SECTION 5; PYTHON STATEMENTS, 1 hour 15 mins, 7 Parts

- **2/7 For Loops in Python**
  - **-> objects in Python are iterable**
    - we can iterate over element in the object (every element in a string e.g)
    - this is a common term in the Python documentation
    - -> you can iterate over the object
    - -> for every character in the string, you can iterate through the string and do something for that character
    - **-> you can iterate every key in a dictionary / every item in a list**

  - **-> syntax of a for loop**
    - my_iterable = [1,2,3]
    - iterate over every element in the object
    - for item_name in my_iterable:
      - print(item_name) <- this prints the item which is being iterated through
        - -> **i.e for ... in ...:**
          - **run this piece of code (e.g print out what is being iterated through)**
          - **-> it's like a giant calculator**

  - **-> in the .ipynb file**
    - -> datatypes, objects
    - -> constructing logical programs
    - -> my_list = [1,2,3,4,5,6,7,8,9,10]
      - **-> don't call it 'list' - this is a keyword in Python and the code will highlight it**

    - **-> for num in mylist:**
      - print(num) <- it's iterating through the list and printing out each of the elements (called num, in this case) as it iterates through them
      - -> you could also ask it to print 'hello' etc -> you want to choose a variable name (in this case num) relates to what is being iterated through

    - **-> to only print out the even numbers when we iterate through the array**
      - -> **we are adding control flow into the iteration**
      - -> **if num % 2 ==0:**
        - print(num) <- **we are printing the number if it divided by 2 has no remainder (it's even)**
        - **-> then she's done print(f'Odd Number: {num}') <- print the number using an f string literal**

    - **-> multiple loops**
      - list_sum = 0
      - for num in mylist:
        - list_sum = list_sum + num <- we are adding the current number to the previous number -> and printing out the result at the end
      - print(list_sum)

    - **-> strings**
      - mystring = 'Hello World'
      - for letter in mystring:
        - print(letter)
        - -> **you can iterate through the characters in a string**

- -> alt. you could write for letter in 'Hello World':
- **-> instead of letter -> you can also use an _ <- this is common where the variable isn't used**

- **-> tuples**
  - tup = (1,2,3)
  - for item in tup:
    - print(item) <- this prints out each of the terms in the tuple as it iterates through them

  - **-> tuple unpacking and for loops**
    - mylist = [(1,2),(3,4),(5,6),(7,8)]
    - -> **there is a list and in the list are tuples (aka coordinates which are immutable / can't be changed)**
    - -> for item in mylist:
      - print(item) <- **the tuples in the list can be returned**

  - **-> another example is:**
    - for (a,b) in mylist:
      - print(a)
      - print(b)
      - -> it's going through each tuple in the list etc
      - -> tuple unpacking
      - -> **you can also say**
        - **for a, b**

  - **-> another example**
    - -> mylist = [(1,2,3),(5,6,7),(8,9,10)]
    - -> for a,b,c in mylist:
      - print(b)
    - -> you can use this to e.g extract the second number in each of the tuples (stored in the array of tuples)

  - **-> another example**
    - d = {'k1':1,'k2':2,'k3':3} <- dictionary
    - for item in d:
      - print(item) <- **iterating through a dictionary only prints the keys and not the values -> you need to call for item in d.items() and then the values of the item will be returned**
        - **-> this is similar to tuple unpacking**
        - -> iterating through dictionaries
        - -> d.values -> this prints out only the values
        - -> dictionaries are unordered
          - if you have a large dictionary, there is no guarantee that the results will be ordered