Design Document

The transpiler will be split into two major sections comprising of a few smaller, separate, stages. The two major sections will be the frontend and the backend. The frontend operates on the original code and translates it to an annotated syntax tree. This will then be translated to an intermediate code representation which is then passed to the backend. The backend will operate on this code to optimise it and then generate the target language code. These phases are common in compiler design [[1]](#_Bibliography), requiring a few modifications for use in a transpiler rather than a compiler [[2]](#_Bibliography).

The phases that will be implemented for this transpiler will be as follows:

FRONTEND:

Lexer

Parser

Semantic Analysis

Intermediate Representation Generation

BACKEND:

Optimisation

Final python code generation

It is possible that I may generate python code at the IR generation phase and feed this into Cython [[3]](#_Bibliography) or some other python compiler to compile to an EXE.

# Bibliography

|  |  |
| --- | --- |
| [1] | T. Æ. Mogensen, Basics of Compiler Design chapter 1.2. |
| [2] | E. &. N. D. Ilyushin, "On source-to-source compilers," *International Journal of Open Information Technologies,* vol. 4, 2016. |
| [3] | "Cython," [Online]. Available: https://cython.org/. |