Project 2

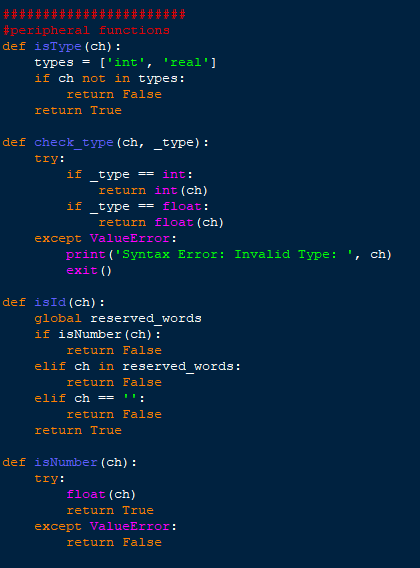
CIS424



Grail login ID: peparian

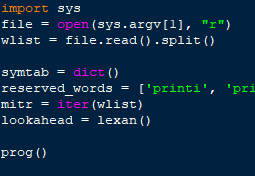
**Description of the Code:**

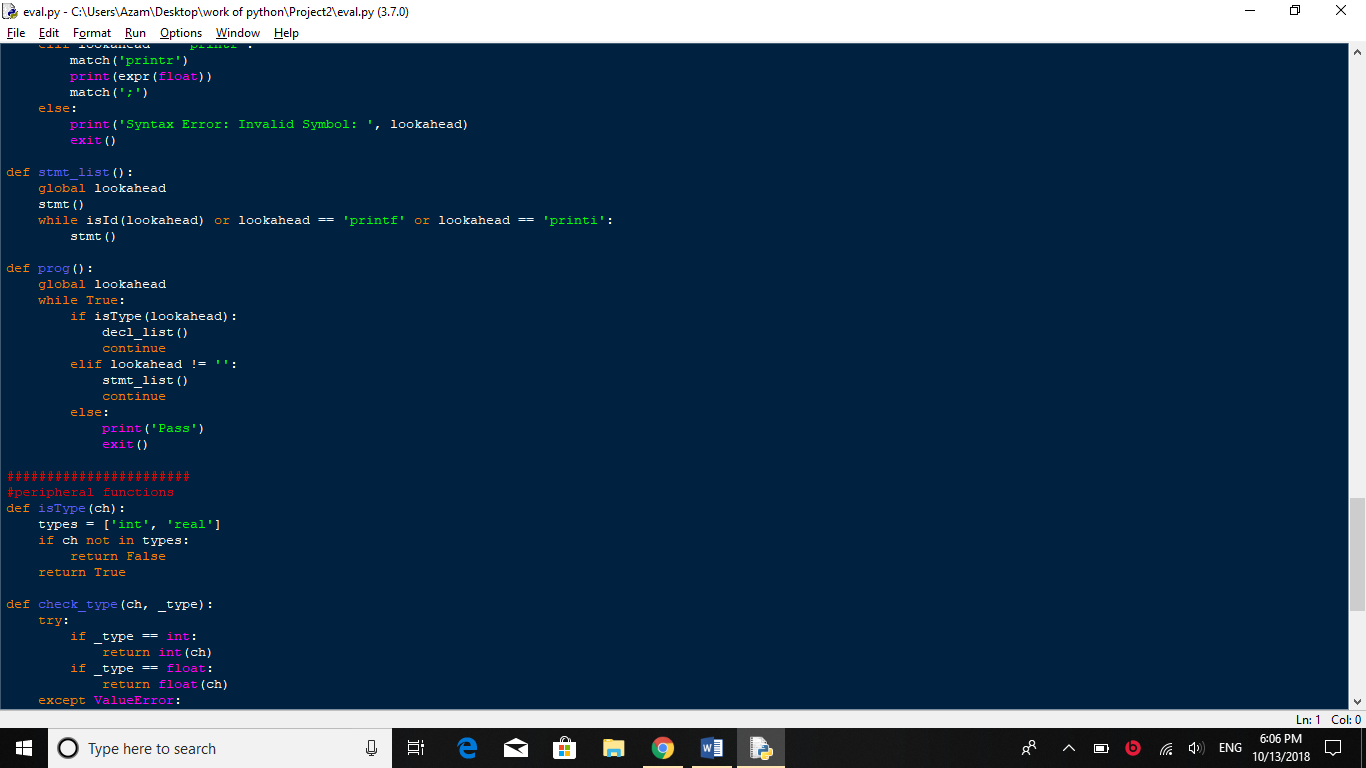
The python IDE used in this project was IDLE, that is prepackaged with version 3.6.



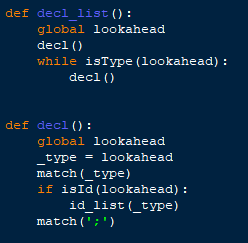
Peripheral functions:

* isType(ch ): checks if the token is “Int ” or “Real”
* isNumber(ch) : uses the try block to convert the token to float which then returns true, otherwise false.
* isId(ch) : checks if the token is a number in reserved words or an empty string.
* check\_type (ch, \_type) : compares the type of the token to \_type (\_type is an inherited attribute from stmt,statement) otherwise it will print “Syntax Error”.

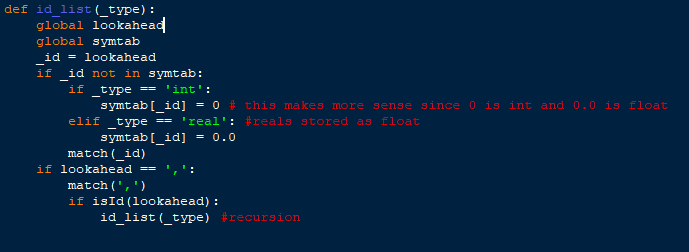




The program is structured with the entry point of the program prog(). This takes is to next line of the global level calling global lookahead. In Prog(), entering a while an infinite loop true, checking the type of lookahead entry.



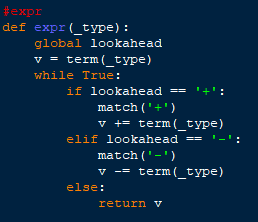
Continuing with *dec\_list()* checks and matches up the lookahead is *isType()*. That where *isType()* is created in *decl\_list()*. Now we check if the lookahead is *isID()*, then returns true. It moves onto the next token.



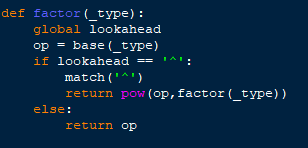
In *id\_list(),* next token is transferred over as lookahead and checks for its *\_id* in *symtab*. In if else statement, if type is “int” store it in *symtab*, or if it is “real” it stores is *symtab.*

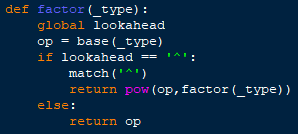
The statement *stmt()* function runs through global entry and symtab. If the id of the entry matched with id of symtab dictionary, then it retrieves the type of the value stored in symtab, stores it in \_id. After matching ‘=’, we store the returned value of expr(\_type) in the value of \_id. In *stmt(),* the “if” and “else” tokens are being matched and checked for its condition Boolean value returned from cond(). This value is used to determine whether to use the next call of expr() to replace the value of the \_id, or continue with the current. After this ‘;’ is matched.

After the *if, else* statements are completed, it checks if the lookahead is either printi or printr, when will pass in a different type for expr() inherited value. If none of these value are found, then print “Invalid Symbol”.

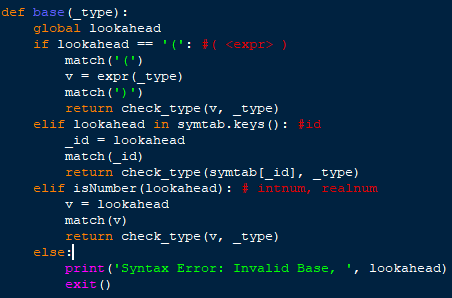


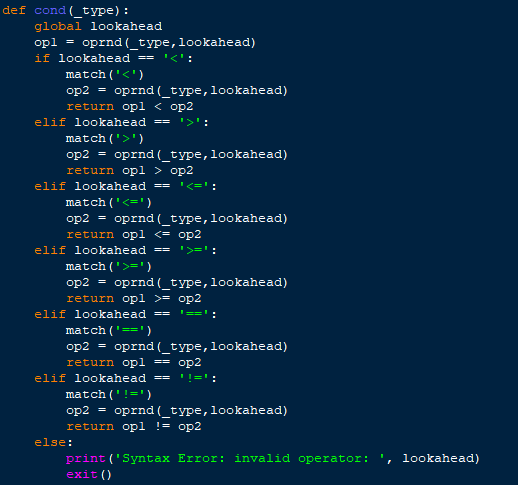
In *expr*, expression will call term recursively until there are no more plus or minus token

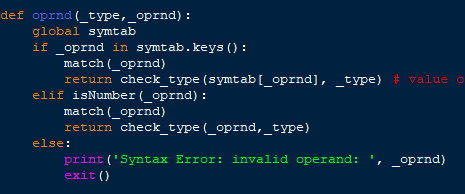
   
  
In *term*, term will call factor infinitively until there are no more plus or minus tokens.



In *factor,* factor will call *base* and then recusively call itself when lookahead is ‘^’ until a base is returned. After a token ‘^’, the return value is op^(factor()).

Base determines whether it is in expressions in parentheses, checks for \_id in symtab, or isNumber, then it matches this value, and returns the value returned check\_type. If the token not matched with a parentheses it will print “Syntax Error: Invalid Base”.

In *cond()*, condition lookahead token is checked for operands \_type. It matches with different operands like such “<, >, =>, <=, ==” and it will returns it’s whichever is greater, less than, or equals to.

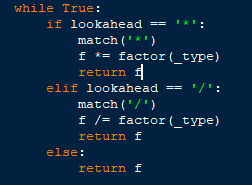


In operand, *oprnd*, parameters *\_type and \_oprnd* and also symtab dictionary are passed over to the function. In if statement, if the *\_oprnd* is in symtab keys dictionary, or if the operand is in isNumber, then matches it and operand moves to next token and returns the converted token which is actually the operand. If the check\_type(symtab[\_oprnd]) is true, it literally call the operand key value from the dictionary converted into the correct type. Else it prints “Syntax Error: invalid operand”.

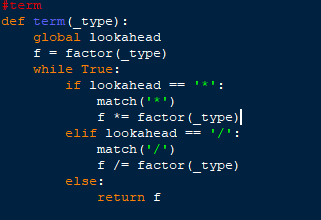
**Problems:**

**The list of tokens are strings, this needed to be compensated when converting to int or float.**

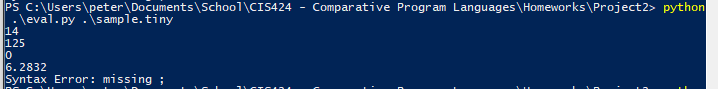
**Needed to change the dictionary pointing to a list back to to a regular dictionary. I did not need to store the type, I will just read the type directly**



**To**



**Because**



**Had to change expr() for the same reason listed above.**