Lecture 3

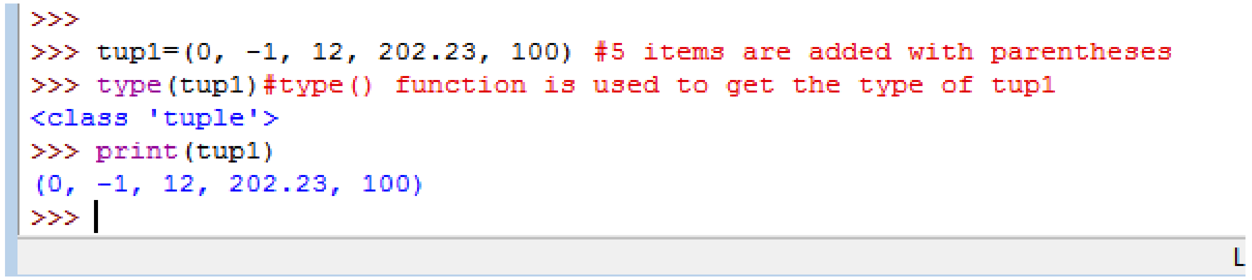
Python Tuples, Lists, Set, and Dictionaries

Tuples

- A tuple is a container which holds a series of comma-separated values (items or elements) between parentheses.
- Tuples are immutable and can hold mix data types.

Creating Tuples

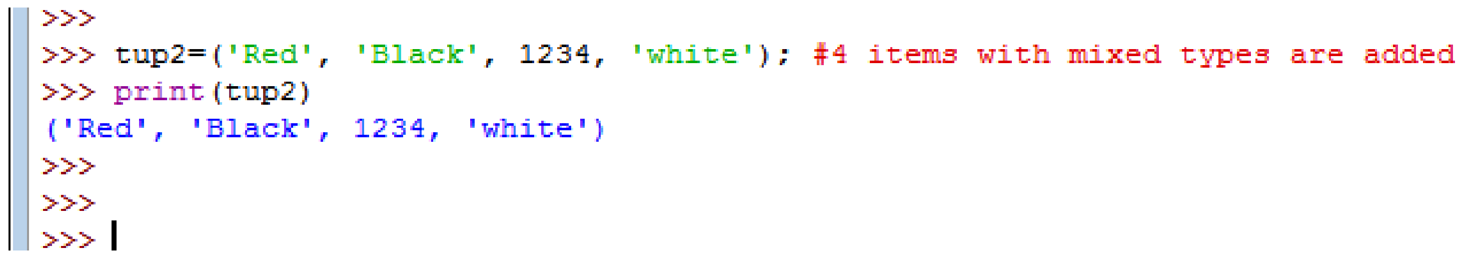
- Items (with same data type) are added with parentheses



```
>>>
>>> tup1=(0, -1, 12, 202.23, 100) #5 items are added with parentheses
>>> type(tup1)#type() function is used to get the type of tup1
<class 'tuple'>
>>> print(tup1)
(0, -1, 12, 202.23, 100)
>>> 
```

Creating Tuples

- Items (with mix data type) are added with parentheses

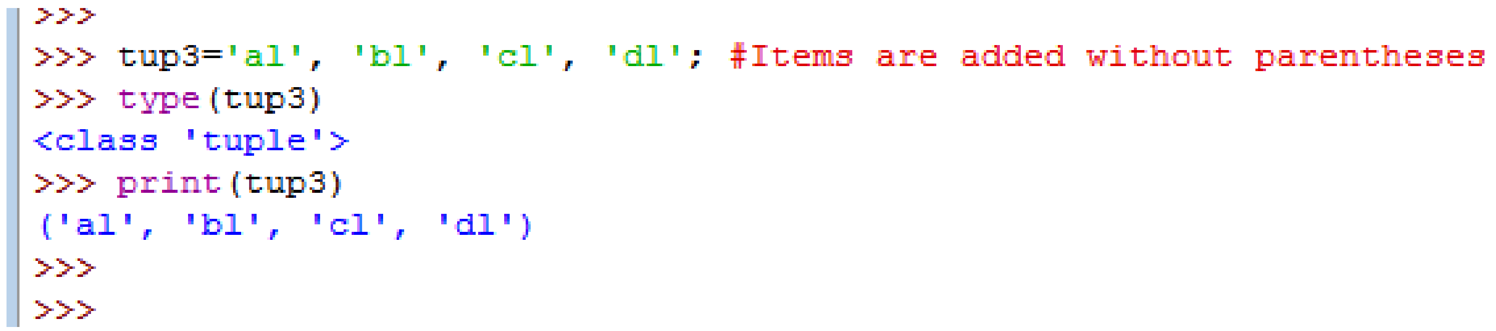


```
>>>
>>> tup2=('Red', 'Black', 1234, 'white'); #4 items with mixed types are added
>>> print(tup2)
('Red', 'Black', 1234, 'white')
>>>
>>>
>>> |
```

KNOW ING

Creating Tuples

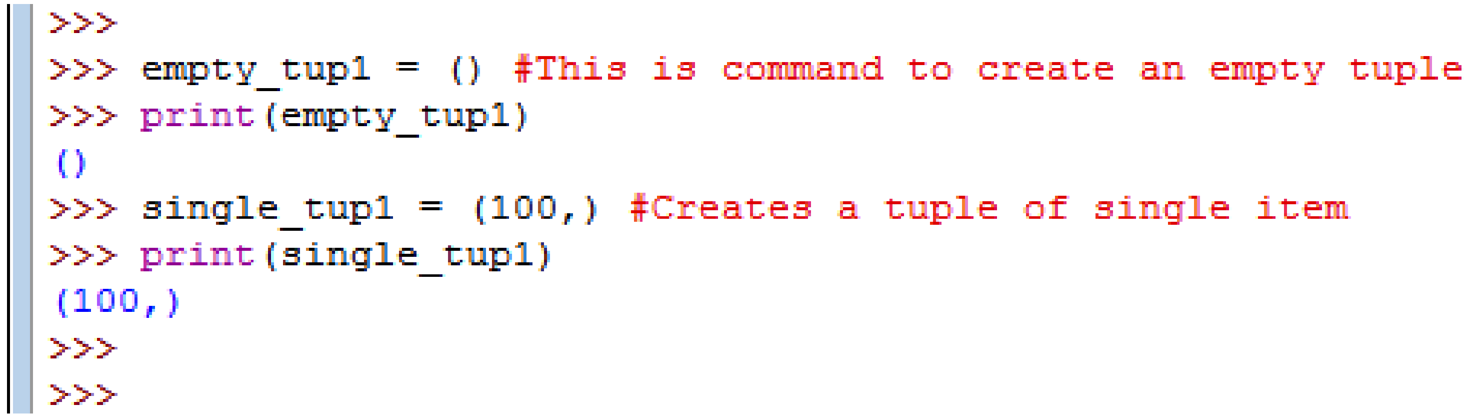
- Items are added without parentheses



```
>>>
>>> tup3='al', 'bl', 'cl', 'dl'; #Items are added without parentheses
>>> type(tup3)
<class 'tuple'>
>>> print(tup3)
('al', 'bl', 'cl', 'dl')
>>>
>>>
```

Creating Tuples

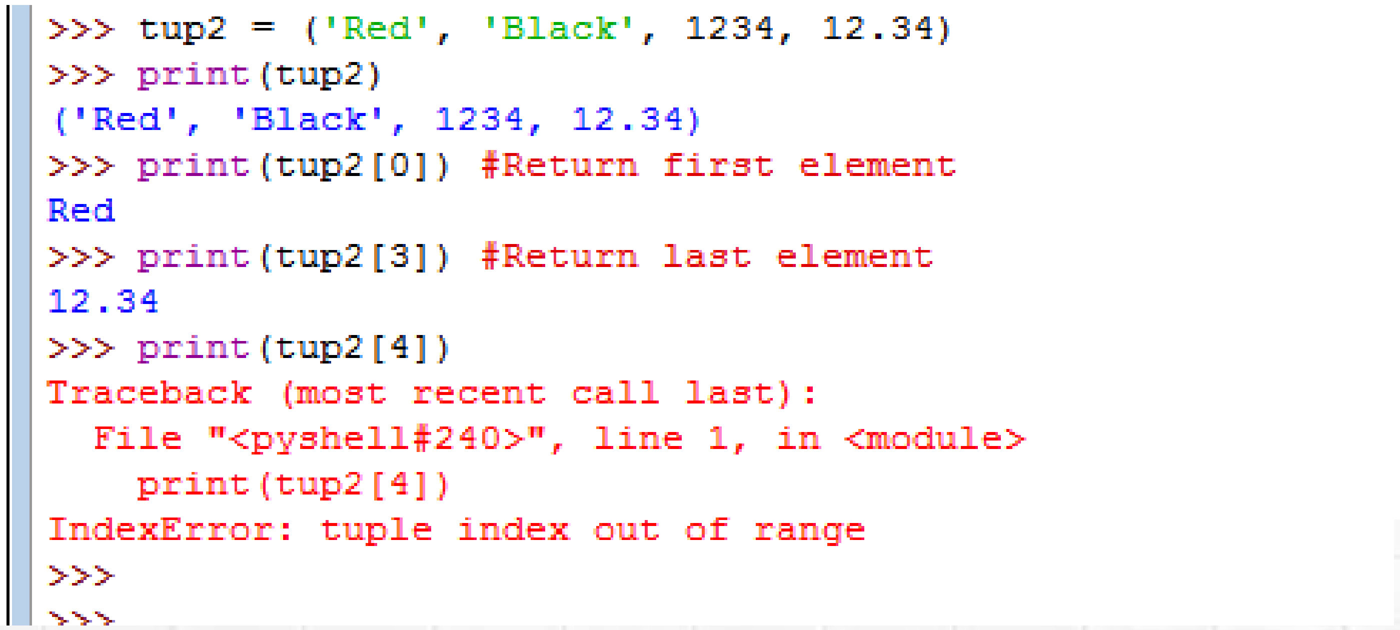
- To create an empty tuple or create a tuple with single element.



```
>>>
>>> empty_tup1 = () #This is command to create an empty tuple
>>> print(empty_tup1)
()
>>> single_tup1 = (100,) #Creates a tuple of single item
>>> print(single_tup1)
(100,)
>>>
>>>
```

Creating Tuples

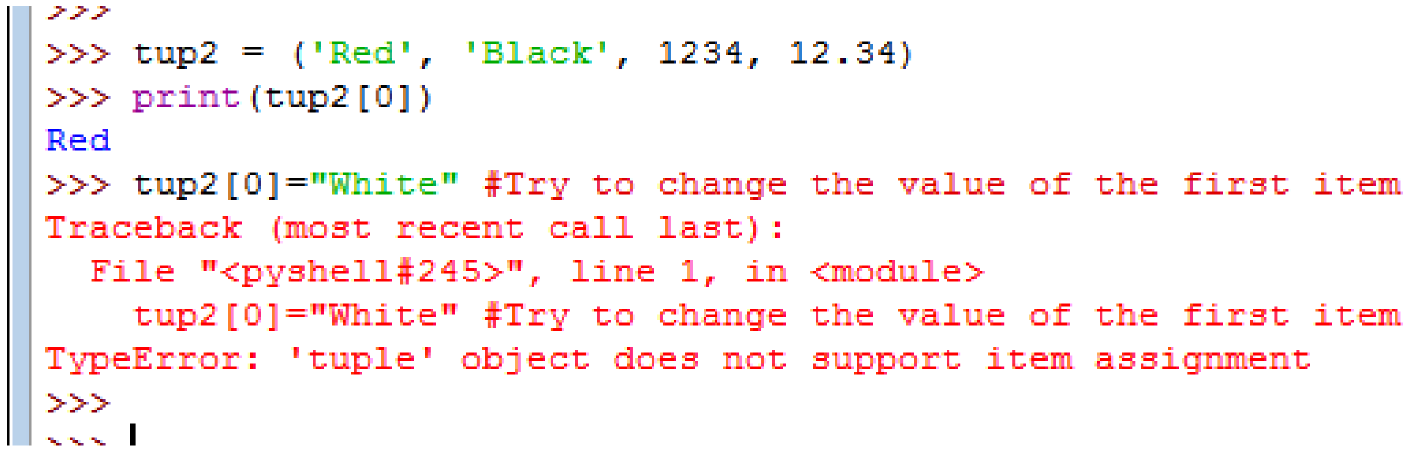
- Elements of a tuple are indexed like other sequences. The tuple indices start at 0.



```
>>> tup2 = ('Red', 'Black', 1234, 12.34)
>>> print(tup2)
('Red', 'Black', 1234, 12.34)
>>> print(tup2[0]) #Return first element
Red
>>> print(tup2[3]) #Return last element
12.34
>>> print(tup2[4])
Traceback (most recent call last):
  File "<pyshell#240>", line 1, in <module>
    print(tup2[4])
IndexError: tuple index out of range
>>>
>>>
```

Creating Tuples

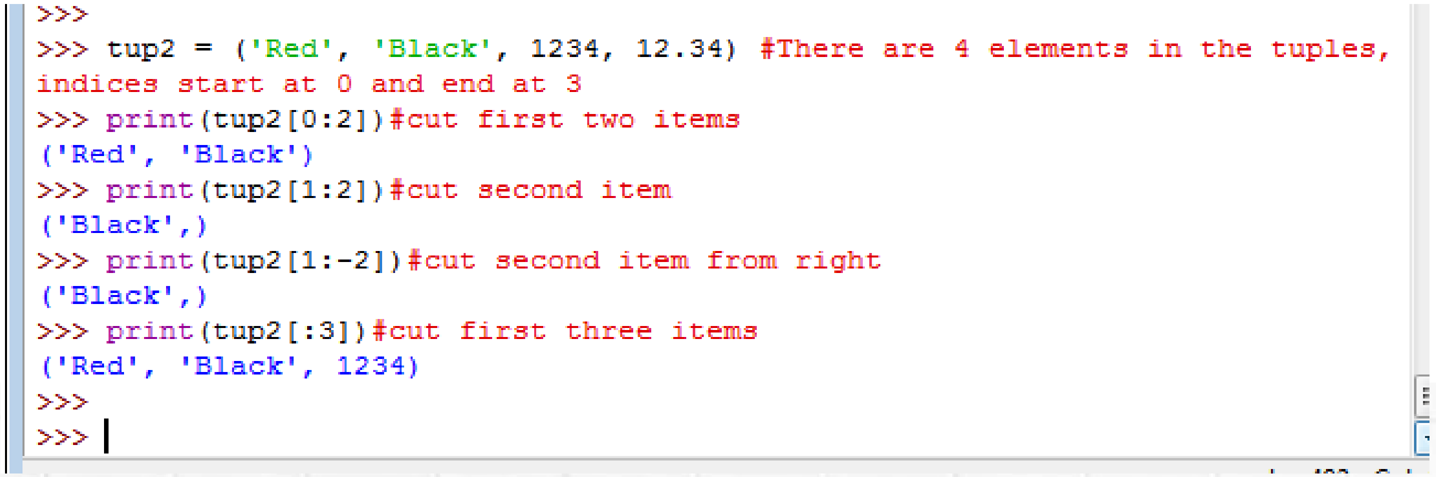
- Tuples are immutable which means it's items values are unchangeable.



```
>>> tup2 = ('Red', 'Black', 1234, 12.34)
>>> print(tup2[0])
Red
>>> tup2[0]="White" #Try to change the value of the first item
Traceback (most recent call last):
  File "<pyshell#245>", line 1, in <module>
    tup2[0]="White" #Try to change the value of the first item
TypeError: 'tuple' object does not support item assignment
>>>
```

Slicing a tuple

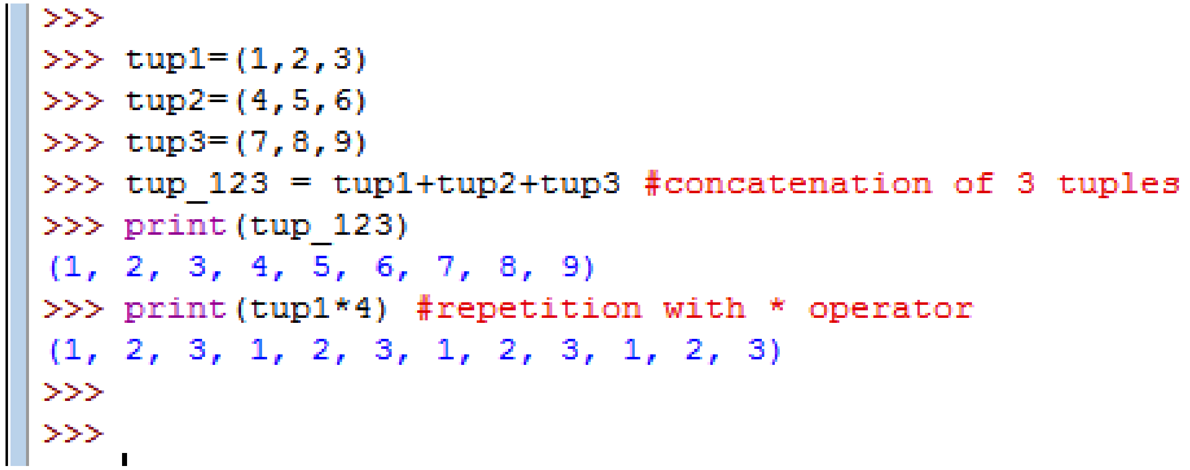
- Like other sequences like strings, tuples can be sliced.
- Slicing a tuple creates a new tuple but it does not change the original tuple.



```
>>>
>>> tup2 = ('Red', 'Black', 1234, 12.34) #There are 4 elements in the tuples,
indices start at 0 and end at 3
>>> print(tup2[0:2])#cut first two items
('Red', 'Black')
>>> print(tup2[1:2])#cut second item
('Black',)
>>> print(tup2[1:-2])#cut second item from right
('Black',)
>>> print(tup2[:3])#cut first three items
('Red', 'Black', 1234)
>>>
>>> |
```

Using + and * operators in Tuples

- Use + operator to create a new tuple that is a concatenation of tuples and use * operator to repeat a tuple.

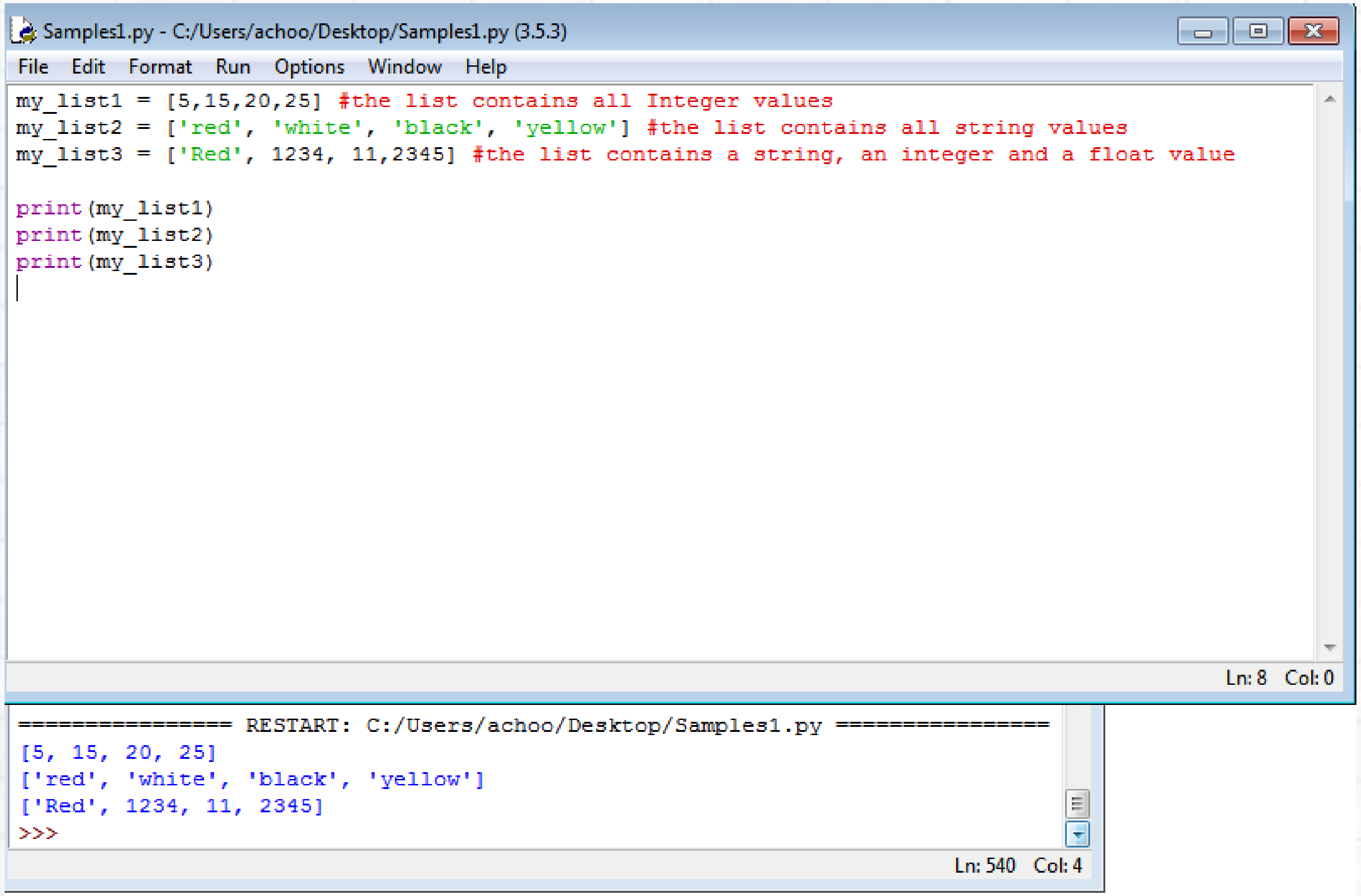


```
>>>
>>> tup1=(1,2,3)
>>> tup2=(4,5,6)
>>> tup3=(7,8,9)
>>> tup_123 = tup1+tup2+tup3 #concatenation of 3 tuples
>>> print(tup_123)
(1, 2, 3, 4, 5, 6, 7, 8, 9)
>>> print(tup1*4) #repetition with * operator
(1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3)
>>>
>>>
```

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


Samples1.py - C:/Users/achoo/Desktop/Samples1.py (3.5.3)
File Edit Format Run Options Window Help
Ln: 8 Col: 0
Ln: 540 Col: 4

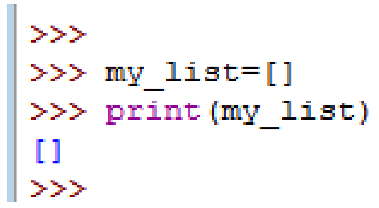
```
my_list1 = [5,15,20,25] #the list contains all Integer values
my_list2 = ['red', 'white', 'black', 'yellow'] #the list contains all string values
my_list3 = ['Red', 1234, 11,2345] #the list contains a string, an integer and a float value

print(my_list1)
print(my_list2)
print(my_list3)
```

```
================= RESTART: C:/Users/achoo/Desktop/Samples1.py =================
[5, 15, 20, 25]
['red', 'white', 'black', 'yellow']
['Red', 1234, 11, 2345]
>>>
```

Creating Lists

- A list without any element is called an empty list.



```
>>>
>>> my_list=[]
>>> print(my_list)
[]
>>>
```

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.

List indices

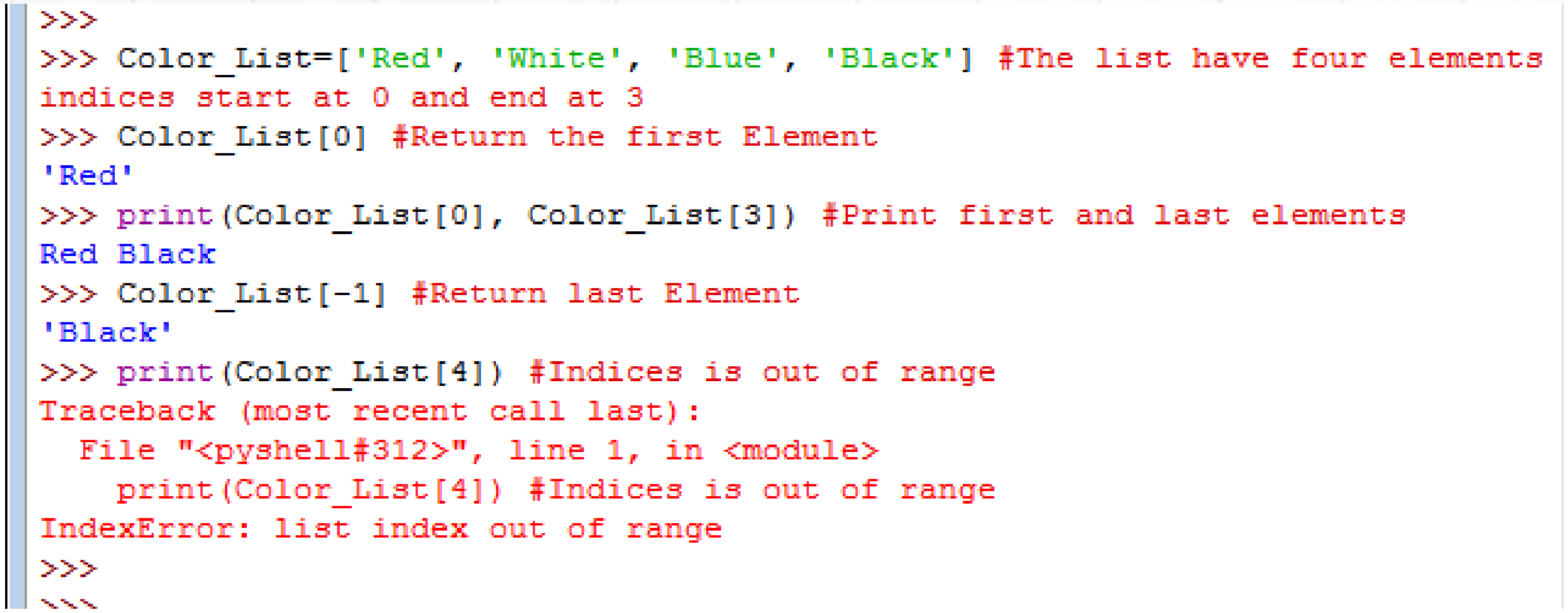
- Let create a list called Color_List with four(4) items.
- Color_List=[‘Red’, ‘White’, ‘Blue’, ‘Black’]

Item	Red	White	Blue	Black
Index(from left)	0	1	2	3
Index(from right)	-4	-3	-2	-1

- If give any index value which is out of range then interpreter creates an error message.

KNOW ING

List indices



```
>>>
>>> Color_List=['Red', 'White', 'Blue', '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 elements
Red Black
>>> Color_List[-1] #Return last Element
'Black'
>>> print(Color_List[4]) #Indices is out of range
Traceback (most recent call last):
  File "<pyshell#312>", line 1, in <module>
    print(Color_List[4]) #Indices is out of range
IndexError: list index out of range
>>>
```

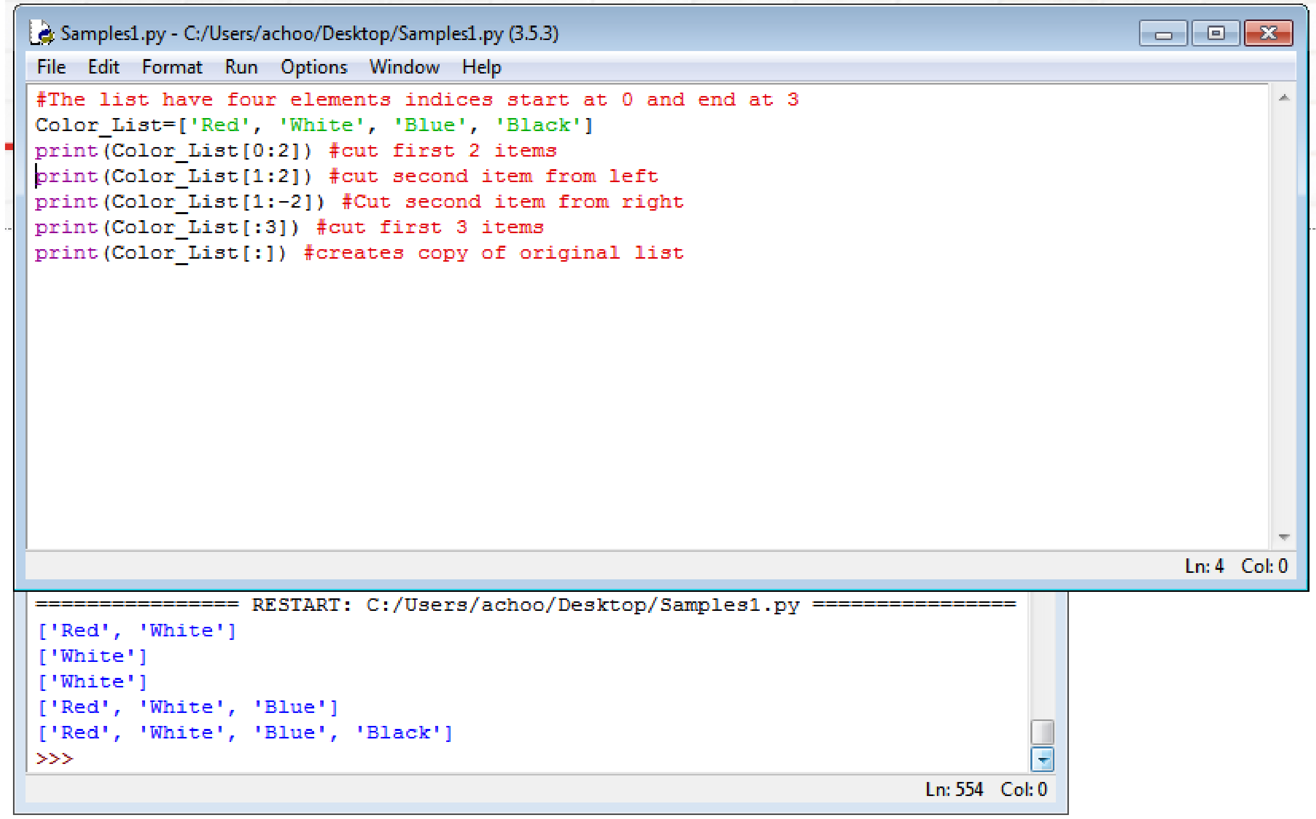
List Slices

- Lists can be sliced like strings and other sequences. The syntax of list slices is easy:

  Sliced_list = List_Name(startIndex:endIndex)

- This refers to the items of a list starting at index startIndex and stopping just before index endIndex.
- The default values for list are 0 (startIndex) and the end (endIndex) of the list.
- If you omit both indices, the slice makes a copy of the original list.

List Slices

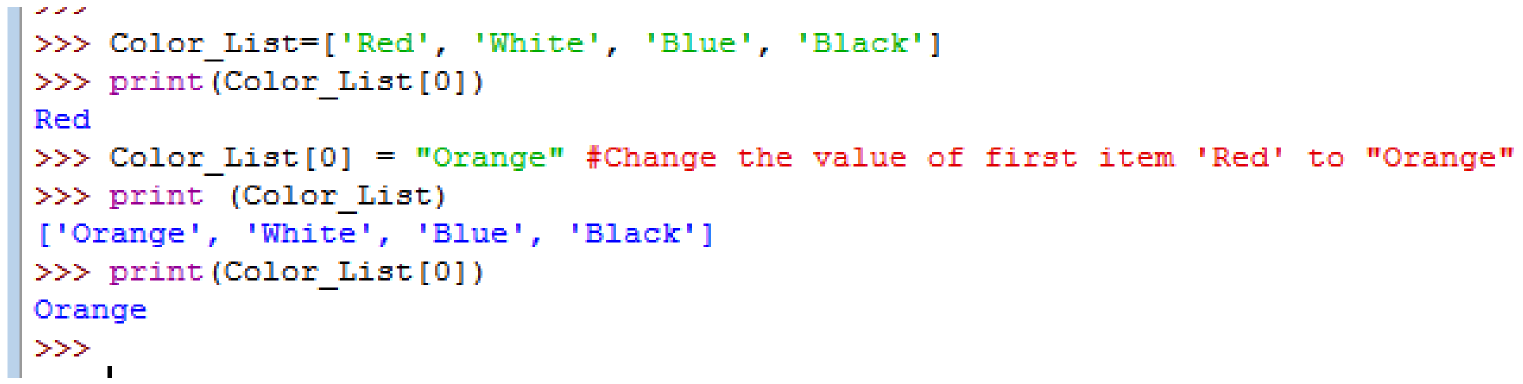

Samples1.py - C:/Users/achoo/Desktop/Samples1.py (3.5.3)
File Edit Format Run Options Window Help
Ln: 4 Col: 0
Ln: 554 Col: 0

```
#The list have four elements indices start at 0 and end at 3
Color_List=['Red', 'White', 'Blue', 'Black']
print(Color_List[0:2]) #cut first 2 items
print(Color_List[1:2]) #cut second item from left
print(Color_List[1:-2]) #Cut second item from right
print(Color_List[:3]) #cut first 3 items
print(Color_List[:]) #creates copy of original list
```

```
================ RESTART: C:/Users/achoo/Desktop/Samples1.py ================
['Red', 'White']
['White']
['White']
['Red', 'White', 'Blue']
['Red', 'White', 'Blue', 'Black']
>>>
```

Lists are Mutable

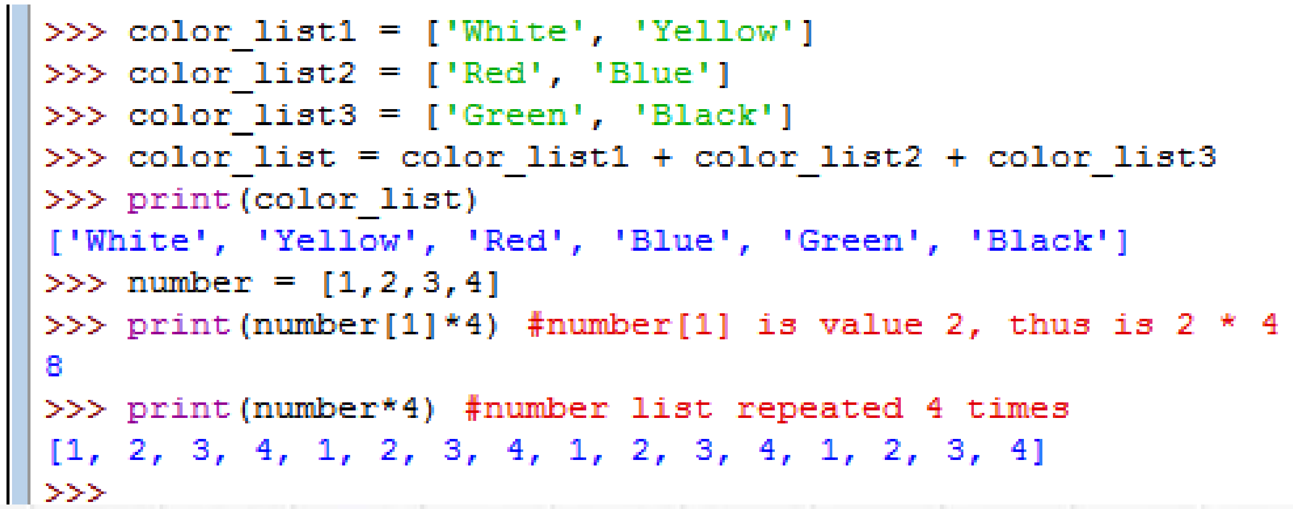
- Items in the list are mutable i.e. after creating a list you can change any item in the list.



```
>>> Color_List=['Red', 'White', 'Blue', 'Black']
>>> print(Color_List[0])
Red
>>> Color_List[0] = "Orange" #Change the value of first item 'Red' to "Orange"
>>> print (Color_List)
['Orange', 'White', 'Blue', 'Black']
>>> print(Color_List[0])
Orange
>>>
```

Using + and * operators in List

- Use + operator to create a new list that is a concatenation of two lists and use * operator to repeat a list.

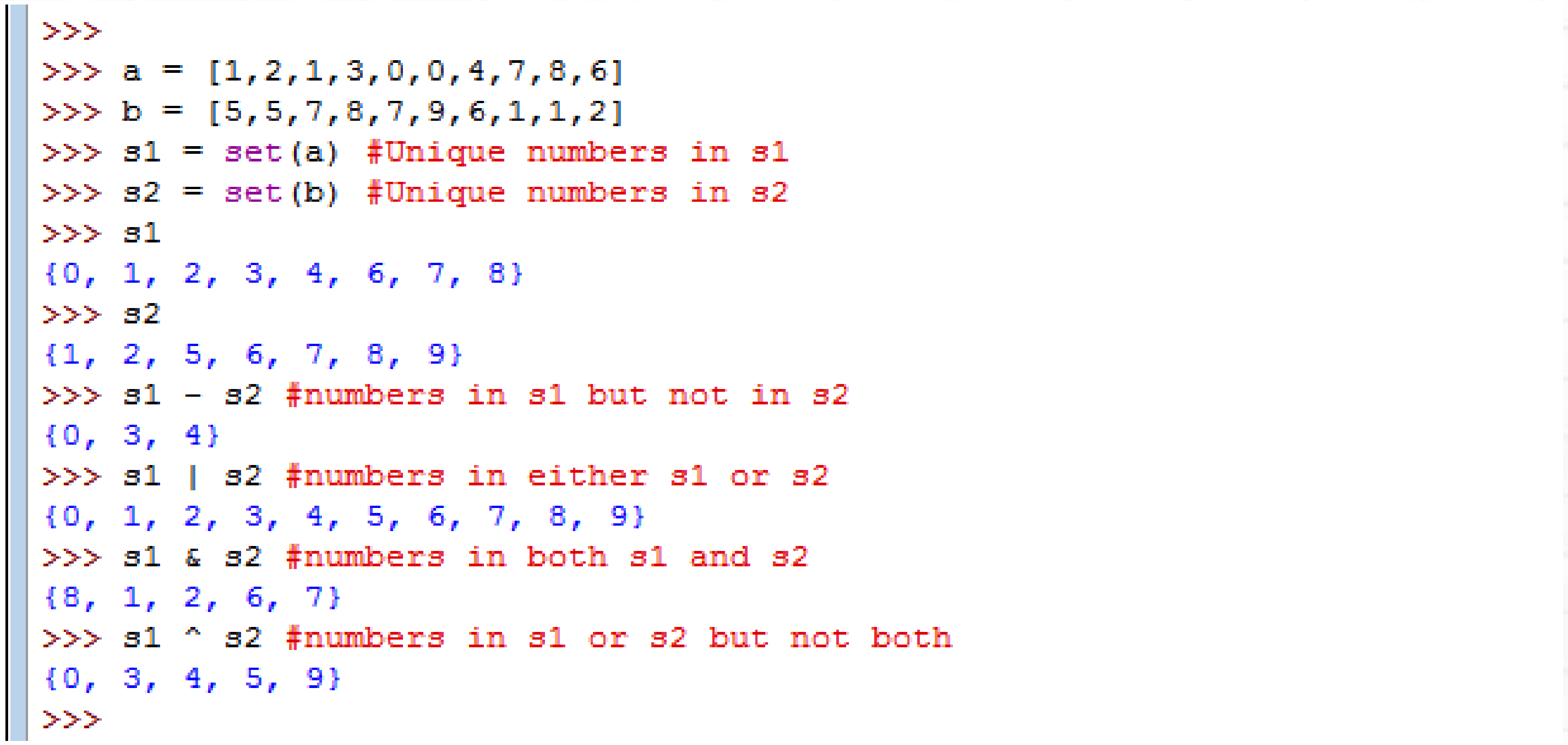


```
>>> color_list1 = ['White', 'Yellow']
>>> color_list2 = ['Red', 'Blue']
>>> color_list3 = ['Green', 'Black']
>>> color_list = color_list1 + color_list2 + color_list3
>>> print(color_list)
['White', 'Yellow', 'Red', 'Blue', 'Green', 'Black']
>>> number = [1,2,3,4]
>>> print(number[1]*4) #number[1] is value 2, thus is 2 * 4
8
>>> print(number*4) #number list repeated 4 times
[1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4]
>>>
```

Sets

- A set is an unordered collection of unique elements.
- Basic uses include dealing with set theory (which support mathematical operations like union, intersection, difference, and symmetric difference) or eliminating duplicate entries

Sets



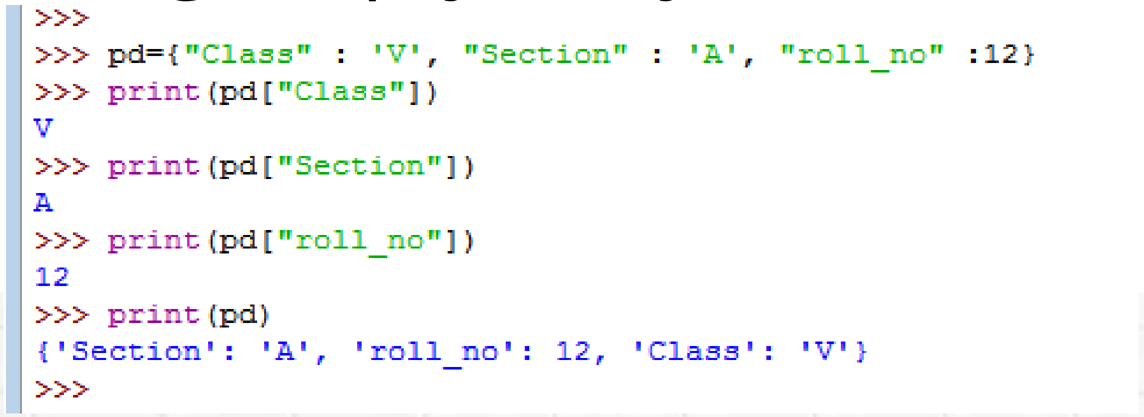
```
>>>
>>> a = [1,2,1,3,0,0,4,7,8,6]
>>> b = [5,5,7,8,7,9,6,1,1,2]
>>> s1 = set(a) #Unique numbers in s1
>>> s2 = set(b) #Unique numbers in s2
>>> s1
{0, 1, 2, 3, 4, 6, 7, 8}
>>> s2
{1, 2, 5, 6, 7, 8, 9}
>>> s1 - s2 #numbers in s1 but not in s2
{0, 3, 4}
>>> s1 | s2 #numbers in either s1 or s2
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
>>> s1 & s2 #numbers in both s1 and s2
{8, 1, 2, 6, 7}
>>> s1 ^ s2 #numbers in s1 or s2 but not both
{0, 3, 4, 5, 9}
>>>
```

Dictionaries

- Python dictionary is a container of the unordered set of objects like lists.
- The objects are surrounded by curly braces {}.
- The items in a dictionary are a comma-separated list of key:value pairs where keys and values are Python data type.
- Each object or value accessed by key and keys are unique in the dictionary.

Dictionaries

- As keys are used of indexing, they must be the immutable type (string, number, or tuple).
- You are create an empty dictionary using empty curly braces.



```
>>>
>>> pd={"Class" : 'V', "Section" : 'A', "roll_no" :12}
>>> print(pd["Class"])
V
>>> print(pd["Section"])
A
>>> print(pd["roll_no"])
12
>>> print(pd)
{'Section': 'A', 'roll_no': 12, 'Class': 'V'}
>>>
```

Making Decisions in a Program

- Three basic control structures
  - Sequence
  - Selection
  - Repetition
- All procedures in an application are written using one of more of these structures.

Testing and Debugging

- Test an application using some sample data
  - Use both valid and invalid data
- **Valid data**: Data that application is expecting
- **Invalid data**: Data that application is not expecting
- **Debugging**: Process of locating and correcting errors
- Errors can be related to either syntax or logic.

KNOW ING

Testing and Debugging

- **Syntax error**: Occurs when a rule of programming language is broken.
- **Logic error**: Occurs when syntax is correct but outcome is not what was desired.
  - Causes may include missing instructions, instructions out of order, or wrong type of instruction.

KNOW ING