Introduction to Python

Feature Summary, Session 4

FUNCTIONS / OPERATIONS

lists, tuples, sets

- **len()** function: ***return integer length of a container***
  argument: list, tuple or set
  return value: integer

```
items = ['th','tha','oth']
print(len(items))    # 3
```

- **sum()**, **min()**, **max()** functions: ***derive a value from numbers***
  argument: list, tuple or set
  return value: integer or float

```
items = [0.2, 0.3, 0.4]
print(sum(items))    # 0.9
```

- **sorted()** function: ***return a sorted list of items***
  argument: list, tuple or set
  return value: a list of the items in the container, sorted by value

```
items = [0.3, 1.3, 0.6]
sx = sorted(items)
print(sx)  # [0.3, 0.6, 1.3]
```

- **for** looping: ***iterate over each item in a container***
  as Python iterates over each item in a container, it assigns the item to the loop variable (at right, ***item****), then executes the block. The assignment and block execution happen once for each item in the container.*
  argument: N/A (iterates over list, tuple or set)
  return value: with each iteration, object from the container is assigned to the loop variable

```
items = ['a', 'b', 'c']
for item in items:
    print(item)
```

- **in** membership test: ***return*** *True* ***if item is in a container,*** *False* ***otherwise***
  "arguments": value to search for, and container (list, set, tuple)
  return value: **True** if object value is in the container;
  False if the value is not in the container

```
mylist = ['a', 'b', 'c']
if 'a' in mylist:
    print("it's there")
```

lists and tuples

- **+** operator: ***concatenate two lists or tuples***
  "arguments": two lists or two tuples
  "return value": a new list or tuple combining both

```
mya = ['a', 'b']
myb = ['c', 'd']
myc = mya + myb
      # ['a', 'b', 'c', 'd']
```

- **subscript:** ***select an item from a list or tuple***
  index: int position of item counting from 0
  return value: whatever object is referenced in the selected item of the list

```
items = ['th', 'tha','oth']
second = items[1]    #'tha'
```

- **slice:** ***select items from a list or tuple by index***
  a new list or tuple is returned based on indices of list or tuple items
  indices: integers representing the first item (lower bound) and one index past the last item (upper bound) in the resulting slice
  return value: a list or tuple (same type as source) with selected items

```
myd = ['a', 'b', 'c', 'd']
mys = myd[1:3]
print(mys)    # ['b', 'c']
```

- **range()**: ***generate an iterator of integers***
  generate an "iterator" that allows looping over a sequence of consecutive integers. To generate a list of such integers, pass to **list()**

```
numrange = range(1, 5)
for num in numrange:
    print(num)  # prints 1-4
numlist = list(range(1, 5))
print(numlist) #[1, 2, 3, 4]
```

METHODS

- list **append()** method: ***add an item to end of list***
  argument: object to append to this list
  return value: None

```
myl = ['a', 'b']
myl.append('c')
print(myl)   # ['a','b','c']
```

- set **add()** method: ***add an item to end of list***
  argument: object to add to this set
  return value: None

```
mys = {'a', 'b'}
mys.add('c')
print(mys)   # {'a','b','c'}
```