Have these as references.

List -     <https://www.youtube.com/watch?v=ohCDWZgNIU0>

Set -     <https://www.youtube.com/watch?v=sBvaPopWOmQ>

Tuple -  <https://www.youtube.com/watch?v=NI26dqhs2Rk> (don't worry about sizeof and timeit)

*# List  
# list is a data type which holds multiple values (each value itself could be of different types)  
  
# Creation*list1 = [1, 2, 3, 4, 5]  
list2 = [1, **"hello"**, True] *# contains different types*list3 = [] *# empty list*list4 = [[1, 2, 3], [4, 5, 6]] *# list with a list  
  
# Access items in list*list5 = [**"hello"**, **"this"**, **"is"**, **"a"**, **"list"**] *# list with a list***print**(list5[0]) *# access the first element (starts with 0)  
# Update value*list5[0] = **"hi"  
print**(list5[0])  
  
*# Slicing  
# format somelist[start:end], it will produce items from start to end-1 (end is not inclusive)***print**(**"slicing"**)  
**print**(list5[2:4])  
**print**(list5[:4]) *# omitting start assumes start is 0***print**(list5[2:]) *# omitting end assumes end is len(list)***print**(list5[:]) *# produces a copy of the same list. Note that it is a copy and not the same list.***print**(**"for loop"**)  
**for** item **in** list5:  
 **print**(item)  
  
**print**(**"while loop"**)  
i = 0  
**while** i < len(list5):  
 **print**(list5[i])  
 i = i + 1 *# if you forget this, it will run forever saying "hi"***print**(**"Assignment"**)  
*# Assigning list to multiple variables*a = [1, 2, 3]  
b = a  
**print**(a)  
**print**(b)  
a[0] = 4  
**print**(a)  
**print**(b)  
*# note that updating a updates b also. This is because a and b are referencing to the same list.  
# variables are basically naming. Imaging somehow having two names. It is a single person but can be referenced in  
# different ways. if you want b to have a copy use slicing to copy b = a[:]  
  
  
# Using functions on list*list1 = [1, 2, 3]  
list2 = [**"a"**, **"b"**, **"c"**]  
*# Append*list1.append(5) *# adds 5 to list1.***print**(list1)  
*# notice is it a little different from normal functions since you usually call functions with out  
# `variable.function\_name()` this is because append is a function as part of the list itself. (more about this in  
# object oriented programming later)***print**(list1) *# list1 has 5 added at the end  
# if you call append with list2 then it adds item to list2.*list2.append(**"d"**)  
**print**(list2)  
  
*# Count***print**(list1.count(3)) *# returns the no.of times 3 occurs in the list. In this case 1.***print**(list2.count(**"x"**)) *# 0  
  
# Extend*list1.extend([5, 6, 7]) *# similar to append, but takes a list. Adds 5, 6, 7 to list1 at the end.***print**(list1)

List1.append([5,6,7]) #different output  
  
*# Reverse*list1.reverse() *# reverses the list. Note that it does not return anything. It updates the list itself.***print**(list1)  
  
*# Remove*list1.remove(7) *# removes the first occurrence of 7.***print**(list1)  
*# error if it is not in list  
# list2.remove("x")  
# so you can check and remove***if "x" in** list2:  
 list2.remove(**"x"**)  
  
*# Index***print**(list1.index(1)) *# returns the position of the item in list. Error if not present. #0,1,2  
  
# Sort*list1.sort() *# doesn't return anything, but sorts the list itself.***print**(list1) #check google for further reference  
  
*# Many of the functions like append, extend, reverse, remove, sort updates the list itself.  
# so if you have something like*list3 = list1  
**print**(list3)  
list1.reverse()  
**print**(list3) *# it reverses list3 also since they both reference the same list  
  
# Operator like + can act on two lists*list3 = list1 + list2  
*# this does not update the list1 or list2.  
# now updating list1 or list2 does not update list3. In this case, list 1 and list 2 are copied as different lists before adding them.***print**(list3)  
list1.reverse()  
**print**(list3)  
  
*# convert a string to list*word = **"hello"**word\_list = list(word)  
**print**(word\_list)

Word=”hello”

Word\_list=[word]

*# Set  
# set is like a list but does not contain duplicates or have positions  
# Creation*set1 = {1, 2, 3}  
set2 = {}  
*# set3 = {{1, 2, 3}, [4, 5, 6]} Note that you can not store set in a set or list in a set.  
# Just remember for now that only basic data types can be stored in set*

*#But you can add tuple in set. Because you can’t modify a tuple once created.  
  
# You can't access a single element in set  
# set1[0] error  
# you can only check if some element is present or not***if** 1 **in** set1:  
 **print**(**"yes"**)  
  
**print**(len(set1)) *# total items in set  
  
# loop***for** i **in** set1:  
 **print**(i)  
*# note the items printed need not be in the same order as created  
  
  
# Using functions on set*set1.add(4) *# adds to set (it does not return anything but updates set1 directly)*set1.add(4) *# adding again does nothing as 4 is already in set.***print**(set1)  
  
set1.remove(1) *# removes an item if it is present  
# set1.remove(0) # error since 0 is not present***if** 0 **in** set1:  
 set1.remove(0)  
*# or use discard*set1.discard(0)  
  
set1.clear() *# removes all items from list***print**(set1)  
  
set1 = {1, 2, 3, 4}  
set2 = {1, 2, 5}  
*# These operations do not update set1 or set2, it returns the result***print**(set1.difference(set2)) *# difference is {3,4}***print**(set1.intersection(set2)) *# intersection is {1,2}***print**(set1.union(set2)) *# union is {1,2,3,4,5}  
  
# set2 is same as set1. So any updates like discard, remove, add, clear will be reflected in set2 also.*set2 = set1  
set1.clear()  
**print**(set2)  
  
*# There are few other methods on set, but won't need it that often. You can learn it yourself.  
# | add(...)  
# | Add an element to a set.  
# | This has no effect if the element is already present.  
# |   
# | clear(...)  
# | Remove all elements from this set.  
# |   
# | copy(...)  
# | Return a shallow copy of a set.  
# |   
# | difference(...)  
# | Return the difference of two or more sets as a new set.  
# | (i.e. all elements that are in this set but not the others.)  
# |   
# | difference\_update(...)  
# | Remove all elements of another set from this set.  
# |   
# | discard(...)  
# | Remove an element from a set if it is a member.  
# | If the element is not a member, do nothing.  
# |   
# | intersection(...)  
# | Return the intersection of two or more sets as a new set.  
# | (i.e. elements that are common to all of the sets.)  
# |   
# | intersection\_update(...)  
# | Update a set with the intersection of itself and another.  
# |   
# | isdisjoint(...)  
# | Return True if two sets have a null intersection.  
# |   
# | issubset(...)  
# | Report whether another set contains this set.  
# |   
# | issuperset(...)  
# | Report whether this set contains another set.  
# |   
# | pop(...)  
# | Remove and return an arbitrary set element.  
# | Raises KeyError if the set is empty.  
# |   
# | remove(...)  
# | Remove an element from a set; it must be a member.  
# | If the element is not a member, raise a KeyError.  
# |   
# | symmetric\_difference(...)  
# | Return the symmetric difference of two sets as a new set.  
# | (i.e. all elements that are in exactly one of the sets.)  
# |   
# | symmetric\_difference\_update(...)  
# | Update a set with the symmetric difference of itself and another.  
# |   
# | union(...)  
# | Return the union of sets as a new set.  
# | (i.e. all elements that are in either set.)  
# |   
# | update(...)  
# | Update a set with the union of itself and others.*

*# Tuple  
# tuples are similar to list, but can not be modified once created. (read only)*t1 = (1, 2, 3)  
t2 = (1,)  
t3 = tuple()  
t4 = ((1, 2, 3), [**"hello"**], True) *# mixed types as list  
  
# Access using index***print**(t1[0])  
*# print(t1[3]) error since there is only 3 items. You are trying to access 4th element.  
# t1[0] = 5 Assignment is error, since it can not be modified***print**(len(t1)) *# total items in tuple  
  
# loop through items***for** i **in** t1:  
 **print**(i)  
  
*# functions on tuple***print**(t1.index(1)) *# returns the position of first occurence of 1. 0 in this case  
# print(t1.index(5)) # error if not found.***if** 5 **in** t1: *# or use if to check before calling index* t1.index(5)  
  
t1.count(5) *# no.of times item occurs in tuple  
  
  
# Tuples are usually used when returning multiple values from function***def** min\_max(numbers):  
 **return** min(numbers), max(numbers)  
  
  
a = min\_max([1, 2, 3, 4, 5])  
**print**(a)  
*# or you can call directly***print**(min\_max([1, 2, 3, 4, 5]))  
*# can split the items in tuple to multiple variables*a, b = min\_max([1, 2, 3])  
c, d = (1, 2)

c,d=d,c #1,2=d,c