List Slicing

In this notebook, we will discuss the slicing and the del operator in the context of lists.

Like in the case of strings, we can use the slicing feature of Python to extract or otherwise manipulate portions of a list. `[start_val:end_val:step_val]` . Recall that the slicing notation allows us to extract subsets of a list or string The slice begins with the element whose index is given by `start_val` (default 0), until 1 less than the `end_val` , in increments of the `step_val` (default is 1).

If the `end_val` is not provided, the default is until the end of the list. As seen earlier (in the case of strings), each of the parameters can be negative values as well.

The `del` operator can used together with list slicing to remove one or more elements from a list.

In [1]:

```
'''
The following are a few examples of list slicing
'''
my_lst = [1, 2, 3, 4, 5, 6, 7, 8]
print('my_lst[2:4]:', my_lst[2:4])           # from element at index 2 to element at index 3
print('my_lst[5:]:', my_lst[5:])             # from element at index 5 to end of list
print('my_lst[:5]:', my_lst[:5])             # from beginnning of list (element at index 0) to elem
ent at index 4
print('my_lst[:]:', my_lst[:])               # all elements in a list
print('my_lst[-1:-8:-1]:', my_lst[-1:-8:-1]) # from last element of list back to element at index -
8+1=-7 or len(list)-8+1
print('my_lst[-1:-9:-1]:', my_lst[-1:-9:-1]) # from last element of list back to element at index -
9+1=-8 or len(list)-9+1
```

```
my_lst[2:4]: [3, 4]
my_lst[5:]: [6, 7, 8]
my_lst[:5]: [1, 2, 3, 4, 5]
my_lst[:]: [1, 2, 3, 4, 5, 6, 7, 8]
my_lst[-1:-8:-1]: [8, 7, 6, 5, 4, 3, 2]
my_lst[-1:-9:-1]: [8, 7, 6, 5, 4, 3, 2, 1]
```

The `del` operator can be used together with slicing to delete multiple elements in a list. Note that this operator changes the original list.

In [3]:

```
'''
The del operator can be used together with slicing to delete multiple elements in a list.
Note that this operator
changes the original list.
'''
elements = ["A", "B", "C", "D"]
del elements[:1]                  # from beginning of list to element at index 0 -> first element
print(elements)

elements = ["A", "B", "C", "D"]
del elements[1:3]                 # from element at index 1 to element at index 2
print(elements)

elements = ["A", "B", "C", "D"]
del elements[1]                   # elemenet at index 1
print(elements)
```

```
['B', 'C', 'D']
['A', 'D']
['A', 'C', 'D']
```

In [4]:

```
'''
You can copy the contents of one list to another using slicing.
```

```
'''
elements = ["A", "B", "C", "D"]
new_elements = elements[:]
print(new_elements)
```

```
['A', 'B', 'C', 'D']
```

List can also be modified using list slicing. This is known as slice assignment and is done by using an appropriate slice expression on the left hand side of the assignment operator. In this notebook, we discuss a few possibilities. Note that slice assignment is only applicable with mutable objects like lists. Although tuple slicing and string slicing can be done, slice assignment is not possible with strings and tuples since they are immutable objects.

In [4]:

```
'''
if the slice expression on the left hand side has a single subscript,
then the length of the list is retained
but the value at that subscript is changed
'''
my_lst = [5, 8, 'hi', 12, 'why']
my_lst[2] = 'this'
print(my_lst)
```

```
[5, 8, 'this', 12, 'why']
```

In [5]:

```
'''
if the slice expression has a single subscript, then the element at that index is replaced by a si
ngle element
'''
my_lst = [5, 8, 'hi', 12, 'why']
my_lst[2] = ['hello', 'why','there']  #The RHS is added as a single element replacing the element
at index 2 in the original list
print(my_lst)
```

```
[5, 8, ['hello', 'why', 'there'], 12, 'why']
```

In [10]:

```
'''
if the slice expression is a range of values, those values are replaced by the values on the RHS.
Note that the
number of elements on the two sides need not match.
'''
my_lst = [5, 8, 'image', 'hi', 'guide', 12, 'why']
my_lst[3:4] = [9, 15, 78, 34] #The number of elements on the LHS (1) does not match the number of
elements on the RHS (4)
print(my_lst)
```

```
[5, 8, 'image', 9, 15, 78, 34, 'guide', 12, 'why']
```

In [8]:

```
'''
if the slice extracted by the slice expression is an empty list, then all elements in
the orginal list are retained with some new elements  being inserted into the list.

'''
my_lst = ['one', 'two', 'three', 'seven', 'eight']
my_lst[3:3] = ['four', 'five', 'six'] #insert ['four','five','six'] at index 3
print(my_lst)
```

```
['one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight']
```

In [9]:

```
'''
```

```
you can use slice assignment to delete portions of a list
'''
my_lst = ['one', 'dos', 'tres', 'cuatro', 'two', 'three', 'four', 'five']
my_lst[1:4] = []
print(my_lst)
```

```
['one', 'two', 'three', 'four', 'five']
```