* https://github.com/Starwort/SPLIWACA -- This repository holds the specification for the language that I am going to attempt to write a compiler for.
* https://en.wikipedia.org/wiki/Compiler -- I can use the wikipedia page on compilers to get basic information about how compilers work and find more academic sources to use in my research.
* https://www.youtube.com/channel/UCUVahoidFA7F3Asfvamrm7w -- This youtube channel has a tutorial about how to write a basic interpreter in Python. I can use these tutorials to get a basic idea of some implementation logic and structure for my product.
* I may use http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.735.4640&rep=rep1&type=pdf because it gives me information about the phases and structure of a transpiler. The concepts shown here are essentially the same as those used for other types of compilers and interpreters because the program has to go through similar initial steps before being able to branch out into different directions to produce the required output.
* https://www.youtube.com/watch?v=QXjU9qTsYCc&safe=active -- This youtube video provides the basics of how a compiler is structured and what it does, so I can use this source to provide a basic outline of the different stages of my product.
* I may read Basics of Compiler Design by Torben Ægidius Mogensen because this book is intended for use in an introductory compiler course so will be at a low enough level that I can easily understand the concepts it talks about and use them to help me with my own project.

I will also use other academic papers and articles I find on google scholar or other similar search tools to provide me with more

I am using a large variety of academic and non-academic work to allow me to get a broad overview of the subject area and base my decisions for my own code off what seems to be the best approach from all that I study, and also get a balanced view of the field with minimal amounts of inaccurate information.

I first became properly interested in creating my own programming language about 9 months ago after watching a video about how a programming language should be structured: <https://www.youtube.com/watch?v=TH9VCN6UkyQ>. This video highlighted to me many issues with current programming languages and inspired me (briefly) to try to create my own. However, I did not get very far with it, following almost line-for-line the beginning of a tutorial on YouTube before giving up. Going into A-level computer science, this project now represents a smaller challenge that I can do to expand my understanding of how programming works and why programming languages are structured in a certain way. I realise that the language that I am going to create an interpreter for is unlikely to be well structured or practical for larger applications but it is simpler than most other languages so it will be a good way to learn how an interpreter is made. I want to apply to do computer science or physics at university, and computer programming is incredibly useful to both, so this project will give me a better knowledge of how programming works that I can transfer to work at university level.

I intend to create a computer program that will translate code from SPLIWACA into Python that can then be run or compiled using a python interpreter or compiler. I will also produce a 1000- to 1500-word report on the success of the project and what I learned from researching for it and writing the code for it. I will support my report with a bibliography detailing what sources I used in research and what I used them for, either referenced as notes in the report or as annotations next to the source referencing particular areas of the codebase that I used that source to create.

In researching to create the product, I will keep a record of what sources I use as I go along. I will look initially at the differences between compilers, transpilers and interpreters and then more specifically at how different stages of compilation are typically handled by the different programs, if there is any difference. It may be interesting also to note where there are few differences as this may indicate areas which are either underdeveloped or almost as efficient as possible. I will decide after looking at the differences between compilers, transpilers and interpreters which specifically I will aim to produce, as each poses different challenges that may or may not be possible to surmount in the time I have to complete this project.

As I produce the program for the project, I will create pseudocode for complicated algorithms that may be difficult to understand otherwise, which will also help me write the code for the algorithms quicker and more easily. The pseudocode for the algorithms will be either in SPLIWACA or in plain text detailing the steps that need to be completed, and, if necessary, how each step will be completed as another algorithm. However, each algorithm should be easy to understand as a single unit without any extra details.