As Cython is a typed language(this means the variables need to be initialized with a type) I will build a compiler with the help of a symbol table to store the initialization of all the variables in the input code

For this I will use dynamic 2D lists. The purpose of the dynamic table is to grow constantly by adding a register each time the program finds a variable definition.

Python has the advantage that lists are always dynamic because it is a dynamic language, also are tuples.

The solution I propose to represent the symbol table is with a list of tuples, where the tuples contains the tokens the variable definition is made up of.

Example:

symbol=[(cdef, int , a , = ,1),(cdef ,int ,b,=, 2),(cdef, float, c, =, 0.99)]

Inserting new registers with the *append* function python provides

new_tuple=(cdef, int, f,=,0)

```
symbol.append(new_tuple)
```