**Kazil, J & Jarmul, K. (2016) Data Wrangling with Python. O'Reilly. Media Inc.**

**Chapter 6 – Acquiring and Storing Data**

“Not all data is created equal” – some tips for checking your data, answer yes to three and it’s probably ok;

* Is the author a source I can contact if I have questions?
* Is the data updated and checked for errors?
* Is there information on how the data was acquired and how?
* Any other sources it can be verified against?
* Does the data seem plausible?

Factcheck your data to look further into the above is key, source methods and updates, comparative sources, speaking to an expert; and further research – are all helpful

Once fact checked, it is easier to script and determine validity in the future.

**Readability, Cleanliness and Longevity**

Illegible data can be cleaned and read by Python. Data may already have been cleaned if from a reputable source. Can aim to understand that process and how frequently it is updated, what is the longevity. Is it regularly collected and updated.

If data comes from standardized and rigorous sources, you are likely to be able to script the cleaning process and it will be unlikely to change for years. These means in the future being able to skip the cleaning and go straight to analysis.

**Where to find data**

If you see data in word, PDF or excel. Then a human has been involved, speak to that person and you may be able to get the source data. Raw data in CSV or from a database is easier to parse. You can also ask them questions about data collection methods, source, timeliness and abbreviations or definitions.

Government agencies and other organisations (NGOs, Universities etc) have open source data available.

Crowdsourced data and APIs (application programming interfaces) can directly take data from web services. Pro’s include access to data, large quantities and can just take directly from service so no need to store. Cons include, unreliability of mass API and reliance on access being allowed and potential data overload.

***Amount of data available is enormous. Sorting through sources and deciding what to use can be challenging. Get an idea of questions you want to answer, approaches to doing that and then understanding the data available.***

***Case Studies: Example Data Investigations***

**Ebola Crisis**

Example flow of questioning and investigating if looking into Ebola in West Africa:

* Google Ebola, find many international organisations tracking and providing access to tools.
* Find WHO site with latest stats, interactive maps, KPIs all updated weekly. They provide data in CSV and JSON – reliable source verifiable and updated.
* Find Github repositories with raw data from many sources and a contract to speak to about data updating and collection formats. Data in formats that can be parsed.
* Decide on a question around burials being safe and find a report with a named person to try and speak to.

Results: Good sources, from verifiable organisations and someone to speak to about the research

**Train Safety**

Looking at negative factors affecting train safety in US:

* Look at previous research, find Federal Railroad Admin (DRA), reports say most due to poor track conditions or human error.
* Looking at human side, identify that sleep and issues relating to drugs and alcohol could contribute.
* Narrow questions down to looking at these factors, you have some initial datasets from research and contacts to ask for more.

**Football Salaries**

Looking at EPL salaries

* Find a series of website with player salaries and endorsements listed, contact the authors to find out more and if data is updated.
* Find a range of playing stats on he official EPL website, can obtain these by web scraping.
* So can start to compare the player salary data vs the playing stats data.

**Child Labour**

Focus on international crisis in child labour.

* Find UNICEF’s open data site dedicated to child labour.
* This has entire datasets on women’s wellbeing and child wellbeing / status
* Allows us to consider question: “Does early marriage affect child labour rates?”
* Government data from the US on rates across the world allows us to cross reference vs UNICEF data.
* Also find ILO trend report on child labour. This has links to many different historic dataset.

So obtained many datasets from reputable sources that can be compared and cross referenced and considered how to refine questions and find sources to answer them.

**Chapter 7 – Data Cleanup: Investigation, Matching and Formatting.**

Data cleaning is essential, it requires precision and good subject and data knowledge. Python can save hours with a good script replacing repetitive problems.

**Why?**

Most data is not well formatted or ready to use. Even cleaned data usually has inconsistencies or readability issues. If from more than one dataset will likely not join so would need formatting and standardizing.

Also, easier to sore, search and re-use if cleaned. Standardized fields (such as phone numbers / email) – remove or clean records that don’t fit then more consistency when you use later.

Want to document steps of data cleaning so can be reproduced and robustly defending data cleaning decisions when used in analysis.

**Data cleanup exercise:**

The activity works through replacing coded / short format text headers with human readable ones from a second CSV file. Using DictReader library in python then a list generator function. It then repeats the process with the zip function, addressing mismatched data.

The process matches most of the data but but two headers remain unmatched – this demonstrates a decision point in data cleaning – are these headers or data needed – If so - what can we do to clean or match them? Otherwise, we could drop if they are not essential for the analysis. Is the effort worth the output, can you make cleaning reputably to save time in the future.

**Formatting data**

One of most common cleanup methods, could be to turn from machine readable to human readable for reporting or so that data is in particular format for machine reading or API to work. Many methods of formatting in Python.

Date formatting can be done easily in Python, a very common process. There are many applications.

**Outliers and bad data**

When considering outliers and bad data, the aim is to clean data not manipulate or change it. Time should be taken to explore and consider options, with removal for clear reason / intention to normalise it.

Some easy ways to check data:

* First clue is the source – how was data collected, by who, has it already been cleaned or processed? Source can determine bias in dataset which could affect conclusions.
* Potential errors when looking whether any data points do not fit. Are values missing, do data types not make sense, are there many not applicable responses which would distort a sample.

Ask, can we account for the answers, such as NA or missing values. For example is it a female only question and survey is for everyone so half would be missing.

Checking for variation in data types in rows and columns can help us understand where there could be errors and devise a cleaning strategy, are the questions important? What are the errors, are they missing values, poorly inputted or something else. Once we understand the data can make a plan to fix it.

**Duplicates**

Many methods of duplicate removal, identifying a unique key in data can be important. This could be a reference number but when no obvious unique key due to messy data will have to determine one, example given includes birth date and address.

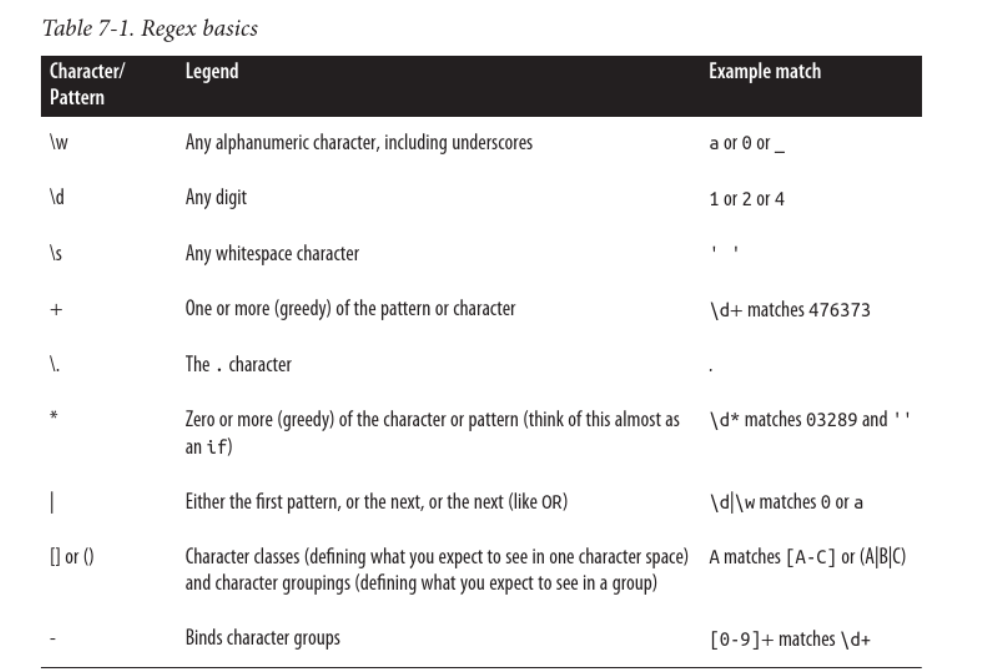
**Fuzzy matching**

Allows determination of whether two strings are the same, not as in-depth as natural language processing or or machine learning it can help relate data items with similar meaning. Various python libraires support this. Could be unclean data that has been sloppily input such as sloppily input or deviations in synaptic or format. Knowledge of data and complexities can help with assessing the quality of the fuzzy matches.

**RegEx Matching**

RegEx means Regular Expression. Allow computer to match, find or eliminate patterns in data defined in the code. For example, identify contact details from a string of text. Regex can become complex and hard to read but they are useful.

Regex invaluable for parsing really messy data like from webscraping, can create patterns then identify data with them.



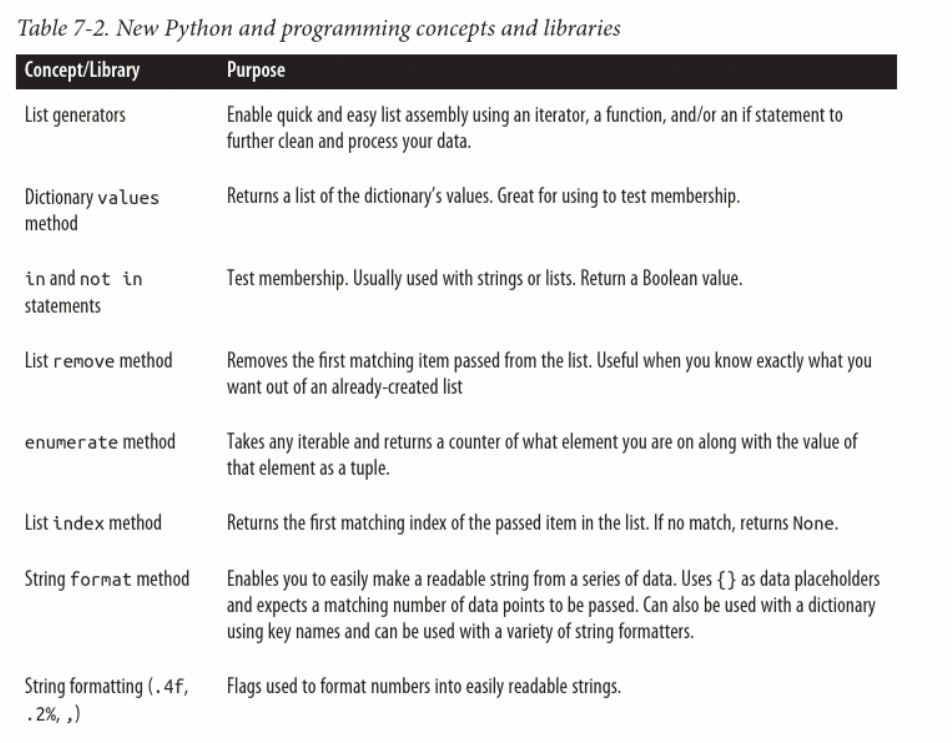
Source: Kazil, J & Jarmul, K. (2016) Data Wrangling with Python. O'Reilly. Media Inc.

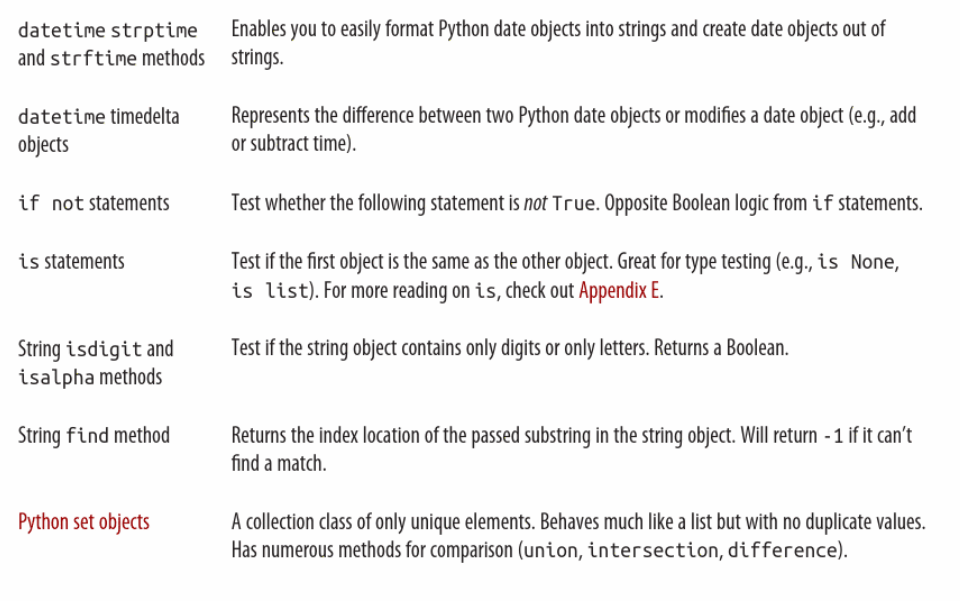
Python has a built in RegEx module. Uses the likes of match and search. Match looks at start of string, whilst search looks right to end of the string.

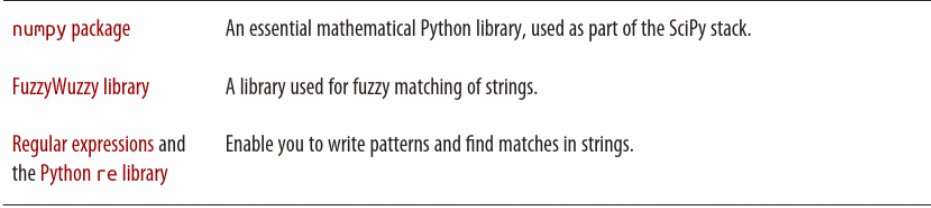
**Duplicate records**

Sometimes may wish to combine duplicate records if they contain different information. If the dataset has exact duplicate rows, likely to just remove. An example could be combining individuals interviewed from one household into household data.

**Concepts from chapter:**

****

****

****

**Reflections:**

The text in the book did not always result in successfully run code, so trouble shooting was required throughout. This may be due to being written in Python 2. Examples include being asked to slice a zipped object which is not allowed by Python 3, a considerable amount of time was spent problem solving, whilst this meant a lot of Python skills were learnt it was frustrating in terms of progress.

The chapter overall showed some practical examples of data cleaning and considerations when working to ensure data is fit for purpose. It highlights that even well-structured data can have issues and readability problems and needs cleaning. Problems like duplicates and nulls need to be considered as to how they will impact on analysis and how they should be dealt with.

**Chapter 8: Data Cleanup: Standardizing and Scripting**

**Normlaising and Standardising data**

Cleanup of data can be automated; data can be standardized and normalised. This means calculating new values using the values you currently have, or it might mean applying standardization across data.

Statistically, normalisation often means calculating new values from a dataset to a standard scale. For example scores across a season given a value of between 0 – 1. Normalize can help see percentiles across different groups (or cohorts). Can help with identification of normal range and outliers.

Standarization is basically what is the normal range and what is outside. Could be standard deviation.

Sometimes these both require the removal of outliers. Helps to see patterns and distribution of data. Sometimes known as trimming of outliers.

**Determining what data cleanup is right for your project**

Depending on nature of data and how often it will be analysed, you might choose a different path for your cleanup. Full scripting may not be possible if data is haphazard or infrequently used, scripting may take more time than worth.

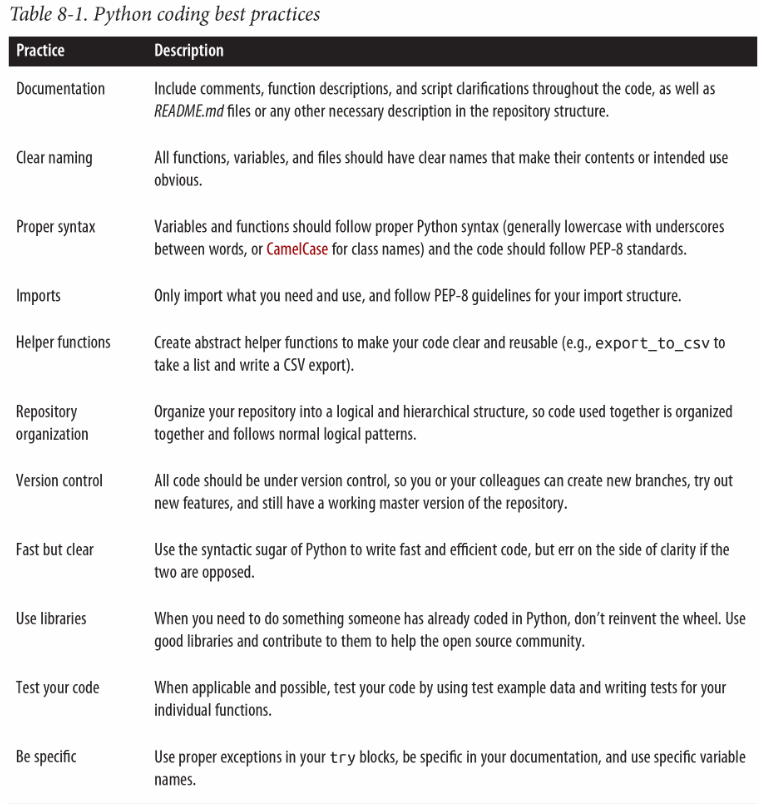
If cleanup onerous, may want to create a repository of helper scripts, so even if can’t fully script can use these functions to process faster. For example, import or duplicate cleanup that are reusable.

**Scripting your cleanup**

If cleanup has a determined pattern and is unlikely to change, could scripty entire cleanup process.

Key part of this is making sure code is clear and readable in terms of function naming, comments – make it clear to you and others (Zen of Python rules! – explicit, clear and practicable as possible)

Re-usable code can become a function, then can write generic simple code:



Documentation of code is hugely important for self and team – what makes sense to you or now, may not in the future. Also, auditable.

Good practice to describe step by step what code is doing in a readme.md

In simple terms, a script will group many lines of data into one script

**Reflections:**

Data processes require a great deal of time spent cleaning data and making it fit for purpose. Scripting this process so they are automatic and save time. We need to ensure the code is written as clearly as possible and fully documented so it can be re-used in the future and understood by others. Creating functions that are portable to other processes is hugely beneficial for saving time and being more productive.

Normalising and Standardizing can help identify outliers in data to improve the quality of it or aid analysis.