Creation of List

Lists in Python can be created by just placing the sequence inside the square brackets[]. A list may contain duplicate values with their distinct positions and hence, multiple distinct or duplicate values can be passed as a sequence at the time of list creation.

In [2]:

```
my_first_list =  [2,25,17,30,31]
```

In [3]:

```
my_first_list
```

Out[3]:

```
[2, 25, 17, 30, 31]
```

In [4]:

```
# Creating a List
List = []
print("Intial blank List: ")
print(List)
```

```
Intial blank List:
[]
```

In [5]:

```
# Creating a List with
# the use of a String
List = ['DataScience']
print("\nList with the use of String: ")
print(List)
```

```
List with the use of String:
['DataScience']
```

In [11]:

```
# Creating a List with
# the use of multiple values
List = ["Python", "For", "Beginners"]
print("\nList containing multiple values: ")
print(List[0])
print(List[2])
```

```
List containing multiple values:
Python
Beginners
```

In [13]:

```
type(List[0])
```

Out[13]:

```
str
```

In [14]:

```
# Creating a Multi-Dimensional List
# (By Nesting a list inside a List)
List = [['Data', 'Science'] , ['Python']]
print("\nMulti-Dimensional List: ")
print(List)
```

```
Multi-Dimensional List:
[['Data', 'Science'], ['Python']]
```

In [22]:

```
type(List[1])
```

Out[22]:

```
list
```

In [23]:

```
# Creating a List with
# the use of Numbers
# (Having duplicate values)
List = [1, 2, 4, 4, 3, 3, 3, 6, 5]
print("\nList with the use of Numbers: ")
print(List)
```

```
List with the use of Numbers:
[1, 2, 4, 4, 3, 3, 3, 6, 5]
```

In [24]:

```
# Creating a List with
# mixed type of values
# (Having numbers and strings)
List = [1, 2, 'Python', 4, 'For', 6, 'Beginners']
print("\nList with the use of Mixed Values: ")
print(List)
```

```
List with the use of Mixed Values:
[1, 2, 'Python', 4, 'For', 6, 'Beginners']
```

In [29]:

```
type(List[2])
```

Out[29]:

```
str
```

In [30]:

```
List
```

Out[30]:

```
[1, 2, 'Python', 4, 'For', 6, 'Beginners']
```

Adding Elements to a List

Elements can be added to the List by using built-in append() function. Only one element at a time can be added to the list by using append() method, for addition of multiple elements with the append() method, loops are used. Tuples can also be added to the List with the use of append method because tuples are immutable. Unlike Sets, Lists can also be added to the existing list with the use of append() method. append() method only works for addition of elements at the end of the List, for addition of element at the desired position, insert() method is used. Unlike append() which takes only one argument, insert() method requires two arguments(position, value). Other than append() and insert() methods, there’s one more method for Addition of elements, extend(), this method is used to add multiple elements at the same time at the end of the list.

Note – append() and extend() methods can only add elements at the end.

In [35]:

```
#List.append('and')
#List.append('DataScience')
List
```

Out[35]:

```
[1, 2, 'Python', 4, 'For', 6, 'Beginners', 'and', 'and', 'DataScience']
```

In [39]:

```
# Python program to demonstrate
# Addition of elements in a List

# Creating a List
List = []
print("Intial blank List: ")
print(List)
```

```
Intial blank List:
[]
```

In [37]:

```
# Addition of Elements
# in the List
List.append(1)
List.append(2)
List.append(4)
print("\nList after Addition of Three elements: ")
print(List)
```

```
List after Addition of Three elements:
[1, 2, 4]
```

In [40]:

```
# Adding elements to the List
# using Iterator
for i in range(1, 4):
    List.append(i)
print("\nList after Addition of elements from 1-3: ")
print(List)
```

```
List after Addition of elements from 1-3: 
[1, 2, 3]
```

In [41]:

```
# Adding Tuples to the List
List.append((5, 6))
print("\nList after Addition of a Tuple: ")
print(List)
```

```
List after Addition of a Tuple: 
[1, 2, 3, (5, 6)]
```

In [45]:

```
List[3][1]
```

Out[45]:

```
6
```

In [46]:

```
# Addition of List to a List
List2 = ['For', 'Data']
List.append(List2)
print("\nList after Addition of a List: ")
print(List)
```

```
List after Addition of a List: 
[1, 2, 3, (5, 6), ['For', 'Data']]
```

In [53]:

```
#List.extend(List2)
List
```

Out[53]:

```
[1, 2, 3, (5, 6), ['For', 'Data'], 'For', 'Data']
```

In [54]:

```
# Addition of Element at
# specific Position
# (using Insert Method)
List.insert(3, 12)
List2.insert(0, 'Python')
print("\nList after performing Insert Operation: ")
print(List)
```

```
List after performing Insert Operation:
[1, 2, 3, 12, (5, 6), ['Python', 'For', 'Data'], 'For', 'Data']
```

In [56]:

```
List
```

Out[56]:

```
[1, 2, 3, 12, (5, 6), ['Python', 'For', 'Data'], 'For', 'Data']
```

In [57]:

```
# Addition of multiple elements
# to the List at the end
# (using Extend Method)
List.extend([8, 'Python', 'Always'])
print("\nList after performing Extend Operation: ")
print(List)
```

```
List after performing Extend Operation:
[1, 2, 3, 12, (5, 6), ['Python', 'For', 'Data'], 'For', 'Data', 8, 'Pytho
n', 'Always']
```

Accessing elements from the List

In order to access the list items refer to the index number.Use the index operator [ ] to access an item in a list.The index must be an integer.Nested list are accessed using nested indexing.

In [58]:

```
# Python program to demonstrate
# accessing of element from list

# Creating a List with
# the use of multiple values
List = ["Python", "For", "Beginners"]

# accessing a element from the
# list using index number
print("Accessing a element from the list")
print(List[0])
print(List[2])
```

```
Accessing a element from the list
Python
Beginners
```

In [59]:

```
# Creating a Multi-Dimensional List
# (By Nesting a list inside a List)
List = [['Data', 'For'] , ['Analyst']]
```

In [60]:

```
# accessing a element from the
# Multi-Dimensional List using
# index number
print("Acessing a element from a Multi-Dimensional list")
print(List[0][1])
print(List[1][0])
```

```
Acessing a element from a Multi-Dimensional list
For
Analyst
```

In [61]:

```
List = [1, 2, 'Python', 4, 'For', 6, 'Data Science']

# accessing a element using
# negative indexing
print("Acessing element using negative indexing")

# print the last element of list
print(List[-1])

# print the third last element of list
print(List[-3])
```

```
Acessing element using negative indexing
Data Science
For
```

Removing Elements from the List

Elements can be removed from the List by using built-in remove() function but an Error arises if element doesn't exist in the set. Remove() method only removes one element at a time, to remove range of elements, iterator is used. Pop() function can also be used to remove and return an element from the set, but by default it removes only the last element of the set, to remove element from a specific position of the List, index of the element is passed as an argument to the pop() method.

Note – Remove method in List will only remove the first occurrence of the searched element.

In [62]:

```
# Python program to demonstrate
# Removal of elements in a List

# Creating a List
List = [1, 2, 3, 4, 5, 6,
                7, 8, 9, 10, 11, 12]
print("Intial List: ")
print(List)
```

```
Intial List:
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
```

In [64]:

```
# Removing elements from List
# using Remove() method
List.remove(5)
List.remove(6)
print("\nList after Removal of two elements: ")
print(List)
```

```
---------------------------------------------------------------------------
-
ValueError                                Traceback (most recent call las
t)
<ipython-input-64-e385fe2339b1> in <module>
      1 # Removing elements from List
      2 # using Remove() method
----> 3 List.remove(5)
      4 List.remove(6)
      5 print("\nList after Removal of two elements: ")

ValueError: list.remove(x): x not in list
```

In [65]:

```
List
```

Out[65]:

```
[1, 2, 3, 4, 7, 8, 9, 10, 11, 12]
```

In [66]:

```
# Removing elements from List
# using iterator method
for i in range(1, 5):
    List.remove(i)
print("\nList after Removing a range of elements: ")
print(List)
```

```
List after Removing a range of elements:
[7, 8, 9, 10, 11, 12]
```

In [67]:

```
# Removing element from the
# Set using the pop() method
List.pop()
print("\nList after popping an element: ")
print(List)
```

```

List after popping an element:
[7, 8, 9, 10, 11]
```

In [68]:

```
# Removing element at a
# specific location from the
# Set using the pop() method
List.pop(2)
print("\nList after popping a specific element: ")
print(List)
```

```

List after popping a specific element:
[7, 8, 10, 11]
```

Slicing of a List

In Python List, there are multiple ways to print the whole List with all the elements, but to print a specific range of elements from the list, we use Slice operation. Slice operation is performed on Lists with the use of colon(:). To print elements from beginning to a range use [:Index], to print elements from end use [:-Index], to print elements from specific Index till the end use [Index:], to print elements within a range, use [Start Index:End Index] and to print whole List with the use of slicing operation, use [:]. Further, to print whole List in reverse order, use [::-1].

Note – To print elements of List from rear end, use Negative Indexes.

In [69]:

```
# Python program to demonstrate
# Removal of elements in a List

# Creating a List
List = ['G','E','E','K','S','F',
                'O','R','G','E','E','K','S']
print("Intial List: ")
print(List)
```

```
Intial List:
['G', 'E', 'E', 'K', 'S', 'F', 'O', 'R', 'G', 'E', 'E', 'K', 'S']
```

In [70]:

```
# Print elements of a range
# using Slice operation
Sliced_List = List[3:8]
print("\nSlicing elements in a range 3-8: ")
print(Sliced_List)
```

```
Slicing elements in a range 3-8:
['K', 'S', 'F', 'O', 'R']
```

In [71]:

```
# Print elements from beginning
# to a pre-defined point using Slice
Sliced_List = List[:-6]
print("\nElements sliced till 6th element from last: ")
print(Sliced_List)
```

```
Elements sliced till 6th element from last:
['G', 'E', 'E', 'K', 'S', 'F', 'O']
```

In [72]:

```
List
```

Out[72]:

```
['G', 'E', 'E', 'K', 'S', 'F', 'O', 'R', 'G', 'E', 'E', 'K', 'S']
```

In [75]:

```
# Print elements from a
# pre-defined point to end
Sliced_List = List[5:]
print("\nElements sliced from 5th "
        "element till the end: ")
print(Sliced_List)
```

```
Elements sliced from 5th element till the end:
['F', 'O', 'R', 'G', 'E', 'E', 'K', 'S']
```

In [76]:

```
# Printing elements from
# beginning till end
Sliced_List = List[:]
print("\nPrinting all elements using slice operation: ")
print(Sliced_List)
```

```
Printing all elements using slice operation:
['G', 'E', 'E', 'K', 'S', 'F', 'O', 'R', 'G', 'E', 'E', 'K', 'S']
```

In [77]:

```
# Printing elements in reverse
# using Slice operation
Sliced_List = List[::-1]
print("\nPrinting List in reverse: ")
print(Sliced_List)
```

```
Printing List in reverse:
['S', 'K', 'E', 'E', 'G', 'R', 'O', 'F', 'S', 'K', 'E', 'E', 'G']
```