SECTION 13: PYTHON GENERATORS - 17 minutes, 3 parts

1/3 Generators with Python

- **generators (concept)**
  - we can create functions with def and return
  - generator functions <- we can write a function which sends back a value and picks up where it left off
  - instead of creating an entire sequence and holding it in memory
  - **use of memory**
  - **it's a function**
  - **we can generate a sequence of values over time**
  - **yield**
  - a generator function becomes an object that supports an iteration protocol when it's compiled
  - **the generator function becomes an object**
  - it's not computing a series of values all at the same time -> it's waiting for the next value to be called for
  - **the range() function generates numbers, for example <- it takes the last number and adds n to it (rather than remembering the entire thing, it can just run a function to generate them)**

- **generators (application)**
  - he's defined a function to cube the argument
  - then he's defined another function which uses the yield keyword
  - it's a bit like return
  - **it's a generator object**

  - **you can ask it to print(next(g)) for example**
    - -> remember what the previous one was, now generate the next one
    - -> don't hold everything in memory
    - -> we can generate different terms in the sequence by running the function
    - -> and then it will return an error when it reaches the highest one
    - **-> if you call a for loop on a generator function, it will just carry on asking for the next one until it reaches the end of the range, at which point an error message will be returned**
    - **-> it doesn't stop as a normal for loop would <- it returns an error message when it reaches the end of the loop**
    - -> the generator function remembers the last value and returns the next value
    - -> all the values have been yielded

  - **iter()**
    - -> this allows us to iterate through a normal object that you wouldn't normally expect
    - -> we can iterate through the letters in a string
    - -> you can't do this with the next() generator method
    - -> you need to first concert the string into a generator, then it will work
    - **-> you have to convert objects which are itterable into iterators**
    - -> the yield keyword
    - -> to create our own generators with the yield keyword