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

- **29/29 Methods and Functions Homework - Solutions**
  - **-> in the .ipynb file**
    - -> put an equation into code e.g -> you can import pi from import math, from math import pi
    - -> **you can put the equation and return on the same line**
    - -> in another example
      - for item in range(0,5)L
        - print(item)

    - **-> 5 in range (1,10) <- this is asking if 5 is in that range of numbers**
      - -> so to put this into a function, it's
      - def ran_check(num, low, high):
        - num in range(low, high+1) <- you can also put this entire thing into an if / else block. **She's printed the result in an f string literal**

    - **-> a Python function which returns the number of upper and lowercase elements in the string**
      - **-> we need a loop in there**
      - -> she's initialised two counters -> one for the number of upper case strings in the input string and a similar one for the number of lowercase strings in it
      - -> we're iterating through the characters in the input string
        - if it's .upper() <- then we're increasing the uppercase counter by 1, it is itself +=1
      - -> then she's printing out the results in an f string literal
        - **and testing the function**
      - **-> you can also set up all of the initialised variables into a dictionary (e.g d={'upper':0,'lower':0}**

    - **-> a function which takes a string and removes the repeated elements in it**
      - def unique)list(list):
        - return list(set(list)) <- **sets remove the repeated elements**
      - another way of doing this is
        - x = []
        - for number in list:
          - if number not in seen_numbers:
            - seen_numbers.append(number)
        - return seen_numbers <- if seen numbers isn't in the list then it hasn't been repeated

    - **-> another example**
      - def multiply(numbers):
        - total = 1
        - for num in numbers:
          - total = total * num
        - return total <- we're taking an array of numbers, iterating through them and then retuning the total product
        - -> **then she's testing the function for different use cases again once it's been written**

    - **-> a function to check if the string is itself written backwards (palindrome)**

- checking is the string == itself spelt backwards
- she's done s = s.replace(' ','')
    - -> the first stage is to remove the spaces
    - **-> s == s[::-1] <- we are checking if s is itself backwards (step size going back wards, stepping -1)**
    - -> then return s == s[::-1]

- **-> another one is a Python function to check is a string is a pangram or not**
    - a word or sentence containing every letter of the alphabet at least once
    - -> it should return for 'the quick brown fox jumps over the lazy dog'
    - -> she creates a set of the entire alphabet
    - -> removes spaces from the input string
    - -> converts the input string into all lowercase strings
    - -> **a lot of it is first making sure the input is in the right format**
    - -> making that into a set -> so we have no repeats
    - -> the set of the alphabet
    - -> checking is the alphabet set is equal to the input
    - -> remove any spaces from the input string
    - -> converting everything into lowercase
    - -> **she's converted both of them into sets, and then is comparing them)**
    - -> another way to visualise it is through print statements
    - -> debugging / understanding what is going on in larger functions