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

- **4/29 Basics of Python Functions**
  - **-> functions <- the name of the JN notebook (this includes tuple unpacking)**
  - **in the .ipynb file**
    - def say_hello(name='Jose'): <- **name is the argument, and the default name is Jose -> without a default then when the function is called without the argument, it will return an error message**
      - **about**
        - anything inside this indentation is called in the code
        - -> if you put a lot of different print statements in this and then call the function -> all of them get printed out on separate lines
        - -> **to call the function its say_hello() not say_hello <- or else it will just return that this thing is a function**
      - **print(f'Hello {name}') <- this is a string literal**

    - **another example**
      - **def add_num(num1, num2):**
        - **return num1, num2 <- this prints out the result when the function is called, and the result of that can be set equal to a variable (in comparison to using print)**
          - **-> if you used print and not return then you couldn't set that function call equal to a variable (it's a NoneType), the value isn't returned - something is just printed out**

    - **you can use a print and a return statement when you define the function**
      - -> then when its called the result is printed and you can set the function call equal to a variable

    - **if you need to check the type of the input of the data into the function**
      - **-> e.g if you define a function which adds two numbers together -> there is a case where strings are put as arguments into the function instead of numbers**
      - **-> you need to check the type of the data before returning the result (in this case - adding numbers not strings)**