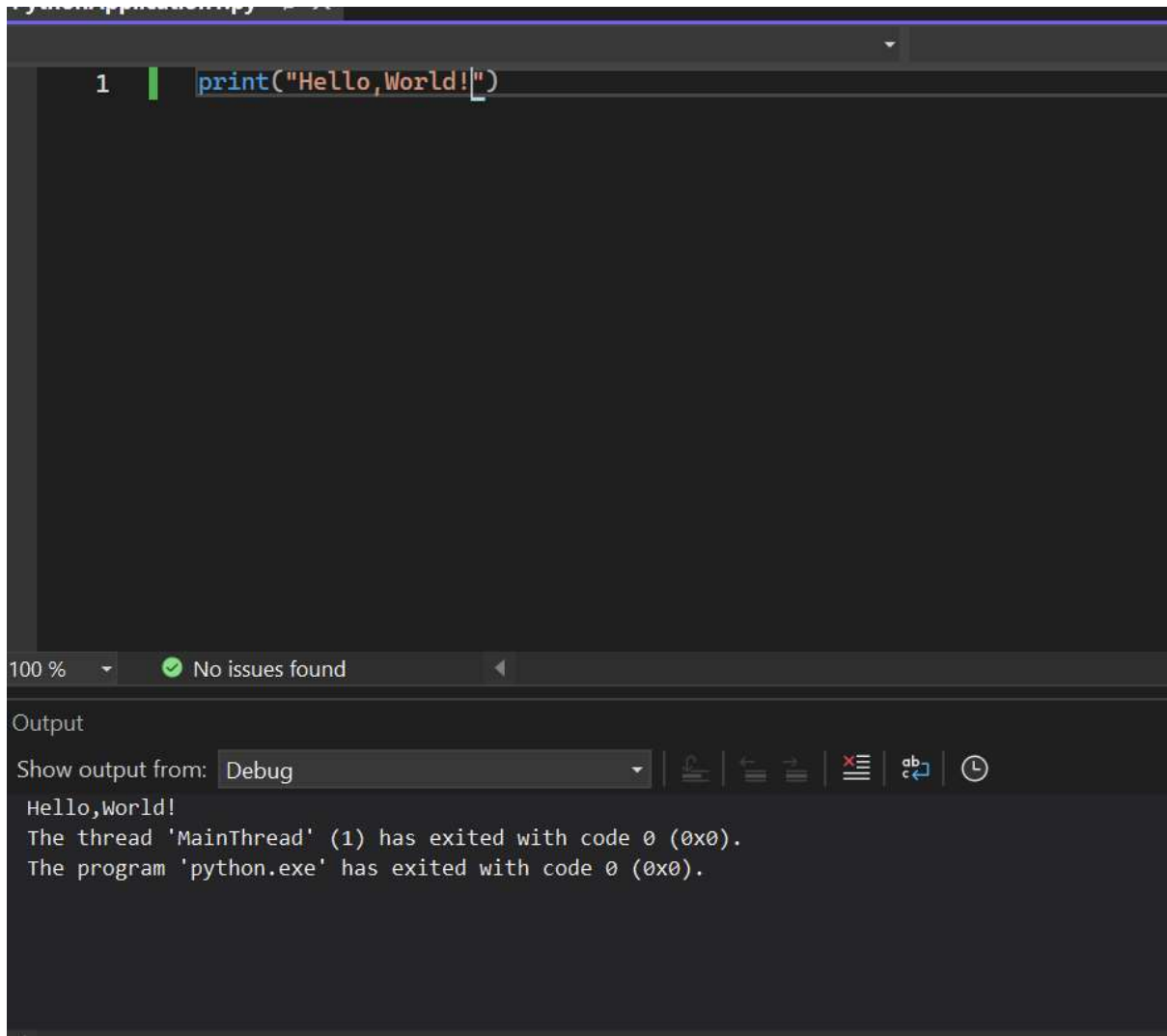


Programming Syntax:



The screenshot shows a Python IDE with a dark theme. The editor window contains a single line of code: `print("Hello,World!")`. The line number 1 is visible on the left. Below the editor, the status bar shows "100 %" and "No issues found". The Output window is open, showing the output of the code: "Hello,World!". Below the output, there are two lines of text: "The thread 'MainThread' (1) has exited with code 0 (0x0)." and "The program 'python.exe' has exited with code 0 (0x0)".

```
1 | print("Hello,World!")
```

100 % | No issues found

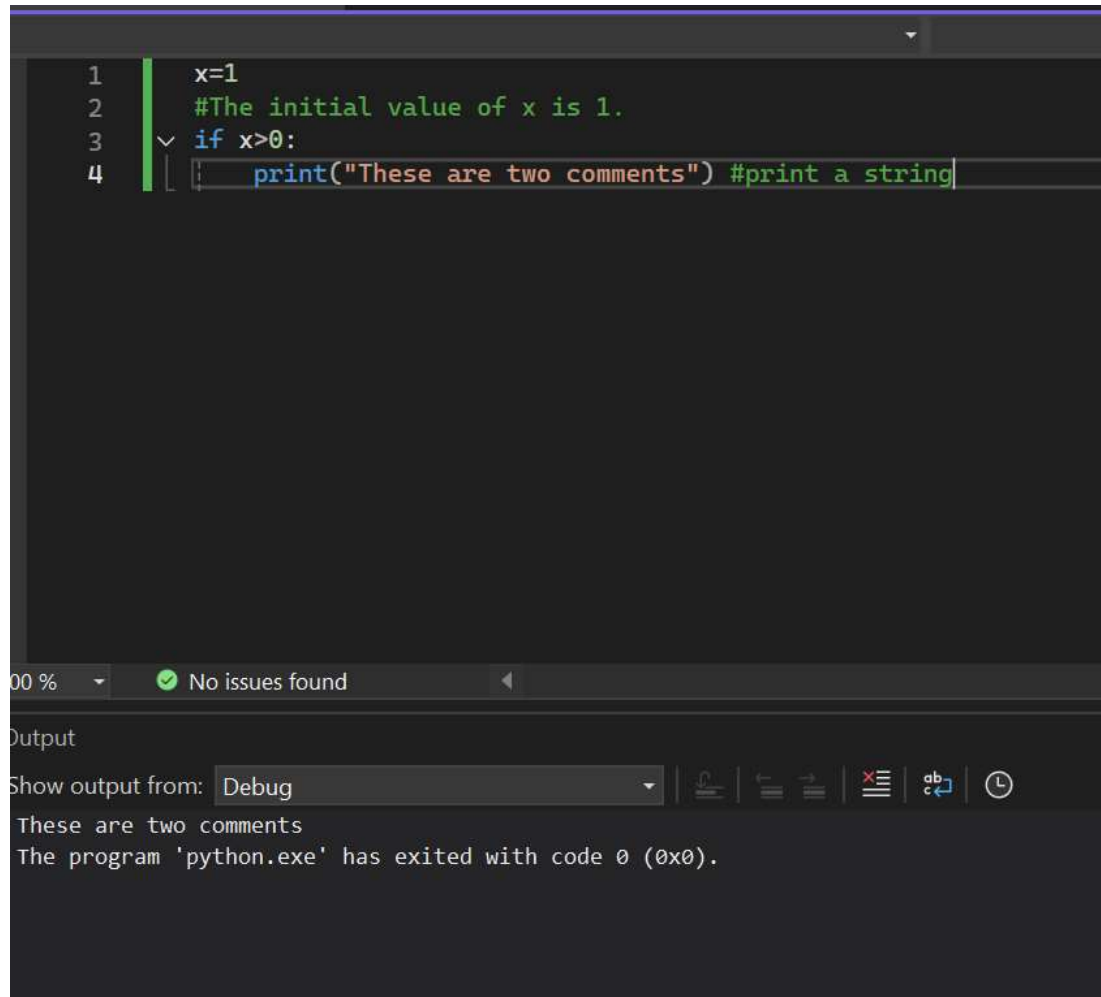
Output

Show output from: Debug

Hello,World!
The thread 'MainThread' (1) has exited with code 0 (0x0).
The program 'python.exe' has exited with code 0 (0x0).

Comments in Python:

Programming Syntax:



The screenshot shows a Python IDE with a dark theme. The editor window contains the following code:

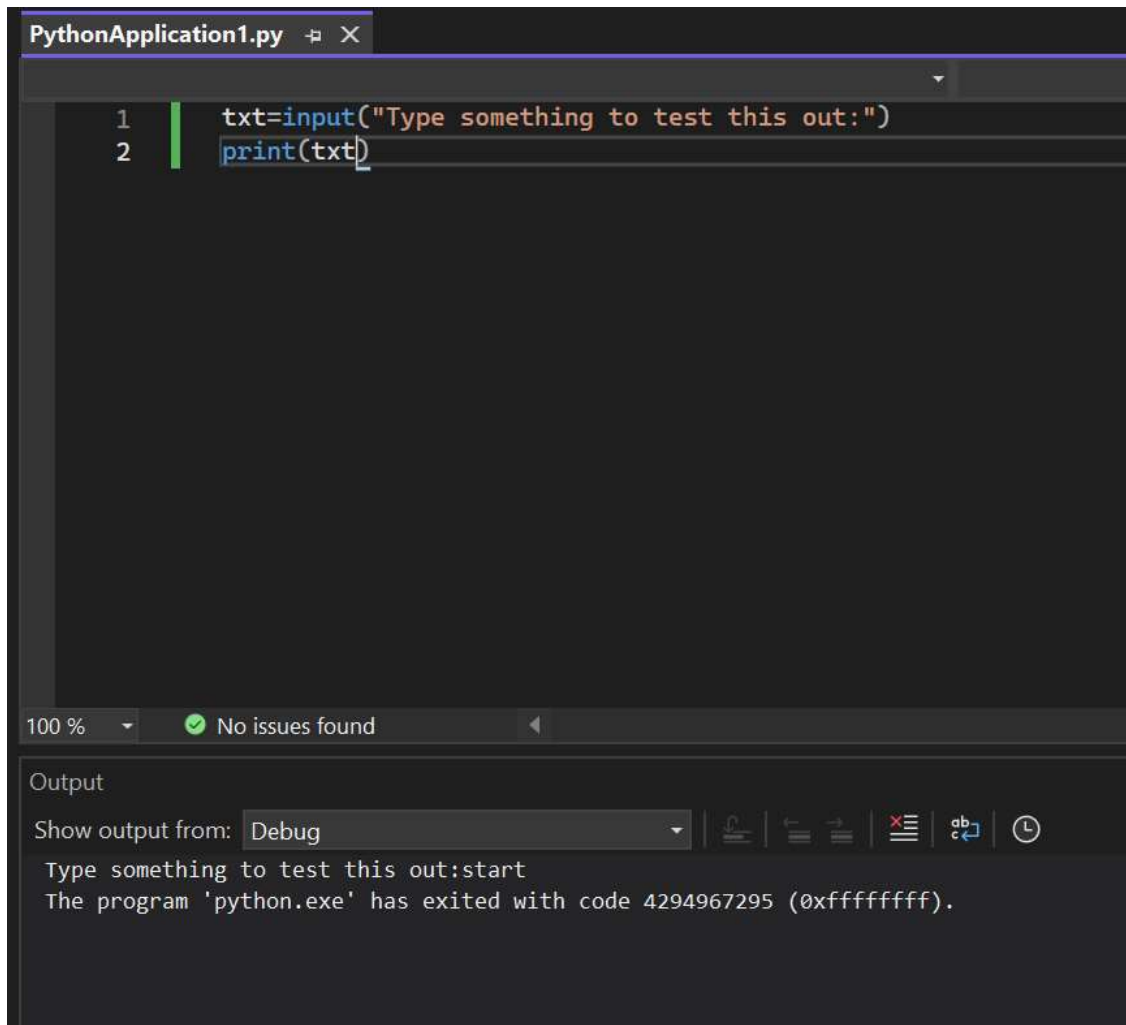
```
1 x=1
2 #The initial value of x is 1.
3 if x>0:
4     print("These are two comments") #print a string
```

Below the editor, a status bar indicates "00 %" and "No issues found". The "Output" panel is visible, showing the output of the program:

```
These are two comments
The program 'python.exe' has exited with code 0 (0x0).
```

Input/Output:

Programming Syntax:



```
PythonApplication1.py  + X
```

```
1  txt=input("Type something to test this out:")
2  print(txt)
```

100 % ✓ No issues found

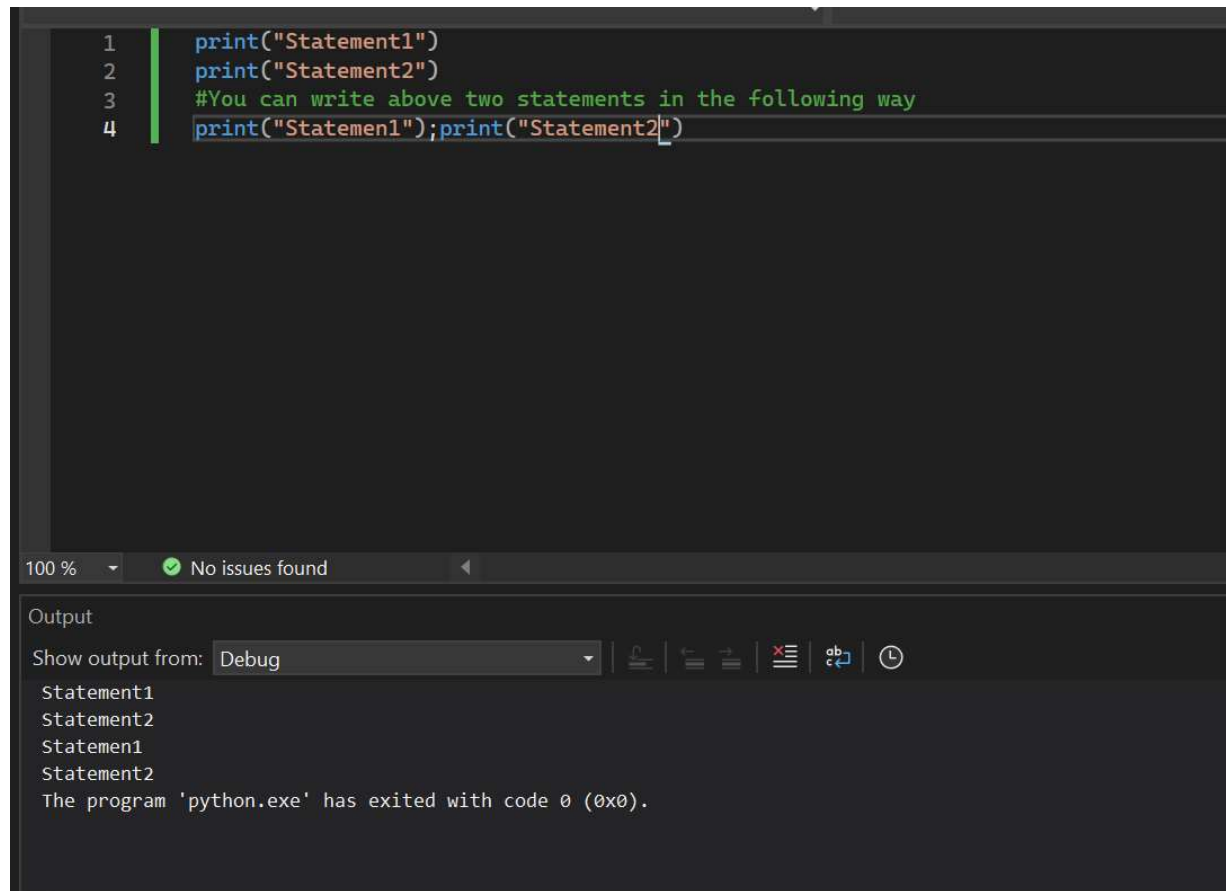
Output

Show output from: Debug

Type something to test this out:start
The program 'python.exe' has exited with code 4294967295 (0xffffffff).

Multiple Statements on a Single Line:

Programming Syntax:



The screenshot shows a Python IDE with a dark theme. The editor window contains four lines of code: `print("Statement1")`, `print("Statement2")`, a comment `#You can write above two statements in the following way`, and `print("Statemen1");print("Statement2")`. The status bar at the bottom indicates '100 %' zoom and 'No issues found'. The output console shows the results of the execution: 'Statement1', 'Statement2', 'Statemen1', 'Statement2', and a message stating 'The program 'python.exe' has exited with code 0 (0x0)'.

```
1 print("Statement1")
2 print("Statement2")
3 #You can write above two statements in the following way
4 print("Statemen1");print("Statement2")
```

100 % No issues found

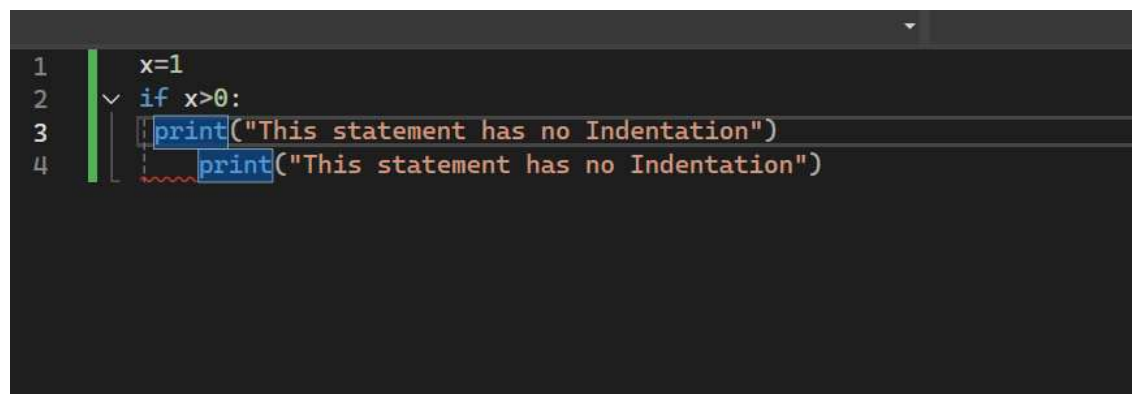
Output

Show output from: Debug

Statement1
Statement2
Statemen1
Statement2
The program 'python.exe' has exited with code 0 (0x0).

Indentation:

Input:



The screenshot shows a Python IDE with a dark theme. The editor window contains four lines of code: `x=1`, `if x>0:`, `print("This statement has no Indentation")`, and `print("This statement has no Indentation")`. The code is syntactically incorrect due to missing indentation for the print statements. The IDE highlights the errors with red squiggly lines under the print statements.

```
1 x=1
2 if x>0:
3     print("This statement has no Indentation")
4     print("This statement has no Indentation")
```

Output:

Programming Syntax:

```
File "c:\program files\microsoft visual studio\2022\community\common7\ide\extensions\microsoft\python\core\debugpy\launcher\...\debugpy\server\cli.py", line 501, in main
    run()
    ~~~~
File "c:\program files\microsoft visual studio\2022\community\common7\ide\extensions\microsoft\python\core\debugpy\launcher\...\debugpy\server\cli.py", line 351, in run_file
    runpy.run_path(target, run_name="__main__")
    ~~~~~~
File "c:\program files\microsoft visual studio\2022\community\common7\ide\extensions\microsoft\python\core\debugpy\venv\pydevd\pydevd_bundle\pydevd_runpy.py", line 309, in run_path
    code, fname = _get_code_from_file(run_name, path_name)
    ~~~~~~
File "c:\program files\microsoft visual studio\2022\community\common7\ide\extensions\microsoft\python\core\debugpy\venv\pydevd\pydevd_bundle\pydevd_runpy.py", line 283, in _get_code_from_file
    code = compile(f.read(), fname, "exec")
    ~~~~~~
File "C:\Users\Barina\source\repos\PythonApplication1\PythonApplication1\PythonApplication1.py", line 4
    print("This statement has no Indentation")
IndentationError: unexpected indent
Press any key to continue . . .
```

Single Space Indentation:

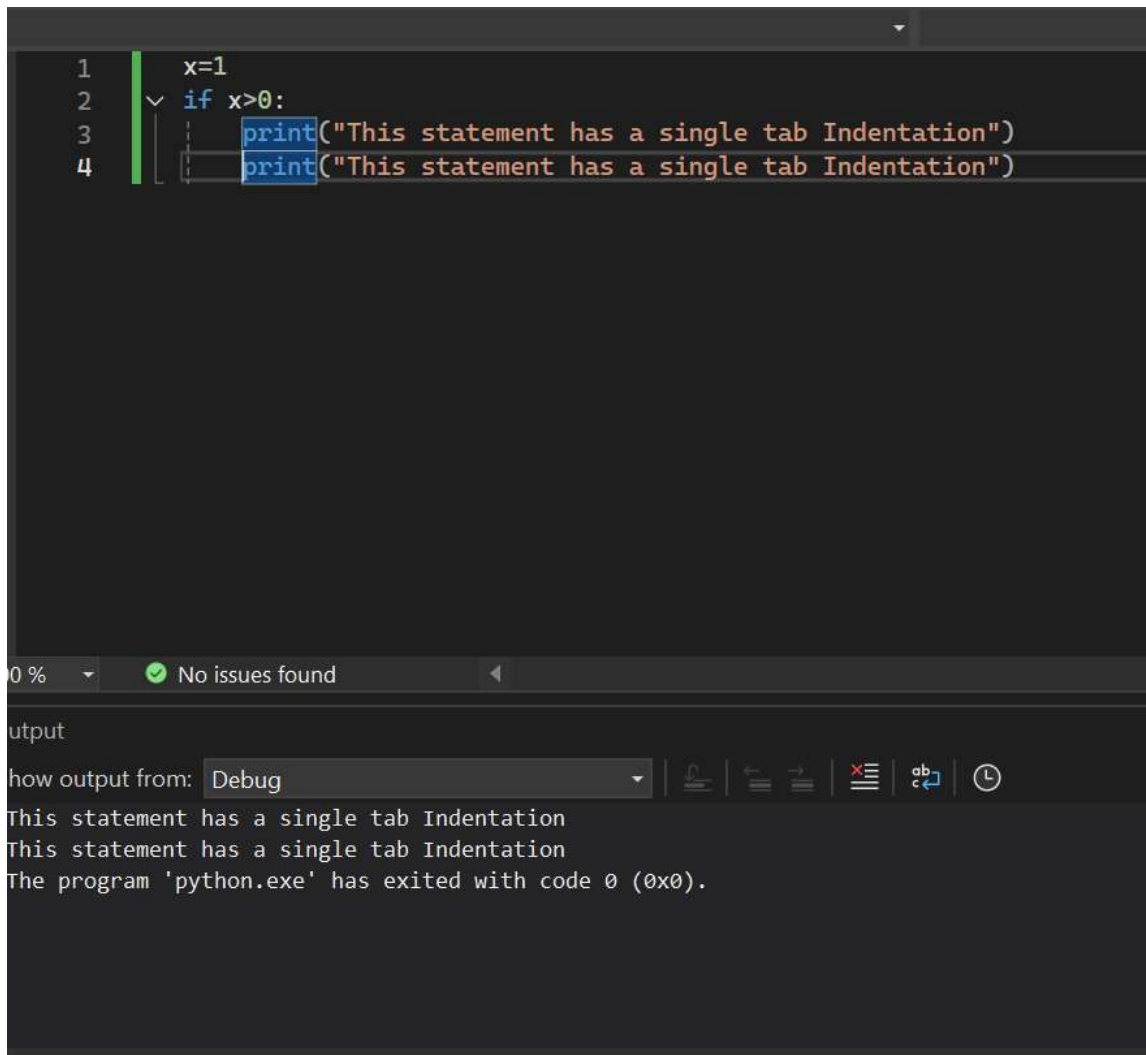
```
PythonApplication1.py  X
1  x=1
2  if x>0:
3      print("This statement has a single space Indentation")
4      print("This statement has a single space Indentation")

100 %  No issues found

Output
Show output from:  Debug
This statement has a single space Indentation
This statement has a single space Indentation
The program 'python.exe' has exited with code 0 (0x0).
```

Single Tab Indentation:

Programming Syntax:



The screenshot shows a Python IDE with a dark theme. The editor window contains the following code:

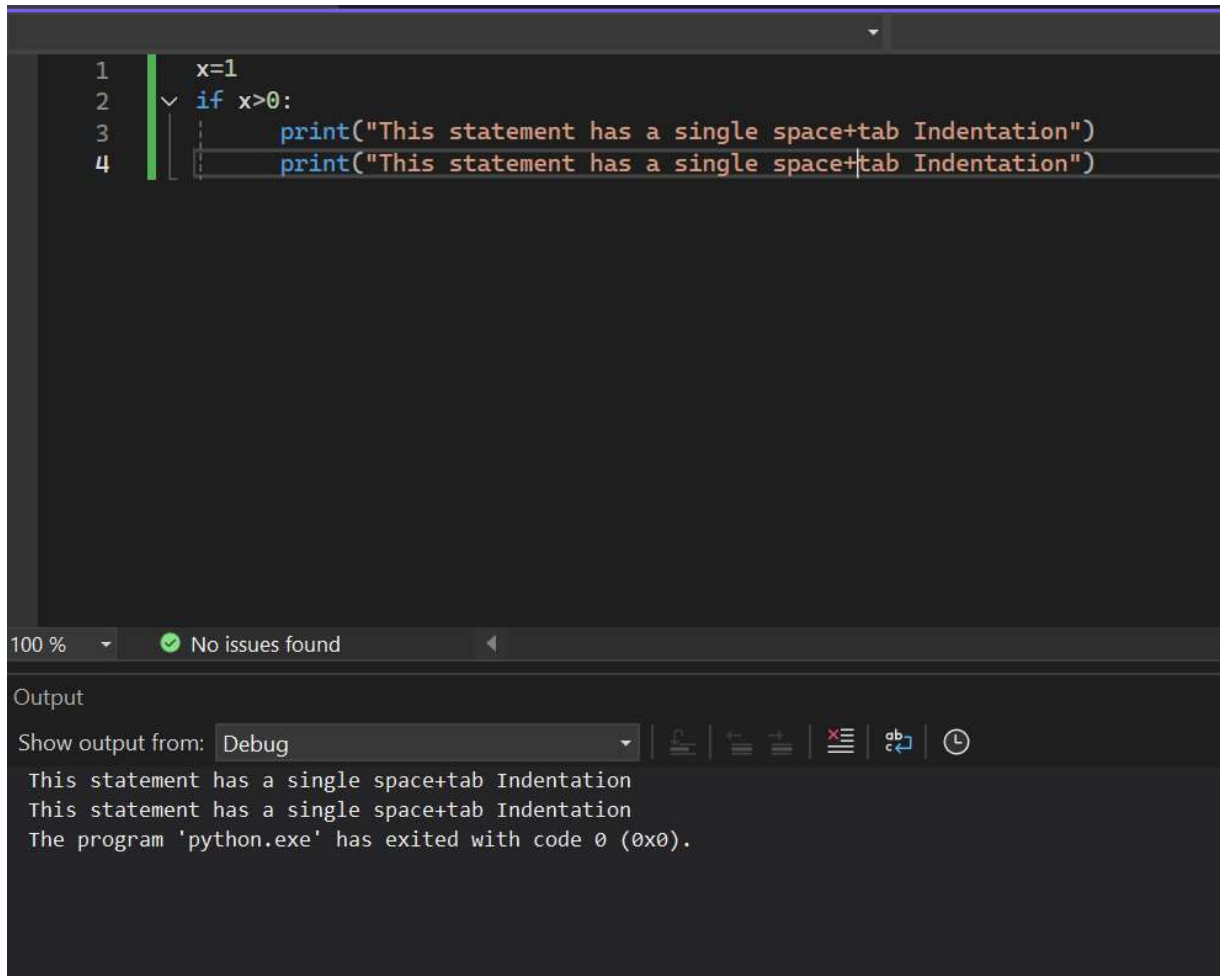
```
1 x=1
2 if x>0:
3     print("This statement has a single tab Indentation")
4     print("This statement has a single tab Indentation")
```

The code is syntactically correct, as indicated by the status bar at the bottom of the editor which says "No issues found". Below the editor is an output window titled "output". It shows the output of the program:

```
how output from: Debug
This statement has a single tab Indentation
This statement has a single tab Indentation
The program 'python.exe' has exited with code 0 (0x0).
```

Single space + Single tab Indentation:

Programming Syntax:



The screenshot shows a Python IDE with a dark theme. The editor window contains the following code:

```
1 x=1
2 if x>0:
3     print("This statement has a single space+tab Indentation")
4     print("This statement has a single space+tab Indentation")
```

Below the editor, the status bar shows "100 %" and "No issues found". The output window is titled "Output" and shows the following text:

```
Show output from: Debug
This statement has a single space+tab Indentation
This statement has a single space+tab Indentation
The program 'python.exe' has exited with code 0 (0x0).
```

Data types and Type casting:

Numbers:

Programming Syntax:

```
1  a=1452
2  type(a)
3  <class 'int'>
4  b=(-4587)
5  type(b)
6  <class 'int'>
7  c=0
8  type(c)
9  <class 'int'>
10 g=1.03
11 type(g)
12 <class 'float'>
13 h=-11.23
14 type(h)
15 <class 'float'>
16 i=.34
17 type(i)
18 <class 'float'>
```

```
26 type(x)
27 <class 'complex'>
28 print(x)
29 (1+2j)
30 z=1+2j
31 type(z)
32 <class 'complex'>
33 z=1+2j
34 type(z)
35 <class 'complex'>
```

Boolean(bool):

Programming Syntax:

```
1 x=True
2 type(x)
3 <class 'bool'>
4 y=False
5 type(y)
6 <class 'bool'>
```

Strings:

```
1 str1="String" #Strings start and end with double quotes
2 print(str1)
3
4 str2='String' #Strings start and end with single quotes
5 print(str2)
6
7 str3="String' #Strings start with double quote and end with single quote
8 #SyntaxError
9
10 str4='String" #Strings start with single quote and end with double quote
11 #SyntaxError
12
13 str5="Day's" #Single quote within double quotes
14 print(str5)
15
16 str6='Days"s' #Double quote within single quotes
```

Special characters in strings:

Programming Syntax:

```
1 print("This is a backslash(\) mark.")
2 #This is a backslash (\) mark.
3 print("This is tab \t key")
4 #This is tab \t key.
5 print("These are \'single quotes\'")
6 #These are 'single quotes'.
7 print("These are \"double quotes\"")
8 print("Thus is a new line\nNew line")
9 #This is a new line.
10 #New line
```

String indices and accessing string elements:

```
PythonApplication1.py*  X
1 string1="PYTHON TUTORIAL"
2 print(string1[0]) #Print first character(P)
3 print(string1[-15]) #Print first character(P)
4 print(string1[14]) #Print last character(L)
5 print(string1[-1]) #Print last character(L)
6 print(string1[4]) #Print 4th character(O)
7 print(string1[-11]) #Print 4th character(O)
8 print(string1[16]) #Out of index page
9
```

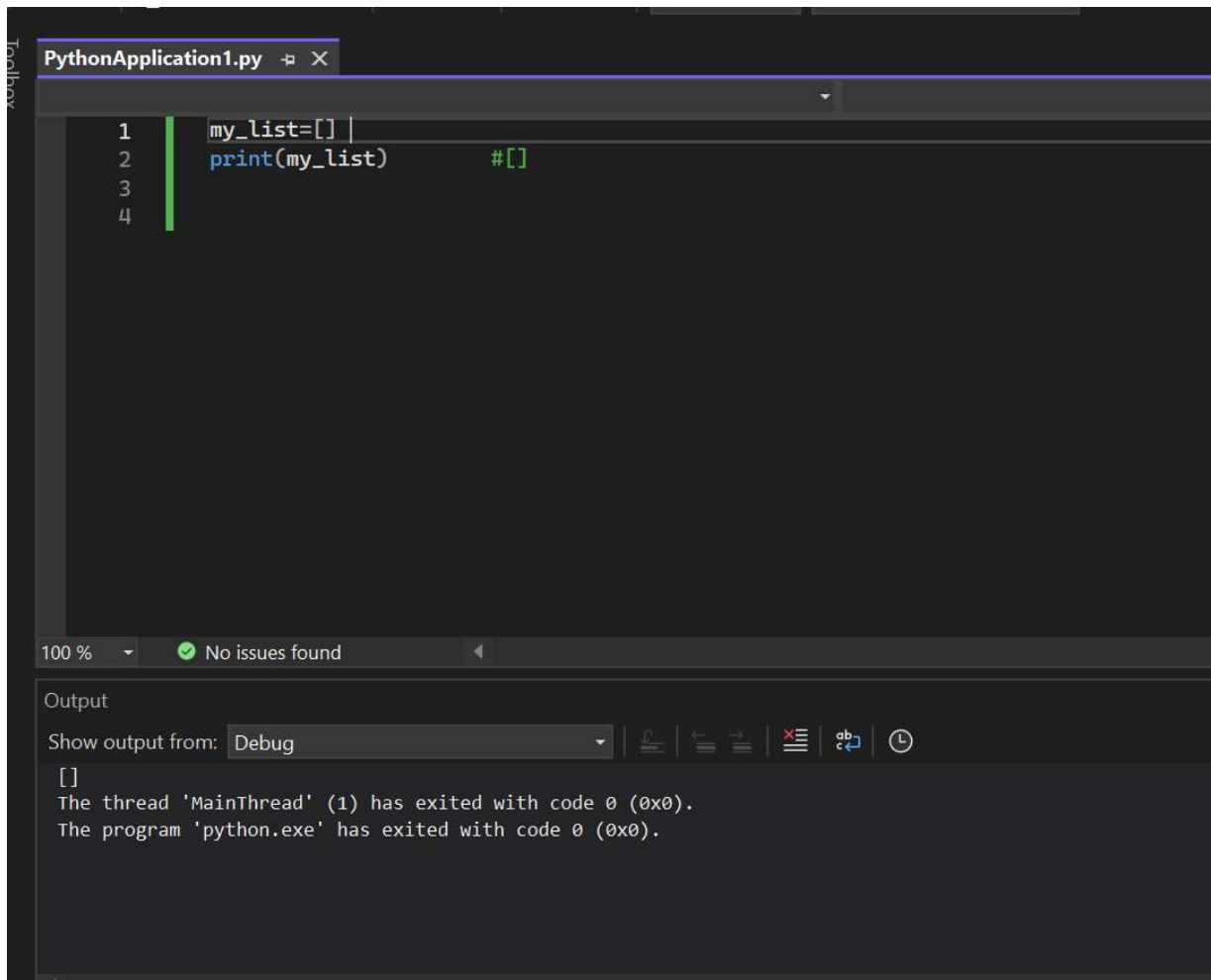
LISTS:

Creating Lists:

```
PythonApplication1.py*  X
1 my_list=[5,12,13,14] #The list contains all integer values
2 print(my_list)      #[5,12,13,14]
3
4 my_list2=['red','blue','black','white'] #the list contain all string values
5 print(my_list2)     #[red', 'blue', 'black', 'white']
6
7 my_list3=['red',12,112.12] #the list contains a string, an integer and a float value
8 print(my_list3)     #[red', 12, 112.12]
```

Programming Syntax:

List Indices:

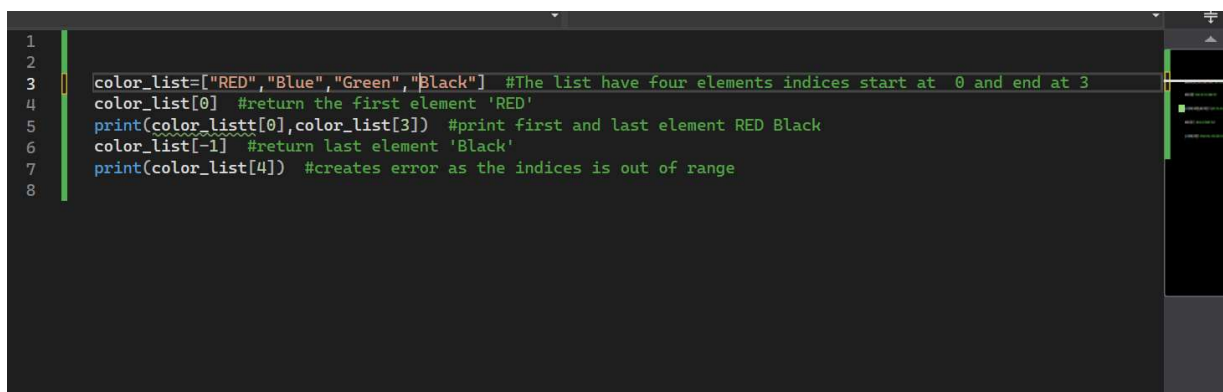


The screenshot shows a Python IDE window titled "PythonApplication1.py". The code editor contains the following Python code:

```
1 my_list=[]  
2 print(my_list) #[]  
3  
4
```

Below the code editor, the "Output" panel is visible, showing the execution results:

```
[]  
The thread 'MainThread' (1) has exited with code 0 (0x0).  
The program 'python.exe' has exited with code 0 (0x0).
```

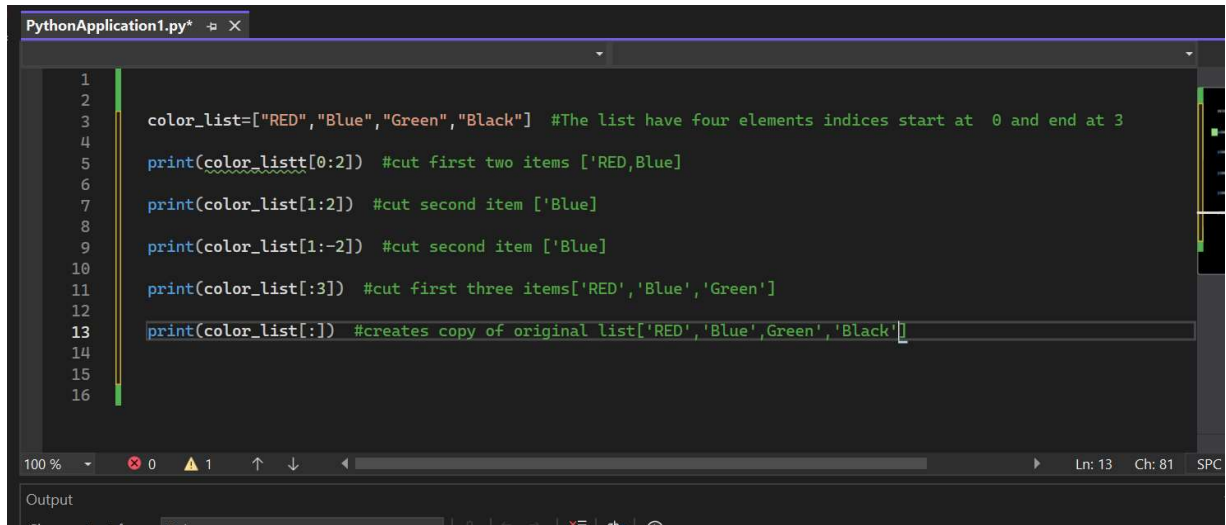


The screenshot shows a Python IDE window with the following Python code:

```
1  
2  
3 color_list=["RED","Blue","Green","Black"] #The list have four elements indices start at 0 and end at 3  
4 color_list[0] #return the first element 'RED'  
5 print(color_list[0],color_list[3]) #print first and last element RED Black  
6 color_list[-1] #return last element 'Black'  
7 print(color_list[4]) #creates error as the indices is out of range  
8
```

List Slice:

Programming Syntax:



The screenshot shows a Python IDE window titled "PythonApplication1.py*". The code is as follows:

```
1
2
3 color_list=["RED","Blue","Green","Black"] #The list have four elements indices start at 0 and end at 3
4
5 print(color_listt[0:2]) #cut first two items ['RED,Blue]
6
7 print(color_list[1:2]) #cut second item ['Blue]
8
9 print(color_list[1:-2]) #cut second item ['Blue]
10
11 print(color_list[:3]) #cut first three items['RED','Blue','Green']
12
13 print(color_list[:]) #creates copy of original list['RED','Blue','Green','Black']
14
15
16
```

The IDE interface includes a status bar at the bottom showing "100 %", "0" errors, "1" warning, and a scrollbar. The output pane at the bottom is labeled "Output".