SECTION 6: METHODS AND FUNCTIONS, 2 hours 54 mins, 30 parts

- **17/29 *args and **kwargs in Python**
  - ***args and **kwargs are functional arguments -> arguments and keyword arguments**
  - **-> you want to accept an arbitrary number of arguments**
  - **in JN**
    - -> def myfunc(a,b):
      - return 0.05*(a+b)
    - -> a and b are positional arguments
    - **-> if you want a third argument added to the function**
      - **you can define other input parameters (e.g d=0)**
      - **alternatively, def myfunc(*args):**
        - **-> this treats it as a tuple of arguments**
        - **return sum(args) * 0.05**
      - **myfunc(40,60,....) e.g <- *args is for all arguments (all the arguments which are entered) -> def(*args)**
      - **-> instead of *args, you could also enter *spam, e.g**
        - **then you -> for item in args, e.g you can run a for loop**

  - **-> for a dictionary of key value pairs when defining a function**
    - **def myfunc(**kwargs) <- for any amount of arguments**
    - **-> for 'fruit' in kwargs: <- e.g**
      - -> then indented a formal
      - -> print("....{}".format(kwargs['fruit']))
      - **-> then you can have any amount of inputs and it's creating a dictionary**
      - **-> ** creates a dictionary and * doesn't**

    - **-> another one is def myfunc(*args, **kwargs):**
      - print('I would like {} {}'.format(args[0],kwargs['food']))
      - -> then you can input numbers and then text e.g -> the first ones entered are args and the second is kwargs
      - -> this is useful for outside libraries