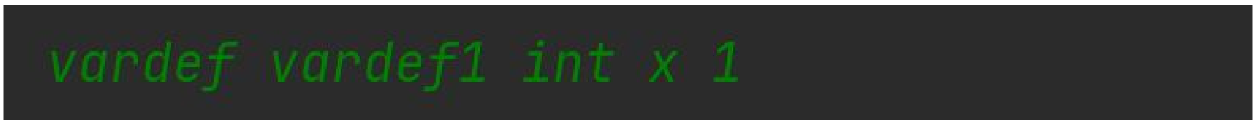
Group 23
User Manual

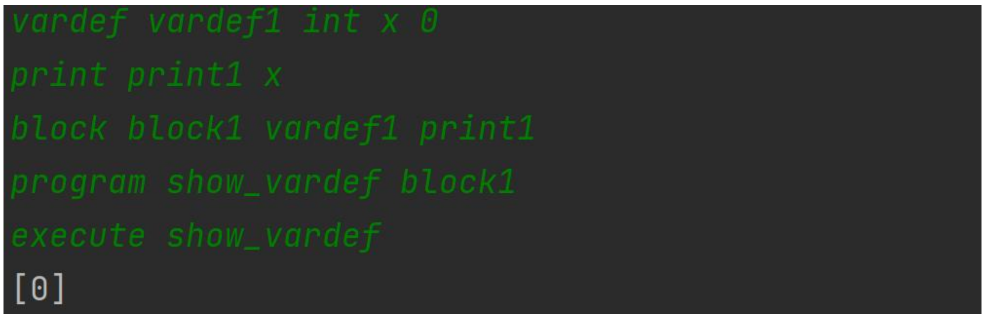
Note: When typing the commands, you should only type a word and there should be a space between each word.

1.[vardef]

‘vardef’ keyword is used to define the data type of the variable. For instance, keyword ‘int’ is inputted to define ‘x’ as an integer value.



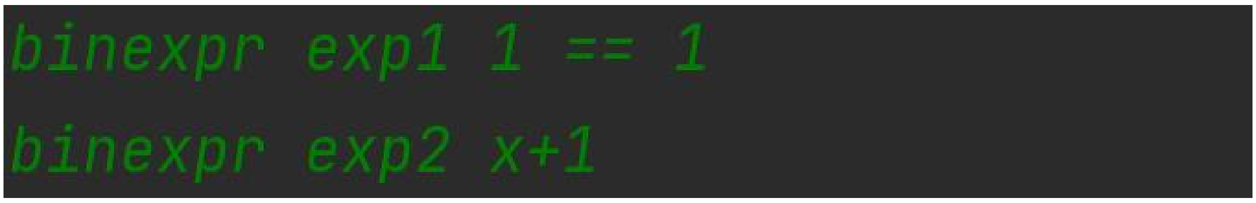
```
vardef vardef1 int x 1
```



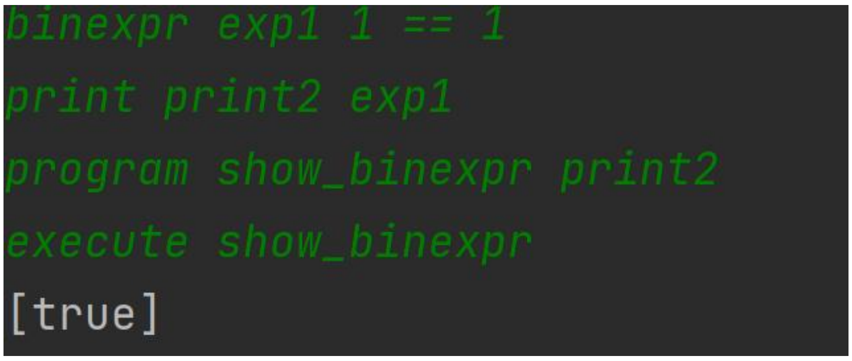
```
vardef vardef1 int x 0
print print1 x
block block1 vardef1 print1
program show_vardef block1
execute show_vardef
[0]
```

2.[binexpr]

‘binexpr’ keyword is used to distinguish where binary expression exists in the command line.Users can use both boolean or integer operators in between left and right operand as demonstrated in the image below.



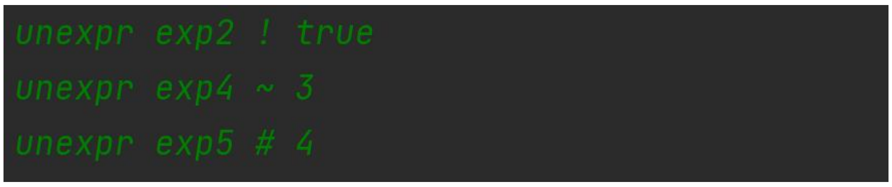
```
binexpr exp1 1 == 1
binexpr exp2 x+1
```



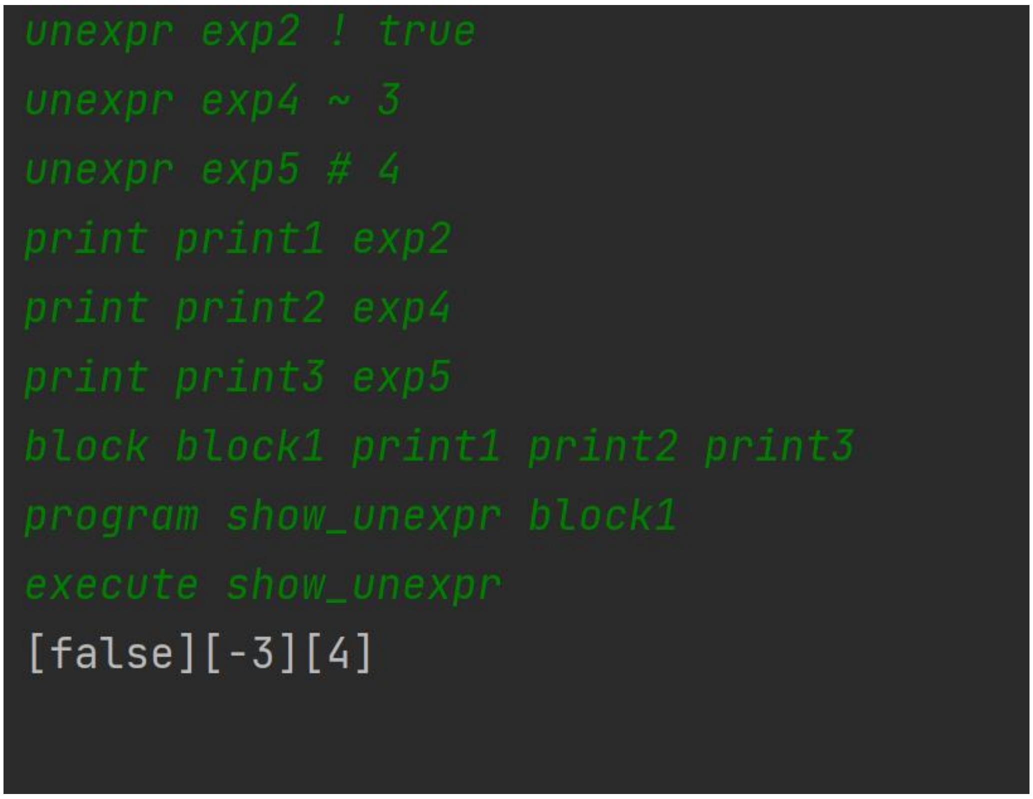
```
binexpr exp1 1 == 1
print print2 exp1
program show_binexpr print2
execute show_binexpr
[true]
```

3.[unexpr]

Similar to 'binexpr', 'unexpr' keyword is used to distinguish where unary expression exists in the command line.Users can use both boolean or integer operators after the left operand as demonstrated in the image below.



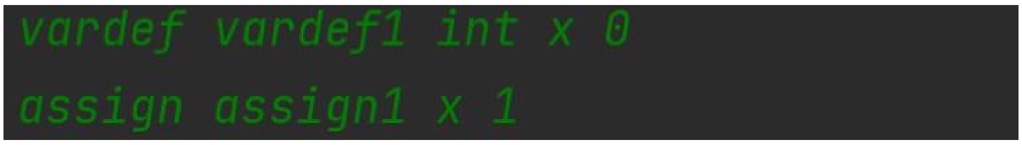
```
unexpr exp2 ! true
unexpr exp4 ~ 3
unexpr exp5 # 4
```



```
unexpr exp2 ! true
unexpr exp4 ~ 3
unexpr exp5 # 4
print print1 exp2
print print2 exp4
print print3 exp5
block block1 print1 print2 print3
program show_unexpr block1
execute show_unexpr
[false][-3][4]
```

4.[assign]

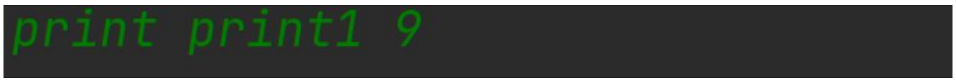
'assign' keyword is used to assign the value to the variable. This can be done after entering the command line as shown in the image below.



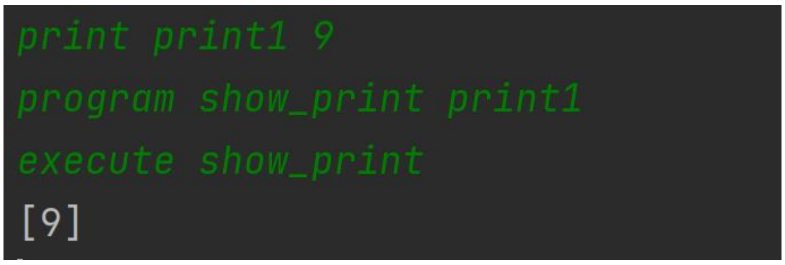
```
vardef vardef1 int x 0
assign assign1 x 1
```

[5.print]

'print' keyword is used to output the variable or value. This operation can be done after entering the command line as shown below:



```
print print1 9
```

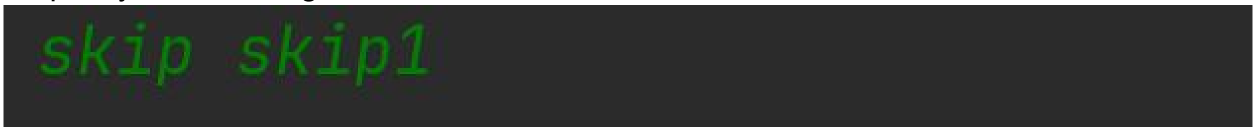


```
print print1 9
program show_print print1
execute show_print
[9]
```

The above shows the usage of print command.

[6.skip]

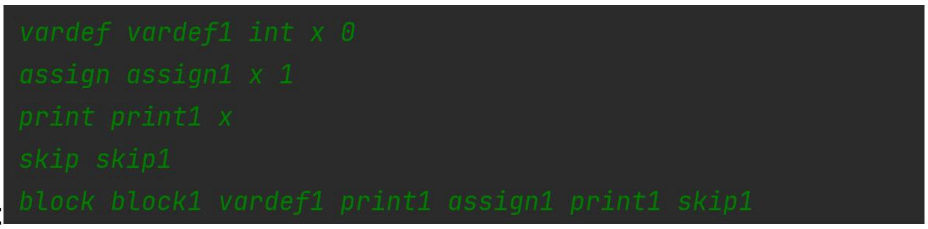
‘skip’ keyword is assigned for the discontinuation to the current iteration once.



```
skip skip1
```

[7.block]

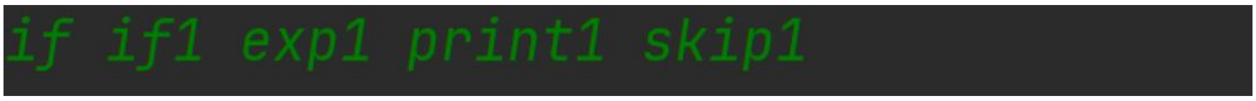
‘block’ keyword is used to store all the statements that need to be executed into a block. This operation can be done as shown in the command below:



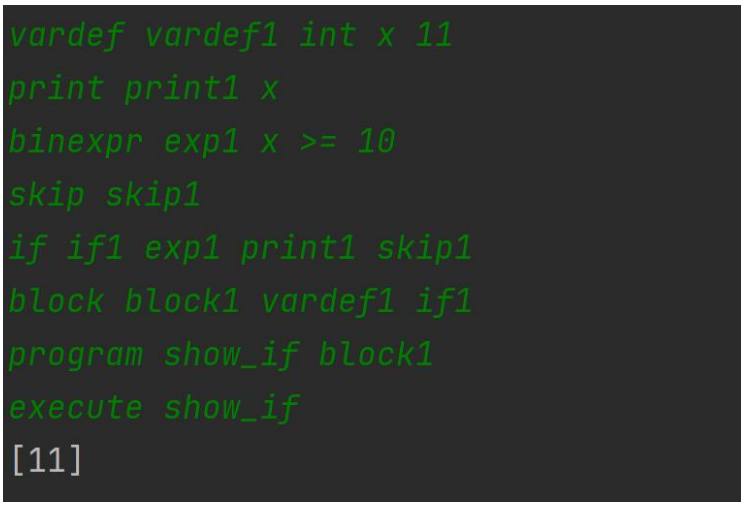
```
vardef vardef1 int x 0
assign assign1 x 1
print print1 x
skip skip1
block block1 vardef1 print1 assign1 print1 skip1
```

[8.if]

‘if’ is the decision-making keyword based on the condition of the expression. For example, if the condition of ‘exp1’ in the image below is ‘true’, it will proceed to ‘print1’, which is the output command mentioned previously. Likewise, if the condition is ‘false’, it will proceed to ‘skip1’, which is the command ‘skip’ mentioned earlier.

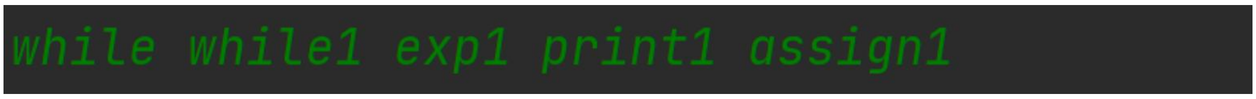


```
if if1 exp1 print1 skip1
```

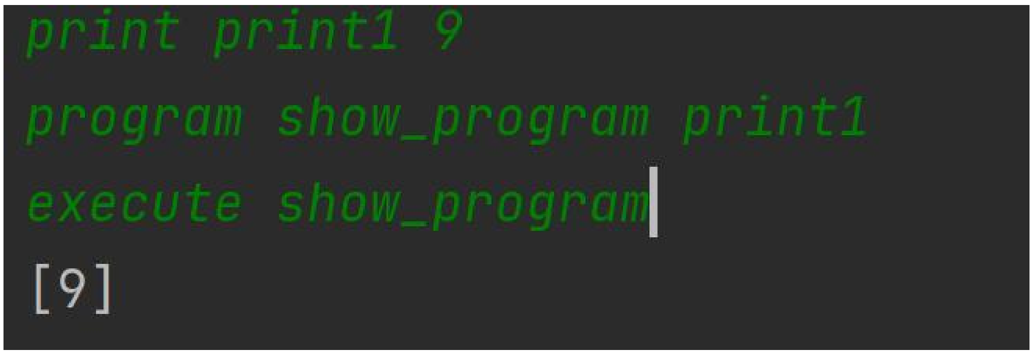


```
vardef vardef1 int x 11
print print1 x
binexpr exp1 x >= 10
skip skip1
if if1 exp1 print1 skip1
block block1 vardef1 if1
program show_if block1
execute show_if
[11]
```

[9.while]
'while' keyword is used to perform an iteration of statements as long as the condition is true. For example, the image below shows the iteration of expression 'exp1'. It will perform 'print1' while the condition is true. Otherwise, 'assign1' will be used if the condition is
false.

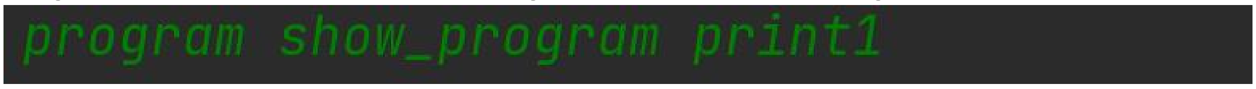


```
while while1 exp1 print1 assign1
```



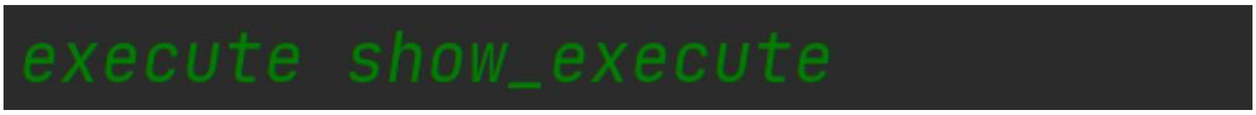
```
print print1 9
program show_program print1
execute show_program
[9]
```

[10.program]
The 'program' keyword is used to combine the statement blocks into a program. According to the image below, 'show_program' is the name of the program and 'print1' is where the program should start running.

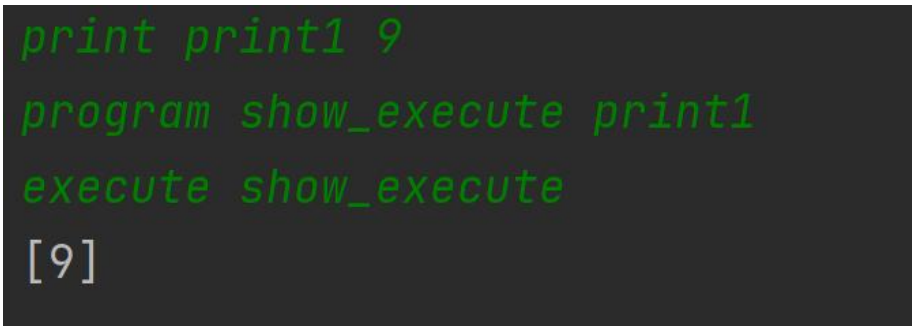


```
program show_program print1
```

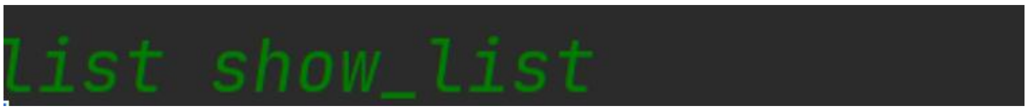
[11.execute]
'execute' keyword will be used to execute the program. This can be done as shown in the image below.

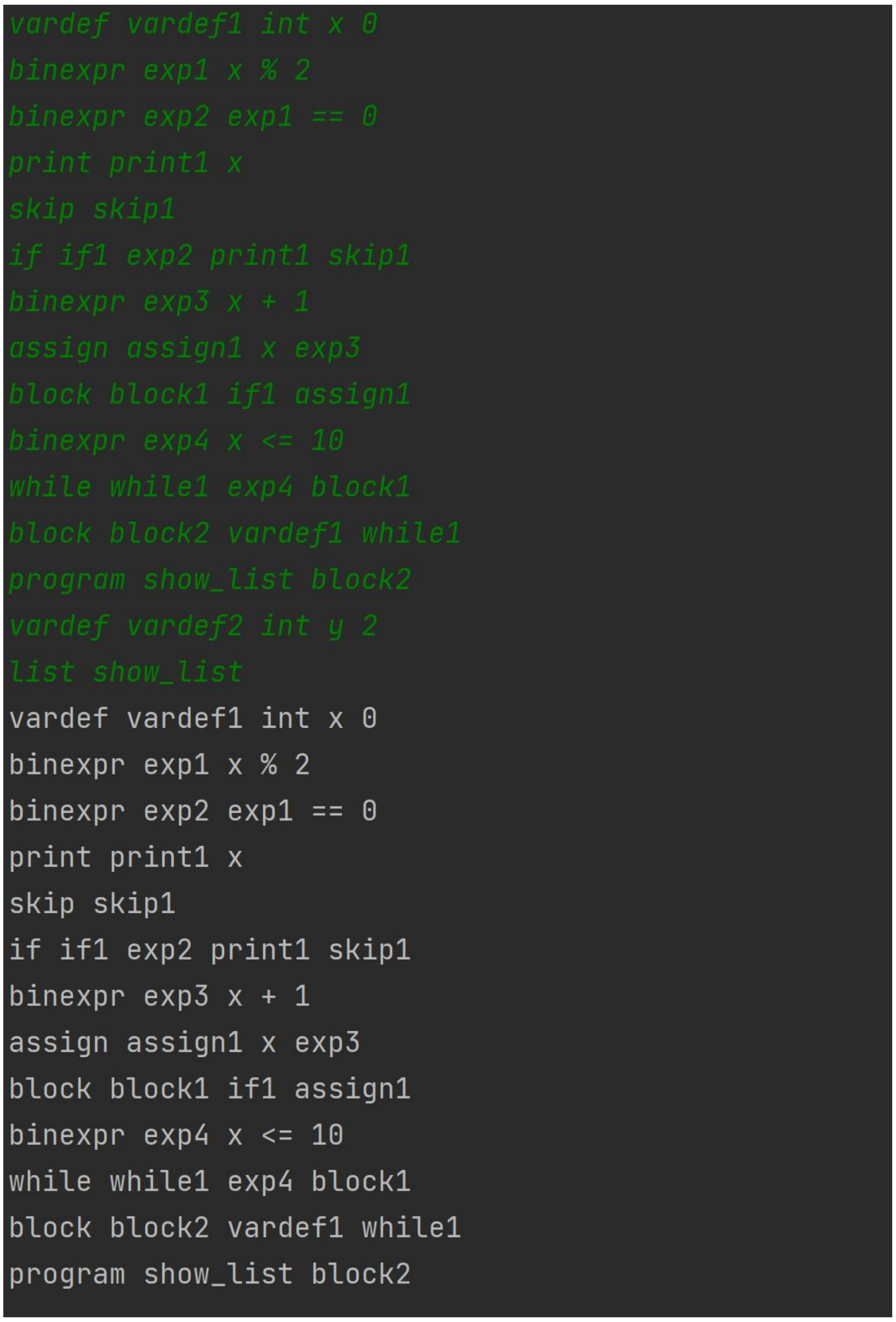


```
execute show_execute
```


```
print print1 9
program show_execute print1
execute show_execute
[9]
```

[12.list]
‘list’ keyword will output all the commands that are used in the program. For example, the commands for ‘show_list’ will be printed out once executed.

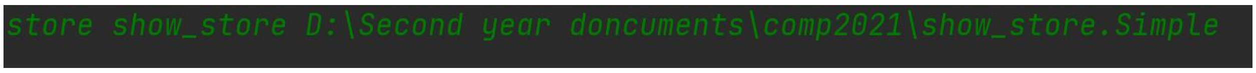

```
list show_list
```



```
vardef vardef1 int x 0
binexpr exp1 x % 2
binexpr exp2 exp1 == 0
print print1 x
skip skip1
if if1 exp2 print1 skip1
binexpr exp3 x + 1
assign assign1 x exp3
block block1 if1 assign1
binexpr exp4 x <= 10
while while1 exp4 block1
block block2 vardef1 while1
program show_list block2
vardef vardef2 int y 2
list show_list
vardef vardef1 int x 0
binexpr exp1 x % 2
binexpr exp2 exp1 == 0
print print1 x
skip skip1
if if1 exp2 print1 skip1
binexpr exp3 x + 1
assign assign1 x exp3
block block1 if1 assign1
binexpr exp4 x <= 10
while while1 exp4 block1
block block2 vardef1 while1
program show_list block2
```

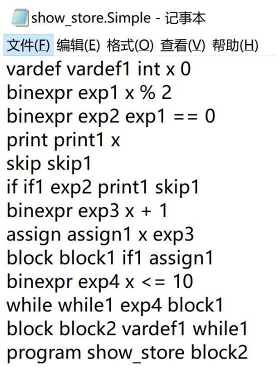
The above image shows the usage of list command,while all the unrelenting commands will be ignored.

[13.store]
'store' keyword is used to store the program as a .Simple document. As shown in the image below, 'show_store' is the name of the file, and the file will be store based on the specified location on the right.



```
store show_store D:\Second year doncuments\comp2021\show_store.Simple
```

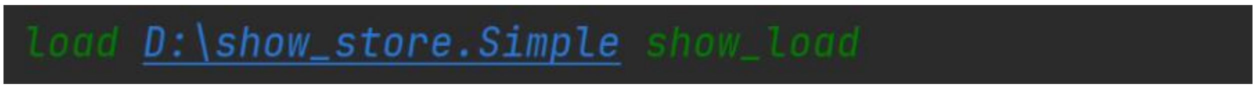
```
show_store.Simple          2022/11/18 15:58          SIMPLE 文件          1 KB
```



show_store.Simple - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

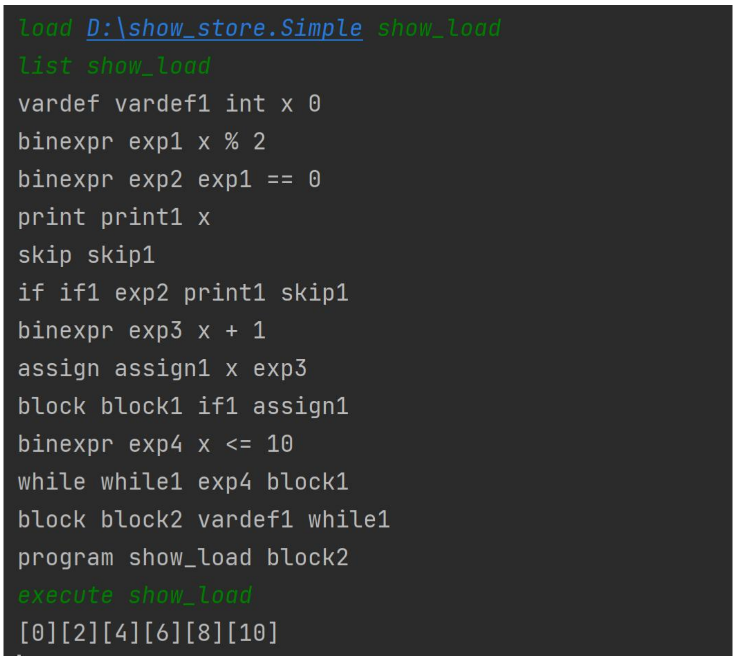
```
vardef vardef1 int x 0
binexpr exp1 x % 2
binexpr exp2 exp1 == 0
print print1 x
skip skip1
if if1 exp2 print1 skip1
binexpr exp3 x + 1
assign assign1 x exp3
block block1 if1 assign1
binexpr exp4 x <= 10
while while1 exp4 block1
block block2 vardef1 while1
program show_store block2
```

[14.load]



```
load D:\show_store.Simple show_load
```

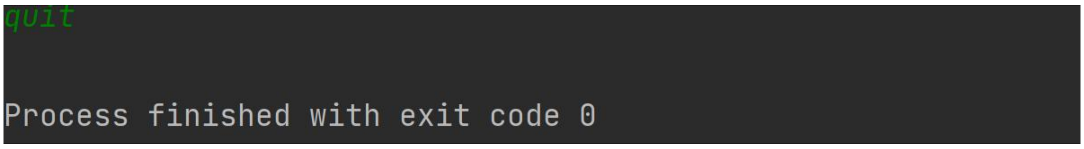
The above image displays the command for loading the saved file. It will load the file based on the entered file path (blue), 'show_load' is the keyword to run the program.



```
load D:\show_store.Simple show_load
list show_load
vardef vardef1 int x 0
binexpr exp1 x % 2
binexpr exp2 exp1 == 0
print print1 x
skip skip1
if if1 exp2 print1 skip1
binexpr exp3 x + 1
assign assign1 x exp3
block block1 if1 assign1
binexpr exp4 x <= 10
while while1 exp4 block1
block block2 vardef1 while1
program show_load block2
execute show_load
[0][2][4][6][8][10]
```

[15.quit]

'quit' keyword will be used to exit the interpreter.

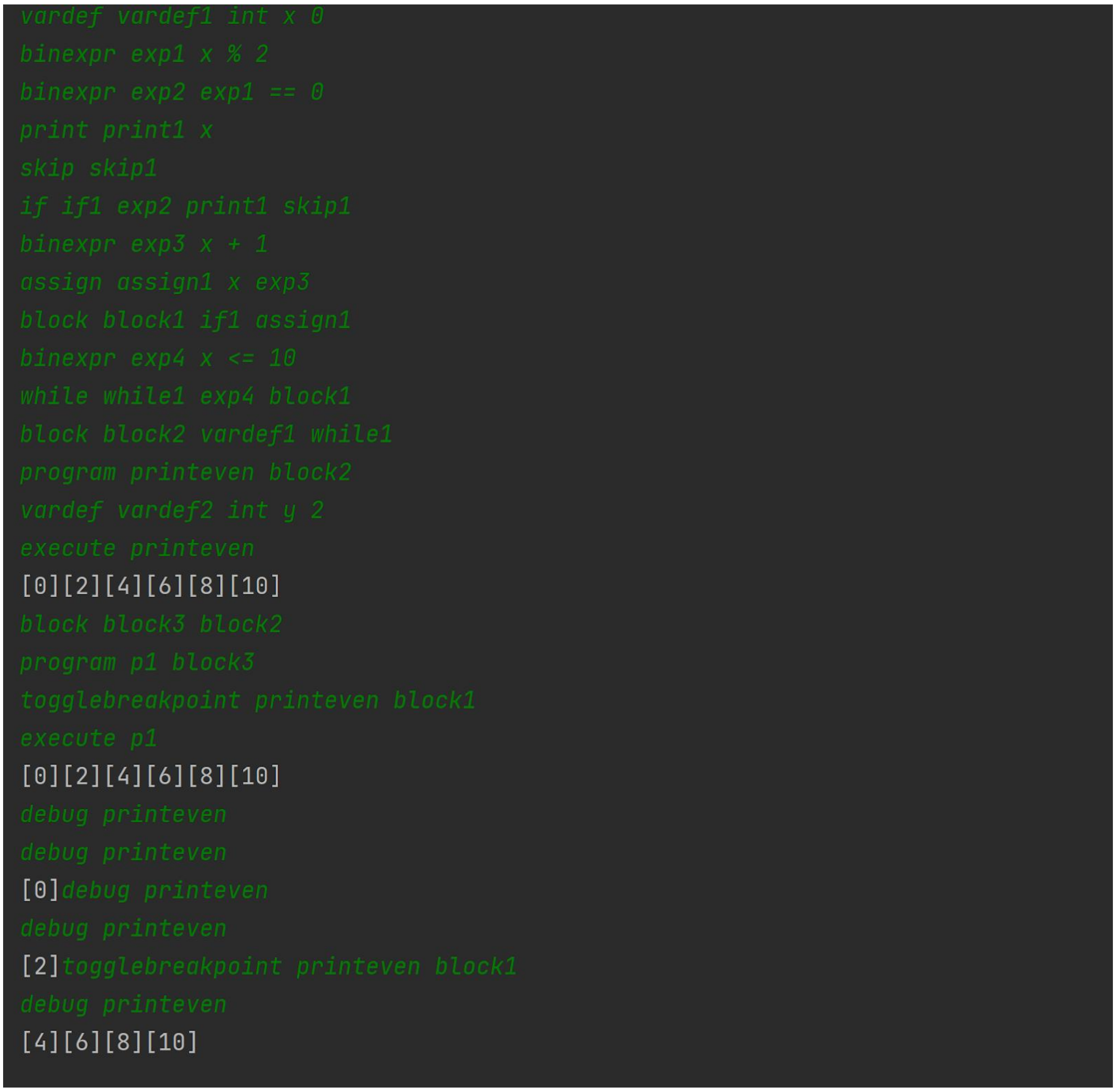


```
quit

Process finished with exit code 0
```

[bonus]

[1.debug]

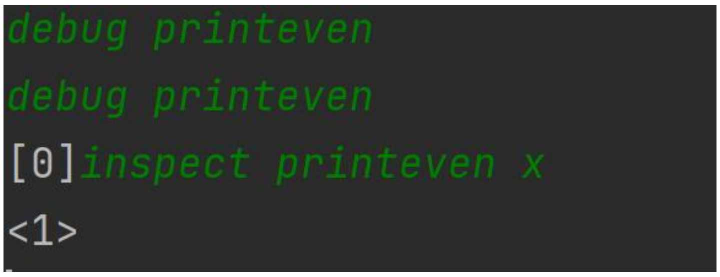


```
vardef vardef1 int x 0
binexpr exp1 x % 2
binexpr exp2 exp1 == 0
print print1 x
skip skip1
if if1 exp2 print1 skip1
binexpr exp3 x + 1
assign assign1 x exp3
block block1 if1 assign1
binexpr exp4 x <= 10
while while1 exp4 block1
block block2 vardef1 while1
program printeven block2
vardef vardef2 int y 2
execute printeven
[0][2][4][6][8][10]
block block3 block2
program p1 block3
togglebreakpoint printeven block1
execute p1
[0][2][4][6][8][10]
debug printeven
debug printeven
[0]debug printeven
debug printeven
[2]togglebreakpoint printeven block1
debug printeven
[4][6][8][10]
```

This screenshot demonstrate how to use debugger in our program:
The first output shows the normal execution of the program. For better demonstration,   block3 is created and it will store block2 inside of it, also p1 will be created to store block3. ‘P1’ will be functioned as the keyword to run the program. “togglebreakpoint printeven block1” is used to set a breakpoint at block1 for the program ‘printeven’. Program p1 will be executed and printed as normal, proving that it won’t affect the programs that share the similar keyword after setting a breakpoint in the program.

To ensure the program ‘printeven’ is in debugging mode, type the command ‘debug printeven’. The program will run and stop at the breakpoint. Otherwise, to debug the program without stopping at the breakpoint, type the command “togglebreakpoint printeven block1” and it will remove the breakpoint.

Normal execution is not allowed for programs that has breakpoint(s).

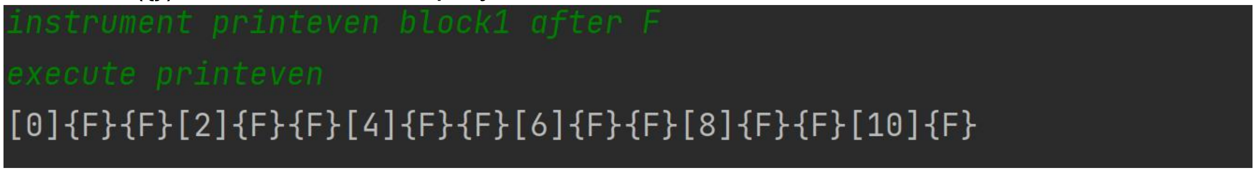


```
debug printeven
debug printeven
[0]inspect printeven x
<1>
```

The ‘inspect’ keyword can be used to check the current value of the variable when the program is in debug mode.

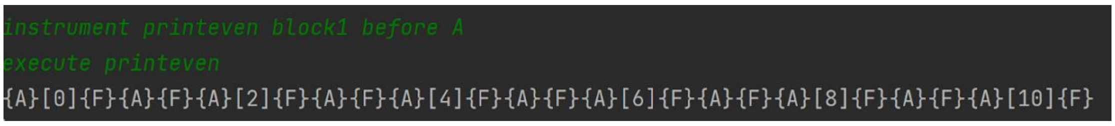
[Instrument]

‘Instrument’ keyword will print out the expression reference alongside the curly brackets ({}). Picture below display the function of ‘instrument’.



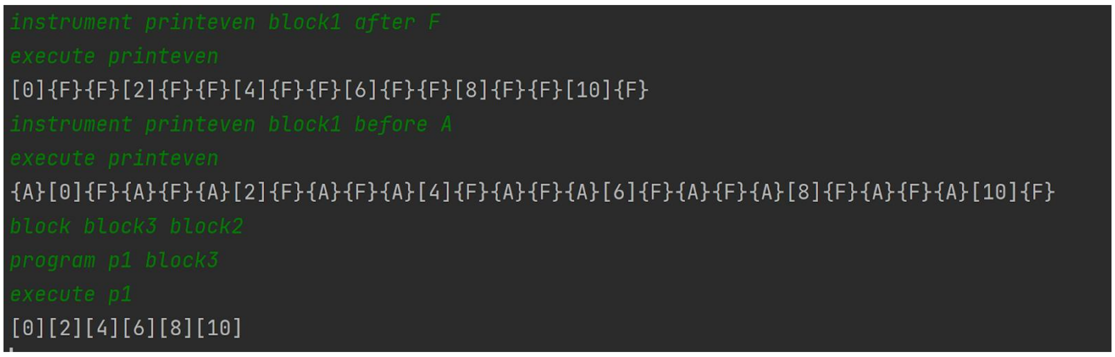
```
instrument printeven block1 after F
execute printeven
[0]{F}{F}[2]{F}{F}[4]{F}{F}[6]{F}{F}[8]{F}{F}[10]{F}
```

To use the command “instrument printeven block1 after F”, set the instrument value as F, the key word here is ‘after’, so it will print {F} after the execution of block1.



```
instrument printeven block1 before A
execute printeven
{A}[0]{F}{A}{F}{A}[2]{F}{A}{F}{A}[4]{F}{A}{F}{A}[6]{F}{A}{F}{A}[8]{F}{A}{F}{A}[10]{F}
```

Then, if we set an instrument value as ‘A’ ,it will print A before the execution of block1.



```
instrument printeven block1 after F
execute printeven
[0]{F}{F}[2]{F}{F}[4]{F}{F}[6]{F}{F}[8]{F}{F}[10]{F}
instrument printeven block1 before A
execute printeven
{A}[0]{F}{A}{F}{A}[2]{F}{A}{F}{A}[4]{F}{A}{F}{A}[6]{F}{A}{F}{A}[8]{F}{A}{F}{A}[10]{F}
block block3 block2
program p1 block3
execute p1
[0][2][4][6][8][10]
```

Still, it won’t affect the execution of the program.