Lists

In [1]:
```
my_empty_list = []
print('empty list: {}, type: {}'.format(my_empty_list, type(my_empty_list)))
```

```
empty list: [], type: <class 'list'>
```

In [2]:
```
list_of_ints = [1, 2, 6, 7]
list_of_misc = [0.2, 5, 'Python', 'is', 'still fun', '!']
print('lengths: {} and {}'.format(len(list_of_ints), len(list_of_misc)))
```

```
lengths: 4 and 6
```

Accessing values

In [3]:
```
my_list = ['Python', 'is', 'still', 'cool']
print(my_list[0])
print(my_list[3])
```

```
Python
cool
```

In [4]:
```
coordinates = [[12.0, 13.3], [0.6, 18.0], [88.0, 1.1]]  # two dimensional
print('first coordinate: {}'.format(coordinates[0]))
print('second element of first coordinate: {}'.format(coordinates[0][1]))
```

```
first coordinate: [12.0, 13.3]
second element of first coordinate: 13.3
```

Updating values

In [5]:
```
my_list = [0, 1, 2, 3, 4, 5]
my_list[0] = 99
print(my_list)

# remove first value
del my_list[0]
print(my_list)
```

```
[99, 1, 2, 3, 4, 5]
[1, 2, 3, 4, 5]
```

Checking if certain value is present in list

In [6]:
```
languages = ['Java', 'C++', 'Go', 'Python', 'JavaScript']
if 'Python' in languages:
    print('Python is there!')
```

```
Python is there!
```

In [7]:
```
if 6 not in [1, 2, 3, 7]:
    print('number 6 is not present')
```

```
number 6 is not present
```

List are mutable

In [8]:
```
original = [1, 2, 3]
modified = original
modified[0] = 99
print('original: {}, modified: {}'.format(original, modified))
```

```
original: [99, 2, 3], modified: [99, 2, 3]
```

You can get around this by creating new `list`:

In [9]:
```
original = [1, 2, 3]
modified = list(original)  # Note list()
# Alternatively, you can use copy method
# modified = original.copy()
modified[0] = 99
print('original: {}, modified: {}'.format(original, modified))
```

```
original: [1, 2, 3], modified: [99, 2, 3]
```

`list.append()`

In [10]:
```
my_list = [1]
my_list.append('ham')
print(my_list)
```

```
[1, 'ham']
```

`list.remove()`

In [14]:
```
my_list = ['Python', 'is', 'sometimes', 'fun']
my_list.remove('sometimes')
print(my_list)

# If you are not sure that the value is in list, better to check first:
if 'Java' in my_list:
    my_list.remove('Java')
else:
    print('Java is not part of this story.')
#my_list
```

```
['Java', 'is', 'fun']
```

Out[14]:
```
['is', 'fun']
```

`list.sort()`

In [15]:
```
numbers = [8, 1, 6, 5, 10]
numbers.sort()
print('numbers: {}'.format(numbers))

numbers.sort(reverse=True)
print('numbers reversed: {}'.format(numbers))

words = ['this', 'is', 'a', 'list', 'of', 'words']
words.sort()
print('words: {}'.format(words))
```

```
numbers: [1, 5, 6, 8, 10]
numbers reversed: [10, 8, 6, 5, 1]
words: ['a', 'is', 'list', 'of', 'this', 'words']
```

`sorted(list)`

While `list.sort()` sorts the list in-place, `sorted(list)` returns a new list and leaves the original untouched:

In [16]:
```
numbers = [8, 1, 6, 5, 10]
sorted_numbers = sorted(numbers)
print('numbers: {}, sorted: {}'.format(numbers, sorted_numbers))
```

```
numbers: [8, 1, 6, 5, 10], sorted: [1, 5, 6, 8, 10]
```

`list.extend()`

In [19]:
```
first_list = ['beef', 'ham']
second_list = ['potatoes',1 ,3]
first_list.extend(second_list)
print('first: {}, second: {}'.format(first_list, second_list))
```

```
None
first: ['beef', 'ham', 'potatoes', 1, 3], second: ['potatoes', 1, 3]
```

Alternatively you can also extend lists by summing them:

In [21]:
```
first = [1, 2, 3]
second = [4, 5]
first += second  # same as: first = first + second
print('first: {}'.format(first))

# If you need a new list
summed = first + second
print('summed: {}'.format(summed))
```

```
first: [1, 2, 3, 4, 5]
summed: [1, 2, 3, 4, 5, 4, 5]
```

`list.reverse()`

In [20]:
```
my_list = ['a', 'b', 'ham']
my_list.reverse()
print(my_list)
```

```
['ham', 'b', 'a']
```

In [ ]:
```
name :poonam Pardeshi
```