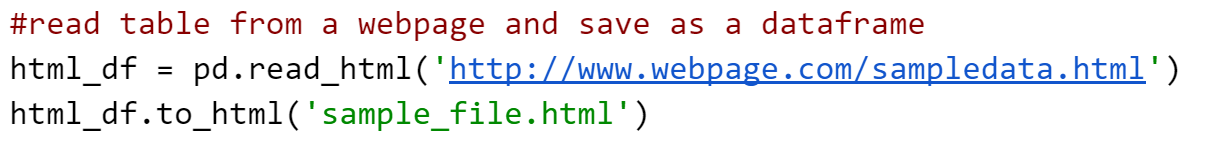
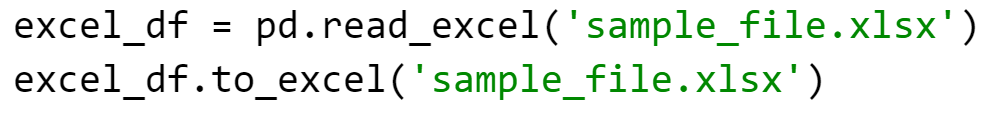
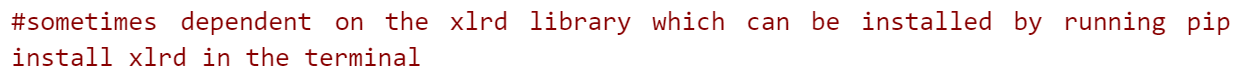
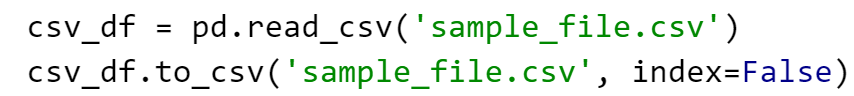
Often data used for analysis in real life scenarios is incomplete as a result of omission, faulty devices and many other factors. Pandas represent missing values as NA or NaN which can be filled, removed and detected with functions like fillna(), dropna(), isnull(), notnull(), replace().



**Data Types and Data Wrangling**

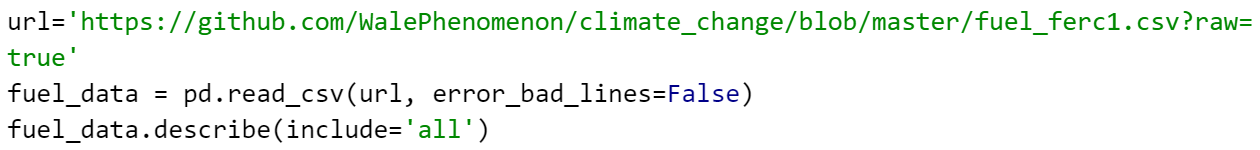
Working with different types of data: text files, CSV, JSON objects, HTML and databases.

The pandas library is vast enough to read data from and save to several file formats such as CSV, JSON, HTML and even databases.

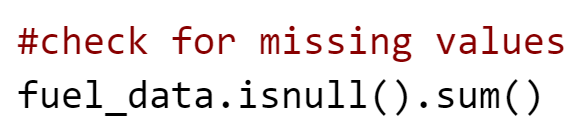


Pandas can connect to databases, get data with queries and save in a dataframe.

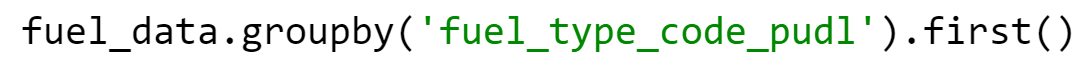
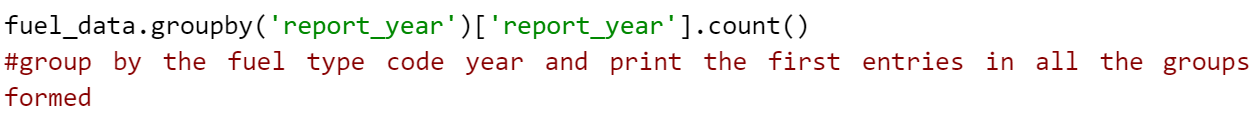
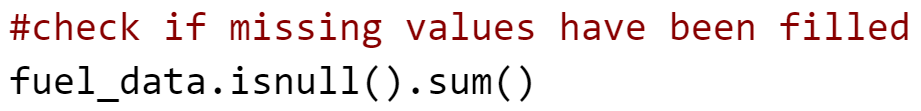
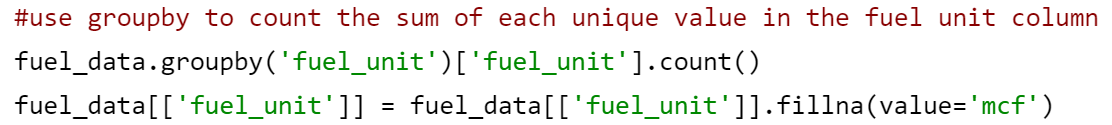
In the later part of this section, we will use [this](https://github.com/WalePhenomenon/climate_change/blob/master/fuel_ferc1.csv?raw=true) fuel dataset to perform some data wrangling operations which can be found in the example notebook for this module. In our notebook, after reading the csv file, we proceed to get a summary of the dataset using the describe function.



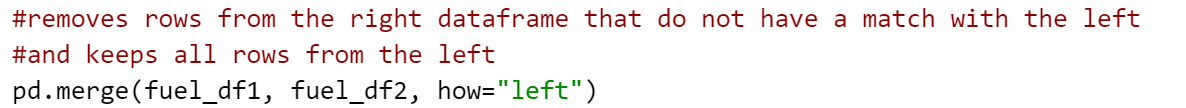
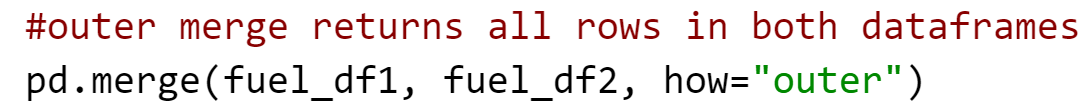
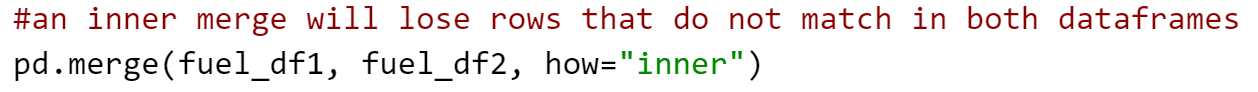
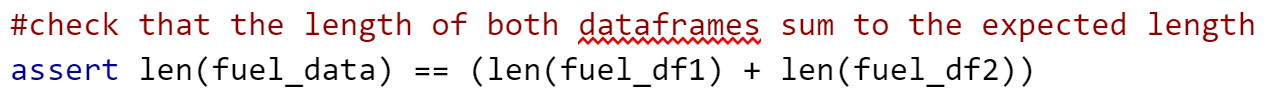
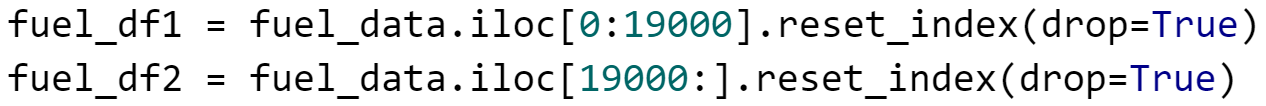
Our analysis shows that there are 180 missing values in the fuel data column. We handle this by filling with the most common value in the column - mcf.



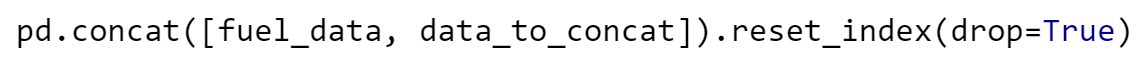
A dataframe can be easily categorised into different segments based on a given criteria using the groupby() function. This initially splits the dataframe into the groups then applies a function to the groups after which the results are combined.



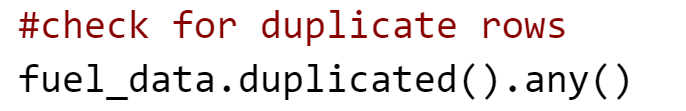
Merging in Pandas can be likened to join operations in relational databases like SQL. Left, inner, right and outer are the merging methods available to the merge() function. The left method can be likened to SQL left outer join,  inner to  SQL inner join, right to SQL left outer join and outer to SQL full outer join. In our analysis, we split the fuel data into two groups and merge using different methods.



Concatenation is performed with the concat() function by combining series or dataframes while keeping the indices of the individual unit irrespective of duplicate indices. In the notebook, we created a dummy dataframe data\_to\_concat which we concatenated to the fuel\_data as below:



Duplicates are a common occurrence in datasets which alter the results of data analysis. Hence, in practice, removing duplicate values is very important. The duplicated() function is used in Pandas to check for and handle duplicates.



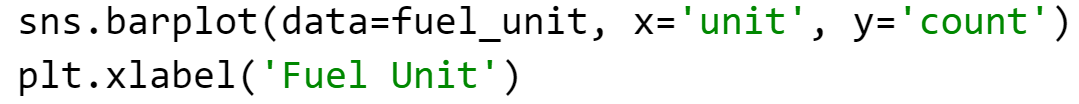
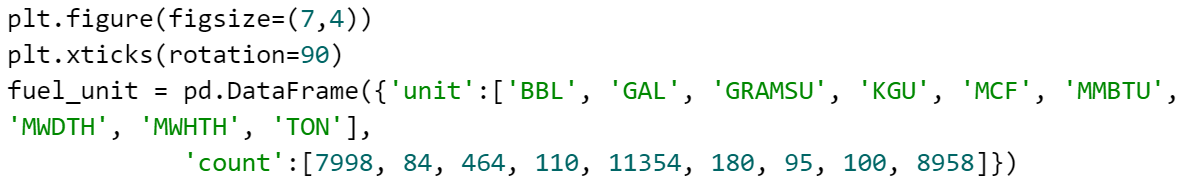
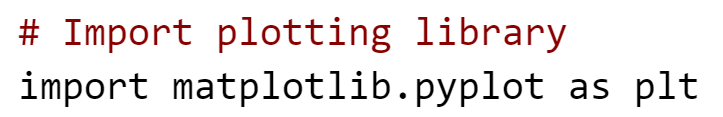
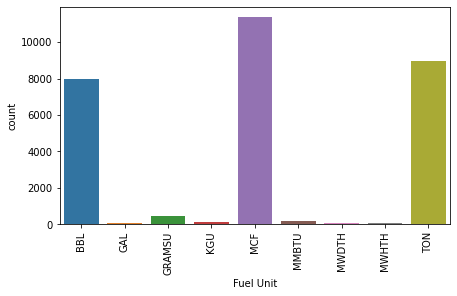
**Data Visualization and Representation in Python**

The Anscombe Quartet and the importance of visualizing data.

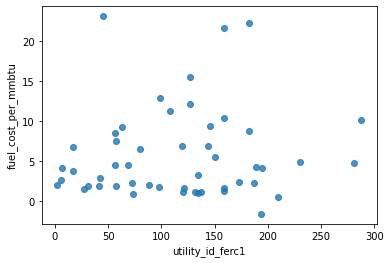
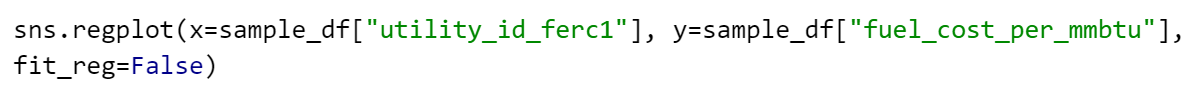
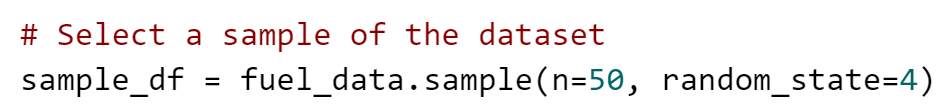
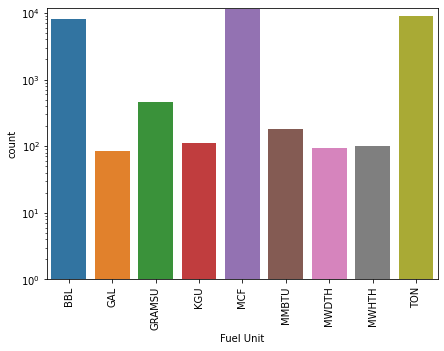
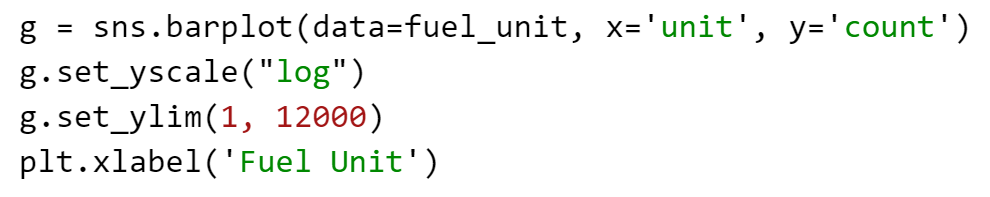
Assuming we have collected data on energy consumption across different states in a  country and how much people earn in these states, calculating some summary statistics can give quick insights to which state consumes the most energy, how much energy is used on average in the country, the [correlation](https://hamoye.com/app/search/details/Q186290/correlation) between people’s earnings and energy consumed in their states and many others. It is essential to note that while these statistics are important and give a description of the dataset, it is not sufficient to use the results alone without plotting the data to obtain a holistic view of the overall distribution. Anscombe Quartet identifies that different datasets can have the same or very identical statistical properties such that they can be labelled the same but when graphed, they are seen to have different distributions.

Plotting: Area plots, Histograms, Bar charts, pie charts and scatter charts

There are several types of charts used in data visualisation which are selected based on the data and the information aimed to be communicated. Area plots, histograms, bar charts, pie charts and scatter charts are some of the simple and common graphs used in data analysis. Graphs assist in understanding data when performing EDA and in conveying insights easily.  A line graph is a basic plot that displays the relationship between two variables on each axis by connecting data points together with straight lines. To show magnitude, the segment between the line and the x-axis is filled which results in an area graph. Histograms and bar charts are completely different plots that can be mistaken as similar. The former are charts used to represent the distribution of a group and use adjacent rectangular bars to display the frequency of intervals while the latter are charts that represent categories using equally spaced rectangular bars.



Because of the extreme range of the values for the fuel unit, we can plot the barchart by taking the logarithm of the y-axis as follows:

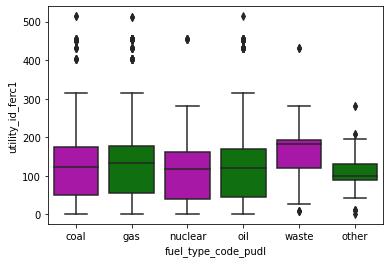


Another common chart is the pie chart which displays a circular representation of the contribution of each proportion of the categories in a dataset. The sum of each proportion is always 100 (percentage). Finally, scatter charts are simple charts similar to the line charts, however, the markers used to represent data points are not joined with lines instead, they are scattered on the charts and easily display the correlation between the variables.

Advanced plotting: Kernel Density Estimate plots, box plots and violin plots

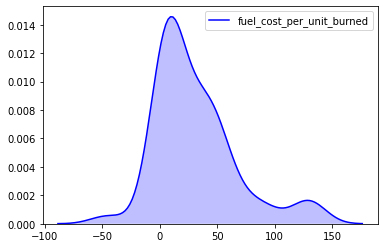
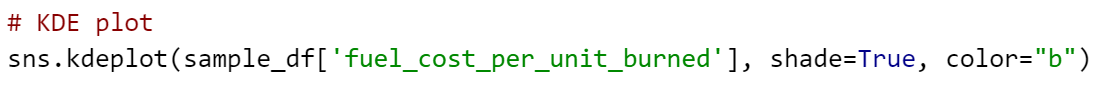
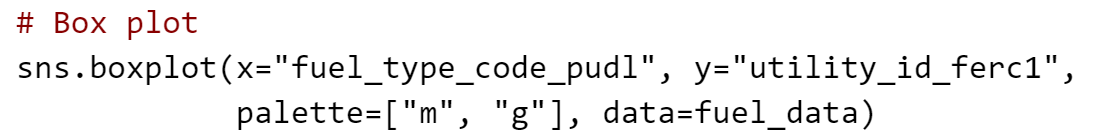
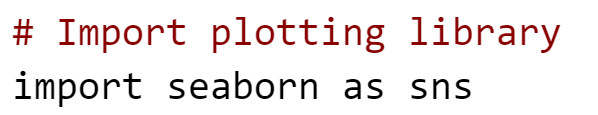
A box plot also called a box and whisker plot is a representation of data that displays the distribution and summary statistics such as the median and the interquartile range of the dataset. This plot shows outliers and makes it easy to compare across different categories.

Although a violin plot is very similar to a box plot such that it  displays the distribution of the data, it provides more information by also showing the probability density of the data rotated on each side. In simple terms, the violin

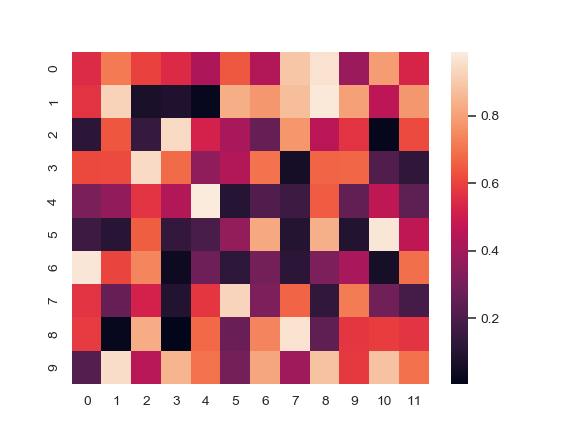


plot displays more information than the box plot however, it is influenced by the sample size. Using Seaborn,  we create a box plot showing the fuel type groups

and, the KDE plot of a sample of fuel cost per unit burned in the fuel data:



Heatmaps and regression plots



A heatmap is a representation of data that uses a spectrum of colours to indicate different values. It gives quick summaries and identifies patterns especially in large datasets. Alternatively, heatmaps can be described as table visualisations where the colour of each cell relates the values. The image above is an example of a heatmap.

Further reading:

1. [The Missing Data Conundrum](https://medium.com/ibm-data-science-experience/missing-data-conundrum-exploration-and-imputation-techniques-9f40abe0fd87) by Wale Akinfaderin.

2. [Python Data Science Handbook](https://jakevdp.github.io/PythonDataScienceHandbook/) by Jake VanderPlas.

3. [Technical Notes on Data Science and Machine Learning](https://chrisalbon.com/) by Chris Albon

#### [IMPORTANT] Instructions

## **Please read the following instructions carefully:**

For the Graded Assessment, you are expected to make use of this dataset with link below:

<https://bit.ly/HDSC-StageOneDataset>

You are to attempt all questions in the Graded Quiz within the stipulated time.

After completion of the assessment, you are required to  submit the link to your code using the Google Form below.

<https://bit.ly/HDSC-StageOneAssessment>

Please note that plagiarism is highly prohibited and you will be disqualified if found guilty of it.

#### Dataset Description

The data provided in this [Github link](https://bit.ly/HDSC-StageOneDataset) is the fuel quality data from the Federal Energy Regulatory Commission which is provided by the United States Energy Information Administration. The data consists of the following columns:

'Record\_id' : record id

'Utility\_id\_ferc1': Utility id assigned by the FERC

'Report\_year': year of report

'Plant\_name\_ferc1': the name of the plant

'Fuel\_type\_code\_pudl': the type of fuel

'Fuel\_unit': the unit of fuel

'Fuel\_qty\_burned': the quantity of fuel burned

'Fuel\_mmbtu\_per\_unit': the measure of energy per unit

'fuel\_cost\_per\_unit\_burned': the fuel cost per unit burned

'Fuel\_cost\_per\_unit\_delivered': the cost of fuel delivered per unit

'fuel\_cost\_per\_mmbtu': the cost of fuel per mmbtu