1/ There are 6 phases in a compiler.
Syntax analysis phase is the second phase which takes the input from a lexical analysis in the form of token streams. It analyzes the source code against the written programming language rules to detect any errors in the code.
The output of this phase is a parse tree

2/ The syntactic structure of the language:
I created these block of codes:

- Import libraries and phase 1
- Create necessary functions
- Write syntax rules
- Build the program

3/ a) Sheet variable definition
In my program, I wrote function for variable definition:

```
def p_variable_definition(p):
    '''variable_definition : scalar_definition
                           | range_definition
                           | sheet_definition'''

def p_different_variable_definition(p):
    '''different_variable_definition : variable_definition
                                     | different_variable_definition variable_definition'''
```

Variable definition can be scalar_definition or range_definition or sheet_definition

b) Function call
In my program, functions can be called from other functions like this:

```
def p_function_call(p):
    '''function_call : function_call_print LSQUARE arguments RSQUARE'''

def p_function_call_print(p):
    '''function_call_print : FUNC_IDENT'''
    print('function_call(',p[1],')')
```

c) Sheet variable definition with initialization list (“{…}”)

In my program, initialization list can be defined as another function which can be called in the previous function recursively

```
def p_comma_arguments(p):
    '''comma_arguments : comma_argument
                       | comma_arguments comma_argument  '''

def p_comma_argument(p):
    '''comma_argument : COMMA formal_arg  '''
```

4/ a. Creating a “nested” function is allowed as inner functions is supported in Python. For example:

```
def outer(num1):
    def inner_increment(num1):
        return num1 + 1
    num2 = inner_increment(num1)
    print(num1, num2)

inner_increment(10)
```

b. Performing arithmetic with integers (1+2) is allowed because it is possible to define a rule for arithmetic operation in a new function

c. Initializing a range variable with a decimal value is not allowed as Python's range() only supports integers. Decimal values are unsupported

d. The syntax: xx--yy and –xx is not syntactically possible as Python does not allow “--”

e. Comparisons cannot appear in a sheet variable’s initialization list (sheet SS = { 1.0 < 2.0 }) as it is not supported the decimal values

f. To be ensured that addition/subtraction are done after multiplication/division, rules can be written to give priority for multiplication/division. The other way is to use parenthesis

g. In SheetScript, statements and definitions are not separated by semicolons (like in Java/C++) or line breaks (like in Python). Therefore, tokens and identation are needed for the program to know when something ends and a new thing begins.

For example:

```
def p_another_atom(p):
    '''another_atom : IDENT
```

```
            | cell_ref
            | NUMBER_SIGN range_expr
            | LPAREN scalar_expr RPAREN'"
    if p[1] == None:
        print('atom')
    else:
        print('atom(',p[1],')')
```

The identation and the word if here with the colon at the end means the if condition will start

5/ I implemented all function and subroutine definitions, function and subroutine calls in my program.

6/ This assignment is interesting for me as through this exercise, I know how the syntax analysis phase determines whether or not a text follows the expected format. The main aim of this phase is to make sure that the source code was written by the programmer is correct or not. It is nor too difficult nor too easy for me