Introduction to Python for Big Data Analytics

Welcome to Big Data Analytics course. This course is designed to support you in understanding not just the basics of Big Data analytics but how it is so crucial in various fields.

# WEEK 1: Introduction to Python for Big Data Analytics

## Introduction to Python for Big Data Analytics

### 1.1 Introduction

Welcome to Big Data Analytics course. This course is designed to support you in understanding not just the basics of Big Data analytics but how it is so crucial in various fields.

However, due to its highly specialised content, Big Data Analytics is often left to computer scientists and mathematicians. But does it have to be like that?

First of all, let me introduce what I mean by big data analytics. As you may have already come across, it is a relatively new area motivated by the increasing availability of large quantities of data. But what is data? And why now?

Data is everything. It’s what surrounds us and what our brain needs to function. It’s images, sounds, touch, as well as their (more structured) derivatives, such as music, texts, etc. But the key issue here is that data is not information. It must be turned into information via a set of processes and methods to identify trends and patterns that can pull all the different data together to make something meaningful.

Another important aspect of big data analytics is that it is an applied science. In other words, the power of data analytics is often seen in its applications. In fact, despite being motivated and backed-up by sophisticated mathematical and statistical techniques, we, data scientists, need to experiment, manipulate, and ultimately apply it to the real-world. In fact, big data analytics can be applied to digital humanities, sociology, psychology, criminology, cybersecurity, to name but a few.

This is why, during this short course, we shall consider specific case studies, which hopefully should provide an insightful overview on the main methods and applications of big data analytics.

### 1.2 Welcome and Introduction Video

Video

0:01 Skip to 0 minutes and 1 second

Welcome to the Big Data Analytics course. This course is designed to support you in understanding not just the basics of big data analytics, but how it is so crucial in various fields. However, due to its highly specialised content, big data analytics is often left to computer scientists and mathematicians. But does it have to be like that? Not at all. First of all, let me introduce what I mean by big data analytics. As you may have already come across, it is a relatively new area motivated by the increasing availability of large quantities of data. But what is data and why? Now, data is everything. It’s what surrounds us and what our brain needs to function.

0:58 Skip to 0 minutes and 58 seconds

Its images sounds touch, as well as the more structured derivatives such as music, texts, etc. But the key issue here is that data is not information. It must be turned into information by a set of processes and methods to identify trends and patterns that can pull all the different data together to make something meaningful. Another important aspect of big data analytics is that it is an applied science. In other words, the power of data analytics is often seen in its application. In fact, despite being motivated and backed up by sophisticated mathematical and statistical techniques, we data scientists need to experiment, manipulate and ultimately apply to the real world.

1:57

Skip to 1 minute and 57 seconds

In fact, big data analytics can be applied to social humanities, sociology, psychology, criminology, cybersecurity, to name but a few. This is why during this short course, we shall consider specific case studies which hopefully should provide an insightful overview on the main methods and applications of big data analytics.

### 1.3 Specific methods relevant to data analytics

The main data analytics methods include

* Machine Learning
* Text mining
* Sentiment analysis
* Systematic reviews
* General statistical analysis
* Visualisation

Let me go through them, one by one. Machine learning, as the name suggests, aims to mimic the way we humans are learning. We might observe patterns, shapes, and trends in a given data set. However, we usually are only able to consider small datasets. So, how can we scale up? Can we create methods and frameworks so that the same procedures are carried out automatically?

This is the aim of machine learning. There are two types of approaches to ML. The first one is supervised learning. Based on a labelled dataset (where the expected output related to the corresponding input is given), the learning process is facilitated. In other words, we ‘educate’ our model.

On the other hand, unsupervised ML does not have any labelled data. It will learn as it progresses through the computation. Despite the differences, their objective is identical, that is finding data trends, and/or classifying data. The main ML models fall into the following:

Regression: a trend (usually depicted by a ‘curve’) is identified so that a prediction can be carried out

Classification/clustering: given a dataset, this model will group them into sub-sets based on specific attributes or properties

Dimensionality reduction: attempt to remove uninformative dimensions from a multi-dimensional dataset

We communicate via spoken and written languages. The amount of information shared, stored, and multiplied across texts, books, novels, etc, throughout human history is staggering. Text mining and sentiment analysis are two ML approaches to analyse textual sources.

More specifically, the former aims to extract concepts, relationships, entities, and semantic information, whereas the latter, despite being part of text mining, focuses not only on what we are talking about but how we are talking about it. In fact, often it is not just important to identify linguistic concepts, but the corresponding opinion held by individuals. This is particularly relevant in, for example, marketing, where sentiment analysis is used to analyse how people discuss brands.

## Introduction to Python

### 1.4 Introduction to the Python Jupyter Notebook

The Jupyter Notebook

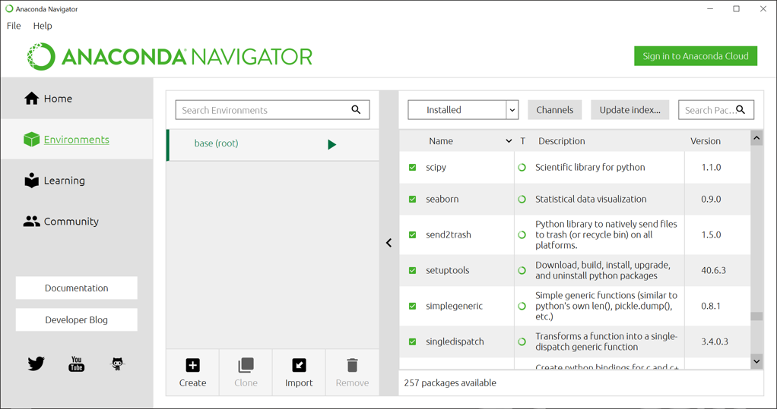
The Jupyter Notebook is an open-source web-based application that you can use to create and share documents that contain live code, equations, visualisations, and text. It is widely used within the Python community as it allows an easy-to-follow visual and real-time implementation. It is particularly effective at sharing code. Let’s look at how it works

Anaconda allows to

* Install/upgrade packages
* Without Administrator privileges
* Without breaking other projects
* Please go to Anaconda Installers to download Anaconda.

As an example, follow the following steps

* Start ‘Anaconda Navigator’
* Click on ‘Environments’ tab
* Check the packages in the ‘base (root)’ environment
* Check that ‘seaborn’ is installed and current (version 0.9.0 or later

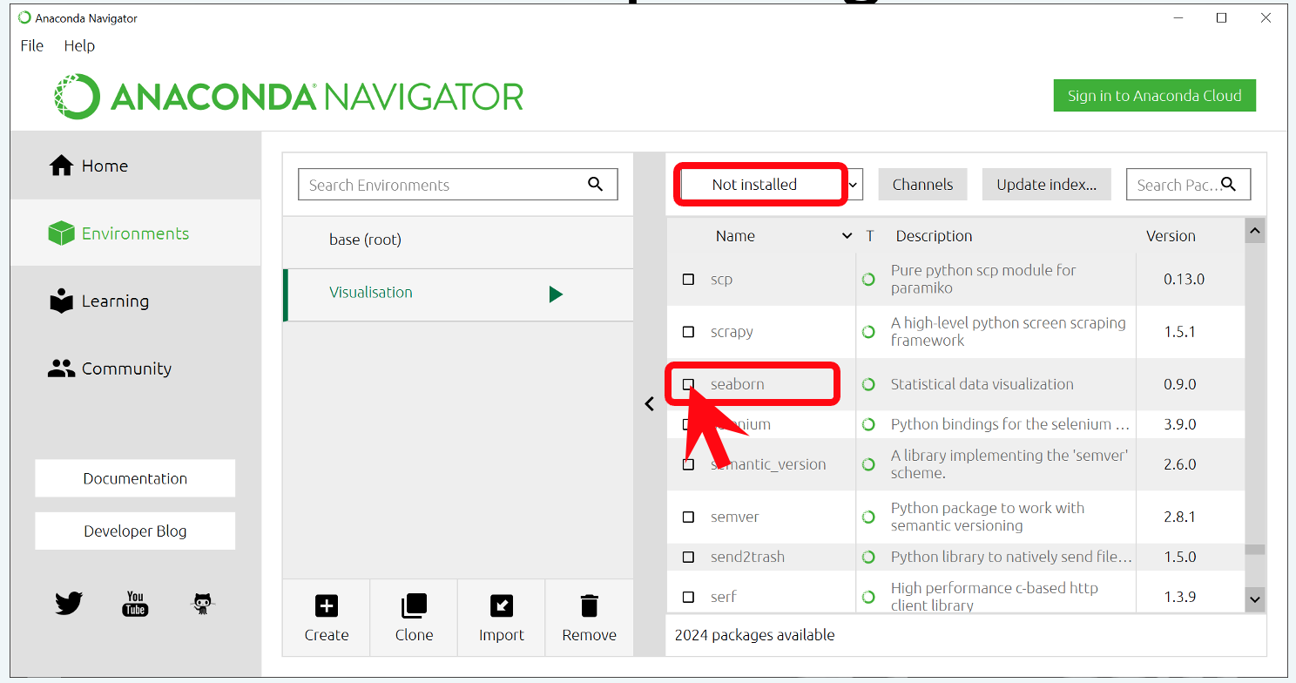


Creating a New Environment

* The ‘base (root)’ Environment requires Administrator privileges
* If you need to install/upgrade packages, you need to create your own new Environment

As an exercise, create the following as depicted in the figure

* Name: Visualisation
* Packages: Python 3.6
* Install the ‘seaborn’ package (and dependencies)



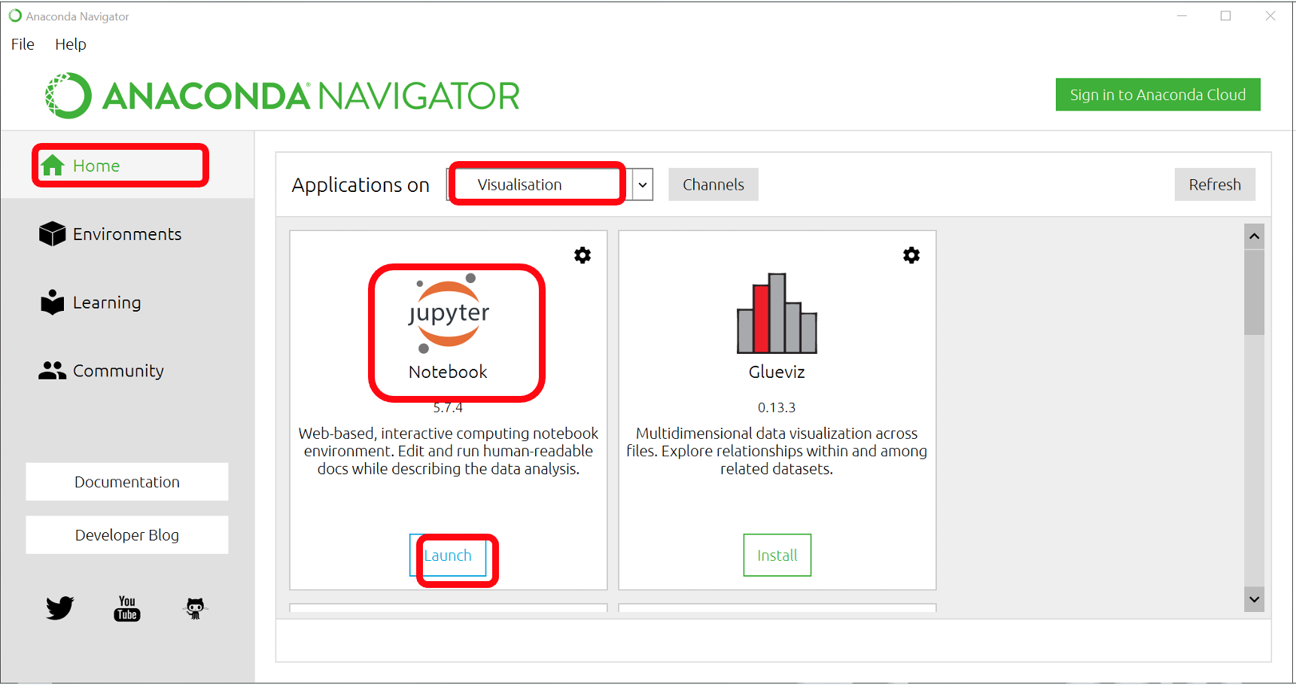
Jupyter Notebook

Jupyter Notebook is a browser-based, interactive Python IDE, especially suitable for

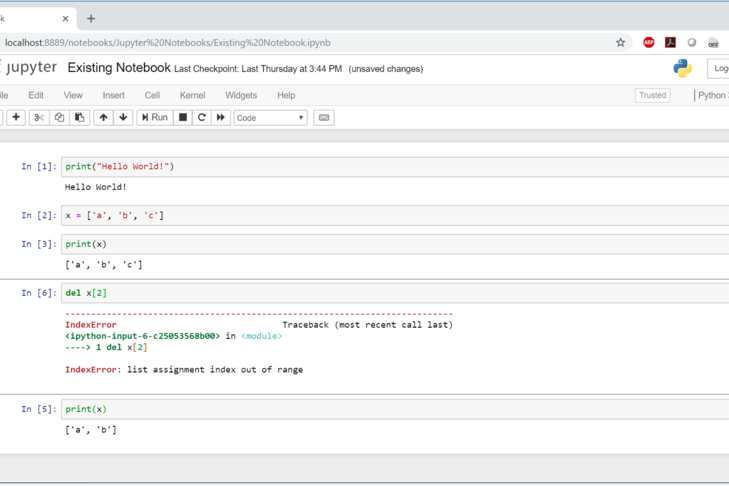
* Prototyping
* Creating documents with interactive code and illustrations
* Developing standalone applications

Do the following steps (as per figure):

* From the Home tab in your Visualisation Environment, install Jupyter Notebook (once)
* Launch Jupyter Notebook in your preferred browser



### 1.5 Jupyter Notebooks



Follow the following steps

* Create a New Python 3 Notebook or open an existing notebook
* Type Python code in an ‘In [ ]:’ cell
* Press Shift+Enter in a cell to run that code

Note that

* Cells are just snippets of codethat you can run in any order
* What happens in a cell doesn’t stay in that cell
* Running a cell can change the state of the notebook (e.g., variables created or changed)
* A cell may work once, but not the next time, if the state of the notebook has changed
* Restart the kernel to reset the state of the notebook

What do you see?

Try some other examples

## General Python Syntax

### 1.6 General Python Syntax

**Introduction to Python**

Python is an interpreted computer language, which has become increasingly popular among data scientists. It was designed to emphasise code readability, by using white­space indentation to define code blocks rather than curly brackets or keywords. In particular, it is widely used both at the prototyping level and professional development.

An example of the most basic operations include:

* Addition: 10 + 4
* Subtraction: 10 - 4
* Multiplication: 10 \* 4
* Exponentiation: 10 \* \*4(=10000)
* Division: 10/4

Note that the division will give you as both numbers are integers. To include the full digits of this operation, it is necessary to perform 10 / float(4)’

Conditional statements are widely used to select specific commands if some conditions are satisfied. The following is an example of conditional statements.

x**=**3

*# if statement*

**if** x **>** 0: *# note the ":" after the conditional.*

*# print "positive" if x is greater than zero. The block of coding, which has to be performed if the conditional is satisfied, is indented*

**print**('positive')

**else**:

**print**('zero or negative')

*# if/elif/else statement*

**if** x **>** 0:

**print**('positive')

**elif** x **==** 0:

**print**('zero')

**else**:

**print**('negative')

*# single-line if statement (sometimes discouraged)*

**if** x **>** 0:

**print**('positive')

Try to understand the above code. What do you notice? Try to remove the indentation of the command print(‘positive’) after the if x>0: statement. What happens?

Experiment with the code and try different conditions. Also, explore different types of conditionals, such as while , etc.

**Lists and Dictionaries**

Lists and dictionaries are also widely used in Python, which has various methods and libraries to allow their creation, manipulation and visualisation. The following code illustrates an example of a list.

*## properties: ordered, iterable, mutable, can contain multiple data types*

*# create an empty list (two ways)*

empty\_list **=** []

empty\_list **=** list()

*# create a list*

simpsons **=** ['homer', 'marge', 'bart']

*# examine a list*

simpsons[0] *# print element 0 ('homer')*

len(simpsons) *# returns the length (3)*

**print**(simpsons) *# print the initial list*

*# modify a list (does not return the list)*

simpsons.append('lisa') *# append element to end*

simpsons.extend(['itchy', 'scratchy']) *# append multiple elements to end*

**print**(simpsons) *# print the list*

simpsons.remove('bart') *# searches for first instance and removes it*

simpsons.pop(0) *# removes element 0 and returns it*

**print**(simpsons) *# print the list*

**del** simpsons[0] *# removes element 0 (does not return it)*

simpsons[0] **=** 'krusty' *# replace element 0*

simpsons.insert(0, 'maggie') *# insert element at index 0 (shifts everything right)*

**print**(simpsons) *# print the list*

*# concatenate lists (slower than 'extend' method)*

neighbors **=** simpsons **+** ['ned','rod','todd']

*# find elements in a list*

simpsons.count('lisa') *# counts the number of instances*

simpsons.index('itchy') *# returns index of first instance*

**print**(simpsons) *# print the list*

Try to find other examples of list operations. The following code provides an example of a dictionary.

*# create an empty dictionary (two ways)*

empty\_dict **=** {}

empty\_dict **=** dict()

*# create a dictionary (two ways)*

family **=** {'dad':'homer', 'mom':'marge', 'size':6}

family **=** dict(dad**=**'homer', mom**=**'marge', size**=**6)

*# convert a list of tuples into a dictionary*

list\_of\_tuples **=** [('dad','homer'), ('mom','marge'), ('size', 6)]

family **=** dict(list\_of\_tuples)

*# examine a dictionary*

family['dad'] *# returns 'homer'*

len(family) *# returns 3*

family.keys() *# returns list: ['dad', 'mom', 'size']*

family.values() *# returns list: ['homer', 'marge', 6]*

family.items() *# returns list of tuples: [('dad', 'homer'), ('mom', 'marge'), ('size', 6)]*

**print**('marge' **in** family) *# returns False (only checks keys)*

**print**(family) *# print family*

*# modify a dictionary (does not return the dictionary)*

family['cat'] **=** 'snowball' *# add a new entry*

family['cat'] **=** 'snowball ii' *# edit an existing entry*

**del** family['cat'] *# delete an entry*

family['kids'] **=** ['bart', 'lisa'] *# value can be a list*

family.pop('dad') *# removes an entry and returns the value ('homer')*

**print**(family) *# print family*

Find examples of other dictionary operations. What are the main differences between lists and dictionaries? Which do you prefer?

**Functions and Loops**

Functions in Python can be used to define a code fragment, which can be called at any time.

*# define a function with one argument and one return value*

**def** **add\_number\_to\_two**(x): *# note the ":"*

**return** x **+** 2 *# the return statements specifies the output of the method. Note the indentation.*

**print**(add\_number\_to\_two(4)) *# This returns 6*

*# return two values from a single function*

**def** **min\_max**(nums):

**return** min(nums), max(nums)

*# return values can be assigned to a single variable as a tuple*

nums **=** [1, 2, 3]

min\_max\_num **=** min\_max(nums) *# min\_max\_num = (1, 3)*

*# Simple loop*

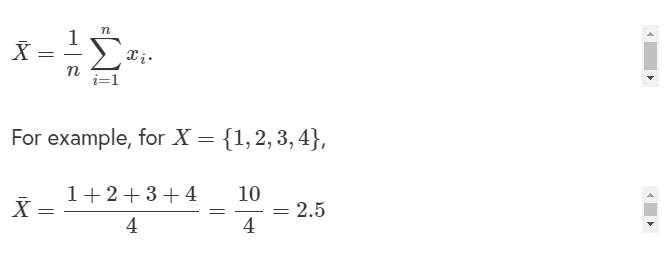
**for** i **in** range(3): *# the range(n) command specifies a list of values from 0 to n-1*

**print**(i)

**for** j **in** range(5):

**print**(add\_number\_to\_two(j))

Find examples of other loops and functions.



Using Python we have

**from** \_\_future\_\_ **import** division

X **=** [1,2,3,4]

temp **=** 0

**for** i **in** range(len(X)):

temp **=** temp **+** X[i]

mean **=** temp**/**len(X)

**print**(mean)

*# Creating a function 'mean'*

**def** **mean**(input\_list):

temp **=** 0

**for** i **in** range(len(input\_list)):

temp **=** temp **+** input\_list[i]

mean **=** temp**/**len(input\_list)

**return** mean

**print** (mean(X))

We can also use specific Python libraries, which provide off-the-shelf functions and modules, such as Numpy

**import** numpy

X **=** [1,2,3,4]

mean **=** numpy.mean(X) *# Find the mean*

std **=** numpy.std(X) *# Find the standard deviation*

**print**(mean)

**print**(std)

Explore what Numpy can do.

Hard exercise! Can you implement the standard deviation for using first principles, rather than the Numpy library?

**Methods in Python**

In Python, we can define fragments of code that can be invoked at any time. These are called methods and defined by the keyword def. Consider, for example, the area of a circle , where and is its radius. For any circle, the only parameter that changes is its radius . So we can write

In other words, by using rather than , we specify that the area is a function of . So, if we want to calculate the area of a circle with radius , the above Equation gives us that . Therefore, our input is and the output will be the number evaluated by . This is how a method (or function) is defined in Python. We specify an input and output based on a sequence of steps. So the area equation can be written as

**def** **A**(r): *# note the ":" and the indentation of the block of commands in the method*

area **=** r**\***r**\***3.141592

**return** area

**print** A(2)

12.566368

Note the return statement. This specifies the output of the method. The area of a square with side is . Define a method to find the area of any square.

**Vectors and Matrices in Python**

Python has various libraries, which allow easy manipulation of vectors and matrices. However, as an example, let us create a Python code to evaluate the sum of two vectors using first principles

**def** **vector\_addition**(v\_1,v\_2):

sum **=** [] *#need to initialise sum as an empty list*

**if** len(v\_1) **==** len(v\_2): *#vector addition makes sense if they are both of the same length*

**for** i **in** range(len(v\_1)): *#consider each component*

temp **=** v\_1[i]**+**v\_2[i] *#add the i-th component*

sum.append(temp) *#add the sum of the two components to the vector sum*

**else**:

**print**("The two vectors must have the same dimension!")

**return** sum

a **=** [1,2,3]

b **=** [**-**1,**-**2,**-**3]

**print**(vector\_addition(a,b))

Modify the above code to evaluate vector subtraction and scalar multiplication.

### 1.7 Introduction to Data Analytics Libraries

**Pandas**

Pandas is a Python package providing fast, flexible, and expressive data structures designed to work with relational or labelled data. It is a fundamental high-level building block for doing practical, real-world data analysis in Python.

Pandas is well suited for:

* 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 actually need not be labelled at all to be placed into a pandas data structure

Key features:

* Easy handling of missing 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 data can be aligned automatically
* Powerful, flexible group by functionality to perform split-apply-combine operations on data sets
* 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 labelling of axes
* Robust IO tools for loading data from flat files, Excel files, databases, and HDF5
* Time-series functionality: date range generation and frequency conversion, moving window statistics, moving window linear regressions, date shifting and lagging, etc.

**Pandas Data Structures: Series**

A series is a single vector of data (like a NumPy array) with an index that labels each element in the vector. If an index is not specified, a default sequence of integers is assigned as the index. A NumPy array comprises the values of the series, while the index is a pandas Index object.

For example

**import** pandas **as** pd

counts **=** pd.Series([632, 1638, 569, 115])

0 632

1 1638

2 569

3 115

dtype: int64

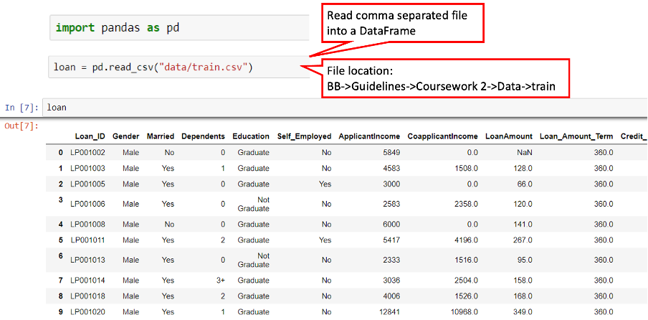
counts.values

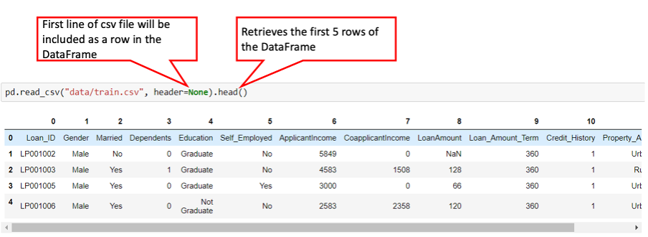
array([632, 1638, 569, 115])

**Pandas Data Structures: DataFrame**

A DataFrame is a tabular data structure, encapsulating multiple series like columns in a spreadsheet. Data are stored internally as a 2-dimensional object, but the DataFrame allows us to represent and manipulate higher-dimensional data.

See, for example the following picture depicting a dataframe extracted from a csv file.





Try the following code

*# import pandas as pd*

**import** pandas **as** pd

input\_users **=** {'Name':['Sarah', 'Lucas', 'Debbie', 'Joanna'],

'Age':[41, 51, 87, 69]}

df **=** pd.DataFrame(input\_users)

**print**(df)

What does it do? Explore the different components and try different examples.

### 1.8 Python Exercises

**Exercise 1**

Go to: Anaconda Website and explore the resources available. Download Anaconda and install it.

**Exercise 2**

Write a program to print the first 50 digits

**Exercise 3**

The Fibonacci numbers are defined as . In other words any number over is given by adding the previous two together Write a program to print the first digits

Have you managed to complete all the exercises?

If not, what was the main challenge(s)?

Discuss the main challenges you have come across

## Week 1 Summary

### Week 1 Summary

**Summary**

This week, the following topics have been discussed

* Introduction to Big Data Analytics
* General introduction to Python
* Python libraries for Data Analytics
* The Anaconda environment and the Jupyter Notebook

Furthermore, some examples have been discussed, specifically addressing general problems that can be found in many Data Analytic scenarios.

### Week 1 Discussion

Think about the following questions and contribute by adding your comments

* What is (and what is not) Big Data analytics?
* What are the main challenges and main limitations that any approach in Big Data analytics will face? -Do you know of any other programming language and/or tool used in Big Data Analytics?

# WEEK 2: The Fundamental Steps for Big Data Analytics

## Introduction

### 2.1 Introduction to Data Analytics Libraries

**Data Analytics and Natural Language Processing**

Data analytics is particularly useful when unstructured data is analysed. There are two types of data, structured and unstructured. The former includes any format that exhibits any highly organised structure, such as numerical data on a spreadsheet. The latter is much more complicated (but more interesting) as there is not a clear structure to use to identify patterns or identify the type of data.

Examples include text, images, sounds, etc. Our brain is highly effective at interpreting, assessing, and extracting usable information from such data. It is something we do with very little effort. However, replicating the same process within a computing environment is extremely hard.

There have been notable advances in image and text recognition, where machines have surpassed humans in correctly identifying specific information. But these are still few and far in between and certainly not the expected outcome. As a consequence, there is a significant effort in developing big data analytics solutions that can address unstructured data.

Natural Language Processing (NLP) is a branch of computer science, which specifically deals with text analysis, whose aim is to facilitate interactions between computers and human language. Its ultimate goal is a computerised understanding of textual data, including the contextual ambiguities of any language. The current-state-of-the-art NLP technology has demonstrated good accuracy in information extraction tasks, which include topic recognition, text summarisation, systematic reviews, sentiment, and social network analysis. A full discussion of NLP goes beyond the scope of this course.

In this context, NLP will be regarded as a set of tools to extract insights from text. In particular, we will analyse some examples to extract specific actionable information that would be otherwise difficult to carry out manually, due to their volume, complexity and diversity.

## Architecture, Analysis and Visualisation Methods

### 2.2 Identification of requirements

Main Requirements

The main requirements in these case studies include the use of some specific Python libraries. More specifically

* NLTK
* Seaborn

[NLTK](https://www.nltk.org/) stands for Natural Language Toolkit and it provides an interface to vast lexical resources, as well as text processing tools for classification, tokenisation, tagging, parsing, etc. Let’s look at what they mean in more details

**Activity**

Open a new Jupyter notebook

Type the following

**import** nltk

sentence **=** "Big Data Analytics is an emergent and significant scientific field, which will drive innovation."

At this stage, all we have is a string which has been stored in the variable sentence. However, we need to specify to the Python interpreter that the string consists of units (or tokens), which are equivalent to words

tokens **=** nltk.word\_tokenize(sentence)

**print**(tokens)

What do you see?

The next step is to attach a lexical tag to each of those words. These tags represent the corresponding lexical properties, for example

* NN: noun, common, singular
* NNP: noun, proper, singular
* NNS: noun, common, plural
* VB: verb, base form, etc.

Run this code

tags **=** nltk.pos\_tag(tokens)

**print**(tags)

What do you see? Search for the different tags and identify what they refer to

What other commands does NLTK have? Spend a few minutes familiarising yourself with the library

**Visualisation via Matplotlib and Seaborn**

Any visualisation approach within Data Analytics, needs to address the following points

* Who is your audience?
* What is the story you want to tell?
* How to present it to optimise the message?

In this course, we will use [matplotlib](https://matplotlib.org/) and [Seaborn](https://seaborn.pydata.org/), which are libraries specifically designed for statistical graphics in Python. Seaborn is based on matplotlib and integrates closely with pandas data structures.

One of the strongest features of Seaborn is that it allows you to explore and better understand your data. It easily plots pandas dataframes and arrays containing whole datasets by automatically perform the necessary pre-processing and statistical aggregation stages to display informative graphs.

Let’s look at some examples. Open a Jupyter notebook and type the following

**import** matplotlib **as** mpl

**import** matplotlib.pyplot **as** plt

**import** numpy **as** np

fig, ax **=** plt.subplots() *# Create a figure containing a single axes.*

ax.plot([1, 2, 3, 4], [**-**1, 2, **-**2, 4]); *# Plot some data on the axes.*

What can you see?

Experiment with other graphs and functions, which are part of matplotlib. Explore scatter plots and try different examples.

Now, let’s consider seaborn. Again, open a Jupyter notebook and type the following

*# Import seaborn*

**import** seaborn **as** sns

*# Apply the default theme*

sns.set\_theme()

*# Load an example dataset*

tips **=** sns.load\_dataset("tips")

*# Let’s plot*

sns.relplot(

data**=**tips,

x**=**"Total Bill", y**=**"Tip", col**=**"time",

hue**=**"smoker", style**=**"smoker", size**=**"size",

)

Understand what the different parameters do.

* What does the relplot() function do?
* What do ‘hue’ and ‘style’ do?
* What is ‘sns.set\_theme()’ ?
* Experiment and try different graphs based on the above dataset

### 2.3 Design of suitable architecture

**Overall Architecture**

In Data Analytics (and computer science in general), an architecture is the organisation of the different components which are used to create a system. They are usually organised sequentially, where each component is likely to contain several other sub-components performing specialised tasks.

In this course, we shall consider a simple architecture, consisting of the following components

* Input
* Data pre-processing
* Model and data analysis
* Visualisation of results and findings

More specifically, the input will comprise the data we need to analyse, which, depending on its size, structure, and any inconsistencies. In fact, data usually contains raw information from sensors, images, texts, etc., which can be corrupted by hardware and software malfunctions, as well as erroneous or missing records due to human error. Data pre-processing can be a lengthy process, and in some cases, it needs to be carried out manually. There are, however, some automated techniques that can speed up the whole process. The model and data analysis stage contains the analytical engine to extract intelligent insights from the data. There are many approaches that can be used, both supervised and unsupervised based on the type of analysis, the available data and the output that needs to be produced. Finally, the results and findings need to be displayed and visualised according to specific constraints, needs and requirements

### 2.4 Evaluate and Discuss the Different Architecture

**Discussion**

Identify suitable resources and critically evaluate further details on

* Input
* Data pre-processing
* Model and data analysis
* Visualisation of results and findings

Discuss and share your findings by identifying pros and cons.

## Data Pre-Processing in Python

### 2.5 Introduction of relevant libraries in Python

**Text Pre-Processing**

Once a text is loaded, the first step to remove any character and provide a uniform input by removing capital letter, digits and any other potential ambiguity. This process is called text normalisation, which includes:

* Converting all letters to lower case
* Changing numbers into words or removing numbers
* Removing punctuations, accent marks, white spaces, etc.
* Expanding (or removing in some cases) abbreviations
* Removing stop words

Try the following code

sentence **=** “Severe weather has been forecast **for** the entire United States.”

new\_sentence **=** sentence.lower()

**print**(new\_sentence)

Explore the *.lower()* function.

Now try the following

sentence **=** “This (sentence) has a lot of punctuation[]!!!”

new\_sentence **=** sentence.translate(string.maketrans(“”,””), string.punctuation)

**print**(new\_sentence)

Another important technique in text pre-processing is tokenisation, which we discussed this earlier on. However, another aspect to consider is remove stop words, such as “the”, “a”, “on”, “is”, “all”. These words tend to occur very frequently, but they are associated with little meaning and so they are usually removed from texts. NLTK allows to do so very easily.

Try the following code

From nltk.tokenize **import** word\_tokenize

input\_sentence **=** “Big Data Analytics **is** a scientific field, which has been expanded over the last decade by several research communities.”

stop\_words **=** set(stopwords.words(‘english’))

tokenized\_words **=** work\_tokenize(input\_sentence)

**for** token **in** tokenized\_words:

**if** token **not** **in** stop\_words:

**print**(token)

### 2.6 Case study via Jupyter Notebook Python

Download the dataset [Tweets.csv](https://github.com/mtrovati/airlines_sentiment/blob/6e367f69d203150b8de78e48e6e5b8567be6a33d/Tweets.csv)

Explore their features via the pandas library

* What are their corresponding sizes?
* What are the main important features? Explain and justify your answer

Discuss and share your findings

## Week 2 Summary

### 2.7 Week 2 Summary

**Summary**

This week we have introduced the main steps to build a Data Analytics solution. More specifically

* Main Data Analytics libraries
* Identification of the solution requirements
* Design and assessment of the appropriate solution
* Preprocessing steps

### 2.8 Week 2 Discussion

Think about the following questions and contribute by adding your comments

* What is the best approach to design a suitable architecture?
* What are the main challenges and main limitations of any preprocessing stage?
* How do you determine the main preprocessing steps?

# WEEK 3: Case Studies

## Architecture design and implementation of study case solution

### 3.1 Analysis of Different Components

Introduction

NLTK and Seaborn are the main components used in the case study presented below. Download the Jupyter Notebook (ADD NAME) and work through it.

Discussion

* How would you use NLTK and Seaborn?
* Can you find other Python libraries that can do a similar job?

Discuss the benefits of NLP and in particular sentiment analysis in your field.

## Airlines Sentiment and Charlottesville Rally: Two Case Studies

### 3.2 Case Studies Introduction

The first case study includes tweets regarding customers’ experience with airlines.

The second case study focuses on the infamous Unite the Right rally, which took place in Charlottesville, Virginia. This event was widely discussed and shared across social media, and in particular Twitter.

In this activity, you will be required to download two datasets containing several pre-processed tweets, which specifically focus on this event, as well as the associated Jupyter notebook to analyse them.

Please download these files by clicking on the links below.

<https://github.com/mtrovati/futurelearn_charlottesville/blob/04815f8646032a20d54c347257f70ec581c09b1d/aug15_sample.csv>

<https://github.com/mtrovati/futurelearn_charlottesville/blob/04815f8646032a20d54c347257f70ec581c09b1d/Charlottesville.ipynb>

<https://github.com/mtrovati/airlines_sentiment/blob/6e367f69d203150b8de78e48e6e5b8567be6a33d/Tweets.csv>

<https://github.com/mtrovati/airlines_sentiment/blob/6e367f69d203150b8de78e48e6e5b8567be6a33d/twitter-us-airline-sentiment.ipynb>

Make sure you follow the steps and use the discussion next to share any issues and difficulties. Solutions will be provided.

### 3.3 The Charlottesville Rally Case Study

Share your findings, difficulties, and questions when attempting the exercise. These types of problems have typically multiple potential solutions, so you are encouraged to identify new ones.

The solutions are provided and can be downloaded using the link at the bottom

<https://github.com/mtrovati/futurelearn_charlottesville_solutions/blob/88c72e71a0d60591f810f7e818b6453e184fce1e/Charlottesville-solutions.ipynb>

## Reflection

### 3.4 Data Science and AI

What is AI?

AI is a field which aims to recreate human intelligence at a computational level. In other words, its motivation is to have systems that can

* Communicate
* Interact with their surroundings
* Help us carry out manual activities, whilst providing unmatched computational capabilities

Ultimately, we want systems that can augment us both physically and intellectually.

Do some independent research on

* Weak and strong AI.
* Which of the above, do you think the activities in this course fall into?
* Do you think Big Data Analytics is part of AI?
* What are the main differences and similarities, if any?

### 3.5 Solutions of the Two Case Studies

I hope you have enjoyed the challenge of the two case studies. You can now download the solution to compare with your own. Remember, this is just one possible solution.

<https://github.com/mtrovati/airlines_sentiment/blob/c0be284bc0597e4391c3bac1f792361ce59c6f4e/twitter-us-airline-sentiment-solution.ipynb>

<https://github.com/mtrovati/futurelearn_charlottesville_solutions/blob/88c72e71a0d60591f810f7e818b6453e184fce1e/Charlottesville-solutions.ipynb>

## Conclusion

### 3.6 Conclusion

Well done! You have completed the Python for Big Data Analytics course! I hope you have enjoyed this course.

During this course you have looked at the main Python components, which are necessary to understand, assess and address real-world Big Data Analytics challenges and problems. You have also looked at some specific implementations, which can provide insightful and actionable information.

Big Data Analytics is continuously evolving due to additional data content created on a daily basis, as well as new techniques and methods, which are the focus of intensive research.

This is a new science that will play an important role in several multidisciplinary fields, where data-driven approaches can drive further innovation.

Check out this related courses: [Data analytics FutureLearn Courses](https://www.futurelearn.com/subjects/business-and-management-courses/data-analytics)

### 3.7 Reflection

Share your thoughts on the points below

* What have you learnt during this course?
* What have you enjoyed?
* Is there something you lack or would like to know more about?

Reflect, comment and discuss within the learning community.

### 3.8 End of Course Quiz

What is preprocessing used for?

* To ensure the data has the correct format and it is free of potential inconsistencies
* To make your life easier!

What is Text Mining?

* It’s the Computer Science discipline to extract information from textual data
* It’s the Computer Science discipline to extract information from visual data

Why do we need Data Visualisation?

* To visualise data and any related analysis, which will enable a better understanding of the different stages and solution
* To produce pretty images.

What is a Data Analytics Architecture?

* It is a Data Analytics application for designing buildings.
* It is the process to identify all the components and mutual relationships of a proposed solution