Lab 1

Introduction to AI and Its Application Using Python

Python is widely used for **artificial intelligence**, with packages for a number of applications including General **AI**, Machine Learning, Natural **Language** Processing and Neural Networks. Haskell is also a very good programming **language** for **AI**. Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. It was created by Guido van Rossum during 1985- 1990. Like Perl, Python source code is also available under the GNU General Public License (GPL).

Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.

Python is Interactive − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python is Object-Oriented − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

Python is a Beginner's Language − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

Programming syntax

Execute Python Syntax

- As we learned in the previous page, Python syntax can be executed by writing directly in the Command Line:

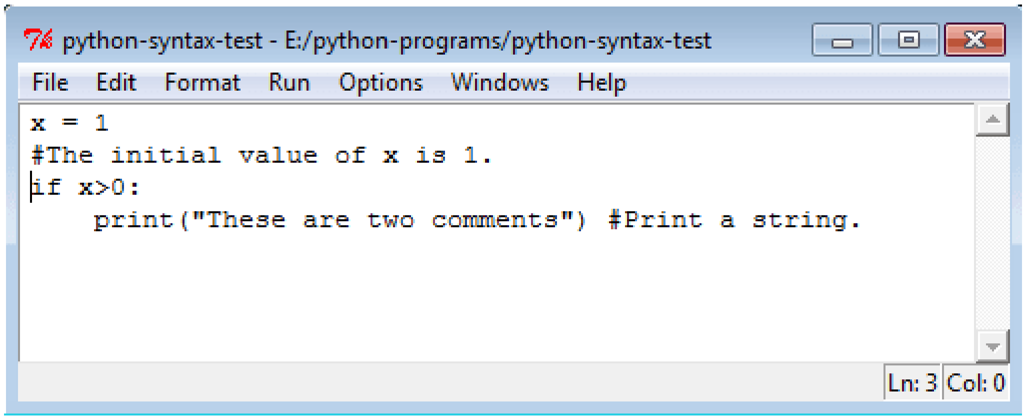
-
  ```
  >>> print("Hello, World!")
  Hello, World!
  ```
- Or by creating a python file on the server, using the .py file extension, and running it in the Command Line:

-
  ```
  C:\Users\Your Name>python myfile.py
  ```

A Python program is read by a parser. Python was designed to be a highly readable language. The syntax of the Python programming language is the set of rules which defines how a Python program will be written.

Comments in Python:

A comment begins with a hash character(#) which is not a part of the string literal and ends at the end of the physical line. All characters after the # character up to the end of the line are part of the comment and the Python interpreter ignores them. See the following example. It should be noted that Python has no multi-lines or block comments facility.


python-syntax-test - E:/python-programs/python-syntax-test
File Edit Format Run Options Windows Help
Ln: 3 Col: 0

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

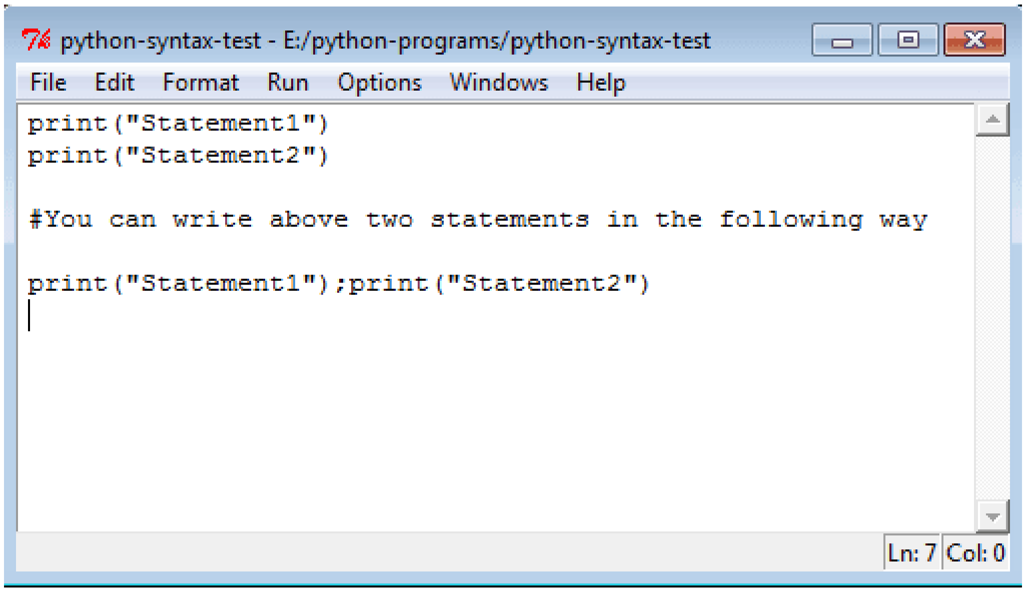
Input / output

txt = input("Type something to test this out: ")

print(txt)

Multiple Statements on a Single Line:

You can write two separate statements into a single line using a semicolon (;) character between two line.


python-syntax-test - E:/python-programs/python-syntax-test
File Edit Format Run Options Windows Help
Ln: 7 Col: 0

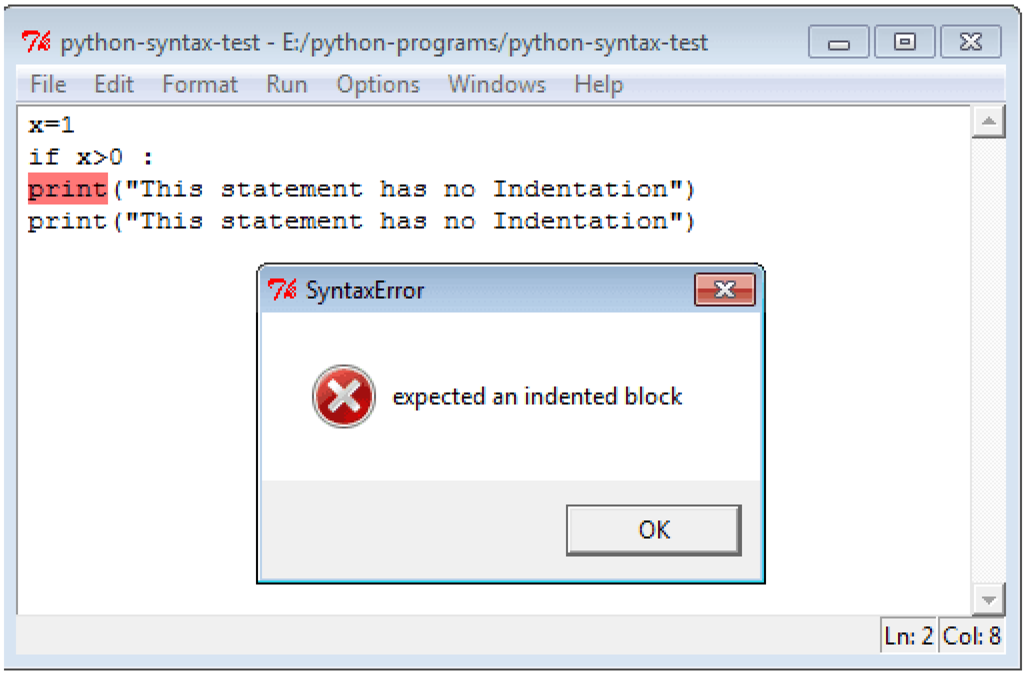
```
print("Statement1")
print("Statement2")

#You can write above two statements in the following way

print("Statement1");print("Statement2")
```

Indentation:

Python uses whitespace (spaces and tabs) to define program blocks whereas other languages like C, C++ use braces ({}) to indicate blocks of codes for class, functions or flow control. The number of whitespaces (spaces and tabs) in the indentation is not fixed, but all statements within the block must be the indented same amount. In the following program, the block statements have no indentation.

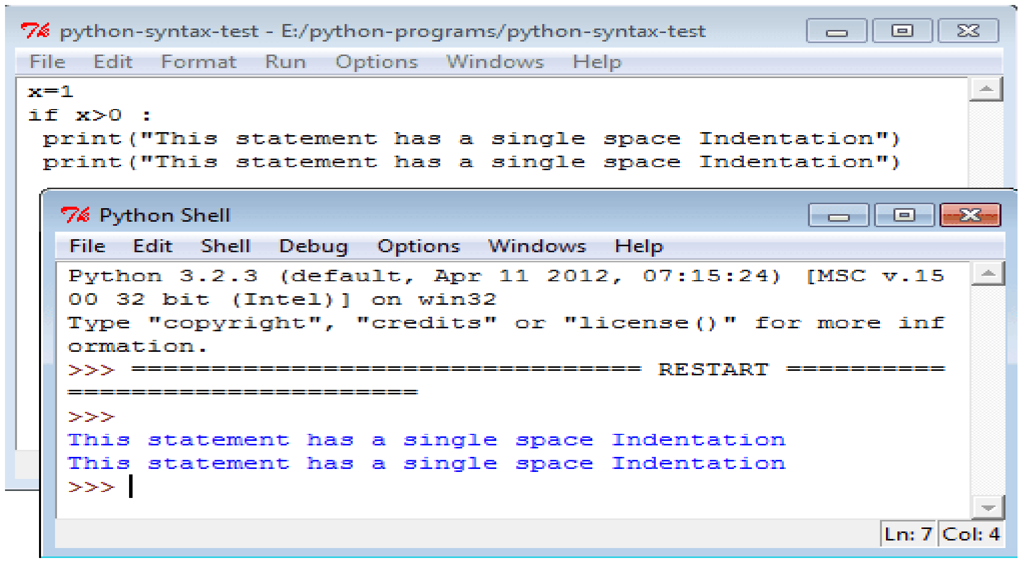


python-syntax-test - E:/python-programs/python-syntax-test
File Edit Format Run Options Windows Help
SyntaxError
OK
Ln: 2 Col: 8

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

expected an indented block

This is a program with single space indentation.

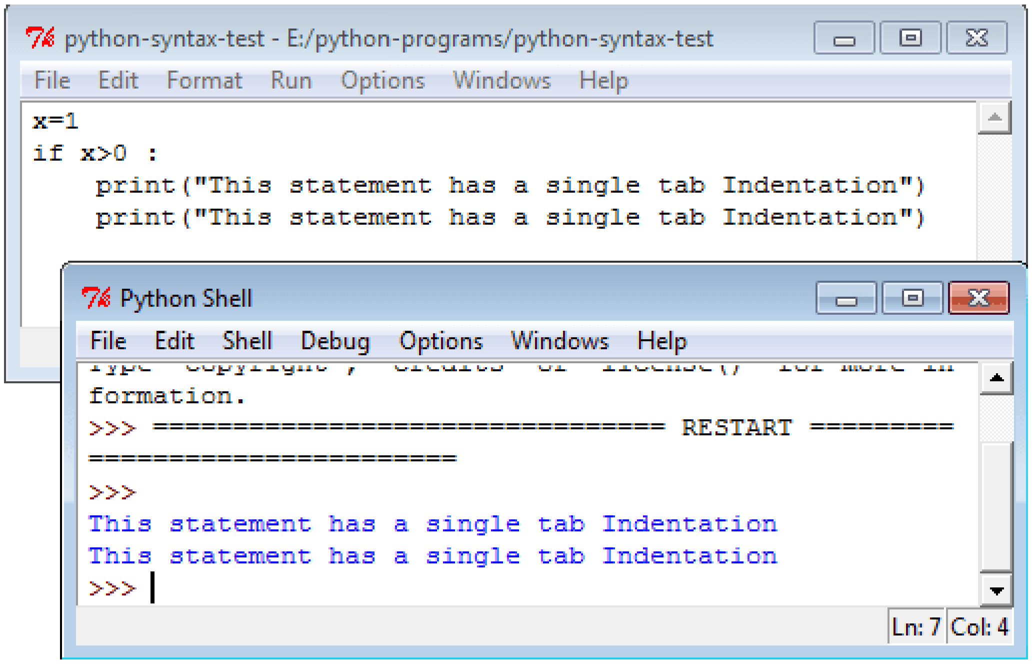


python-syntax-test - E:/python-programs/python-syntax-test
File Edit Format Run Options Windows Help
Python Shell
File Edit Shell Debug Options Windows Help
Ln: 7 Col: 4

```
x=1
if x>0 :
 print("This statement has a single space Indentation")
 print("This statement has a single space Indentation")
```

```
Python 3.2.3 (default, Apr 11 2012, 07:15:24) [MSC v.15
00 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more inf
ormation.
>>> ================================ RESTART ==========
======================
>>>
This statement has a single space Indentation
This statement has a single space Indentation
>>> 
```

This is a program with single tab indentation.

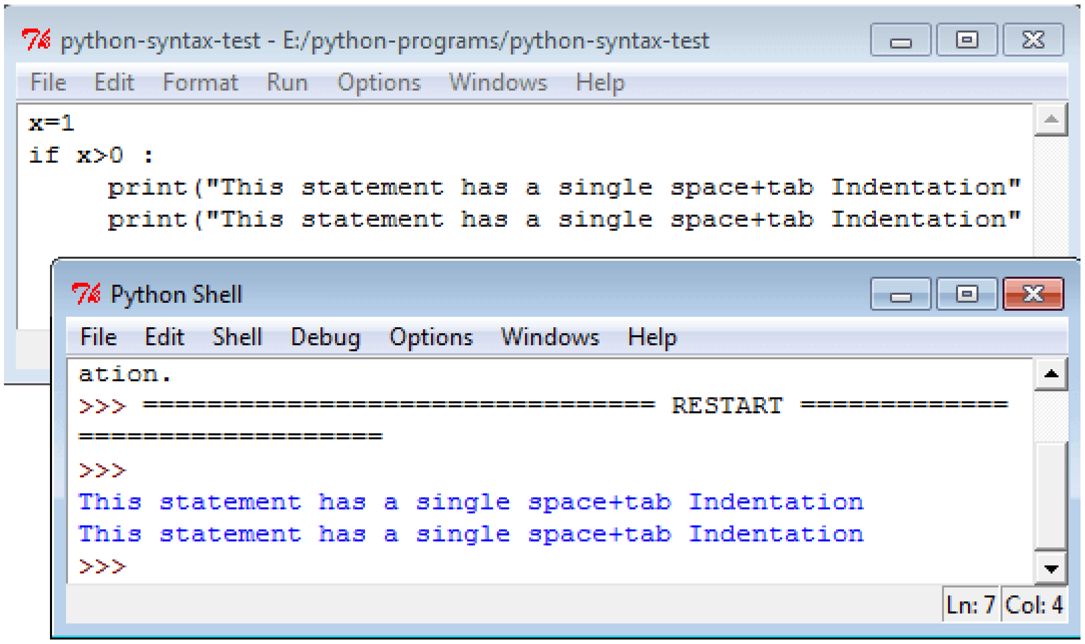

python-syntax-test - E:/python-programs/python-syntax-test
File Edit Format Run Options Windows Help

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

Python Shell
File Edit Shell Debug Options Windows Help

```
formation.
>>> ================================ RESTART =========
=======================
>>> 
This statement has a single tab Indentation
This statement has a single tab Indentation
>>> 
```

Here is an another program with an indentation of a single space + a single tab.


python-syntax-test - E:/python-programs/python-syntax-test
File Edit Format Run Options Windows Help

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

Python Shell
File Edit Shell Debug Options Windows Help

```
ation.
>>> ================================ RESTART =============
===================
>>> 
This statement has a single space+tab Indentation
This statement has a single space+tab Indentation
>>> 
```

Python Coding Style:

- Use 4 spaces per indentation and no tabs.

- Do not mix tabs and spaces. Tabs create confusion and it is recommended to use only spaces.
- Maximum line length : 79 characters which help users with a small display.

Python Reserve words:

The following identifiers are used as reserved words of the language, and cannot be used as ordinary identifiers.

False	class	finally	is	return
None	continue	for	lambda	try
True	def	from	nonlocal	while
and	del	global	not	with
as	el	if	or	yield
assert	else	import	pass	
break	except	in	raise	

- Use blank lines to separate top-level function and class definitions and single blank line to separate methods definitions inside a class and larger blocks of code inside functions.
- When possible, put inline comments (should be complete sentences).
- Use spaces around expressions and statements.

Data types and Type casting

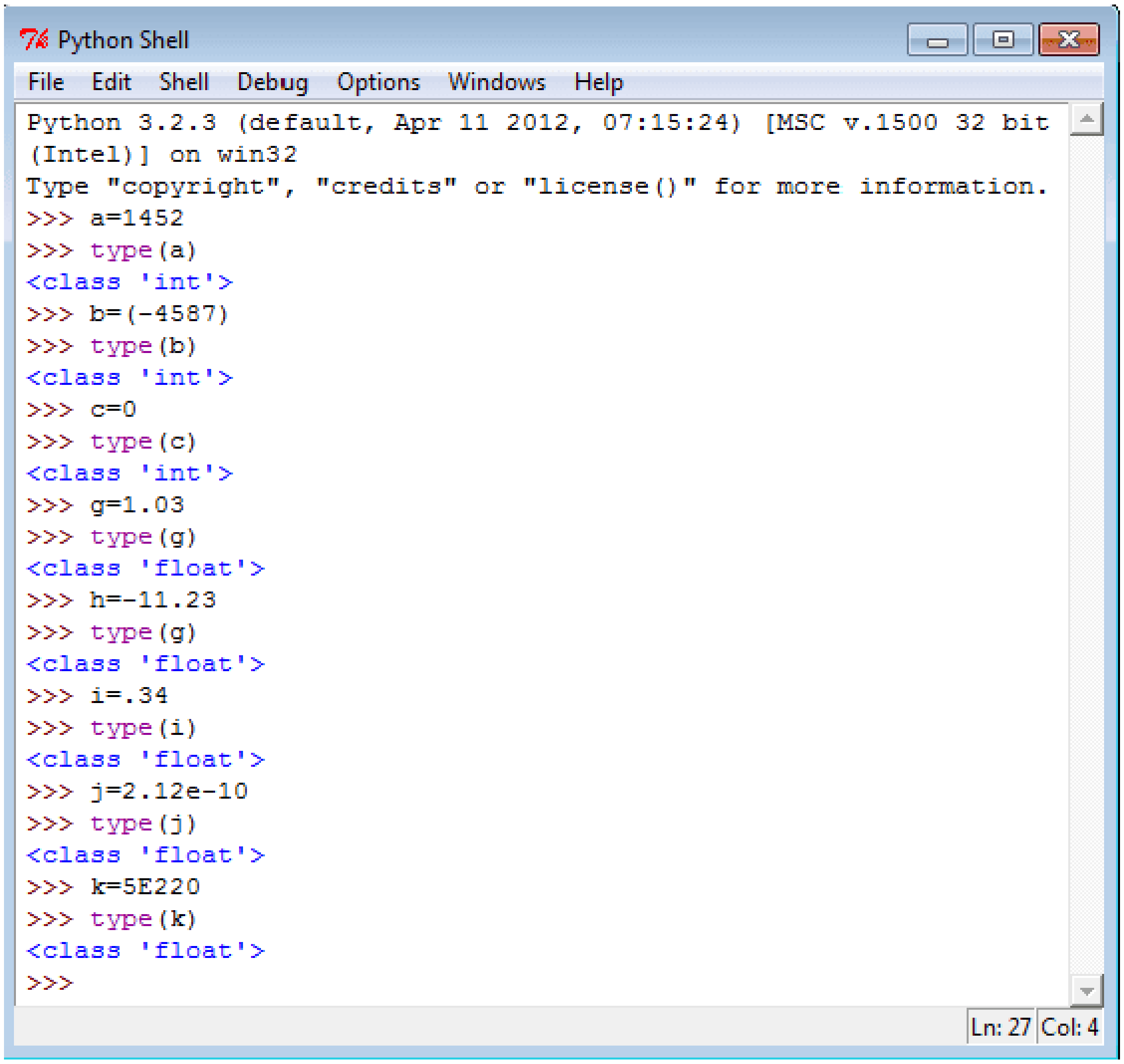
Type represents the kind of value and determines how the value can be used. All data values in Python are encapsulated in relevant object classes. Everything in Python is an object and every object has an identity, a type, and a value. Like another object-oriented language such as Java or C++, there are several data types which are built into Python. Extension modules which are written in C, Java, or other languages can define additional types.

To determine a variable's type in Python you can use the type() function. The value of some objects can be changed. Objects whose value can be changed are called mutable and objects whose value is unchangeable (once they are created) are called immutable. Here are the details of Python data types

Numbers

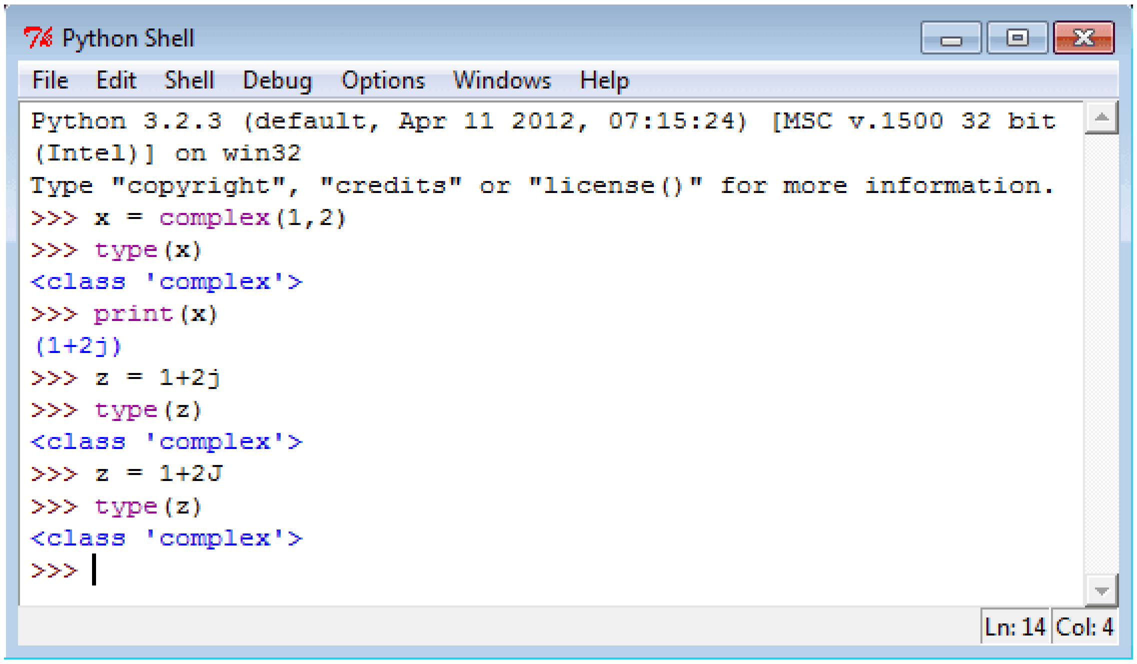
Numbers are created by numeric literals. Numeric objects are immutable, which means when an object is created its value cannot be changed.

Python has three distinct numeric types: ***integers, floating point numbers, and complex numbers***. Integers represent negative and positive integers without fractional parts whereas floating point numbers represents negative and positive numbers with fractional parts. In addition, Booleans are a subtype of plain integers. See the following statements in Python shell.



```
Python 3.2.3 (default, Apr 11 2012, 07:15:24) [MSC v.1500 32 bit
(Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> a=1452
>>> type(a)
<class 'int'>
>>> b=(-4587)
>>> type(b)
<class 'int'>
>>> c=0
>>> type(c)
<class 'int'>
>>> g=1.03
>>> type(g)
<class 'float'>
>>> h=-11.23
>>> type(g)
<class 'float'>
>>> i=.34
>>> type(i)
<class 'float'>
>>> j=2.12e-10
>>> type(j)
<class 'float'>
>>> k=5E220
>>> type(k)
<class 'float'>
>>>
```

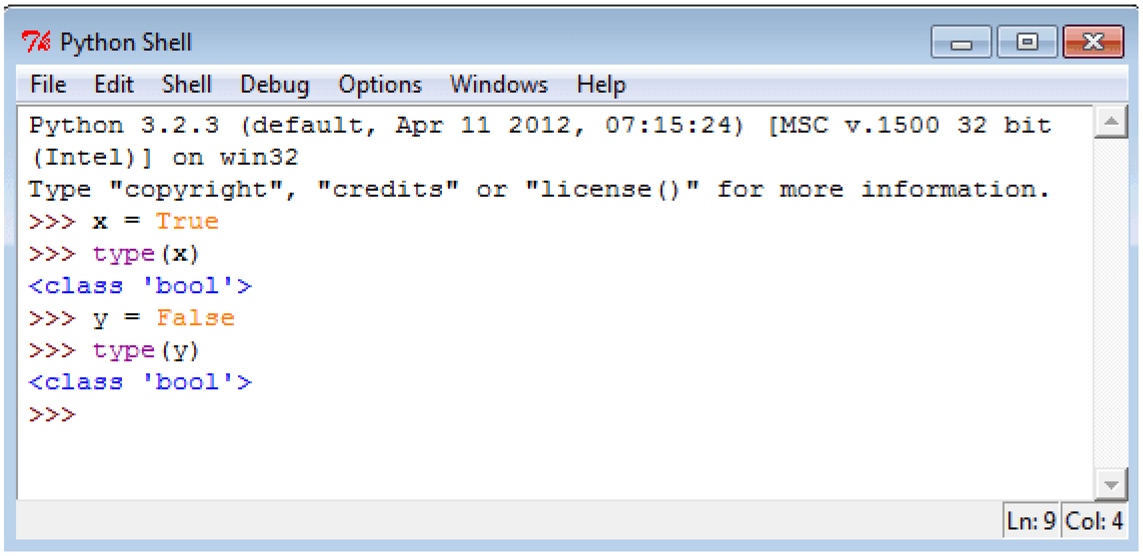
Mathematically, a complex number (generally used in engineering) is a number of the form A+Bi where i is the imaginary number. Complex numbers have a real and imaginary part. Python supports complex numbers either by specifying the number in (real + imag**J**) or (real + imag**j**) form or using a built-in method complex(x, y). See the following statements in Python Shell.



```
Python 3.2.3 (default, Apr 11 2012, 07:15:24) [MSC v.1500 32 bit
(Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> x = complex(1,2)
>>> type(x)
<class 'complex'>
>>> print(x)
(1+2j)
>>> z = 1+2j
>>> type(z)
<class 'complex'>
>>> z = 1+2J
>>> type(z)
<class 'complex'>
>>>
```

Boolean (bool)

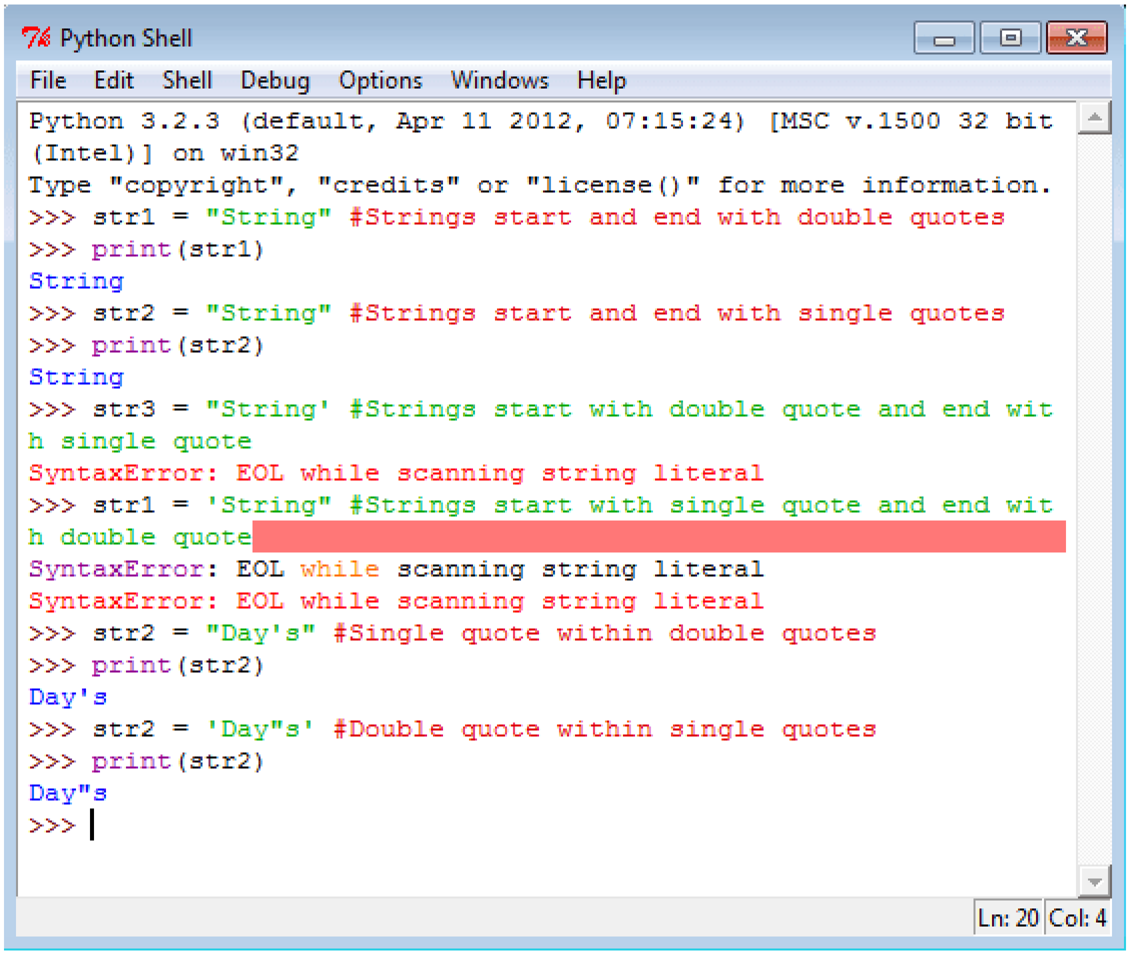
The simplest build-in type in Python is the bool type, it represents the truth values False and True. See the following statements in Python shell.



```
Python 3.2.3 (default, Apr 11 2012, 07:15:24) [MSC v.1500 32 bit
(Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> x = True
>>> type(x)
<class 'bool'>
>>> y = False
>>> type(y)
<class 'bool'>
>>>
```

Strings

In Python, a string type object is a sequence (left-to- right order) of characters. Strings start and end with single or double quotes Python strings are immutable. Single and double quoted strings are same and you can use a single quote within a string when it is surrounded by double quote and vice versa. Declaring a string is simple, see the following statements.



```
Python 3.2.3 (default, Apr 11 2012, 07:15:24) [MSC v.1500 32 bit
(Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> str1 = "String" #Strings start and end with double quotes
>>> print(str1)
String
>>> str2 = "String" #Strings start and end with single quotes
>>> print(str2)
String
>>> str3 = "String' #Strings start with double quote and end wit
h single quote
SyntaxError: EOL while scanning string literal
>>> str1 = 'String" #Strings start with single quote and end wit
h double quote
SyntaxError: EOL while scanning string literal
SyntaxError: EOL while scanning string literal
>>> str2 = "Day's" #Single quote within double quotes
>>> print(str2)
Day's
>>> str2 = 'Day"s' #Double quote within single quotes
>>> print(str2)
Day"s
>>> 
```

Special characters in strings

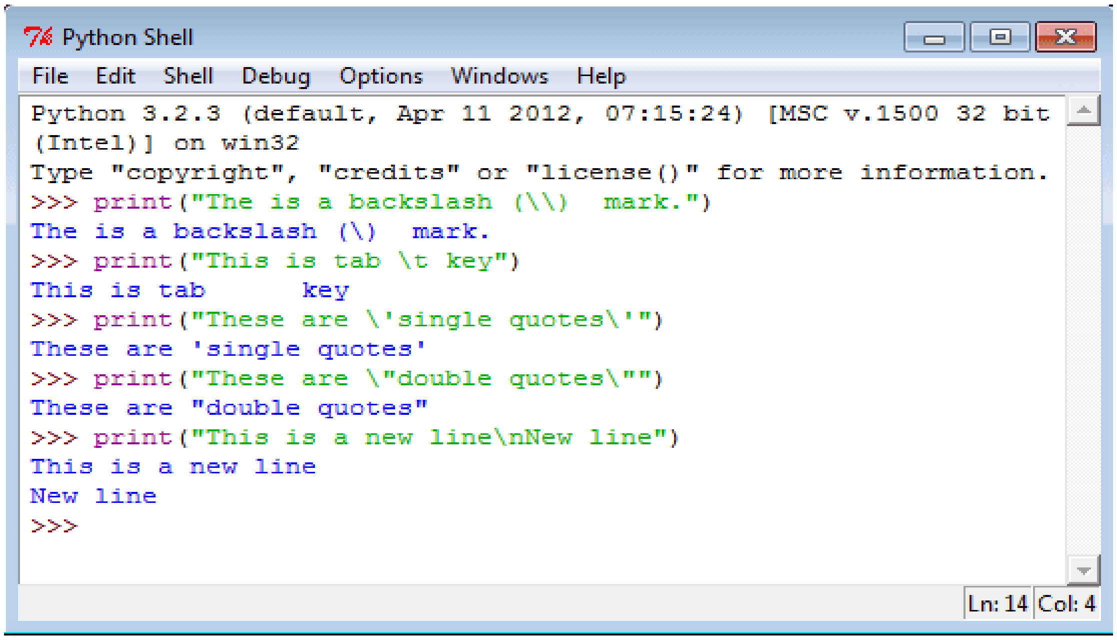
The backslash (\) character is used to introduce a special character. See the following table.

Escape sequence	Meaning
\n	Newline
\t	Horizontal Tab
\\	Backslash

`

\'	Single Quote
\"	Double Quote

See the following statements on special characters.



```
Python 3.2.3 (default, Apr 11 2012, 07:15:24) [MSC v.1500 32 bit
(Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> print("The is a backslash (\\)  mark.")
The is a backslash (\)  mark.
>>> print("This is tab \t key")
This is tab 	key
>>> print("These are \'single quotes\'")
These are 'single quotes'
>>> print("These are \"double quotes\"")
These are "double quotes"
>>> print("This is a new line\nNew line")
This is a new line
New line
>>>
```

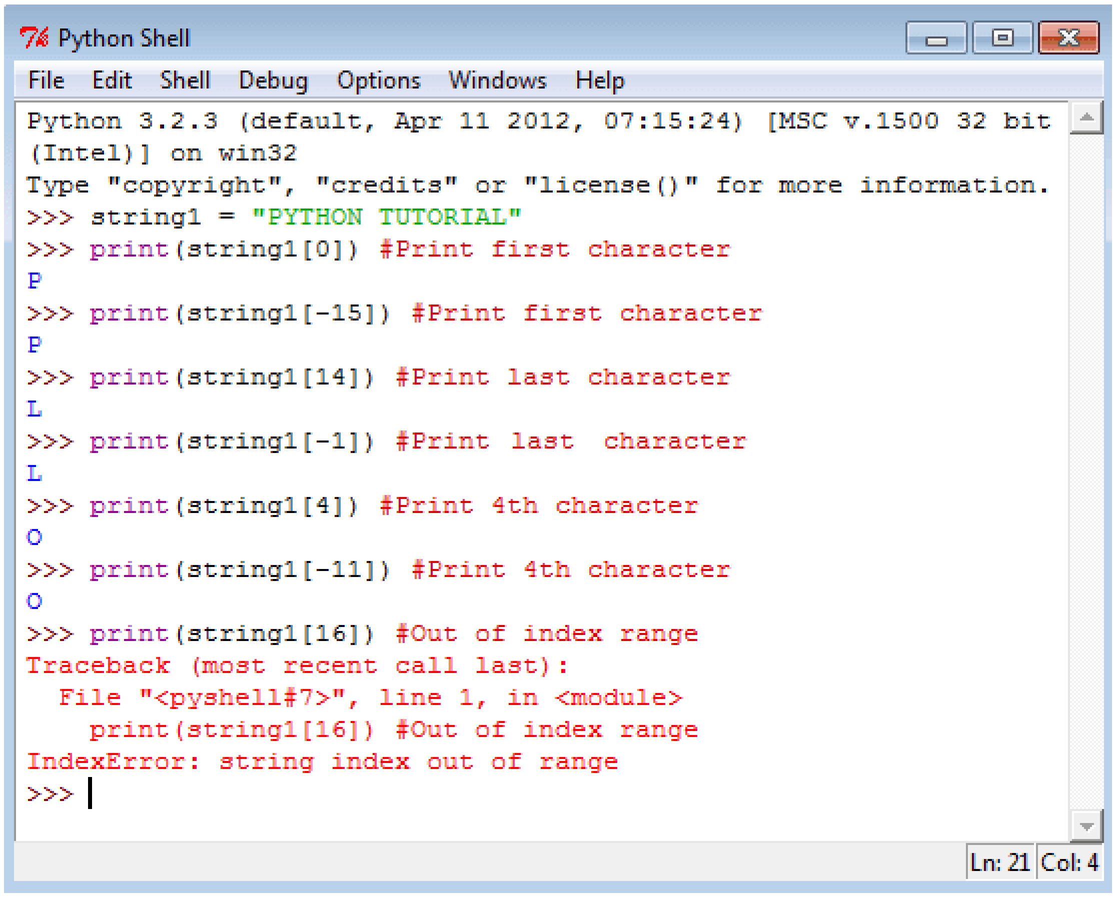
String indices and accessing string elements

- Strings are arrays of characters and elements of an array can be accessed using indexing. Indices start with 0 from left side and -1 when starting from right side.
- string1 ="PYTHON TUTORIAL"

Character	P	Y	T	H	O	N		T	U	T	O	R	I	A	L
Index (from left)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Index (from right)	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

- See the following statements to access single character from various positions.

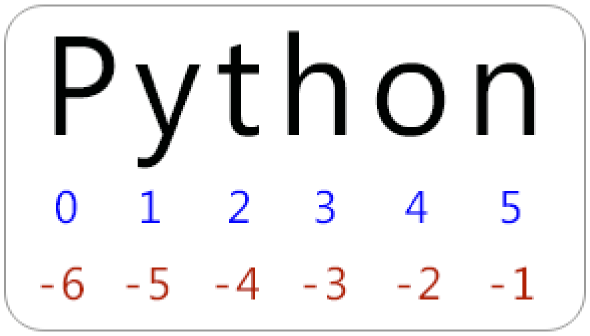
`



```
Python 3.2.3 (default, Apr 11 2012, 07:15:24) [MSC v.1500 32 bit
(Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> string1 = "PYTHON TUTORIAL"
>>> print(string1[0]) #Print first character
P
>>> print(string1[-15]) #Print first character
P
>>> print(string1[14]) #Print last character
L
>>> print(string1[-1]) #Print last  character
L
>>> print(string1[4]) #Print 4th character
O
>>> print(string1[-11]) #Print 4th character
O
>>> print(string1[16]) #Out of index range
Traceback (most recent call last):
  File "<pyshell#7>", line 1, in <module>
    print(string1[16]) #Out of index range
IndexError: string index out of range
>>> 
```

String Slicing

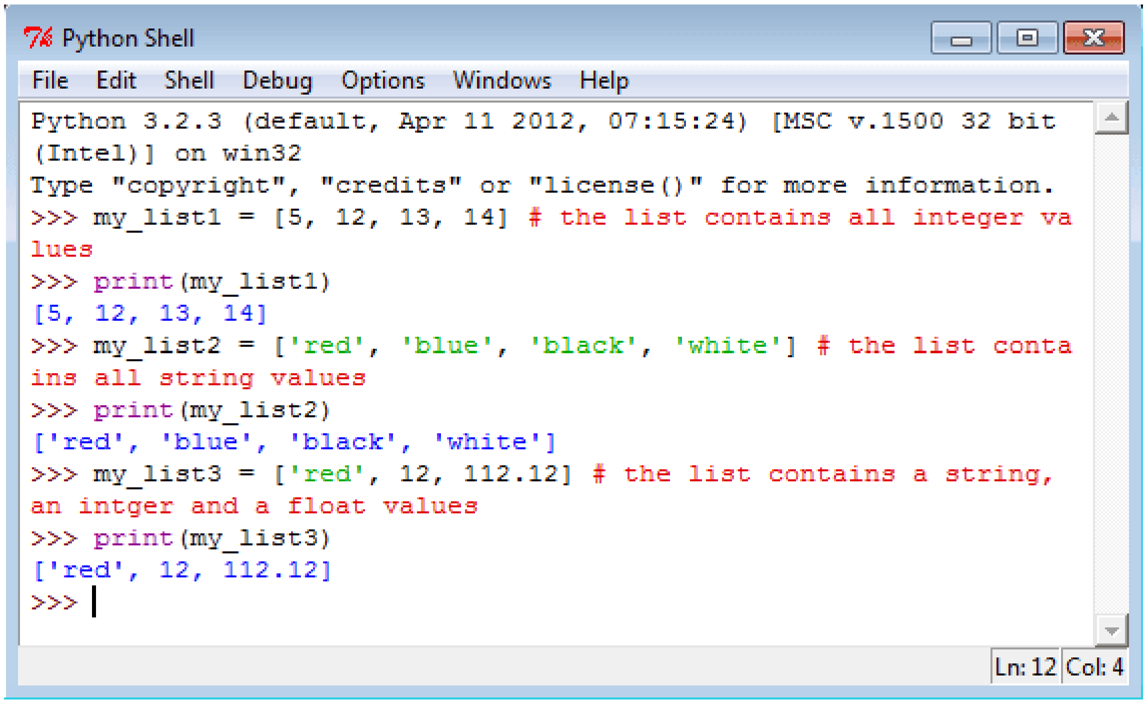
- To cut a substring from a string is called string slicing. Here two indices are used separated by a colon (**:**). A slice 3:7 means indices characters of 3rd, 4th, 5th and 6th positions. The second integer index i.e. 7 is not included. You can use negative indices for slicing. See the following statements.



Lists

A list is a container which holds comma-separated values (items or elements) between square brackets where Items or elements need not all have the same type.

Creating Lists



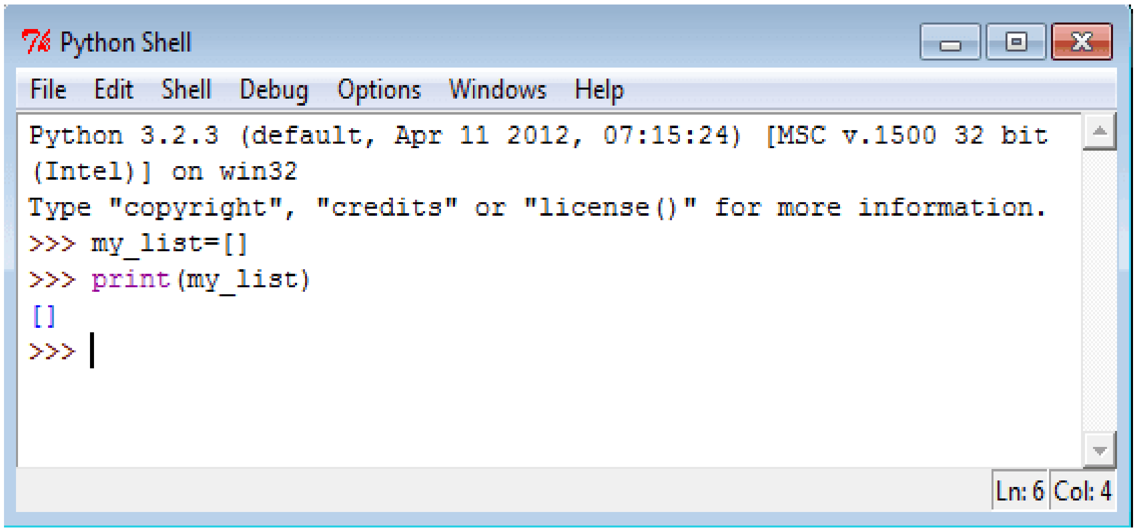
```
Python 3.2.3 (default, Apr 11 2012, 07:15:24) [MSC v.1500 32 bit
(Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> my_list1 = [5, 12, 13, 14] # the list contains all integer va
lues
>>> print(my_list1)
[5, 12, 13, 14]
>>> my_list2 = ['red', 'blue', 'black', 'white'] # the list conta
ins all string values
>>> print(my_list2)
['red', 'blue', 'black', 'white']
>>> my_list3 = ['red', 12, 112.12] # the list contains a string,
an intger and a float values
>>> print(my_list3)
['red', 12, 112.12]
>>> 
```

- A list without any element is called an empty list. See the following statements.

List indices

List indices work the same way as string indices, list indices start at 0. If an index has a positive value it counts from the beginning and similarly it counts backward if the index has a negative value. As positive integers are used to index from the left end and negative integers are used to index from the right end, so every item of a list gives two alternatives indices. Let create a list called color_list with four items.

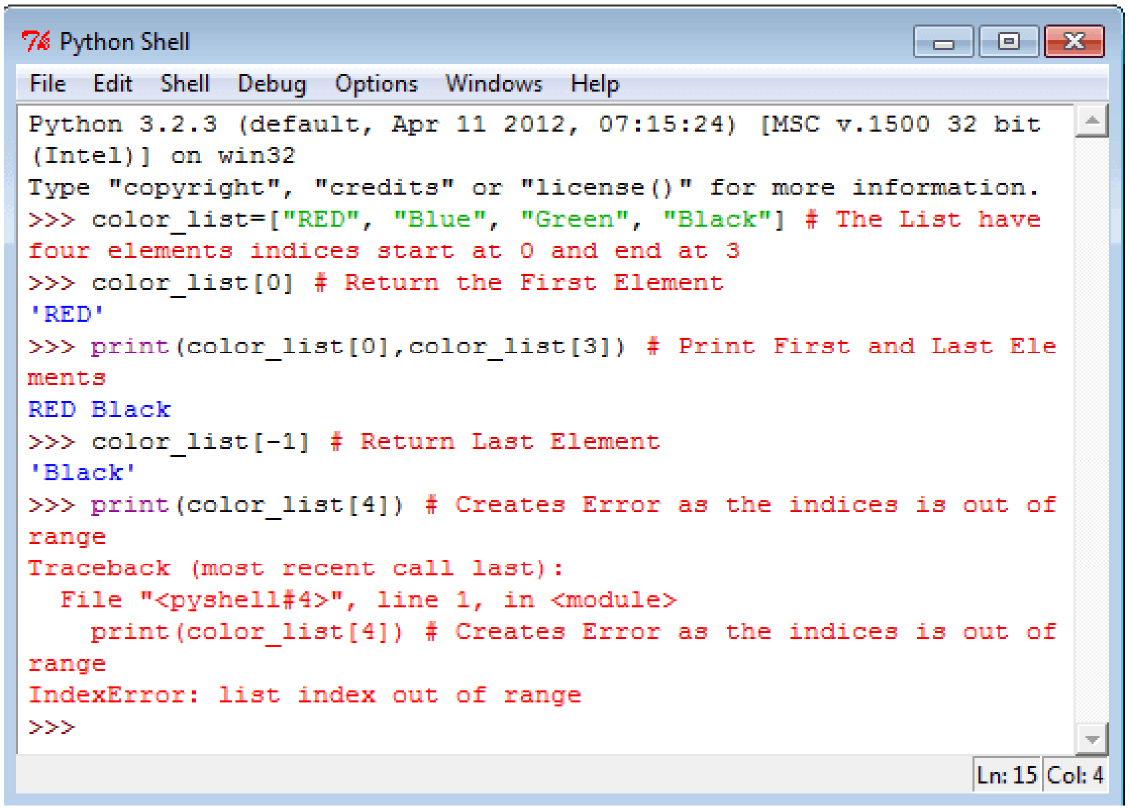
color_list=["RED", "Blue", "Green", "Black"]



```
Python 3.2.3 (default, Apr 11 2012, 07:15:24) [MSC v.1500 32 bit
(Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> my_list=[]
>>> print(my_list)
[]
>>> 
```

Item	RED	Blue	Green	Black
Index (from left)	0	1	2	3
Index (from right)	-4	-3	-2	-1

If you give any index value which is out of range then interpreter creates an error message. See the following statements.



```
Python 3.2.3 (default, Apr 11 2012, 07:15:24) [MSC v.1500 32 bit
(Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> color_list=["RED", "Blue", "Green", "Black"] # The List have
four elements indices start at 0 and end at 3
>>> color_list[0] # Return the First Element
'RED'
>>> print(color_list[0],color_list[3]) # Print First and Last Ele
ments
RED Black
>>> color_list[-1] # Return Last Element
'Black'
>>> print(color_list[4]) # Creates Error as the indices is out of
range
Traceback (most recent call last):
  File "<pyshell#4>", line 1, in <module>
    print(color_list[4]) # Creates Error as the indices is out of
range
IndexError: list index out of range
>>>
```

List Slice

Lists can be sliced like strings and other sequences. The syntax of list slices is easy; sliced_list = List_Name[startIndex:endIndex]

Conditional Statements

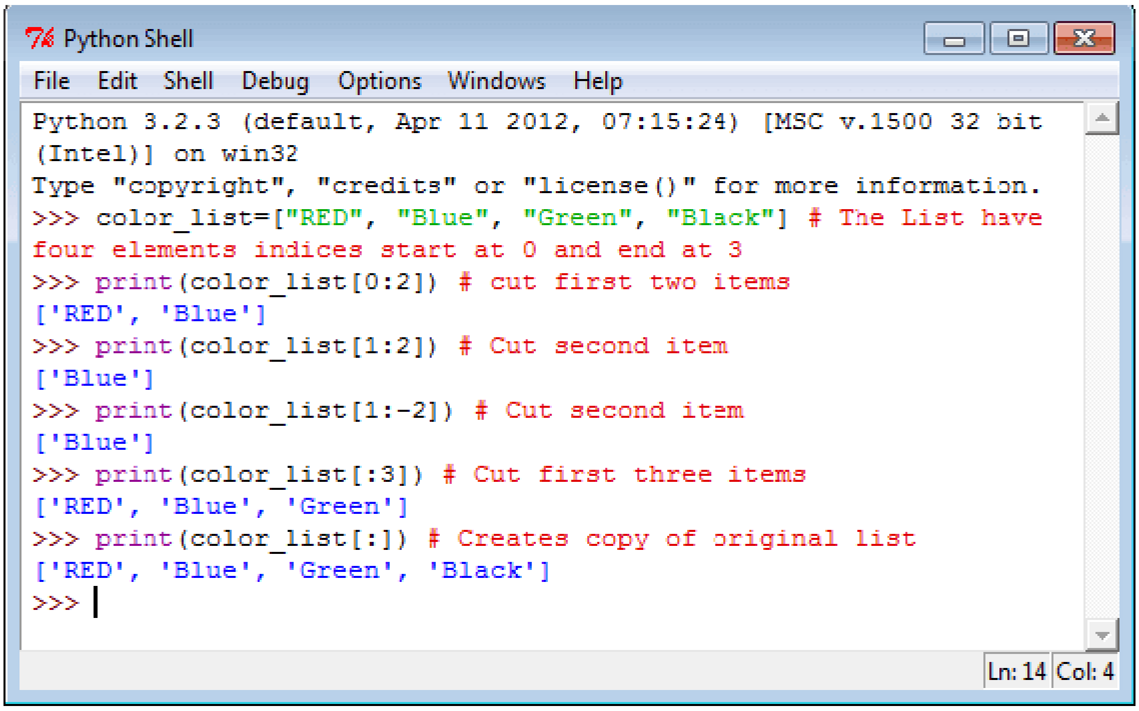
Python supports the usual logical conditions from mathematics:

- Equals: a == b
- Not Equals: a != b
- Less than: a < b
- Less than or equal to: a <= b
- Greater than: a > b
- Greater than or equal to: a >= b

e.g

```
if b > a:

    print("b is greater than a")
```



```
Python 3.2.3 (default, Apr 11 2012, 07:15:24) [MSC v.1500 32 bit
(Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> color_list=["RED", "Blue", "Green", "Black"] # The List have
four elements indices start at 0 and end at 3
>>> print(color_list[0:2]) # cut first two items
['RED', 'Blue']
>>> print(color_list[1:2]) # Cut second item
['Blue']
>>> print(color_list[1:-2]) # Cut second item
['Blue']
>>> print(color_list[:3]) # Cut first three items
['RED', 'Blue', 'Green']
>>> print(color_list[:]) # Creates copy of original list
['RED', 'Blue', 'Green', 'Black']
>>> 
```