Book analysis with AI techniques

Learning Sequence

# (Top summary)

This learning sequence explores text analysis through Natural Language Processing, a significant application of Artificial Intelligence.

Teachers and students are led through a series of video tutorials to develop a Python program that can break down and analyse the content of a complete text, such as Robert Louis Stevenson's *Treasure Island*, and use smart sentiment analysis to attempt to determine the villain(s) and hero(s).

This learning sequence is recommended for **Years 9 and 10** or experienced students in Years 7 and 8. It is not recommended for beginners to General Purpose Programming. Basic understanding of iteration, branching and functions is assumed. (For a learning sequence on Natural Language Processing designed to better suit new programmers in **Years 7 and 8**, try [A Sentimental Chatbot](https://www.digitaltechnologieshub.edu.au/teachers/lesson-ideas/coding-a-sentimental-chatbot-in-python).)

View the [**Overview**](https://publish.viostream.com/play/bxixurbnazus84) video for more information on this learning sequence, including a short lecture on Natural Language Processing. (Check the Resources section for links to more information on advanced concepts such as Part Of Speech Tagging and Lexicon Normalisation.)

# Curriculum links (button)

Links with Digital Technologies Curriculum Area

| **Strand** | **Year** | **Content Description** |
| --- | --- | --- |
| Processes and Production Skills | Year 7-8 | Analyse and visualise data using a range of software to create information, and use structured data to model objects or events ([ACTDIP026](https://www.australiancurriculum.edu.au/Search/?q=ACTDIP026))  Define and decompose real-world problems taking into account functional requirements and economic, environmental, social, technical and usability constraints ([ACTDIP027](https://www.australiancurriculum.edu.au/Search/?q=ACTDIP027))  Design the user experience of a digital system, generating, evaluating and communicating alternative designs ([ACTDIP028](https://www.australiancurriculum.edu.au/Search/?q=ACTDIP028))  Design algorithms represented diagrammatically and in English, and trace algorithms to predict output for a given input and to identify errors ([ACTDIP029](https://www.australiancurriculum.edu.au/Search/?q=ACTDIP029))  Implement and modify programs with user interfaces involving branching, iteration and functions in a general-purpose programming language ([ACTDIP030](https://www.australiancurriculum.edu.au/Search/?q=ACTDIP030)) |
| Year 9-10 | Analyse and visualise data to create information and address complex problems, and model processes, entities and their relationships using structured data ([ACTDIP037](https://www.australiancurriculum.edu.au/Search/?q=ACTDIP037))  Define and decompose real-world problems precisely, taking into account functional and non-functional requirements and including interviewing stakeholders to identify needs ([ACTDIP038](https://www.australiancurriculum.edu.au/Search/?q=ACTDIP038))  Design the user experience of a digital system by evaluating alternative designs against criteria including functionality, accessibility, usability, and aesthetics ([ACTDIP039](https://www.australiancurriculum.edu.au/Search/?q=ACTDIP039))  Design algorithms represented diagrammatically and in structured English and validate algorithms and programs through tracing and test cases ([ACTDIP040](https://www.australiancurriculum.edu.au/Search/?q=ACTDIP040))  Implement modular programs, applying selected algorithms and data structures including using an object-oriented programming language ([ACTDIP041](https://www.australiancurriculum.edu.au/Search/?q=ACTDIP041)) |

# Assessment (button)

Each part of this learning sequence builds on the previous part. Students can be encouraged to type up their own code as demonstrated in the videos. (Links to milestone completed programs are provided after each part, in case of student absences or confusion.)

A number of project ideas are suggested at the end of the sequence, ranging from simple to highly ambitious. Students may collect appropriate text, prepare program designs and/or implement coded programs in response to these prompts.

In assessing code in languages like Python, consider a rubric that brings in important skills for General Purpose Programming.

* Download a sample rubric in [Word](https://www.digitaltechnologieshub.edu.au/docs/default-source/Lesson-Ideas/visual-to-text-coding/rubric-example---software-design-and-development-with-general-purpose-language6baf4a9809f96792a599ff0000f327dd.docx) or [PDF](https://www.digitaltechnologieshub.edu.au/docs/default-source/Lesson-Ideas/visual-to-text-coding/rubric-example---software-design-and-development-with-general-purpose-language.pdf) format.

# PART 1: Setup

View the video[**Setup**](https://publish.viostream.com/play/bxixurbn5bz73d). This will introduce the [repl.it](https://repl.it) coding environment, and the process for obtaining the text for *Alice in Wonderland* from the online repository [Project Gutenberg](https://www.gutenberg.org/).

## Questions for discussion

Q: The presenter uses the site Project Gutenberg to obtain the text for *Alice in Wonderland*. Why is this site used?

A: Copyright law might prohibit the use of many modern books and songs. Project Gutenberg hosts texts that are out of copyright, often due to their age. Note that texts available on the site may not be suitable for all audiences.

Q: The presenter refers to repl.it as a Python IDE. What is an IDE, and why do we use one?

A: IDE stands for Integrated Development Environment. Unless you are coding in Windows Notepad, you are probably using an IDE. IDEs gather the tools a programmer needs to type out code, test the program with the push of a button, and more easily identify errors. Code text is usually colour-coded to help with readability, documentation is readily available and syntax errors are underlined automatically, similarly to how spelling or grammar errors are identified in a modern word processor. Advanced IDEs also provide drag-and-drop tools to create graphical user interfaces.

Q: When obtaining *Alice in Wonderland*, the presenter selects the Plain Text format. Why might other formats like HTML, EPUB or Kindle be unsuitable for text analysis?

A: HTML and other formats include information that is not part of the text itself. This information is used for attractive presentation, such as images, text formatting and page setup. As the name suggests, Plain Text does not have this extra information.

# PART 2: Removing punctuation

View the video [**Removing punctuation**](https://publish.viostream.com/play/bxixurbn5bz73m). In this part, we write and test a function to remove all punctuation from the text, so that our text analysis can focus on words only.

([Completed code](https://repl.it/@digitechhub/Part-2-completed-code) up to this point.)

## Questions for discussion

Q: The goal in this part is to remove all punctuation so that only words remain. Could this hinder our efforts to analyse the text for sentiment?

A: Punctuation can certainly change the *meaning* of a string of words. (eg. "Give to charity. Please no presents." vs. "Give to charity? Please, no. Presents!") The sentiment analysis in this learning sequence is less advanced than other approaches that might be in existence. It will rely on the volume and strength of certain words associated with other words.

Q. The presenter demarcates the string of punctuation characters using triple apostrophes at the start and end. eg, '''#%^&\*'".,\_-=()''' Why was this necessary?

A: Python needs something distinctive to make it clear where a text string starts and ends. Normally this is done with a quote (") or a single apostrophe (') at each end, but the string of punctuation characters already includes both these.

Q: The code for removing punctuation is placed inside a **function**. What are some advantages of this approach?

A: Functions allow a programmer to organise things by separating some code from the main program, so that section of code can be called (run) whenever required. **Parameters** then allow the main program to supply different values to the function. In the video, a parameter st is added so that any book text can be supplied to the remove\_punctuation(st) function.

Q: The presenter names the function remove\_punctuation(…). Are there rules about naming functions and variables in Python?

A: The naming of functions and variables is very important for code readability, so that other programmers can understand your code. One useful approach is to name functions with verbs, since they are performing an action. (Note: Using underscores in names is a Python **convention** rather than a rule. In other languages, like JavaScript, the convention is to use 'camel case', eg. removePunctuation(…))

## Skill review

Functions will be used throughout this learning sequence, including functions with parameters and return variables.

View the video [**Intro to Functions in Python**](https://publish.viostream.com/play/bxixurbnazuser) for a brief introduction to writing and using functions.

Other link

# PART 3: Tokenisation

View the video [**Tokenisation 1**](https://publish.viostream.com/play/bxixurbnazwbd7), noting the minor changes made manually to the text of *Alice in Wonderland* before the coding begins:

* The word CHAPTER has been added before each chapter heading, eg. “**CHAPTER** I--DOWN THE RABBIT-HOLE”).
* The Gutenberg license text has been removed from the end of the file.

In the part, we write new functions to break up the book text into a list of words, or into a list of sentences.

([Completed code](https://repl.it/@digitechhub/Part-3a-completed-code) up to this point.)

## Questions for discussion

Q: When writing the create\_word\_list(…) function, most of the work is done by a built-in string function called split(). What exactly does split() do? *(Hint: you can always look up the* [*official Python documentation*](https://docs.python.org/3/library/stdtypes.html#string-methods)*, or a Python cheat sheet.)*

A: Python's built-in split(…) function breaks up a string, resulting in a list that contains each separate part. If no argument is given, as in the case of creating a list of words, the string is split wherever a space occurs. A different character can be provided as an argument, such as a full stop ('.') to break up the string by sentences.

Q. During the video, the presenter decides to improve the remove\_punctuation(…) function. What change is made?

A: An additional parameter called exception is added. This allows one punctuation character to be designated to not be removed along with all the others. Inside the function, the exception character is immediately removed from the punctuation string, so that it does not get removed from the text itself.

Q: In the improved remove\_punctuation(…) function, what does it mean that the new parameter is written as exception=''

A: When parameters are written with an = sign, this means there is a **default** value. It allows someone to call the function without *having to* supply an argument for that parameter. In this case, exception has a default value of an empty string ''. With this value for the parameter, the punctuation string will remain intact and the removal of punctuation will work as normal.

Now view the video [**Tokenisation 2**](https://publish.viostream.com/play/bxixurbnazus85). In this video, we write two more functions to break up the book text into a list of paragraphs, or into a list of chapters. This completes our library of functions for tokenising the book's text.

([Completed code](https://repl.it/@digitechhub/Part-3b-completed-code) up to this point.)

## Skill building

Working with large bodies of text usually requires a bit of manual editing.

View the video [**Text File Preparation**](https://publish.viostream.com/play/bxixurbnazwb9j) for more details on obtaining and editing the most suitable book text of *Alice in Wonderland*.

# PART 4: Modular programming

View the video [**Modular programming**](https://publish.viostream.com/play/bxixurbnazus86). In this part, the functions we've created are separated into a different file.

([Completed code](https://repl.it/@digitechhub/Part-4-completed-code) up to this point.)

## Questions for discussion

Q: Besides cleaner main programs, what other advantages might come from this modular approach of placing groups of functions into separate files?

A: A modularapproach to coding makes it easier for different programmers to work on a project. Programmer A can use a function from a file written by Programmer B without needing to see the code inside it, as long as there is documentation explaining how to use the function. Programmer B can make changes to the internal code in a function without necessarily disturbing Programmer A.

Q: In the video, the presenter uses the import statement to connect the functions from the new tokenization.py file into the main program. Have you used import to access other modules in the past?

A: A common module imported into Python programs is the random module, which provides functions for generating pseudo-random numbers like rolling a die. Students may also be familiar with the turtle module, used to issue Logo commands to an onscreen turtle.

# LECTURE: Sentiment analysis

View the video[**Lecture – Sentiment Analysis**](https://publish.viostream.com/play/bxixurbnazus89) to:

* discover interesting connections between linguistics and digital technologies,
* import a Python module that makes it easy to incorporate sentiment analysis into your own programs,
* explore two numbers for measuring sentiment: polarity and subjectivity.

*Lectures are primarily intended for a teacher audience, but you may choose to re-view the video with students.*

# PART 5: Testing sentiment analysis

View the video [**Testing Sentiment Analysis**](https://publish.viostream.com/play/bxixurbnazus8z). In this part, the TextBlob module is used to attempt to rate the *polarity* and *subjectivity* of sentences, paragraphs and chapters. (For an explanation of these two concepts, be sure to view the lecture in the previous section.)

([Completed code](https://repl.it/@digitechhub/Part-5-completed-code) up to this point.)

## Questions for discussion

Q: What does polarity measure?

A: Polarity is a measure of how “positive” or “negative” a block of text seems to be. Sentiment Analysis modules like TextBlob draw on large lists of words compiled from research into linguistics.

Q: What is the numerical range of the polarity value?

A: The polarity value can range from -1 (very negative) to 1 (very positive). A value of 0 suggests a block of text that is not weighted in either direction.

Q: What does subjectivity measure?

A: Subjectivity can be thought of as a measure of how **emotive** the language in a block of text seems to be. A high value means that the text contains a high number of adjectives and nouns that are associated with strong feelings from the writer. A low value means that the text can be thought of as more “clinical” or objective, containing fewer emotive words.

Q: What is the numerical range of the subjectivity value?

A: The subjectivity value can range from 0 (completely objective) to 1 (highly subjective).

Q: Do you think this way of analysing text sentiment is foolproof?

A: Despite its basis in linguistics research, this is still a crude means of determining sentiment when compared to a mature human’s analysis. This demonstrates the complexity of human communication. (Try inputting a sarcastic sentence, for example.) Developments in Machine Learning may result in more reliable sentiment analysis results.

# PART 6: How many times does each word appear?

View the video [**Frequency of Words 1**.](https://publish.viostream.com/play/bxixurbnazused) In this part, we write and test a function to store each unique word in the book alongside the number of times that word appears in the book. In PART 7, we’ll rank these to find out the most frequent words in the book.

([Completed code](https://repl.it/@digitechhub/Part-6-completed-code) up to this point.)

## Skill review

A *dictionary* data structure is used to hold the data for the most frequent words. This data structure is sometimes called an *associative array* or a *map* in other programming languages.

View the video [**Intro to Dictionaries**](https://publish.viostream.com/play/bxixurbnazus8s) for a brief introduction to the *dictionary* data structure.

These short exercises will help you practice using Python Dictionaries:

* [Exercise 1](https://repl.it/@digitechhub/Python-Dictionaries-Exercise-1) – accessing elements and printing ([solution](https://repl.it/@digitechhub/Python-Dictionaries-Exercise-1-Solution))
* [Exercise 2](https://repl.it/@digitechhub/Python-Dictionaries-Exercise-2) – creating dictionaries and adding elements ([solution](https://repl.it/@digitechhub/Python-Dictionaries-Exercise-2-Solution))
* [Exercise 3](https://repl.it/@digitechhub/Python-Dictionaries-Exercise-3) – accessing elements with a loop ([solution](https://repl.it/@digitechhub/Python-Dictionaries-Exercise-3-Solution))

## Questions for discussion

Q: Why is the *dictionary* data structure suitable for storing each unique word in the book with its frequency?

A: The *dictionary* data structure consists of a set of pairs – keys and values. Our data is also a set of pairs. The key is the unique word from the book, and the value is the number of times that word appears in the book.

Q. Couldn’t two lists be used instead of a dictionary?

A: It would be feasible to use two parallel lists. One list would contain all the unique words and the other list would contain all the frequencies. This may not be considered an ideal solution because it relies on the lists always being kept in sync. For example, removing an element from one list means the same element must be removed from the other list. This leaves room for a programmer to forget, and then you have a bug.

Q: The function create\_frequency\_dictionary(words) has a single parameter words. What is expected to be given here?

A: The words parameter is expected to be a complete list of all words from the book, ie. the full book split up into its words. We already created a function create\_word\_list(…) to do this job.

Q: The function create\_frequency\_dictionary(words) has two sections of code inside, each with a loop. The first section builds a list of unique words from the complete list of the book’s words. What does the second section do?

A: The second section builds the dictionary by storing each unique word alongside the number of times it appears in the book.

Now view the video [**Frequency of Words 2**](https://publish.viostream.com/play/bxixurbnazus8a). In this video, we make some improvements to our function. Now, the dictionary will be constructed ignoring the case of the words from the book. So, “Rabbit” and “rabbit” will now be considered one unique word, and the frequency value will reflect all instances of both words.

But, if the function’s new second argument cap is set to True, something quite different will happen. The dictionary will be constructed with *only* the Title Case words from the book, such as proper nouns. So, “rabbit” will not be included at all, but words like “Rabbit” and “Alice” will be included.

*During this video, the presenter makes use of a Python shortcut called a List Comprehension, which allows a list to be quickly made from another list without many lines of code. This is not an essential skill and is merely done for convenience. See* [*this external tutorial*](https://www.tutorialspoint.com/python-list-comprehension) *for more information on List Comprehensions.*

# PART 7: Ranking words by frequency

View the video [**Ranking Words by Frequency**](https://publish.viostream.com/play/bxixurbnazwfge). In this short part, we make use of a dictionary created with the function we wrote in PART 6. The dictionary contains unique words from the book alongside how often they appear in the book. Now we will sort those entries. The result is a simple list of the words ordered by frequency.

([Completed code](https://repl.it/@digitechhub/Part-7-completed-code) up to this point.)

## Questions for discussion

Q: What is the purpose of the sorted(…) function used in this video?

A: The sorted(…) function is a very powerful function built into Python (see [this article](https://docs.python.org/3/howto/sorting.html) from the Python documentation for more). In our case, we are using it to sort the unique words from the dictionary containing the unique words and their frequencies. We sort the words from most to least common according to their frequencies. The result is not another dictionary, but a simple list called ranked\_list.

# PART 8: Removing stop words

View the video [**Removing Stop Words**](https://publish.viostream.com/play/bxixurbnazusey). In this part, we write a function to filter out very common English words, and another function to filter out *stop words* like “the”, “I”, “and”. By first removing all these from the list of words in the book, our ranked list of frequent words will be more useful.

([Completed code](https://repl.it/@digitechhub/Part-8-completed-code) up to this point.)

* Click [here](https://bit.ly/gtH1000) to access the webpage with 1000 common words.
* Click [here](https://bit.ly/stopWORDS) to access the webpage with stop words.

## Questions for discussion

Q: What is the difference between the two new functions remove\_common(words) and remove\_stop\_words(words) we created in this video.

A: The remove\_common(words) function looks for the occurrences of 1000 well-known words within the full list of words in the book, removing any that it sees. The remove\_stop\_words(words) function does the same thing, but it only filters out a smaller selection of words like “if”, “and”, “the”.

# PART 9: Heroes and villains

View the video [**Heroes and Villains 1**](https://publish.viostream.com/play/bxixurbnazwggf). By now, we have identified the likely main characters in the book. In this part, we use sentiment analysis to guess whether each character is a hero or villain.

([Completed code](https://repl.it/@digitechhub/Part-9-completed-code) up to this point.)

## Questions for discussion

Q: Given a character’s name, exactly how does our new function try to guess whether that character is a hero or a villain?

A: The function measures the *positivity* of each paragraph in which the character appears by name. Each positivity value is used to increase or decrease an overall score. Finally, a verdict of “hero” or “villain” is pronounced based on the final score.

Q: Do you think this is an effective way to determine whether a character is a hero or villain?

A: Clearly, the function does not always get it right. It is looking for positive or negative words associated with the character by being in the same paragraph.

Now view the final video [Heroes and villains 2](https://publish.viostream.com/play/bxixurbnazusef). In this video, we try a couple of other books, and we tweak the behaviour of the function for judging the main characters.

# Projects / Assessment

### PROJECT IDEA 1

Initially, students may wish to try **reusing** the program already written in this learning sequence:

* choose a different book that can be obtained in Plain Text format,
* try to identify the main characters,
* try to determine which of the main characters are heroes or villains,
* by judging the polarity of each chapter, try to determine if the story has a happy or sad ending.

### PROJECT IDEA 2

Sometimes it can be hard to tell if articles from newspapers and other online sources are **reporting** pieces or **opinion** pieces.

Write a fresh program that uses sentiment analysis to determine if an article is a reportingpiece or an opinion piece, based on the subjectivity of its language. We might expect opinion pieces to have higher subjectivity.

You will need to:

* find at least 3 reporting pieces and 3 opinion pieces, ideally from the same news source,
* obtain or convert each article into Plain Text format (eg. using copy-paste),
* use the TextBlob library to analyse the subjectivity of the article,
* make a decision whether the article is reporting or opinion.

*Note, you may wish to reuse modules or functions written in this learning sequence, but your main program must be freshly written, with appropriate comments.*

### PROJECT IDEA 3

Design and implement a **research project** test a limit of the sentiment analysis approach used in this learning sequence. eg. How well does it respond to different conversational styles?

### PROJECT IDEA 4 (AMBITIOUS)

Write a new tool for analysing dialogue in a movie or play script.

The tool will be able to separate each paragraph, ignore stage/screen directions and notes, and connect each bit of dialogue with a character from a limited cast.

From there, characters can be compared based on volume of dialogue and sentiment analysis of dialogue.

# Resources

* Coding
  + Python [CheatSheet](https://groklearning-cdn.com/resources/cheatsheet-python-1.pdf) (from Grok Learning)
  + Another Python CheatSheet that focuses on String functions (ways to manipulate text).
  + [Visual to Text Coding](https://www.digitaltechnologieshub.edu.au/teachers/lesson-ideas/visual-to-text-coding-index-page) – a series of lessons with videos and exercises to help you and your class transition from visual coding (eg. Scratch) to general purpose programming (eg. Python and JavaScript).
* Natural Language Processing theory
  + [Natural Language Processing](https://en.wikipedia.org/wiki/Natural_language_processing) on Wikipedia.
  + An [article](https://www.freecodecamp.org/news/an-introduction-to-part-of-speech-tagging-and-the-hidden-markov-model-953d45338f24/) on part-of-speech tagging.