Python for Selenium

List

- List type is another sequence type defined by the list class of python.
- List allows you add, delete or process elements in very simple ways.
- List is very similar to arrays.

Creating list in python

```
list1 = list() # Create an empty list
list2 = list([22, 31, 61]) # Create a list with elements 22, 31, 61
list3 = list(["tom", "jerry", "spyke"]) # Create a list with strings
list4 = list("python") # Create a list with characters p, y, t, h, o, n

print(list1)
print(list2)
print(list3)
print(list4)
```

Output:

```
[]
[22, 31, 61]
['tom', 'jerry', 'spyke']
['p', 'y', 't', 'h', 'o', 'n']
```

Accessing elements in list

- You can use index operator ([]) to access individual elements in the list. List index starts from 0.

```
l = [1,2,3,4,5]
print(l[1]) # access second element in the list   2
print(l[0]) # access first element in the list    1
```

List Common Operations

Method name	Description
x in s	True if element x is in sequence s.
x not in s	if element x is not in sequence s.
s1 + s2	Concatenates two sequences s1 and s2
s * n , n * s	n copies of sequence s concatenated
s[i]	ith element in sequence s.

Method name	Description
len(s)	Length of sequence s, i.e. the number of elements in s.
min(s)	Smallest element in sequence s.
max(s)	Largest element in sequence s.
sum(s)	Sum of all numbers in sequence s.
for loop	Traverses elements from left to right in a for loop.

```
list1 = [2, 3, 4, 1, 32]
print(2 in list1) # True
print(33 not in list1) #True
print(len(list1)) # find the number of elements in the list 5
print(max(list1)) # find the largest element in the list 32
print(min(list1)) # find the smallest element in the list 1
print(sum(list1)) # sum of elements in the list 42
```

List slicing

- Slice operator ([start:end]) allows to fetch sublist from the list. It works similar to string.

```
list = [11,33,44,66,788,1]
print(list[0:5]) # this will return list starting from index 0 to index 4
[11,33,44,66,788]
print(list[:3]) #Similar to string start index is optional, if omitted it will be 0.

print(list[2:]) #end index is also optional, if omitted it will be set to the last
index of the list.   # [44,66,788,1]
```

+ and * operators in list

- + operator joins the two list.
- * operator replicates the elements in the list.

```
list1 = [11, 33]
list2 = [1, 9]
list3 = list1 + list2
print(list3) #[11, 33, 1, 9]

list4 = [1, 2, 3, 4]
list5 = list4 * 3
print(list5) # [1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4]
```

in or not in operator

- in operator is used to determine whether the elements exists in the list. On success it returns True on failure it returns False .
- Similarly not in is opposite of in operator.

```
list1 = [11, 22, 44, 16, 77, 98]

print(22 in list1) #True
print(22 not in list1) #False
```

Traversing list using for loop

- We can use for loop to loop through all the elements of the list.

```
list = [1,2,3,4,5]
for i in list:
    print(i, end=" ") #1 2 3 4 5
```

Commonly used list methods with return type

Method name	Description
append(x:object):None	Adds an element x to the end of the list and returns None.
count(x:object):int	Returns the number of times element x appears in the list.
extend(l:list):None	Appends all the elements in l to the list and returns None.
index(x: object):int	Returns the index of the first occurrence of element x in the list
insert(index: int, x:object):None	Inserts an element x at a given index. Note that the first element in the list has index 0 and returns None..
remove(x:object):None	Removes the first occurrence of element x from the list and returns None
reverse():None	Reverse the list and returns None
sort(): None	Sorts the elements in the list in ascending order and returns None.
pop(i): object	Removes the element at the given position and returns it. The parameter i is optional. If it is not specified, pop() removes and returns the last element in the list.

```
list1 = [2, 3, 4, 1, 32, 4]
list1.append(19)
print(list1) #[2, 3, 4, 1, 32, 4, 19]

print(list1.count(4)) # Return the count for number 4 result:2

list2 = [99, 54]
list1.extend(list2)
print(list1) #[2, 3, 4, 1, 32, 4, 19, 99, 54]

print(list1.index(4)) # Return the index of number 4 result:2

list1.insert(1, 25) # Insert 25 at position index 1
print(list1) #result:[2, 25, 3, 4, 1, 32, 4, 19, 99, 54]

list1 = [2, 25, 3, 4, 1, 32, 4, 19, 99, 54]
print(list1.pop(2)) #3
print(list1) #[2, 25, 4, 1, 32, 4, 19, 99, 54]

list1.remove(32) # Remove number 32
print(list1)#[2, 25, 4, 1, 4, 19, 99, 54]

list1.reverse() # Reverse the list
print(list1) #[54, 99, 19, 4, 1, 4, 25, 2]

list1.sort() # Sort the list
print(list1) #[1, 2, 4, 4, 19, 25, 54, 99]
```

List Comprehension

```
list1 = [ x for x in range(10) ]
print(list1) # [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

list2 = [ x + 1 for x in range(10) ]
print(list2) #[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

list3 = [ x for x in range(10) if x % 2 == 0 ]
print(list3) #0, 2, 4, 6, 8]
```

What is Dictionary

- Dictionary is a python data type that is used to store key value pairs.
- It enables you to quickly retrieve, add, remove, modify, values using key.
- Dictionary is very similar to HashMap in Java.
- Dictionaries are mutable.

Creating Dictionary

- Dictionaries can be created using pair of curly braces ({}).
- Each item in the dictionary consist of key, followed by a colon, which is followed by value.
- And each item is separated using commas (,).
- key must be of hashable type, but value can be of any type. Each key in the dictionary must be unique.

```
#Creating Dictionary
friends = {
'tom' : '111-222-333',
'jerry' : '666-33-111'
}
print(friends) #{'tom': '111-222-333', 'jerry': '666-33-111'}
```

Retrieving, modifying and adding elements in the dictionary

```
friends = {'tom' : '111-222-333','jerry' : '666-33-111'}
print(friends) #{'tom': '111-222-333', 'jerry': '666-33-111'}

#Retrieving elements from the dictionary
print(friends['tom']) # 111-222-333

#Adding elements into the dictionary
friends['bob'] = '888-999-666'
print(friends) #{'tom': '111-222-333', 'jerry': '666-33-111', 'bob': '888-999-
666'}

#Modify elements into the dictionary
friends['bob'] = '888-999-777'
print(friends) #{'tom': '111-222-333', 'jerry': '666-33-111', 'bob': '888-999-
777'}

#Delete element from the dictionary
del friends['bob']
print(friends) #{'tom': '111-222-333', 'jerry': '666-33-111'}
```

Looping items in the dictionary

```
friends = {'a' : '100',
           'b' : '200',
           'c' :'300',
           'd' : '400'
           }
for x in friends:
    print(x, ":", friends[x])
```

Find the length of the dictionary

- We can use len() function to find the length of the dictionary.

```
friends = {'a' : '100',
           'b' : '200',
           'c' :'300',
           'd' : '400'
           }
print(len(friends)) #4
```

Equality Tests in dictionary

- == and != operators tells whether dictionary contains same items not.

- **Note**: You can't use other relational operators like < , > , >= , <= to compare dictionaries.

```
d1 = {"mike":41, "bob":3}
d2 = {"bob":3, "mike":41}

print(d1 == d2) #True
print(d1 != d2) # False
```

Dictionary methods

Methods	Description
popitem()	Returns randomly select item from dictionary and also remove the selected item.
clear()	Delete everything from dictionary
keys()	Return keys in dictionary as tuples
values()	Return values in dictionary as tuples
get(key)	Return value of key, if key is not found it returns None, instead on throwing KeyError exception
pop(key)	Remove the item from the dictionary, if key is not found KeyError will be thrown

```
friends = {'tom': '111-222-333', 'bob': '888-999-666', 'jerry': '666-33-111'}

#popitem() Returns randomly select item from dictionary and also remove the selected item.
print(friends.popitem()) #('jerry', '666-33-111')

#clear()   Delete everything from dictionary
print(friends.clear()) #None

friends = {'tom': '111-222-333', 'bob': '888-999-666', 'jerry': '666-33-111'}

#keys()    Return keys in dictionary as tuples
print(friends.keys()) #dict_keys(['tom', 'bob', 'jerry'])

#values()  Return values in dictionary as tuples
print(friends.values()) #dict_values(['111-222-333', '888-999-666', '666-33-111'])

#get(key) Return value of key, if key is not found it returns None, instead on throwing
KeyError exception
print(friends.get('tom')) #111-222-333

#pop(key) Remove the item from the dictionary, if key is not found KeyError will be thrown
print(friends.pop('bob')) #888-999-666
print(friends) #{'tom': '111-222-333', 'jerry': '666-33-111'}
```