Lists and sets

CS195 - Lecture 8
Instructor: Dr. V

Lecture 8

- lists
  - lists vs tuples
  - get value at index, get slice
  - append, extend
  - set value at index
  - pop, index, remove, clear, reverse, count
- sets
  - add/remove
  - union (a|b), intersection (a&b), difference (a-b)
- lists, sets, and tuples
  - commonalities and differences

Python types

- immutable
  - single value:
    - **bool, int, float, None**
  - iterable:
    - **str, tuple**
- mutable
  - iterable:
    - **list, set, dict**

python lists

- lists are very much like tuples, but they are **mutable**
  - a tuple is a finite ordered sequence of items
  - a list is an ordered sequence of items

- tuples are more memory efficient
- changing or appending an item in a list is faster than trying to do the equivalent with tuples
  - to change a tuple you have to create a new tuple and delete the old one

tuples and lists

```python
t = ('apple', 'banana', 'cherry')
l = ['apple', 'banana', 'cherry']
# get length of tuple/list
print( len(t) )
print( len(l) )
# get membership in tuple/list
print( 'date' in t )
print( 'date' in t )
# get item at index in tuple/list
print( t[0] )
print( l[0] )
# get slice of each
print( t[1:3] )
print( l[1:3] )

```

tuples vs lists

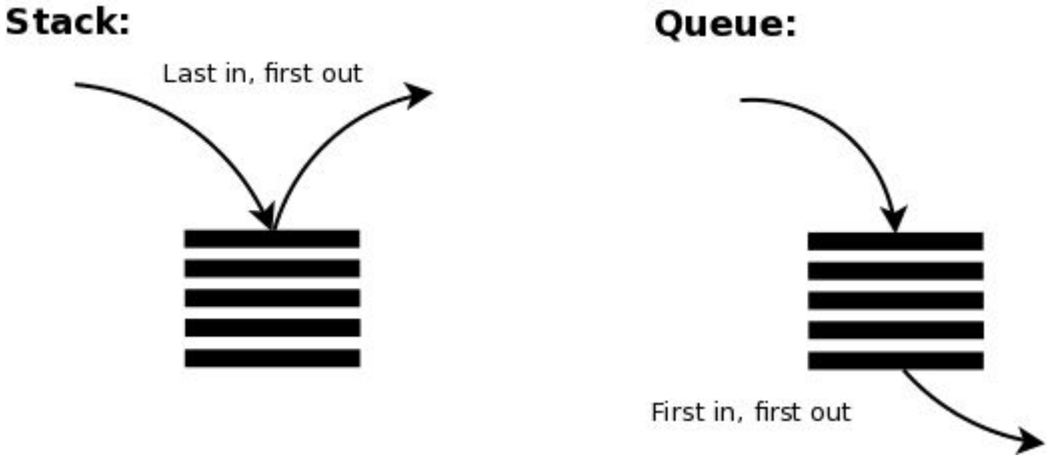
```python
l = ['apple', 'banana', 'cherry']
t = ('apple', 'banana', 'cherry')
# add one item to each
l.append('date')
t += ('date',)
# add multiple items to each
l.extend( ('elderberry','fig') ) # can extend w/ any iterable
t += ('elderberry','fig')        # can only add tuples
# set item at index
l[1] = 'blueberry'
t = t[:1] + ('blueberry',) + t[2:]
# insert item at index
l.insert(1, 'blackberry')
t = t[:1] + ('blackberry',) + t[1:]

```

When should you use tuples vs lists?

list methods

```python
l = ['apple', 'banana', 'cherry']

l.append('date') #add one item
l.extend( ['date','elderberry','fig'] ) #extend list

print( l.pop() ) #pop() removes and returns last item
print( l.pop(0) ) #pop(i) removes and returns item at index i
print( l.index('date') ) #index(item) returns the index of item
print( l.count('date') ) #count(item) returns count of item

l.remove('date') #remove(item) finds item and removes item
del l[1] #remove item at index 1

l.reverse() #reverses the order of items in list
l.clear() #clears entire list
```

LIFO vs FIFO



Stack:

Last in, first out

Queue:

First in, first out

using list as LIFO queue (aka stack)

```python
# what do you think this code prints?
myStack = []
myStack.append('apple')
myStack.append('banana')
myStack.append('cherry')

print( myStack.pop() )
print( myStack.pop() )

myStack.append('date')
myStack.append('elderberry')

print( myStack.pop() )

print( myStack )
```

using list as FIFO queue

```python
# what do you think this code prints?
myQ = []
myQ.append('apple')
myQ.append('banana')
myQ.append('cherry')

print( myQ.pop(0) )
print( myQ.pop(0) )

myQ.append('date')
myQ.append('elderberry')

print( myQ.pop(0) )

print( myQ )
```

python sets

- sets are
  - iterable sequences (like lists)
  - mutable (like lists)
  - **unordered**
  - each item in a set is **unique** and **immutable**

- finding an item in a set is very fast
  - useful for fast membership lookup or deletion
- you cannot look up or change item at index, because sets are unordered

python sets

```python
s = set() # create empty set

s.add('apple')  # add items to set, one at a time
s.add('banana')
s.add('cherry')
s.add('apple')
s.add('banana')

# what do you think this prints?
print(len(s))
print(s)
```

python sets

```python
s = {'banana','apple','cherry'}

s.add('elderberry') # add item to set
s.remove('banana')  # remove item from set

s.update( [1,2,3] ) # update set from sequence
s.update( (1,2), (3,4), (5,6) ) # update set from sequences

s.remove( 40 ) # <--- will throw an Error!
s.discard( 40 ) # remove item, if it exists

# remove multiple items, if they exist
s.difference_update( (4,5,6,7,8) )

s.clear() # remove all items from set
```

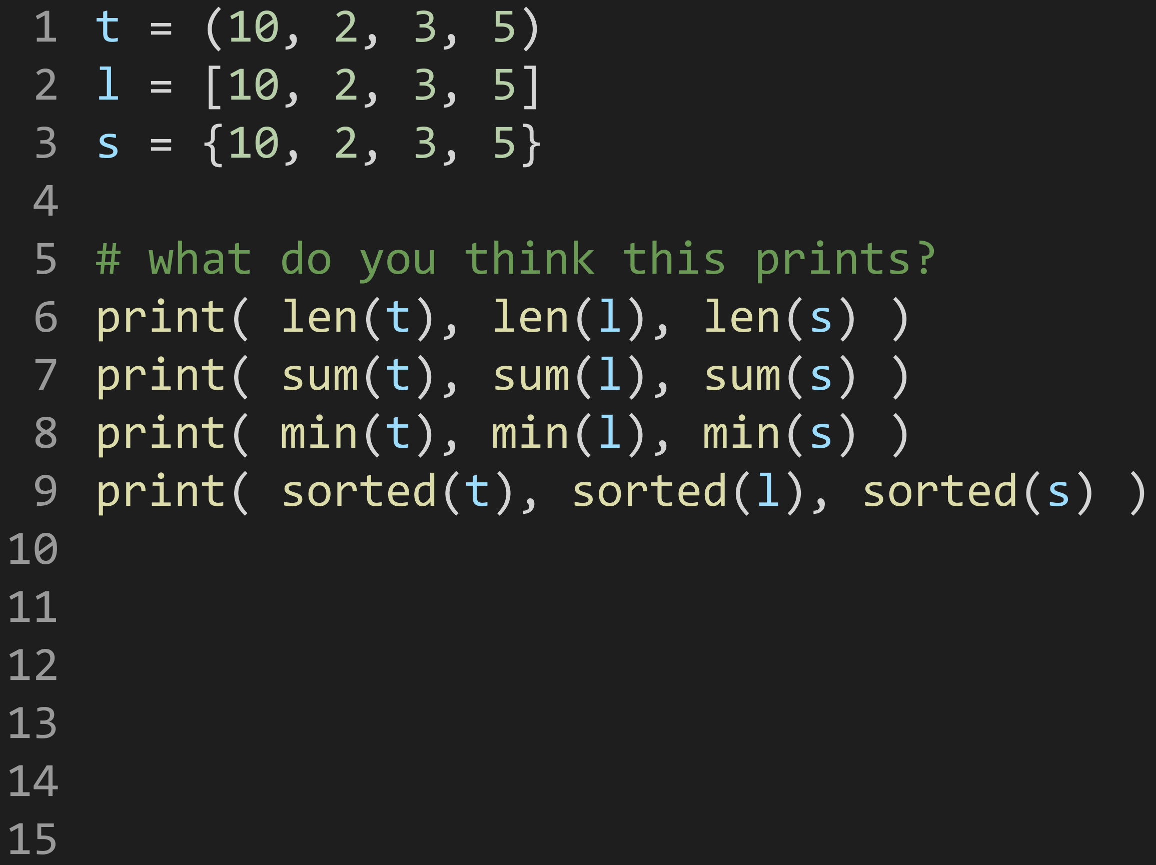
python sets

```python
s1 = {1,2,3,4,5}
s2 = {4,5,6,7,8}

print(s1|s2) # union of two sets
print(s2-s1) # difference between two sets
print(s1&s2) # intersection of two sets

print( s1.union( (4,5,6,7,8) ) )
print( s1.difference((4,5,6)) )
print( s1.intersection((4,5,6)) )
print( s1.symmetric_difference((4,5,6)) )

print( s1.issubset(range(10)) )
print( s1.issuperset([2,3]) )
print( s1.isdisjoint([11,12]) )
```

initiating tuples, lists, and sets

```python
t = ()               # create empty tuple
t = tuple()
t = 1, 2, 3          # create tuple with initial values
t = (1, 2, 3)
t = tuple( (1,2,3) ) # create tuple from sequence

l = []               # create empty list
l = list()
l = [1, 2, 3]        # create list with initial values
l = list( (1,2,3) )  # create list from sequence

s = set()            # create empty set
s = {1, 2, 3}        # create set with initial values
s = set( (1,2,3) )   # create set from sequence

```

len, sum, min,max, sorted



```python
t = (10, 2, 3, 5)
l = [10, 2, 3, 5]
s = {10, 2, 3, 5}

# what do you think this prints?
print( len(t), len(l), len(s) )
print( sum(t), sum(l), sum(s) )
print( min(t), min(l), min(s) )
print( sorted(t), sorted(l), sorted(s) )
```

l.sort() vs sorted(l)

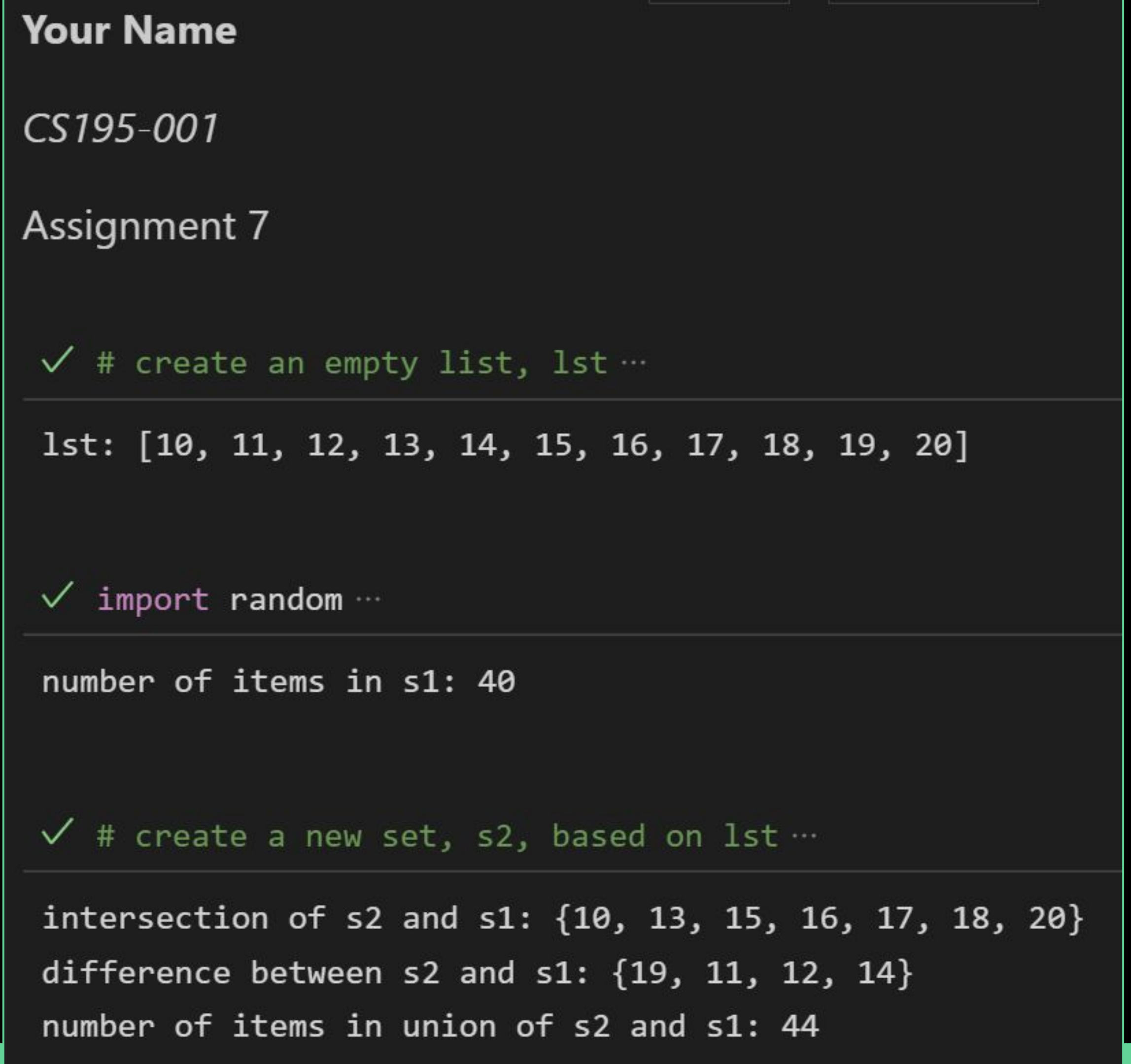
```
l = [10, 2, 3, 5]

# what do you think this prints?

print( sorted(l) ) # does not change l, just makes a sorted copy

print( l )

l.sort() # this is destructive - it actually changes l

print( l )
```

Assignment 7

- create a new file a7.yourLastName.ipynb, open it in VSCode
  - this is a jupyter notebook
- create and run the following four cells
  1. add your name, course/section number, assignment number, change cell type from Python to Markdown
  2. create an empty list, lst, use a for-loop to add numbers 10-20 to lst, print lst, print second item item in lst
  3. create a new empty set, s1, use a for-loop to add 50 random numbers between 1 and 50 to s1, print the number of items in s1
  4. create a new set, s2, based on lst, print out the intersection of s2 and s1, the difference between s2 and s1, and the number of items that would be in the union of s2 and s1

Assignment 7

Your final notebook to-be submitted on blackboard should look something like this:



Your Name

CS195-001

Assignment 7

```
✓ # create an empty list, lst ···

lst: [10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]

✓ import random ···

number of items in s1: 40

✓ # create a new set, s2, based on lst ···

intersection of s2 and s1: {10, 13, 15, 16, 17, 18, 20}
difference between s2 and s1: {19, 11, 12, 14}
number of items in union of s2 and s1: 44
```