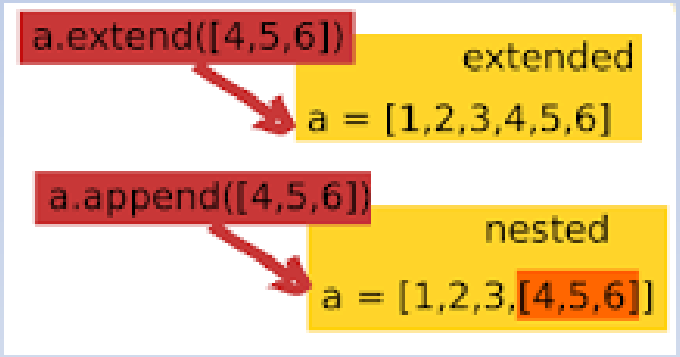
The `list` Object

- A list is an ordered sequence of Python objects
  - Objects can be of any type
  - Objects do not have to all be the same type
  - Constructed by writing items enclosed in square brackets … items separated by comma

```
team = ["Seahawks", 2014, "CenturyLink Field"]
nums = [5, 10, 4, 5]
words = ["spam", "ni"]
```

The `list` Object

- Some common List operations
  - len, max, min, sum, count, index, reverse, extend, append, del, clear, remove, insert, +, *
    - Append Vs Extend
      - Object added at the end Vs individual items added at the end



The `list` Object

```
team = ["Seahawks", 2014, "CenturyLink Field"]
nums = [5, 10, 4, 5]
words = ["spam", "ni"]

print(len(words))
print(max(nums))
print(min(nums))
print(sum(nums))
print(nums.count(5))
print(nums.index(4))
words.reverse()
print(words)
print(team)
nums.extend([1, 2])
print(nums)
nums.append([7, 8])
print(nums)
del(team[-1])
print(team)
team.clear()
print(team)
nums.remove(5)
print(nums)
words.insert(1,"wink")
print(words)
newlist = ['a',1] + [2, 'b']
print(newlist)
newlist = [0] * 3
print(newlist)
```

```
2
10
4
24
2
2
['ni', 'spam']
['Seahawks', 2014, 'CenturyLink Field']
[5, 10, 4, 5, 1, 2]
[5, 10, 4, 5, 1, 2, [7, 8]]
['Seahawks', 2014]
[]
[10, 4, 5, 1, 2, [7, 8]]
['ni', 'wink', 'spam']
['a', 1, 2, 'b']
[0, 0, 0]
```

The `list` Object

- Example: Program requests five grades as input, displays average after dropping two lowest grades

```
## Calculate average of grades.
grades = []    # Create the variable grades and assign it the empty list.
num = float(input("Enter the first grade: "))
grades.append(num)
num = float(input("Enter the second grade: "))
grades.append(num)
num = float(input("Enter the third grade: "))
grades.append(num)
num = float(input("Enter the fourth grade: "))
grades.append(num)
num = float(input("Enter the fifth grade: "))
grades.append(num)
minimumGrade = min(grades)
grades.remove(minimumGrade)
minimumGrade = min(grades)
grades.remove(minimumGrade)
average = sum(grades) / len(grades)
print("Average Grade: {0:.2f}".format(average))
```

Slices

- A *slice* of a list is a sublist specified with colon notation
  - Analogous to a slice of a string
- Meanings of slice notations

Slice Notation	Meaning
list1[*m*:*n*]	list consisting of the items of *list1* having indices *m* through *n* – 1
list1[:]	a new list containing the same items as *list1*
list1[*m*:]	list consisting of the items of *list1* from list1[*m*] through the end of *list1*
list1[:*m*]	list consisting of the items of *list1* from the beginning of *list1* to the element having index *m* – 1

Slices

- Examples of slices where list1 = ['a', 'b', 'c', 'd', 'e', 'f'].

Example	Value
list1[1:3]	['b', 'c']
list1[-4:-2]	['c', 'd']
list1[:4]	['a', 'b', 'c', 'd'']
list1[4:]	['e', 'f']
list1[:]	['a', 'b', 'c', 'd', 'e', 'f']
del list1[1:3]	['a', 'd', 'e', 'f']
list1[2:len(list1)]	['c', 'd', 'e', 'f']
(list1[1:3])[1]	'c' (This expression is usually written as list1[1:3][1])
list1[3:2]	[], the list having no items; that is, the empty list

Printing List Object Vs List Items

```
lst = [100, "test", "printing", 55, 725.30, "entire", 333]
```

- Print List Object

```
print(lst)
```

```
[100, 'test', 'printing', 55, 725.3, 'entire', 333]
```

- Print List Items individually

```
while i < len(lst):
    print(lst[i], end="\t")
    i += 1
```

```
100     test    printing        55      725.3   entire  333
```