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

- **22/36 Dictionaries in Python**
  - theory
    - -> unordered mappings for storing objects
      - key value pairing (rather than indices)
      - {'key1':'value1', 'key2':'value2'}
    - -> dictionaries -> objects retrieved by key name
      - a dictionary can't be sorted -> it puts key value pairs wherever is more efficient
      - -> but lists can be retrieved by location / ordered / indexed / slices
  - in JN
    - -> my_dict = {'key1':'value1', 'key2':'value2'}
    - -> we want to return values from the dictionary
      - -> my_dict['key1'] <- this returns the value stored at the key called key1
      - -> **the indexes are replaced with keys**
      - -> e.g you could store the value of apple at the key apple -> then it would return the cost of apples when it was called
    - -> they can store many different datatypes
      - -> integers
      - -> arrays
      - **-> nested dictionaries -> then to return the value of the element at that index, it would be [][]**
      - -> d['k3'][:'insidekey']
    - -> arrays and dictionaries
      - **you can set an element stored at the key an entire array**
      - **-> to extract this, it's name_of_dictionary['key_is_stored_at']**
      - -> this is the array -> which you can then [extract a value with the index typed in these brackets]
      - -> you can then letter.upper() <- operate on the element you've extracted
      - -> in another example -> he's [][] called two keys next to each other
      - **-> d['key'][2].upper() -> this returns C**
        - **in this example, he's taking the second element in the list stored under 'key' (which is the letter c) and is making it uppercase**
    - -> adding in a new key value pair
      - -> **d = {'k1':100, 'k2':200}**
      - -> if we want to add a k3
        - **d['k3'] = 300**
      - -> you can also change the value stored at k1
        - d['k1']='NEW VALUE'
        - -> then it's appearing when d is called
    - -> methods on dictionaries
      - we have a dictionary called d
      - **d.keys() <- this returns all of the keys**
      - **d.values() <- this returns the values**
      - **d.values() <- this returns all of the items**
    - -> summary
      - the syntax
      - you can change the value of the element stored at the key
      - -> you can nest objects inside a dictionary