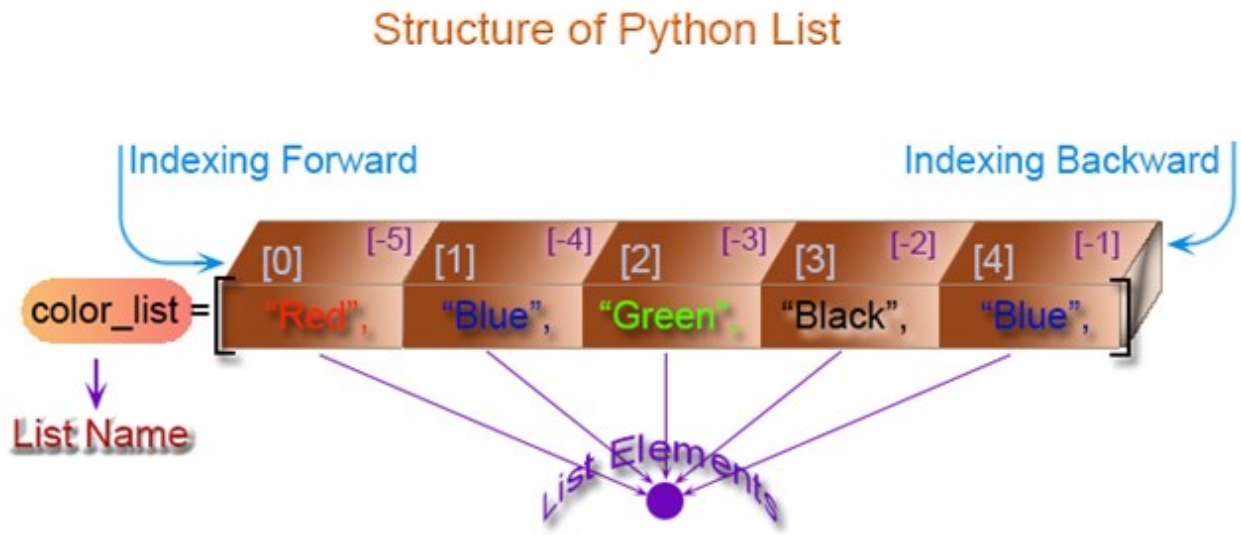
013 Python List

List

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.

In general, we can define a list as an object that contains multiple data items (elements). The contents of a list can be changed during program execution. The size of a list can also change during execution, as elements are added or removed from it.



Note: There are much programming languages which allow us to create arrays, which are objects similar to lists. Lists serve the same purpose as arrays and have many more built-in capabilities. Traditional arrays can not be created in Python.

Examples of lists:

- numbers = [10, 20, 30, 40, 50]
- names = ["Sara", "David", "Warner", "Sandy"]
- student_info = ["Sara", 1, "Chemistry"]

Lists: Commands

```
<list> = <list>[from_inclusive : to_exclusive : ±step_size]

<list>.append(<el>)
# Or: <list> += [<el>]
<list>.extend(<collection>)
 # Or: <list> += <collection>

<list>.sort()
<list>.reverse()
<list> = sorted(<collection>)
<iter> = reversed(<list>)

sum_of_elements  = sum(<collection>)
elementwise_sum  = [sum(pair) for pair in zip(list_a, list_b)]
sorted_by_second = sorted(<collection>, key=lambda el: el[1])
sorted_by_both   = sorted(<collection>, key=lambda el: (el[1],
el[0]))
flatter_list     = list(itertools.chain.from_iterable(<list>))
product_of_elems = functools.reduce(lambda out, x: out * x,
<collection>)
list_of_chars    = list(<str>)

# Returns number of occurrences. Also works on strings.
  <int> = <list>.count(<el>)
# Returns index of first occurrence or raises ValueError.
  index = <list>.index(<el>)
# Inserts item at index and moves the rest to the right.
  <list>.insert(index, <el>)
# Removes and returns item at index or from the end.
  <el> = <list>.pop([index])
# Removes first occurrence of item or raises ValueError.
  <list>.remove(<el>)
# Removes all items. Also works on dictionary and set.
  <list>.clear()
```

Create a Python list

Following list contains all integer values:

```
>>> my_list1 = [5, 12, 13, 14] # the list contains all integer values

>>> print(my_list1)

[5, 12, 13, 14]
```

```
>>>
```

Following list contains all string:

```
>>> my_list2 = ['red', 'blue', 'black', 'white'] # the list contains
all string

values

>>> print(my_list2)

['red', 'blue', 'black', 'white']

>>>
```

Following list contains a string, an integer and a float values:

```
>>> my_list3 = ['red', 12, 112.12] # the list contains a string, an
integer 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.

```
>>> my_list=[]

>>> print(my_list)

[]

>>>
```

Use + operator to create a new list that is a concatenation of two lists and use * operator to repeat a list. See the following statements.

```
>>> 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]
>>> print(number[0]*4)
4
>>> print(number*4)
[1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3]
>>>
```

List indices

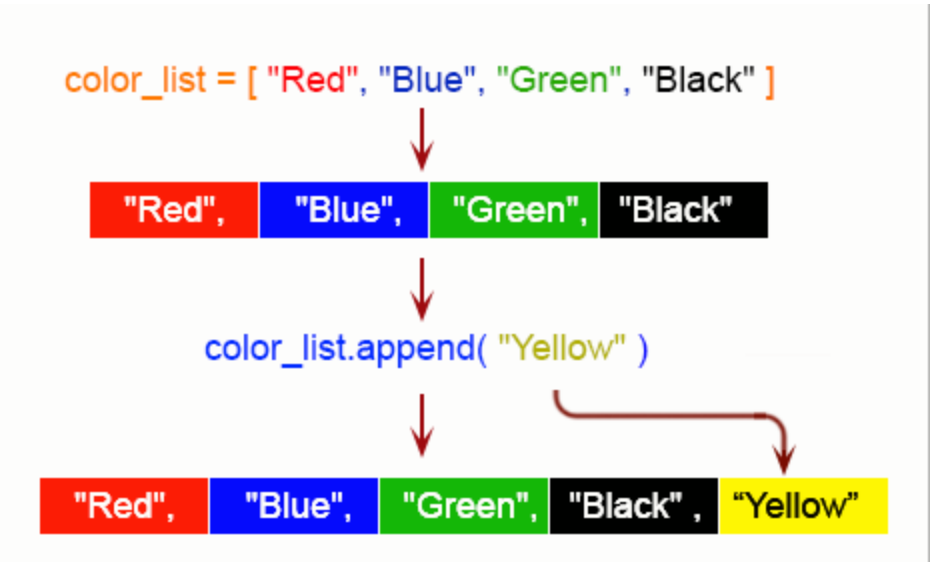
List indices work the same way as string indices, list indices start at 0. If an index has a positive value it counts from the beginning and similarly it counts backward if the index has a negative value. As positive integers are used to index from the left end and negative integers are used to index 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"]

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.

```
>>> 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 Elements

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 "<stdin>", line 1, in <module>

IndexError: list index out of range

>>>
```

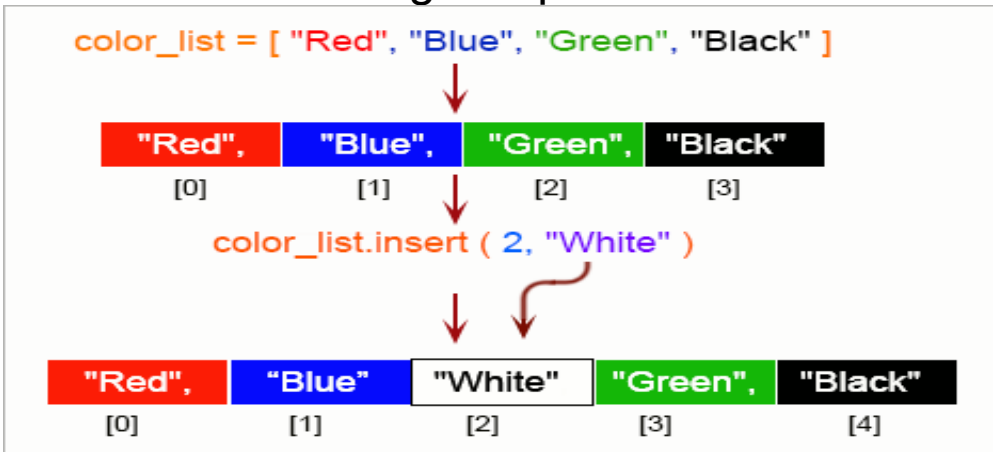
Add an item to the end of the list



See the following statements:

```
>>> color_list=["Red", "Blue", "Green", "Black"]
>>> print(color_list)
['Red', 'Blue', 'Green', 'Black']
>>> color_list.append("Yellow")
>>> print(color_list)
['Red', 'Blue', 'Green', 'Black', 'Yellow']
>>>
```

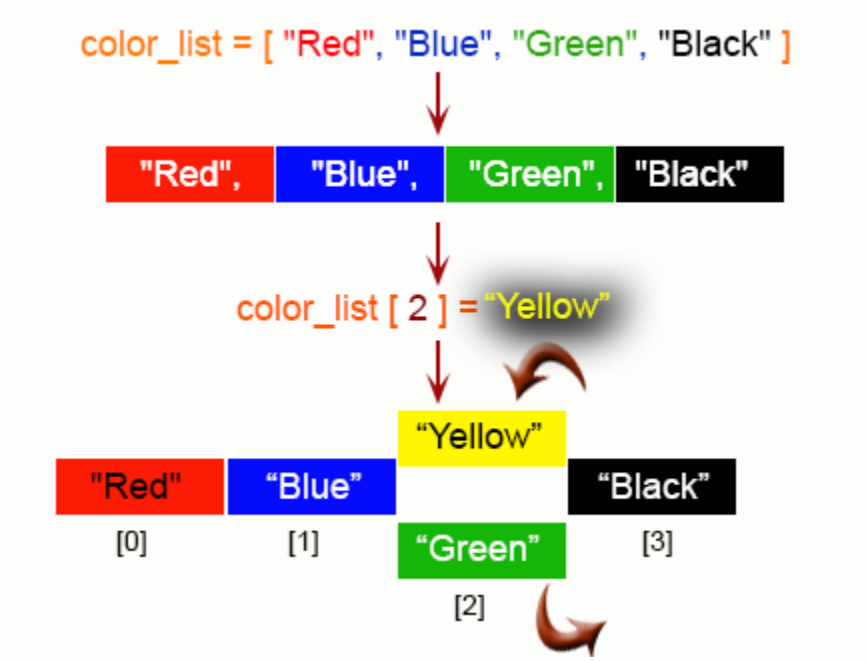
Insert an item at a given position



See the following statements:

```
>>> color_list=["Red", "Blue", "Green", "Black"]
>>> print(color_list)
['Red', 'Blue', 'Green', 'Black']
>>> color_list.insert(2, "White") #Insert an item at third position
>>> print(color_list)
['Red', 'Blue', 'White', 'Green', 'Black']
>>>
```

Modify an element by using the index of the element

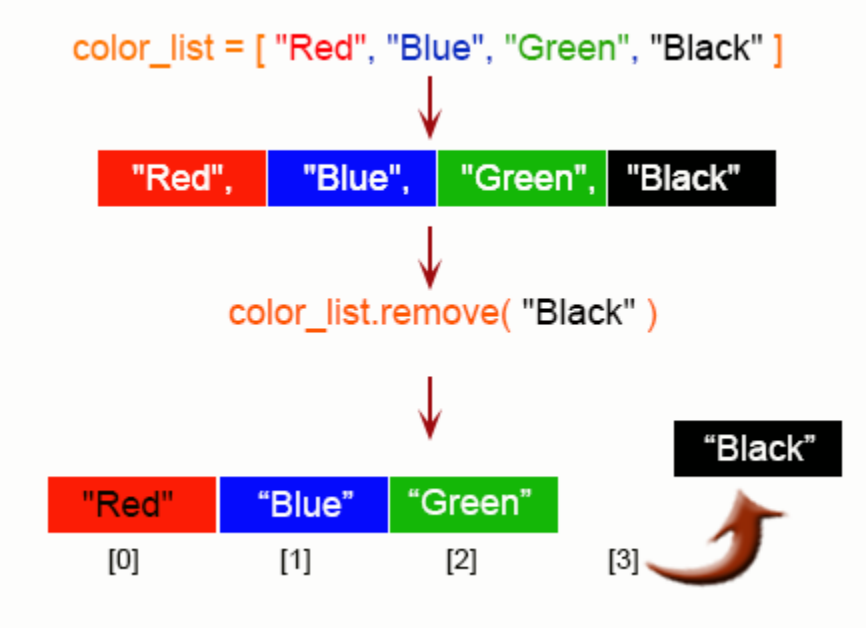


See the following statements:

```
>>> color_list=["Red", "Blue", "Green", "Black"]
>>> print(color_list)
```

```
['Red', 'Blue', 'Green', 'Black']
>>> color_list[2]="Yellow"  #Change the third color
>>> print(color_list)
['Red', 'Blue', 'Yellow', 'Black']
>>>
```

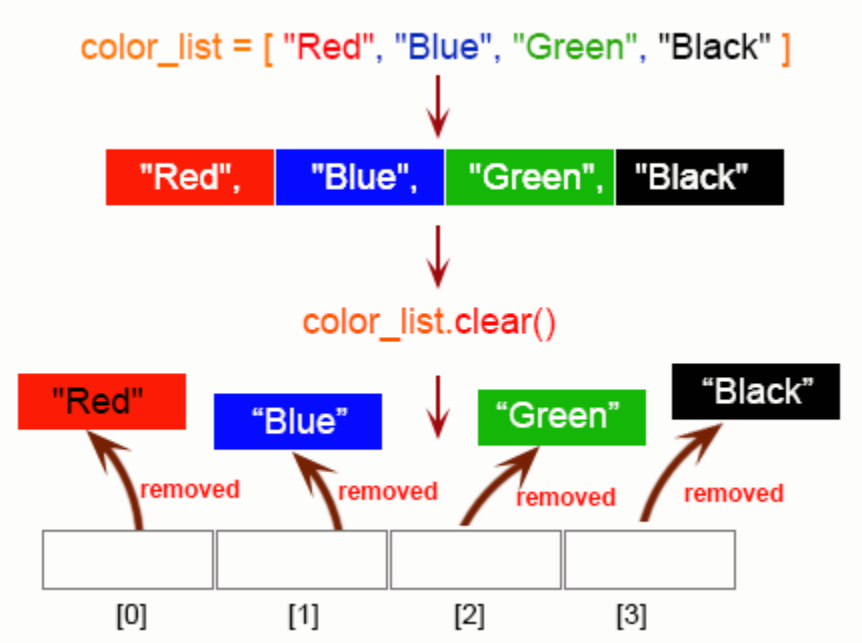
Remove an item from the list



See the following statements:

```
>>> color_list=["Red", "Blue", "Green", "Black"]
>>> print(color_list)
['Red', 'Blue', 'Green', 'Black']
>>> color_list.remove("Black")
>>> print(color_list)
['Red', 'Blue', 'Green']
```

Remove all items from the list



See the following statements:

```
>>> color_list=["Red", "Blue", "Green", "Black"]
>>> print(color_list)
['Red', 'Blue', 'Green', 'Black']
>>> color_list.clear()
>>> print(color_list)
[]
>>>
```

List Slices

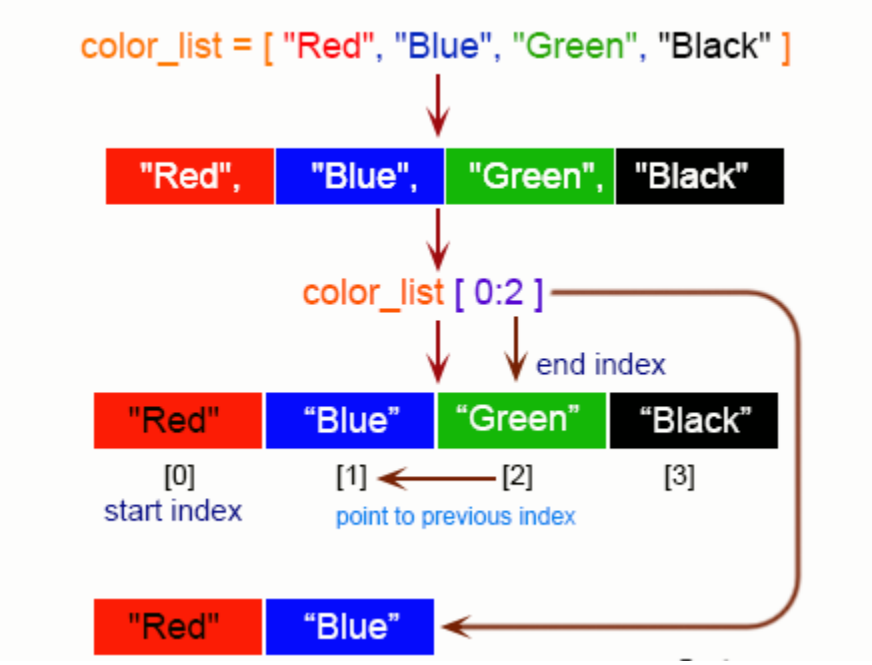
Lists can be sliced like strings and other sequences.

Syntax:

```
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 of the original list.

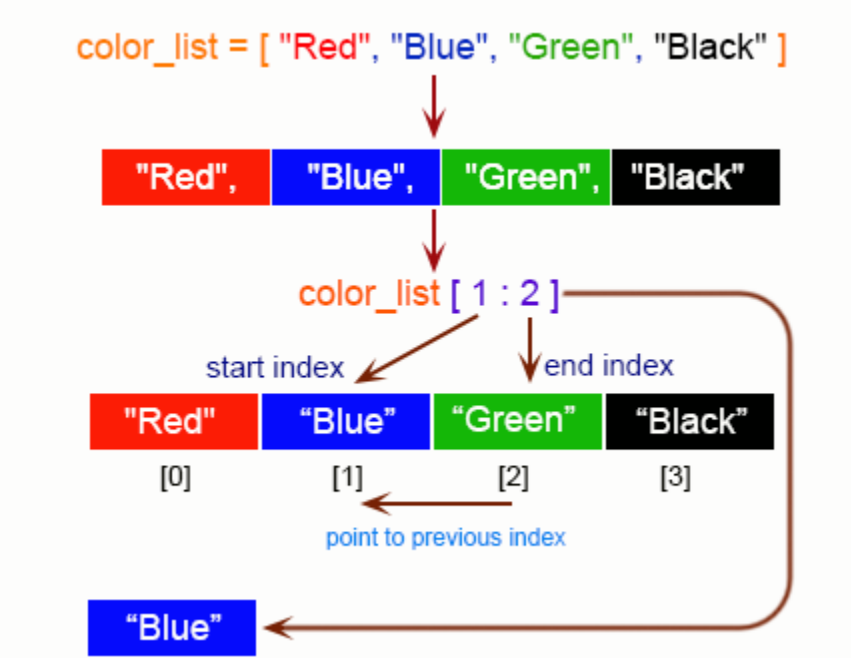
Cut first two items from a list:



See the following statements:

```
>>> 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']

>>>
```

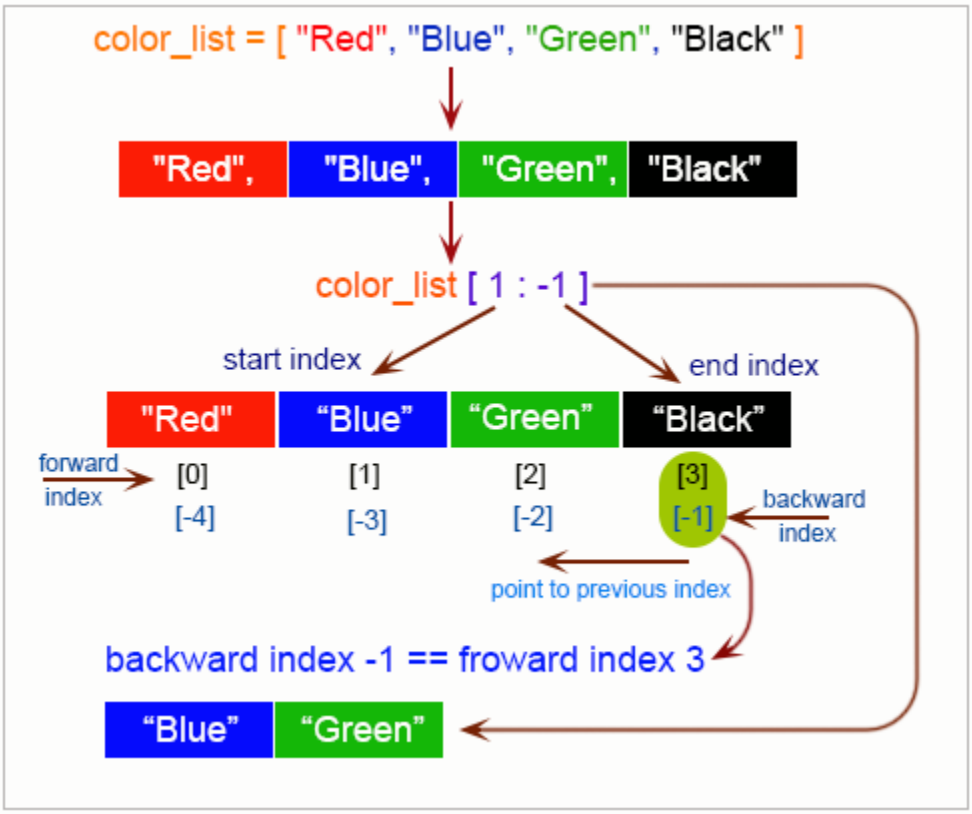
Cut second item from a list:



See the following statements:

```
>>> color_list=["Red", "Blue", "Green", "Black"] # The list have four elements

indices start at 0 and end at 3

>>> print(color_list[1:2])

['Blue']

>>> print(color_list[1:-2])

['Blue']

>>>
```

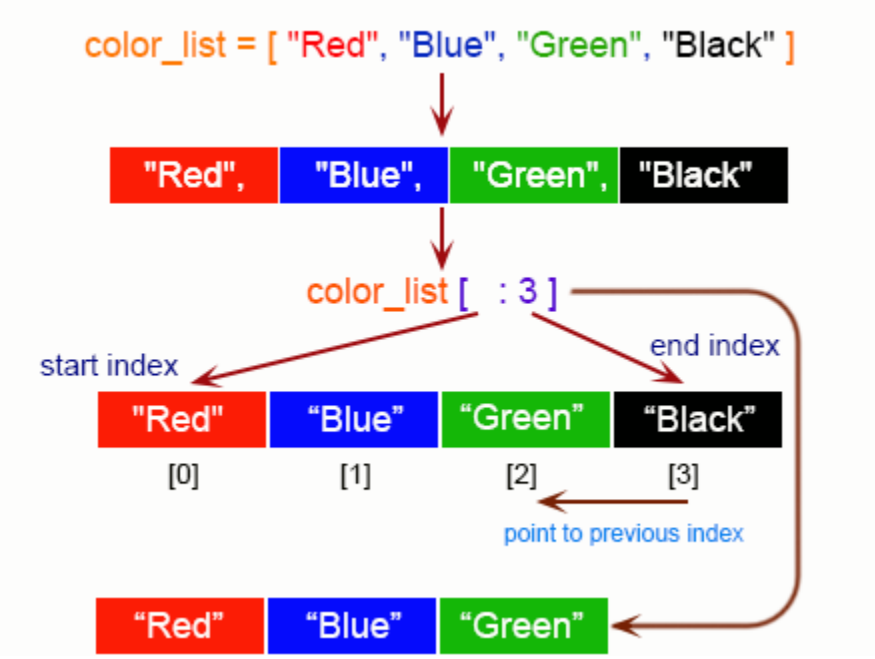
Cut second and third elements from a list:



See the following statements:

```
>>> color_list=["Red", "Blue", "Green", "Black"] # The list have four elements

indices start at 0 and end at 3

>>> print(color_list[1:-1])

['Blue', 'Green']

>>>
```

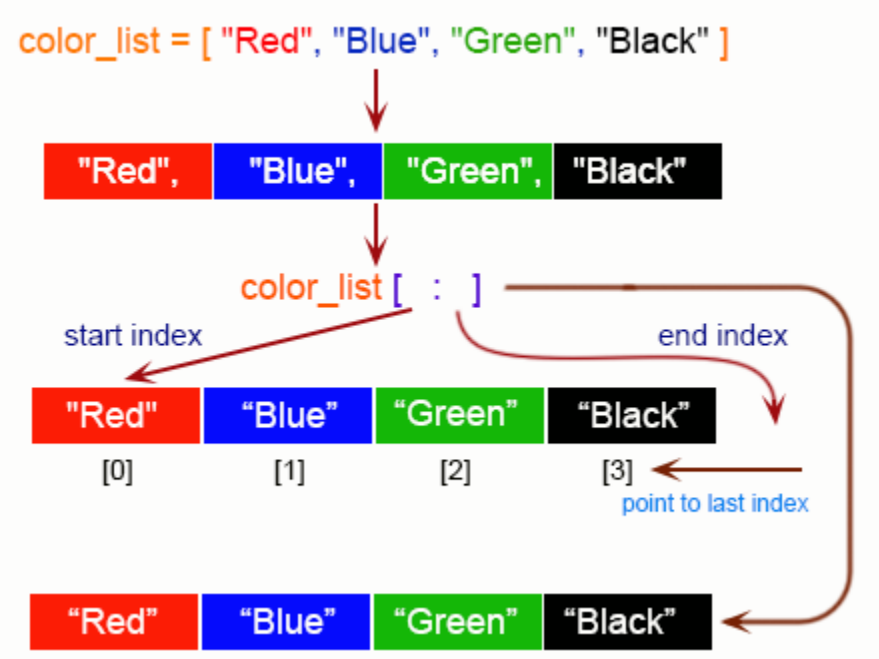
Cut first three items from a list:



See the following statements:

```
>>> color_list=["Red", "Blue", "Green", "Black"] # The list have four elements

indices start at 0 and end at 3

>>> print(color_list[:3]) # cut first three items

['Red', 'Blue', 'Green']

>>>
```

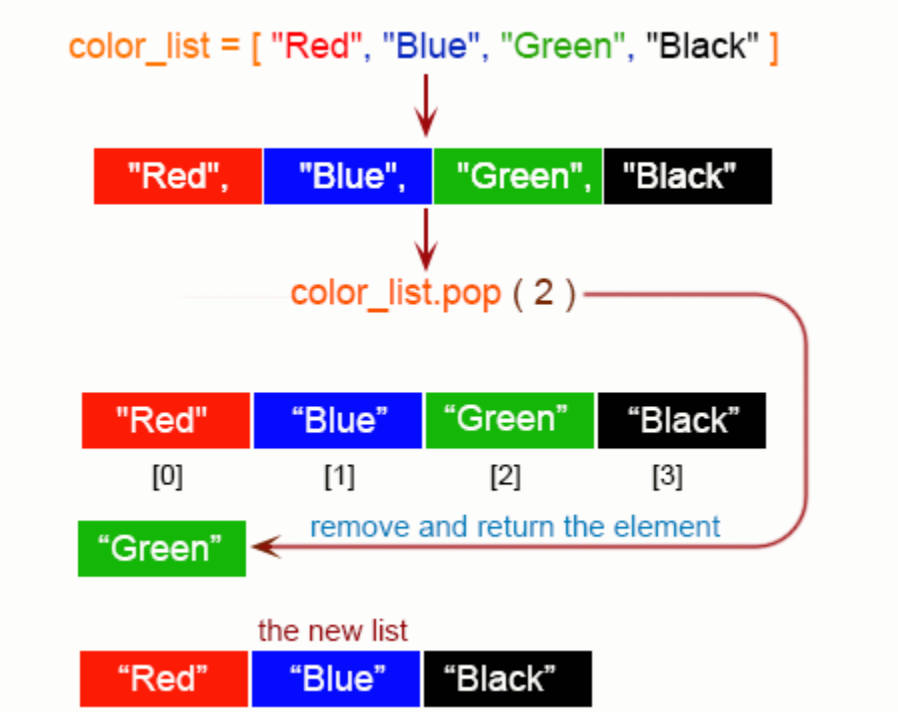
Creates of original list:



See the following statements:

```
>>> color_list=["Red", "Blue", "Green", "Black"] # The list have four elements

indices start at 0 and end at 3

>>> print(color_list[:]) # Creates   of original list

['Red', 'Blue', 'Green', 'Black']

>>>
```

Remove the item at the given position in the list, and return it

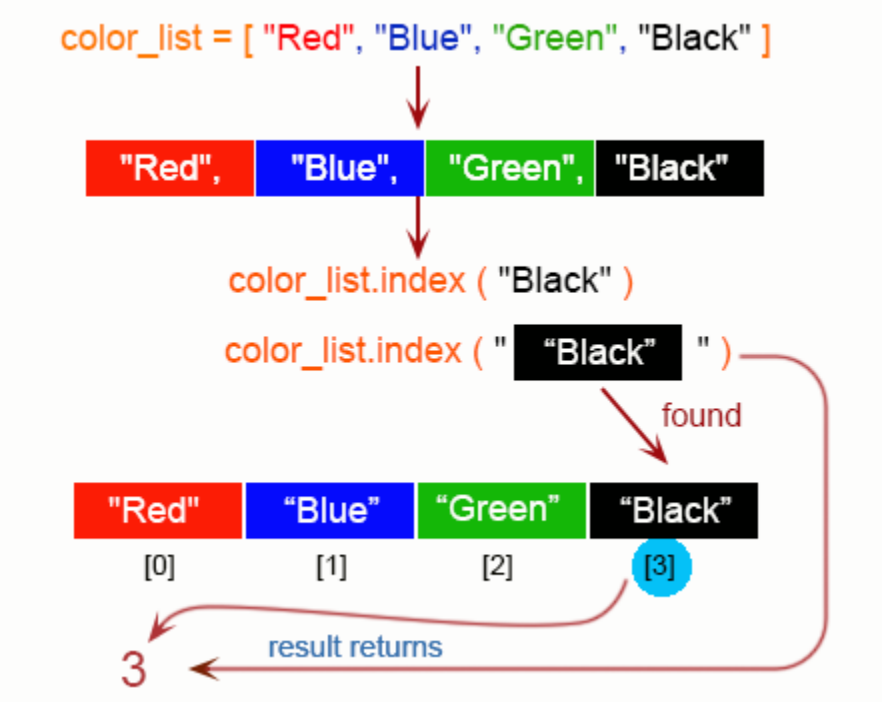


See the following statements:

```
>>> color_list=["Red", "Blue", "Green", "Black"]
>>> print(color_list)
['Red', 'Blue', 'Green', 'Black']
>>> color_list.pop(2) # Remove second item and return it
'Green'
>>> print(color_list)
['Red', 'Blue', 'Black']
>>>
```

Return the index in the list of the first item whose value is

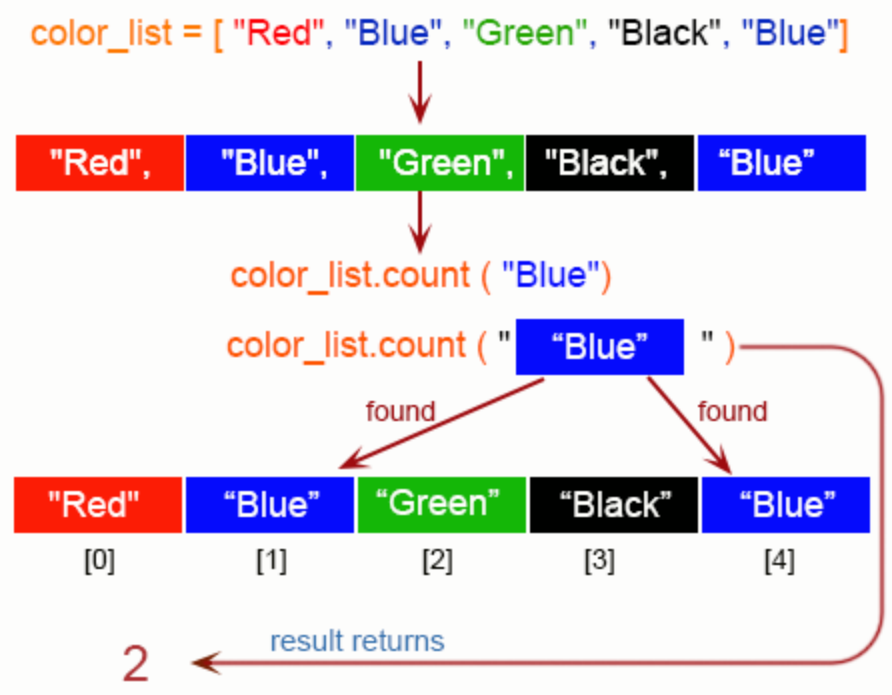
x



See the following statements:

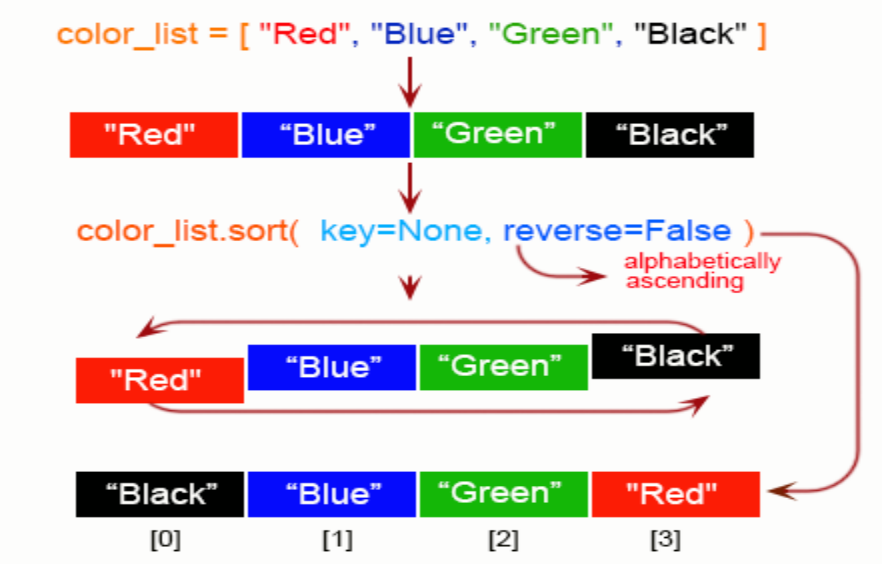
```
>>> color_list=["Red", "Blue", "Green", "Black"]
>>> print(color_list)
['Red', 'Blue', 'Green', 'Black']
>>> color_list.index("Red")
0
>>> color_list.index("Black")
3
>>>
```

Return the number of times 'x' appear in the list



See the following statements:

```
>>> color_list=["Red", "Blue", "Green", "Black"]
>>> print(color_list)
['Red', 'Blue', 'Green', 'Black']
>>> color_list=["Red", "Blue", "Green", "Black", "Blue"]
>>> print(color_list)
['Red', 'Blue', 'Green', 'Black', 'Blue']
>>> color_list.count("Blue")
2
>>>
```



Sort the items of the list in place

See the following statements:

```
>>> color_list=["Red", "Blue", "Green", "Black"]
>>> print(color_list)
['Red', 'Blue', 'Green', 'Black']
>>> color_list.sort(key=None, reverse=False)
>>> print(color_list)
['Black', 'Blue', 'Green', 'Red']
>>>
```

Reverse the elements of the list in place

```
>>> color_list=["Red", "Blue", "Green", "Black"]
>>> print(color_list)
```

```
['Red', 'Blue', 'Green', 'Black']
>>> color_list.reverse()
>>> print(color_list)
['Black', 'Green', 'Blue', 'Red']
>>>
```

Return a shallow of the list

```
>>> color_list=["Red", "Blue", "Green", "Black"]
>>> print(color_list)
['Red', 'Blue', 'Green', 'Black']
>>> color_list. ()
['Red', 'Blue', 'Green', 'Black']
>>>
```

Search the Lists and find Elements

```
>>> color_list=["Red", "Blue", "Green", "Black"]
>>> print(color_list)
['Red', 'Blue', 'Green', 'Black']
>>> color_list.index("Green")
2
>>>
```

Lists are Mutable

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

```
>>> color_list=["Red", "Blue", "Green", "Black"]

>>> print(color_list[0])

Red

>>> color_list[0]="White" # Change the value of first item "Red" to
"White"

>>> print(color_list)

['White', 'Blue', 'Green', 'Black']

>>> print(color_list[0])

White

>>>
```

Convert a list to a tuple in Python

```
>>> listx = [1, 2, 3, 4]

>>> print(listx)

[1, 2, 3, 4]

>>> tuplex = tuple(listx)

>>> print(tuplex)

(1, 2, 3, 4)

>>>
```

How to use the double colon [: :]?

```
>>> listx=[1, 5, 7, 3, 2, 4, 6]

>>> print(listx)

[1, 5, 7, 3, 2, 4, 6]

>>> sublist=listx[2:7:2] #list[start:stop:step], #step specify an
increment

between the elements to cut of the list.

>>> print(sublist)

[7, 2, 6]

>>> sublist=listx[::3] #returns a list with a jump every 2 times.

>>> print(sublist)

[1, 3, 6]

>>> sublist=listx[6:2:-2] #when step is negative the jump is made back

>>> print(sublist)

[6, 2]

>>>
```

Find the largest and the smallest item in a list

```
>>> listx=[5, 10, 3, 25, 7, 4, 15]

>>> print(listx)

[5, 10, 3, 25, 7, 4, 15]

>>> print(max(listx))       # the max() function of built-in allows to
know the highest

value in the list.
```

```
25

>>> print(min(listx)) #the min() function of built-in allows to know
the lowest

value in the list.

3

>>>
```

Compare two lists in Python

```
>>> listx1, listx2=[3, 5, 7, 9], [3, 5, 7, 9]

>>> print (listx1 == listx2)

True

>>> listx1, listx2=[9, 7, 5, 3], [3, 5, 7, 9]    #create two lists
equal, but unsorted.

>>> print(listx1 == listx2)

False

>>> listx1, listx2 =[2, 3, 5, 7], [3, 5, 7, 9]   #create two different
list

>>> print(listx1 == listx2)

False

>>> print(listx1.sort() == listx2.sort()) #order and compare

True

>>>
```

Nested lists in Python

```
>>> listx = [["Hello", "World"], [0, 1, 2, 3, 4, 5]]

>>> print(listx)

[['Hello', 'World'], [0, 1, 2, 3, 4, 5]]

>>> listx = [["Hello", "World"], [0, 1, 2, 3, 3, 5]]

>>> print(listx)

[['Hello', 'World'], [0, 1, 2, 3, 3, 5]]

>>> print(listx[0][1])            #The first [] indicates the index of
the outer list.

World

>>> print(listx[1][3])            #the second [] indicates the index
nested lists.

3

>>> listx.append([True, False])          #add new items

>>> print(listx)

[['Hello', 'World'], [0, 1, 2, 3, 3, 5], [True, False]]

>>> listx[1][2]=4

>>> print(listx)

[['Hello', 'World'], [0, 1, 4, 3, 3, 5], [True, False]]       #update
value items

>>>
```

How can I get the index of an element contained in the list?

```
>>> listy = list("HELLO WORLD")

>>> print(listy)
```

```
['H', 'E', 'L', 'L', 'O', ' ', 'W', 'O', 'R', 'L', 'D']

>>> index = listy.index("L")       #get index of the first item whose value is passed as parameter

>>> print(index)

2

>>> index = listy.index("L", 4)    #define the index from which you want to search

>>> print(index)

9

>>> index = listy.index("O", 3, 5) #define the segment of the list to be searched

>>> print(index)

4

>>>
```

Using Lists as Stacks

```
>>> color_list=["Red", "Blue", "Green", "Black"]

>>> print(color_list)

['Red', 'Blue', 'Green', 'Black']

>>> color_list.append("White")

>>> color_list.append("Yellow")

>>> print(color_list)

['Red', 'Blue', 'Green', 'Black', 'White', 'Yellow']

>>> color_list.pop()

'Yellow'
```

```
>>> color_list.pop()
'White'
>>> color_list.pop()
'Black'
>>> color_list
['Red', 'Blue', 'Green']
>>>
```

Using Lists as Queues

```
>>> from collections import deque
>>> color_list = deque(["Red", "Blue", "Green", "Black"])
>>> color_list.append("White")      # White arrive
>>> print(color_list)
deque(['Red', 'Blue', 'Green', 'Black', 'White'])
>>> color_list.append("Yellow")     # Yellow arrive
>>> print(color_list)
deque(['Red', 'Blue', 'Green', 'Black', 'White', 'Yellow'])
>>> color_list.popleft()            # The first to arrive now leaves
'Red'
>>> print(color_list)
deque(['Blue', 'Green', 'Black', 'White', 'Yellow'])
>>> color_list.popleft()            # The second to arrive now leaves
'Blue'
>>> print(color_list)
```

```
deque(['Green', 'Black', 'White', 'Yellow'])

>>> print(color_list)                # Remaining queue in order of
arrival

deque(['Green', 'Black', 'White', 'Yellow'])

>>>
```