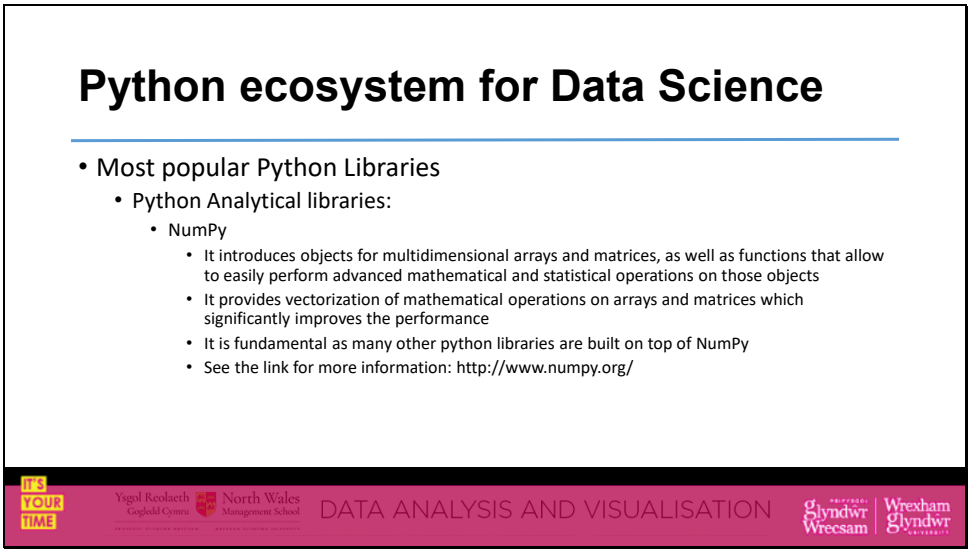
Slide 1

Welcome to lecture 2 , week 6 of data analysis and visualisation.

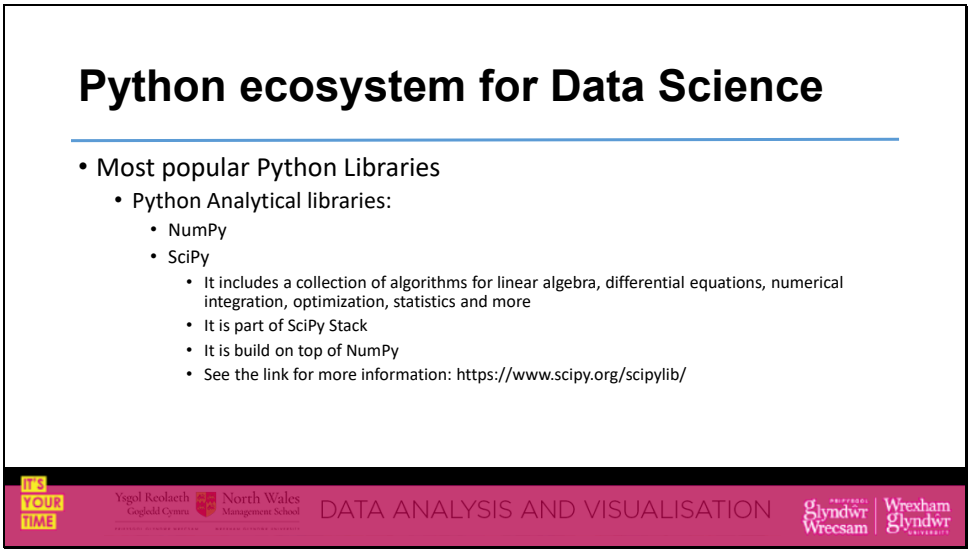
Slide 2



The Python ecosystem of libraries, frameworks, and tools is enormous and growing. Python is used for web scraping, data analysis, web development, internet of things development (IoT), machine learning, DevOps, general scientific computing, and many other computing and scripting uses.

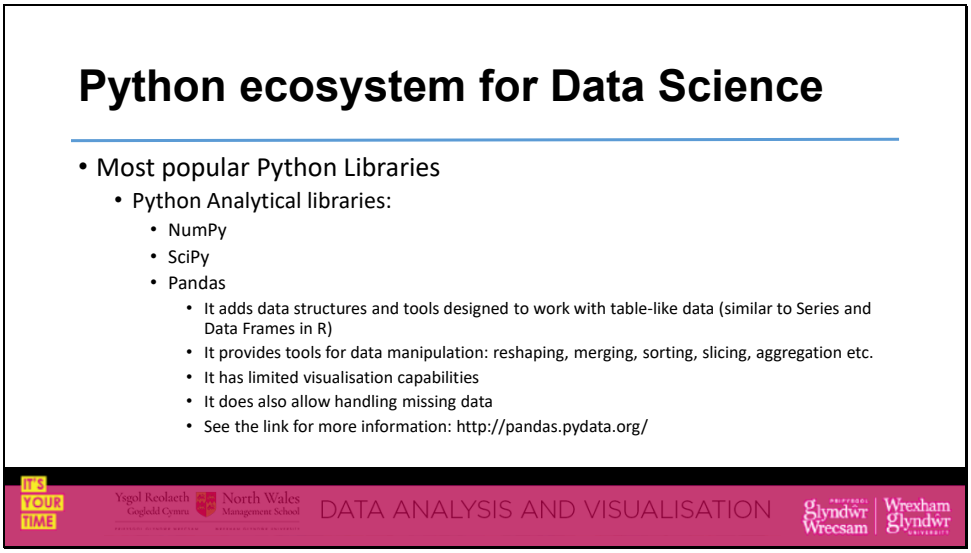
NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

Slide 3



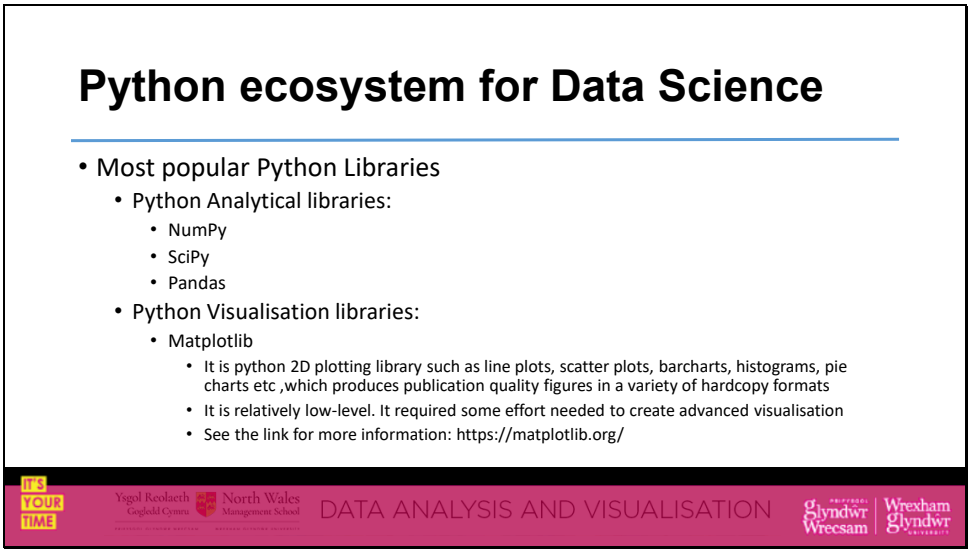
SciPy is a set of open source scientific and numerical tools for Python. It currently supports special functions, integration, ordinary differential equation (ODE) solvers, gradient optimization, parallel programming tools, an expression-to-C++ compiler for fast execution, and others

Slide 4



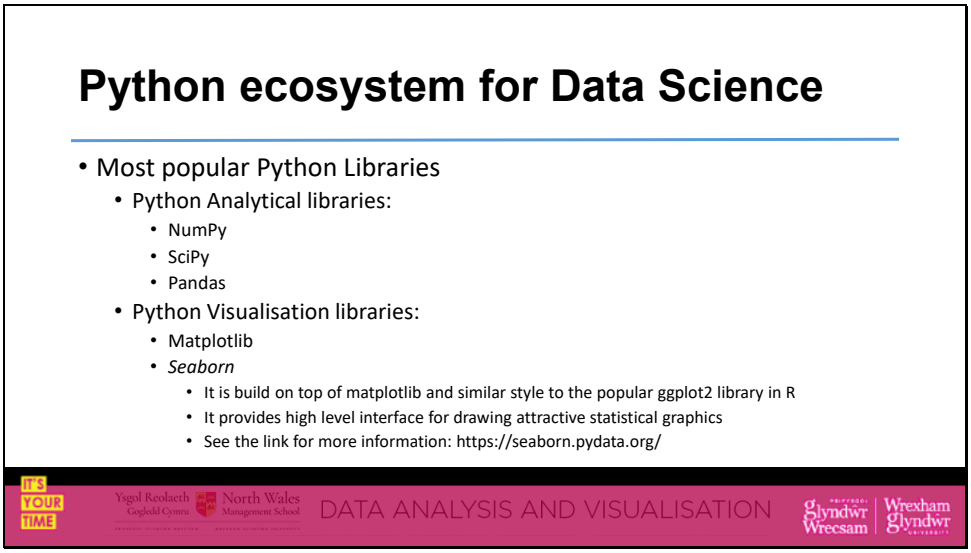
Pandas is another software library written for the Python programming language, it's used for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series.

Slide 5



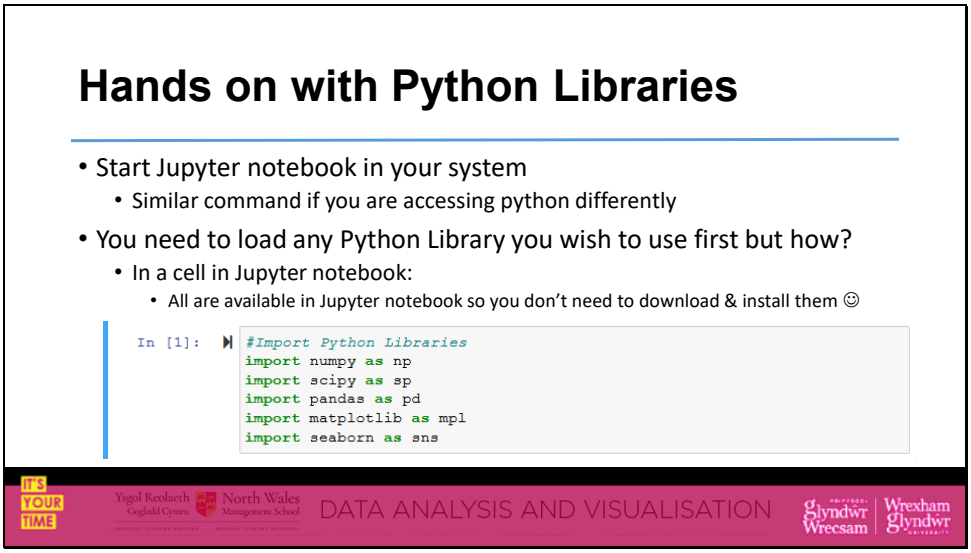
Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits.

Slide 6



Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

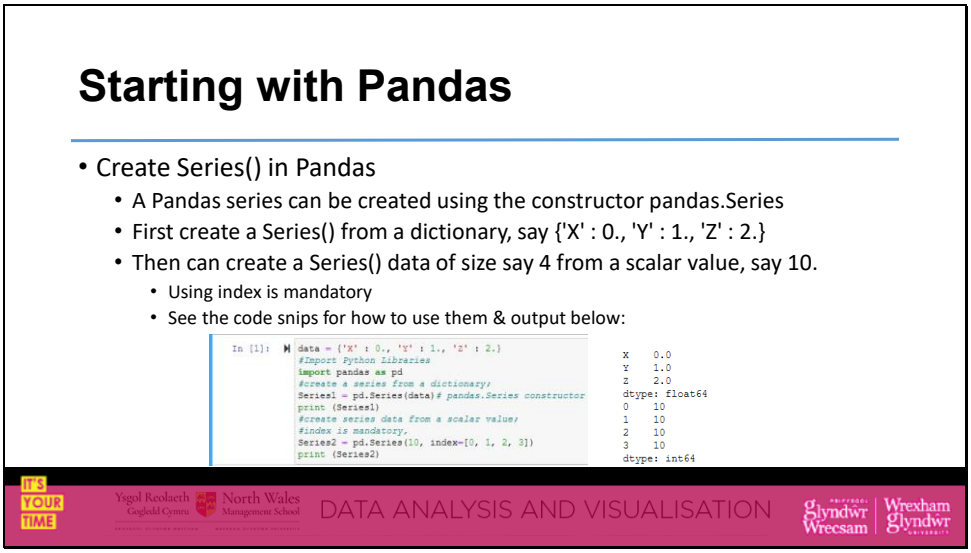
Slide 7



The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

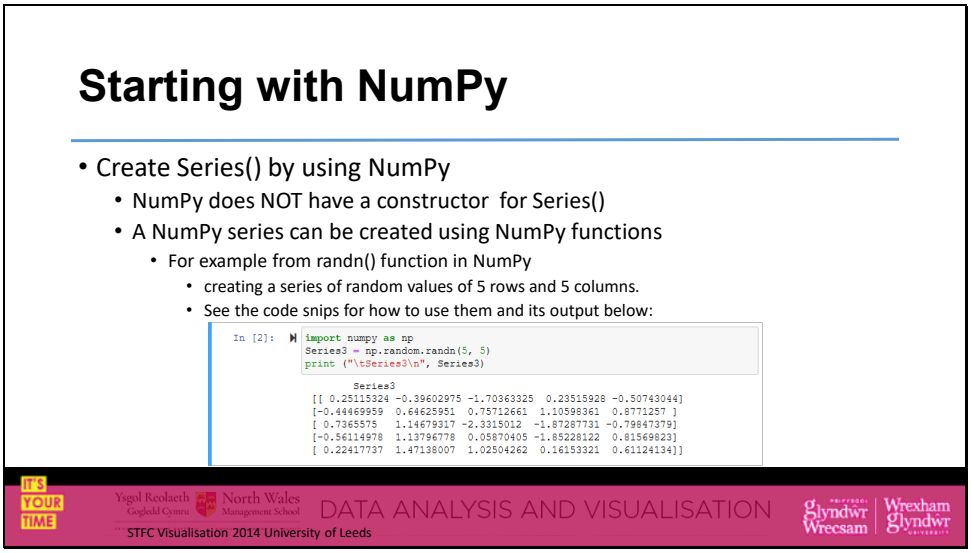
You must import the various libraries you want to use at the start of your code.

Slide 8



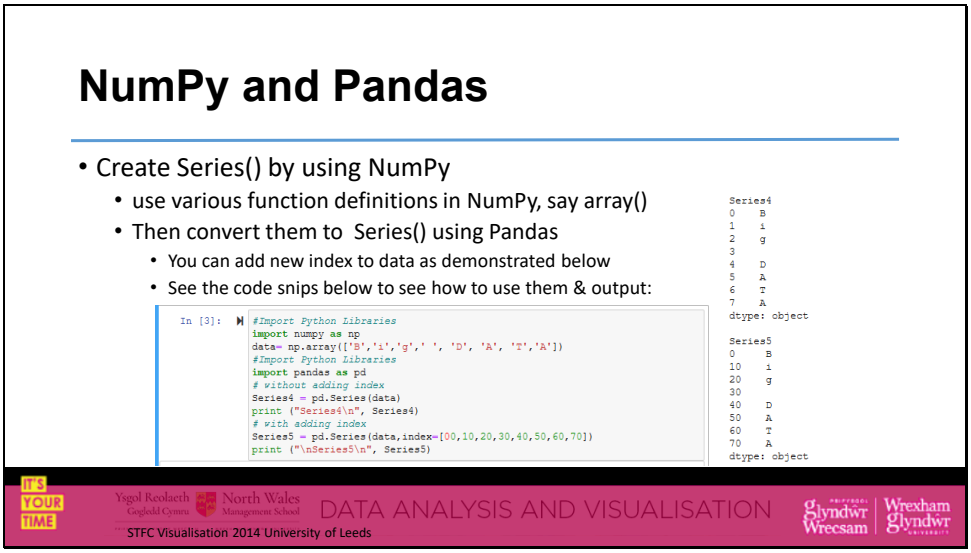
Pandas provides a Series() method that is used to create a series structure.
A series is a sequence of one-dimensional data such as a dictionary, list, array, tuple, and so on.
A serious structure of size n should have an index of length n.
By default Pandas creates indices starting at 0 and ending with n-1.
A Pandas series can be created using the constructor pandas where data could be an array, constant, list, etc.
The series index should be unique and hashable with length n, while dtype is a data type that could be explicitly declared or inferred from the received data.

Slide 9



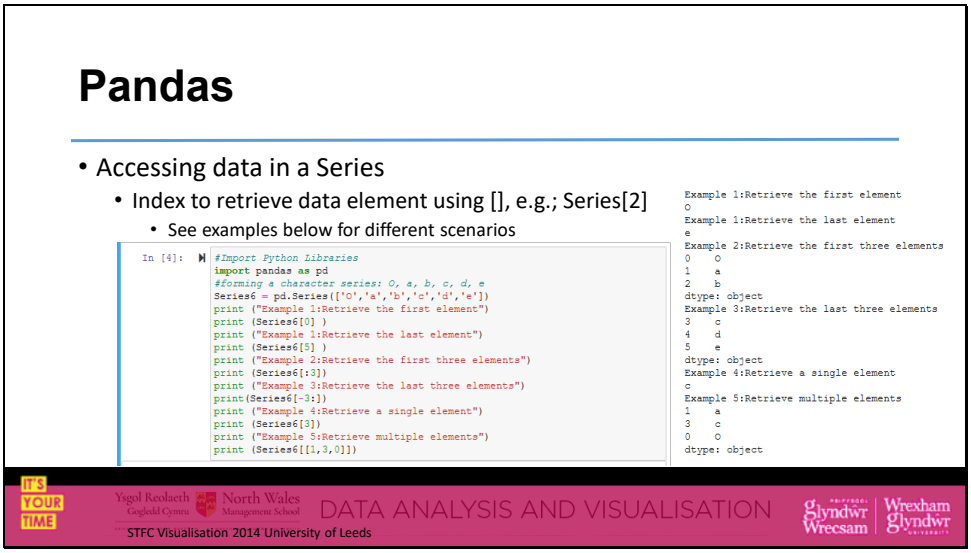
NumPy does NOT have a constructor for Series()
A NumPy series can be created using NumPy functions
For example from randn() function in NumPy creating a series of random values of 5 rows and 5 columns.
See the code snips for how to use them and its output below:

Slide 10



To Create Series() by using NumPy we use various function definitions in NumPy , say array() Then convert them to Series() using Pandas.

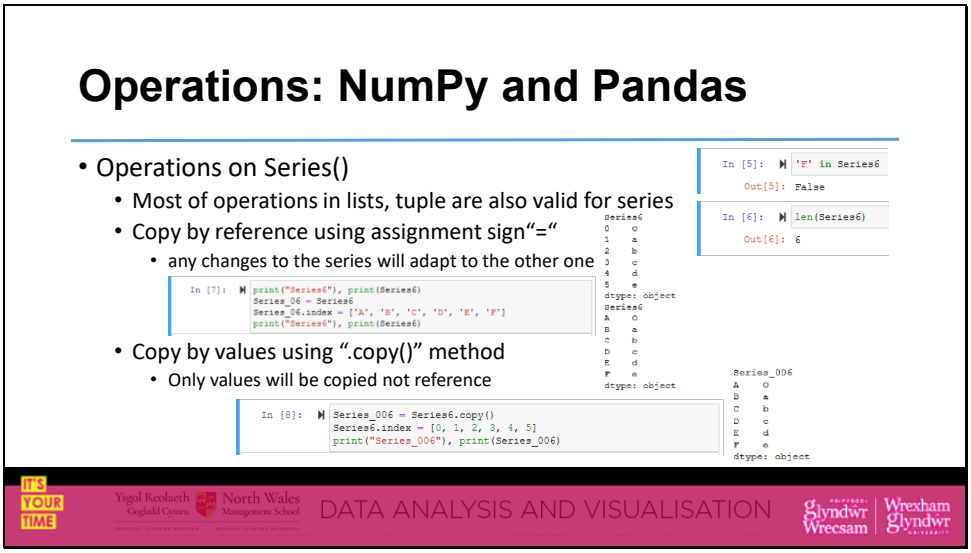
Slide 11



Like lists, you can access a series data via its index value. examples on screen demonstrate different methods of accessing a series of data.

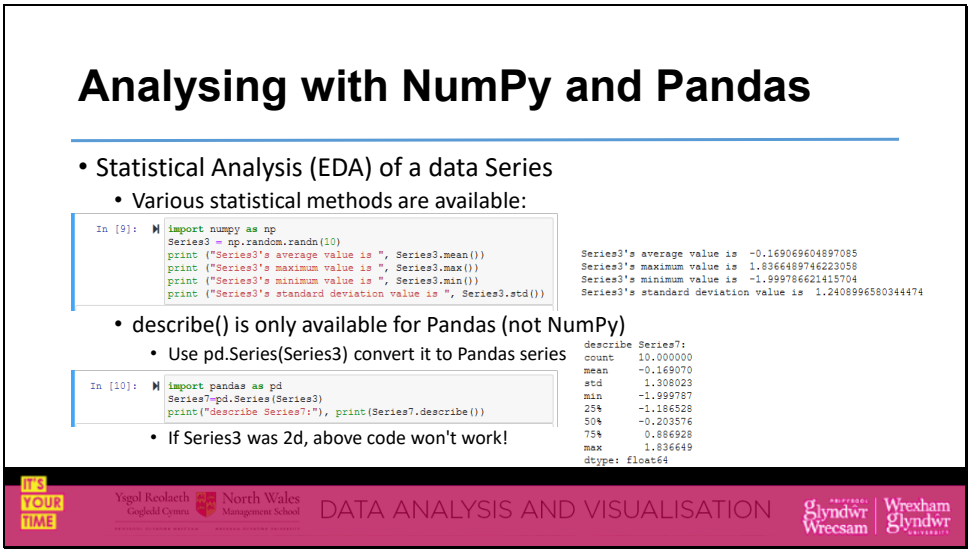
The first example demonstrates retrieving a specific element with index 0.
The second example retrieves indices 0, 1, and 2.
The third example retrieves the last three elements since the starting index is -3 and moves backward to -2, -1.
The fourth and fifth examples retrieve data using the series index labels.

Slide 12



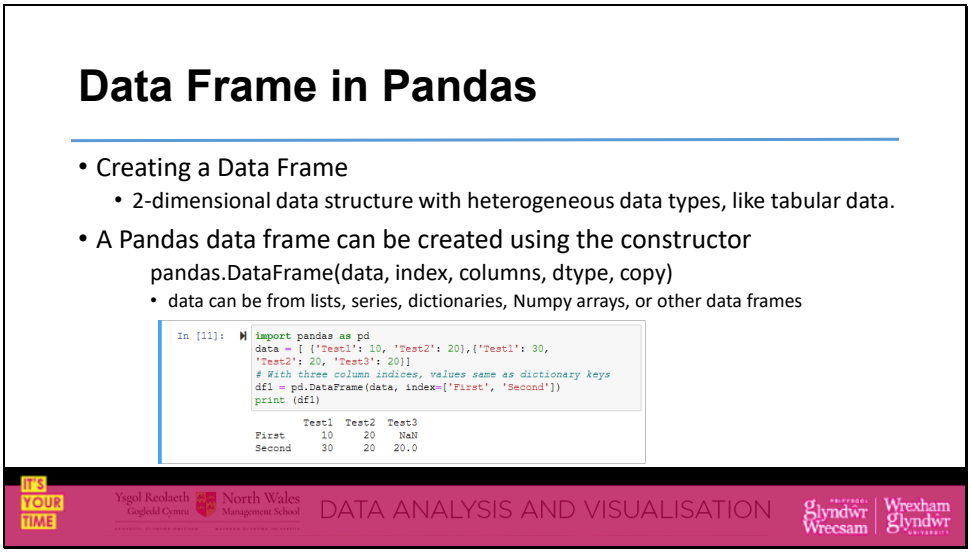
Numerous operations can be implemented on series data. You can check whether an index value is available in a series or not.
Also, you can check all series elements against a specific condition, such as if the series value is less than 8 or not. In addition, you can perform math operations on series data directly or via a defined function.

Slide 13



Numerous statistical methods can be applied directly on a data series. On screen we demonstrate the calculation of mean, max, min, and standard
deviation of a data series.
Also, the .describe() method can be used to give a data description, including quantiles.

Slide 14

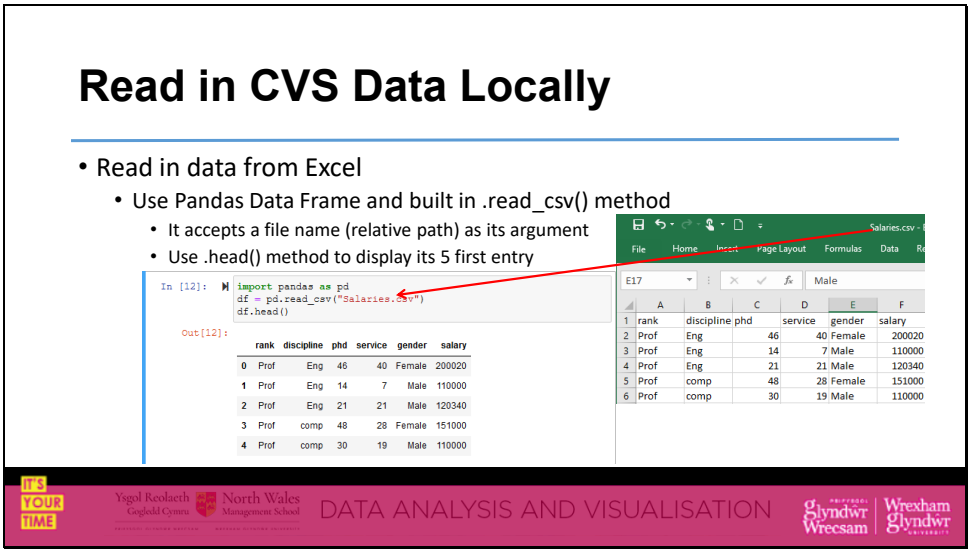


Data Frame it is a two-dimensional data structure with heterogeneous data types, i.e., tabular data.

Pandas can create a data frame using the constructor pandas. DataFrame(data, index, columns, dtype, copy).
A data frame can be created from lists, series, dictionaries, Numpy arrays, or other data frames.
A Pandas data frame not only helps to store tabular data but also performs arithmetic operations on rows and columns of the data frame.

You can create a data frame from dictionaries or arrays, as shown in code snip.
Also, you can set the data frame indices.
However, if you don't set the indices, then the data frame starts with 0 and goes up to n-1, where n is the length of the list.
Column names are taken by default from the dictionary keys.
However, it's possible to set labels for columns as well.

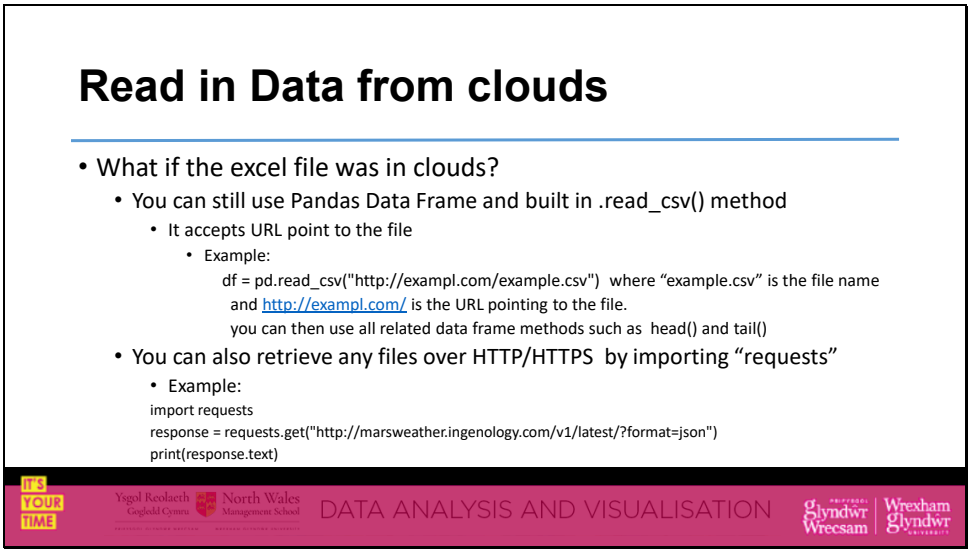
Slide 15



You can read in data from Excel
Use Pandas Data Frame and the built in .read_csv() method, It accepts a file name with relative path as its argument Use .head() method to display its 5 first entry

You can also use print to see both head and tail of CVS file.
In this code snip the file is located in the same directory, however, it can exist anywhere in your computer and you can use relative or absolute path to load it.

Slide 16



What if the excel file was in clouds?

You can still use Pandas Data Frame and built in .read_csv() method

It accepts URL point to the file

Example:

df = pd.read_csv("http://exampl.com/example.csv")

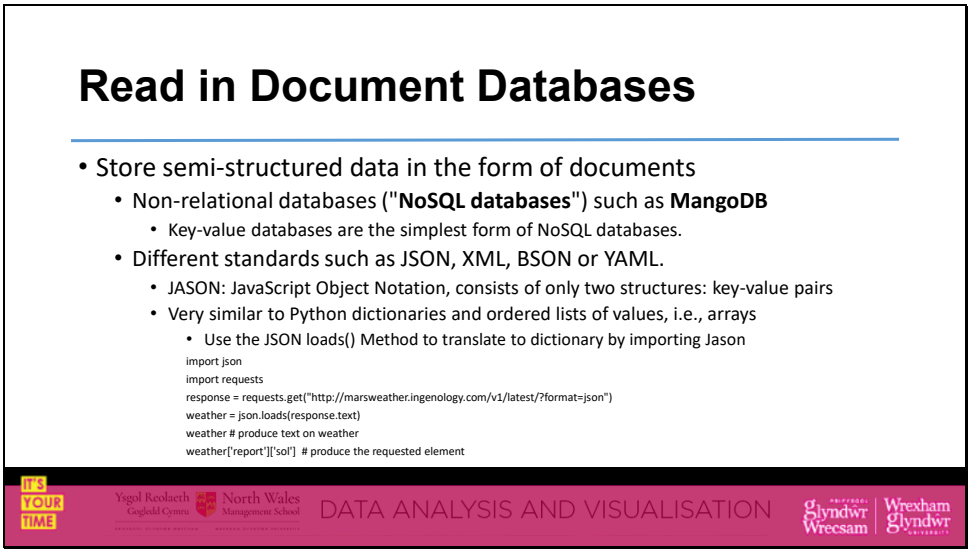
where “example.csv” is the name of the file

and http://exampl.com/ is the URL pointing to the file.

you can then use all related data frame methods such as head() and tail()

You can also retrieve any files over HTTP/HTTPS by importing “requests”

Slide 17

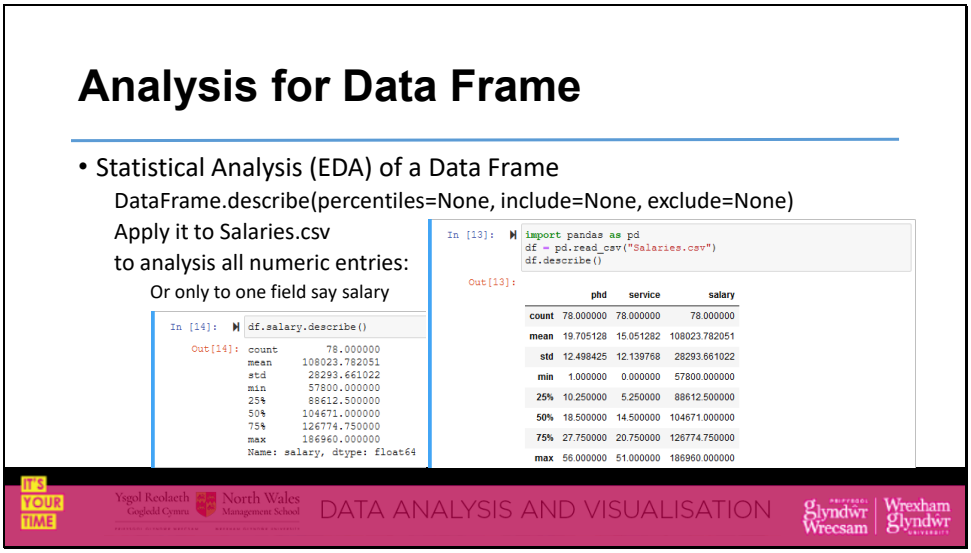


JSON, which stands for JavaScript Object Notation, dates to 1999. It consists of only two structures: key-value pairs, called *structures*, that are very similar to Python dictionaries; and ordered lists of values, called *arrays*, that are very much like Python lists.

JSON is so common that most languages have features to translate JSON to and from native data types. In the case of Python, that feature is the json module, which became part of the standard library with version 2.6.

When you know the URL to use, you can use the requests library to fetch data from an API and either process it on the fly or save it to a file for later processing. The simplest way to do this is exactly like retrieving a file: as already discussed.

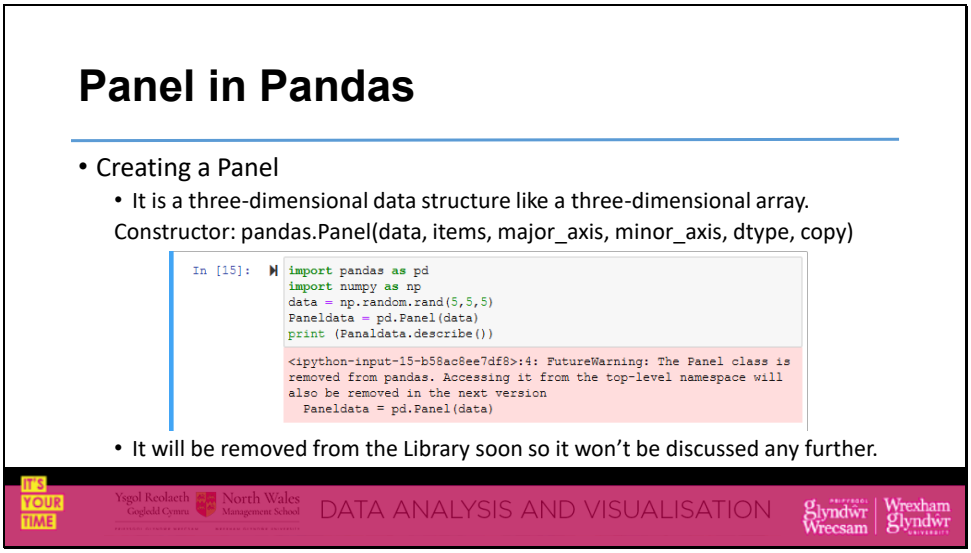
Slide 18



Pandas provides various methods for analyzing data in a data frame. The .describe() method is used to generate descriptive statistics that summarize the central tendency, dispersion, and shape of a data set's distribution.

DataFrame.describe() analyzes both numeric and object series, as well as data frame column sets of mixed data types. The output will vary depending on what is provided.

Slide 19



A *panel* is a three-dimensional data structure like a three-dimensional array.

Pandas creates a panel using the constructor pandas.Panel(data, items, major_axis, minor_axis, dtype, copy).

The panel can be created from a dictionary of data frames and narrays.
The data can take various forms, such as ndarray, series, map, lists, dictionaries, constants, and also another data frames.
It will be removed from the Library soon, so it won't be discussed any further.