SECTION 3: PYTHON OBJECT AND DATA STRUCTURE BASICS, 2 hrs 2 mins, 33 parts

- **18/36 Lists in Python**
  - -> lists are ordered sequences that can hold a variety of object types
  - -> [] brackets -> e.g [1,2,3]
  - -> my_list = [1,2,3]
  - -> you can store multiple different data types in a list
  - -> len(my_list) <- prints the length of the list
  - -> mylist = ['one', 'two', 'three']
  - -> mylist[0] -> 'one' (it's like an array except with words -> the same [] technique applies to find the value of one specific variable)
  - -> mylist[1:] <- ['two','three']
    - -> slicing the list works as it does slicing a string
    - **-> you an also add lists together (aka, concatenate them)**
    - **-> so, indexing, slicing and concatenation**
      - **you can change the elements of a list but you can't change the letters in a string**
  - -> **lists are mutable**
    - **e.g list[0] = 'element'**
  - -> to add items to the end of th elist
    - **-> name_of_list.append('six') <- add the term 'six' to the end of the list**
    - **-> this is in comparison to adding two lists together, which is concatonation**
  - **removing elements in the list**
    - -> pop
      - name_of_list.pop()
      - this is returning seven in this case -> it's popped off / removed the last element of the list
        - **it's returned the last element of the list**
        - and then when you call the name of the list -> the last element is no longer in it
      - **if you give pop an index in the argument, it removes the list element at that index**
  - sort
    - new_list = ['a','c','b']
    - -> new_list.sort()
    - -> it's none type -> not an entire list
      - there is a type of object called None
      - -> placeholders -> return values of methods or functions which don't return anything
    - -> **it's updated new_list -> if you define a variable and set it equal to new_list.sort() it won't work if you try and call it -> because the aim of it is to update the original list**
  - reverse
    - **-> name_of_list.reverse() <- this sets the list equal to the same thing in reverse**
    - -> indexing as well as slicing
    - -> unlike strings, we can do reassignments with lists