Python for data science

Libraries, Web Scraping and Basic Operations

Before we start:

If you missed last week then you probably won’t have python set up on your system.

If this is the case then please let us know at the start of the labs, we can then try and get you up set up

Libraries, modules and packages

A package contains all the files you need for a module.

Modules are Python code libraries you can include in your project.

From now we will refer to them as libraries…

What are Python libraries?

Normally, a library is a collection of books or is a room or place where many books are stored to be used later. Similarly, in the programming world, a library is a collection of precompiled codes that can be used later on in a program for some specific well-defined operations.

It contains bundles of code that can be used repeatedly in different programs. It makes Python Programming simpler and convenient for the programmer. As we don't need to write the same code again and again for different programs. Python libraries play a very vital role in fields of Machine Learning, Data Science, Data Visualization, etc.

Commonly used Python libraries

Numpy: The name “Numpy” stands for “Numerical Python”. It is the commonly used library. It is a popular machine learning library that supports large matrices and multi-dimensional data. It consists of in-built mathematical functions for easy computations.

Pandas: Pandas are an important library for data scientists. It is an open-source machine learning library that provides flexible high-level data structures and a variety of analysis tools. It eases data analysis, data manipulation, and cleaning of data.

Urllib: Urllib package is the URL handling module for python. It is used to fetch URLs (Uniform Resource Locators). It uses the *urlopen* function and is able to fetch URLs using a variety of different protocols.

Beautiful Soup: will identify the needed HTML string and the relevant information within it. Based on predefined criteria and the rules of the parser, it’ll filter and combine the needed information into CSV, JSON, or any other format.

A quick note on installation

- Many of the libraries you'll want to use and already included with your python install
- For other libraries we can use the 'pip' method

In the command line it would look like this:

C:\Users\Your Name\AppData\Local\Programs\Python\Python36-32\Scripts>pip install numpy

In a jupyter notebook just:

```
pip install numpy
```

Using a requirements.txt file

requirements.txt is a file that contains a list of packages or libraries needed to work on a project that can all be installed with the file. It provides a consistent environment and makes collaboration easier.

You should have imported all the required libraries with this file when you set up your environment. If not let us know!

Basic library use

Using Numpy and Pandas as an examples…

NumPy is a Python library.

NumPy is short for "Numerical Python".

NumPy is used for working with arrays.

Numpy arrays are much faster than normal python value lists

Datatypes in Numpy

Standard Python

- `string` - used to represent text data, the text is given under quote marks. e.g. "ABCD"
- `integer` - used to represent integer numbers. e.g. -1, -2, -3
- `float` - used to represent real numbers. e.g. 1.2, 42.42
- `boolean` - used to represent True or False.

Numpy

- `i` - integer
- `b` - boolean
- `u` - unsigned integer
- `f` - float
- `c` - complex float
- `m` - timedelta
- `M` - datetime
- `O` - object
- `S` - string
- `U` - unicode string
- `V` - fixed chunk of memory for other type (void)

How to use / import a library

```
import numpy as np
// This imports the library into your environment


arr = np.array([1, 2, 3, 4, 5])
// we then create the numpy array “arr”, and fill it with 5 values


print(arr)
// we then print this array
```

Example: Using an array index

```
import numpy as np

arr = np.array([1, 2, 3, 4])

print(arr[0])
```

This code will print the first value from the array. Remember we count array indexes from 0 !

Example: Slicing an array

```
import numpy as np

arr = np.array([1, 2, 3, 4, 5, 6, 7])

print(arr[4:])
```

Here we use : to slice the array. So we are only printing values from the 4th index (which here is 5) to the end of the array. If we put the : before the index it will slice to the first value

Example: Joining Arrays with numpy

```
import numpy as np

arr1 = np.array([1, 2, 3])

arr2 = np.array([4, 5, 6])

arr = np.concatenate((arr1, arr2))

print(arr)
```

Here we call the concatenate() function from the numpy library. This joins arr1 and arr2 into a new array called arr. We then print this. Remember [ ] brackets for array values!!!

Converting Datatypes with Numpy

```
import numpy as np

arr = np.array([1.1, 2.1, 3.1])

newarr = arr.astype('i')

print(newarr)

print(newarr.dtype)
```

This code will convert an array of floats (numbers with decimal places) to an array of integers (whole numbers)

Searching arrays

You can search an array for a certain value, and return the indexes that get a match.

```
arr = np.array([1, 2, 3, 4, 5, 4, 4])

x = np.where(arr == 4)

print(x)
```

This will return a tuple: `(array([3, 5, 6],)`. Which means that the value 4 is present at index 3, 5, and 6.

PANDAS use with an example dataset

What is Pandas?

- Pandas is a Python library used for working with data sets.

- It has functions for analyzing, cleaning, exploring, and manipulating data.

- Pandas allows us to analyze big data and make conclusions based on statistical theories.

- Pandas can clean messy data sets, and make them readable and relevant.

Dataframes

A 2 dimensional data structure, like a 2 dimensional array, or a table with rows and columns.

```
data = {
  "calories": [420, 380, 390],
  "duration": [50, 40, 45]
}

df = pd.DataFrame(data, index = ["day1", "day2", "day3"])

print(df)
```

We can isolate the data of a specific index with this code, which is useful in large data sets:

```
print(df.loc["day2"])
```

Data Correlations:

We can find relationships between columns of data using the `corr()` method.

The `corr()` method calculates the relationship between each column in your data set.

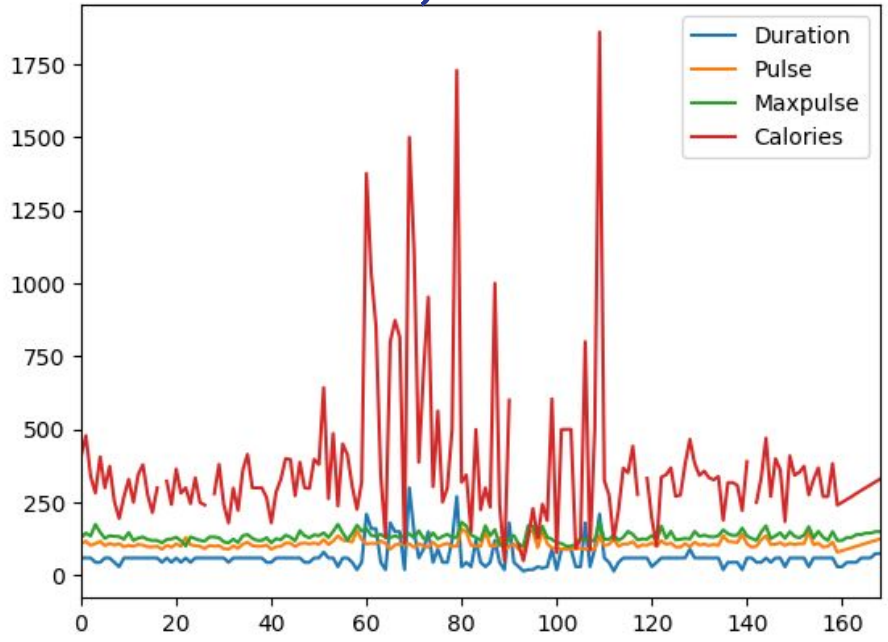
The Result of the `corr()` method is a table with a lot of numbers that represents how well the relationship is between two columns.

The number varies from -1 to 1.

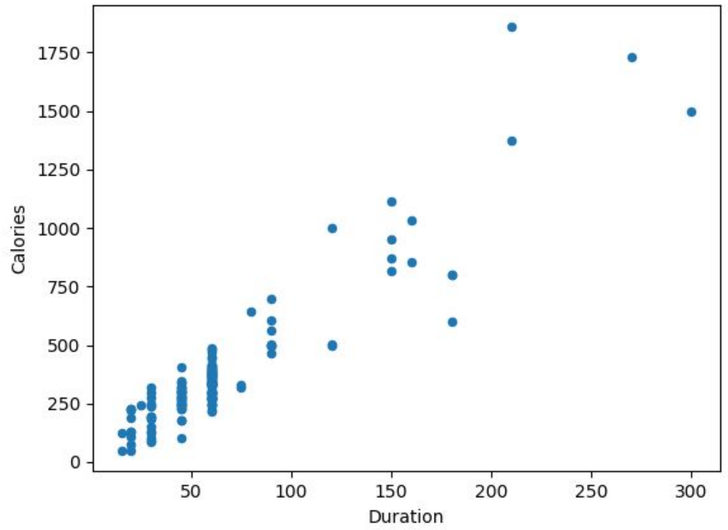
We will look at this in the labs

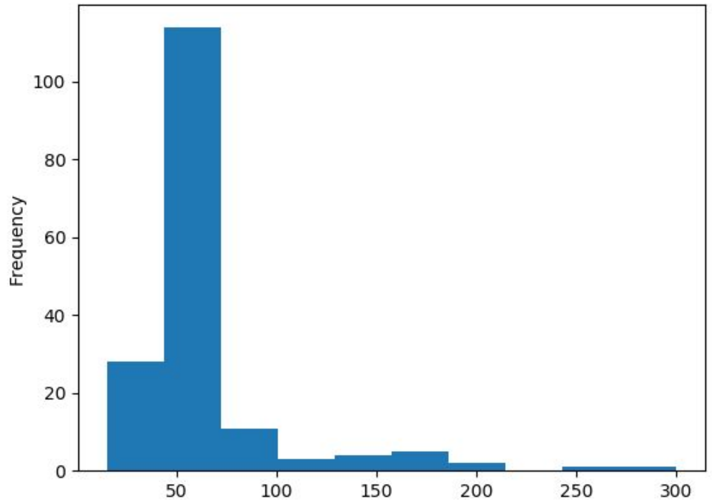
Plotting data (more on this next week!)

```
import pandas as pd

import matplotlib.pyplot as plt


df = pd.read_csv('data.csv')


df.plot()


plt.show()
```



X axis is index number, Y axis is value. We can see here longer exercise means more calories burned





There are different types of graphs we can plot, and we can change which columns to plot them with, it depends what you want to know.

We will look more at this next week!

Web scraping and crawling

Some of the following slides are credited to Terence Broad and Rebecca Fiebrink @ Ual - CCI

Introduction to web scraping

- Web scraping is where we write code to automatically parse and collect data from the internet — usually from websites on the **world wide web**

- We can quickly build very large datasets of text (or other things) using web scraping and web crawling

- We can use write software to **search for** and **extract**

- specific things from specific websites

Uses of web scraping

- Web scraping is the process of collecting unstructured and structured data in an automated manner. Then working with that data to find patterns, correlations, trends ect.

- Some of the main use cases of web scraping include price monitoring, price intelligence, news monitoring and market research among many others.

- In general, it is used by people and businesses who want to make use of publicly available web data to generate valuable insights and make smarter decisions.

Web crawling

- Web crawling is the practice of building software that automatically searches for webpages by navigating through hyperlinks — to scrape data from a whole website (or lots of websites)
- A web crawler can collect vast amounts of data by exhaustively clicking on every hyperlink on a website
- You can set rules for what links to click through and what subdomains you want to retrieve data from

Scraping VS Crawling

- **Web scraping**, also known as web data extraction, is similar to web crawling in that it identifies and locates the target data from web pages. The key difference is that with web scraping, we know the exact data set identifier, e.g., an heading, image or table.

- **Web crawling**, is used to index the information on the page using bots, also known as crawlers. Crawling is essentially what search engines do. It's all about viewing a page as a whole and indexing it. When a bot crawls a website, it goes through every page and every link, until the last line of the website, looking for ANY information.

Technical challenges with web scraping/crawling

- Many websites try to stop web crawling, either by:

  - Not putting content in HTML, or making the format difficult to parse
  - Blocking IP (internet protocol) address that make lots of requests very quickly
  - Adding captchas or other interactive elements to prevent software scripts accessing the page
  - Putting data behind a paywall
  - Regularly changing the way their websites are organised to make previous web scraper obsolete

Ethical issues with web scraping/crawling

- There are a lot of ethical issues to be aware of when web scraping:

  - It is often against a website or platforms **terms of service** to scrape all of their data
  - The dataset may be copyrighted by the website or other party.
  - The data may contain private or personally identifying information. (especially on social media)
  - The data may contain offensive or malicious content
  - The data is full of factually inaccurate content or material

Meta, ethics and web scraping

Last year Meta Platforms sued web scraping firm Bright Data for scraping data from its Facebook and Instagram websites – even though Meta paid Bright Data to scrape data from other websites. They claimed Bright data were selling users information, while Meta claim they were web scraping to find scams.

(https://www.theregister.com/2023/02/02/meta_web_scraping/)

How are websites structured?

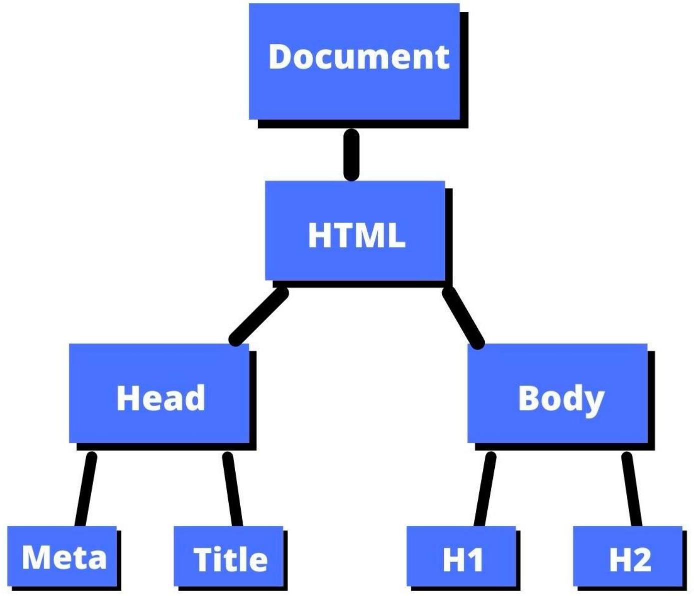
- A website is structured through a hierarchical organisation, with a homepage followed by various interconnected web-pages
- A webpage is the equivalent of a single document
- A website has a domain name (e.g wikipedia.org) and pages all have separate URLs that stem from the domain name (e.g. en.wikipedia.org/wiki/Camberwell)
- We can connect web pages together with hyperlinks

How are web pages structured?

- Web pages are made primarily using three different scripting or markup languages:
- **HTML:** Hypertext Markup Language — Where the content of the web page is stored
- **CSS:** Cascading Stylesheets — Where the instructions on the layout and visual characteristics are stored
- **JS:** Javascript — Code that allows for interactive elements (and other things) on a web page

The document object model (DOM)

- Is how a web page gets represented by the computer
- The DOM is built from the content in the HTML (or XML) file
- It gives a hierarchy and organisation to how the elements in the web page are structured
- Using the DOM specific elements can be manipulated using software scripts
- The DOM is always a 'tree' structure (a network that only branches)



HTML overview

- When we are performing web scraping we are mostly only interested in parsing HTML — as this is where text data is usually stored

- The HTML defines things like:

  The document title

  The headings

  Paragraph

  text Lists

  Images

  Hyperlinks

  Other groupings or id’s that we want to give to elements in the document

HTML basics recap

- In HTML, elements are enclosed in tags
- The opening tag looks like:

  <tag> The closing tag looks

  like: </tag>
- We can put whatever we want between tags (but we may get errors if we do it wrong!)

```
<!DOCTYPE
<html>
  <head>
  <title>This   is  a   title</title>
  </head>
  <body>
    <div>
        <p>Hello  world!</p>
    </div>
  </body>
</html>
```

```
<html     class="client-nojs     vector-feature-language-in-header-enabled    vector-feature-language-in-main-page-header-disabled    vector-feature-sticky-header-disable
vector-feature-page-tools-pinne
<head>
<meta charset="UTF-8">
<tit1e>HTML - Wikipedia</tit1e>
<script>(function(){var className="client-js vector-feature-language-in-header-enabled vector-feature-language-in-main-page-header-disabled vector-feature-sticky-header-di
vector-feat
"wgDefaultDateFormat"."dmy","wgMonthNames".["","January","February","March","April","May","June","July","August","September","October","November","December"],"wgRequestId"
61fe-4b99-4  "Articles  with  specifically  marked  weasel-worded  phrases  from  Harch  2017","All  articles  with  vague  or  ambiguous  time","Vague  or  ambiguous  time  from
2022","Articles                              to                              be                              expanded                              fro
"wgRelevantPageName"."HTML","wgRelevantArticleId".13191,"wgIsProbablyEditable".false,"wgRelevantPageIsProbablyEditable".false,"wgRestrictionEdit":["autoconfirmed"],"wgRest
nMove".(],"
"GEHomepageSuggestedEditsEnableTopics":true,"wgGETopicsMatchModeEnabled":false,"wgGEStructuredTaskRejectionReasonTextInputEnabled":false,"wgGELevelingUpEnabledForUser":fal
STATE=("ski
"ext.centralNotice.startUp","ext.gadget.ReferenceTooltips","ext.gadget.geonotice","ext.gadget.switcher","ext.urlShortener.toolbar","ext.centralauth.centralautologin","mmv.
"mmv.bootst
<script>(RLQ    indow.  Qlltl).push(function()(mw.loader.impl(function()(return[ "user.options#12s5i",
€unction($,jQuery,require,module){mw.user.tokens.set({"patrolToken":"+\\","watchToken":"+\
}];});});</scripts
<link rel="stylesheet"
href="/w/load.php?lang=en&amg;modules=codex-search-styles%7Cext.cite.styles%7Cextpygments%2CwikimediaBadges%7Cext.u1s.inter1anguage%7Cext.visualEditor.desktooArticle
<script async="" src="/w/load.php7lang=en&amp;modules=startup&amp;only=Ocripts&amp;raw=1&amp;skin=vector-2022"></scripts
<meta name="ResourceLoaderDynamicStyles"content="">
<link rel="stylesheet"  href="/w/load.php?lang=en&amp;moduleQ=ext.gadget.SubtleUpdatemarker%2CWatchlistGreenIndicators&amp;only=styles&amp;Okin=vector-2022">
<link rel="stylesheet" href="/w/load.php?lang=en&amg;modules=site.styles&amg;only=styles&amg;skin=vector-2022">
<meta name="generator" content="MediaWiki1.42.0-wmf.1">
<meta name="referrer" content="origin">
<meta name="reIerrer" content="origin-when-cross-origin">
<meta name="robots" content="max-image-preview:standard">
<meta name="format-detection" content="telephone=no">
<meta property="og:image" content="https.//upload.wikimedia.org/wikipedia/commons/thumb/6/61/HTML5_logo_and_wordmark.svg/1200px-HTML5_logo_and_wordmark.svg.png">
<meta property="og:image:width"content="1200">
<meta property="og:image:height" content="1200">
<meta property="og.image" content="https.//upload.wikimedia.org/wikipedia/commons/thumb/6/61/HTML5_logo_and_wordmark.svg/800px-HTML5_logo_and_wordmark.svg.png">
<meta property="og:image:width" content="800">
<meta property="og.image.height"content="800">
<meta property="og:image" content="https://upload.wikimedia.org/wikipedia/commons/thumb/6/61/HTML5_logo_and_wordmark.svg/640px-HTNL5_logo_and_wordmark.svg.png">
<meta property="og.image.width"content="640">
<meta property="og:image:height" content="640">
<meta name="viewport" content="width=1000"°
<meta property="og:title" content="HTML - Wikipedia">
<meta property="og:type" content="website">
<link rel="preconnect" href="//upload.wikimedia.org">
<link rel="alternate" media="only screen and (max-width: 720px)" href="//en.m.wikipedia.org/wiki/HTML">
<link rel="apple-touch-icon" href="/static/apple to h i ipedia.png">
<link rel="icon" href="/static/favicon/wikipedia.ico">
```

Parsing:

Parsing is taking a set of data and extracting the meaningful information from it. With HTML parsing, you're looking to read some html and return a structured set of tags and text

Parsing HTML with Beautiful Soup

- Luckily we don’t have to parse HTML ourselves, the Python library beautifulsoup can do it for us!
- We just need to run:

```
response = requests.get( ‘https://en.wikipedia.org' )

soup = BeautifulSoup(response.text,  ‘html.parser')
```

And we will get a Python object that we can search and extract the information we want out of it

How to find the content you want to scrape?

`<div>` tags are used to divide the content up into

sections Tags can have id’s, e.g:

```
<div   id=‘section-1’>
some   stuff
</div>
```

We can use the `<p>` tag to look for text

We can use the `<h>` tag to look for headings

We can use the `<a>` tag to look for links

Manually going through a HTML file to find the tags you want can be difficult — especially since most HTML files are generated by computer software, not written by people!

Robots.txt files

robots.txt is the filename used for implementing the **Robots Exclusion Protocol**, a standard used by websites to indicate to visiting web crawlers and other web robots which portions of the website they are allowed to visit.

The standard was used in the 1990s to mitigate server overload. In the 2020s many websites began denying bots that collect information for generative artificial intelligence.

https://en.wikipedia.org/robots.txt

Cleaning and sorting data after scraping/crawling

- Generally data collected from a web crawling will be messy and disorganised
- You will normally need to:
  - Remove data from unwanted pages
  - Remove duplicates
  - Remove junk data
  - Extract and reformat data from HTML into a structured format (.txt, .csv, or other data storage type)

Data Cleaning

Data cleaning means fixing bad data in your data set.

Bad data could be:

- Empty cells (eg. all data missing)
- Data in wrong format (eg. string instead of int)
- Wrong data (eg. extra 0 on a result)
- Duplicates (eg. exact same data in two different rows)

Reading a .cvs file with Pandas

A simple way to store big data sets is to use CSV files (comma separated files). CSV files contains plain text format

```
df = pd.read_csv('data.csv')


print(df.to_string())
```

Here ‘data.cvs’ is an example file. You will need to replace this with the correct filename for your chosen data