SECTION 14: ADVANCED PYTHON MODULES, 2 hours 23 minutes, 13 sections
7/13 Python Regular Expressions Part One

- we can search for substrings within a larger string of text
- **"dog" in "the dog said hi"**
    - **-> the limits of this are**
        - you need to know the exact string
        - you need to perform additional operations to account for capitalisation and punctuation
        - -> **if we are looking for something e.g like a phone number or an email but don't know the exact number (just the format / structure of it)**
        - -> **searching through a document of text for patterns in a certain format**

        - -> **e.g searching for emails -> "text" + "@" + "text" + ".com"**
            - if we want to find emails in a document of text
            - -> we are also looking for user@gmail.com
            - -> **these are regular expressions -> we are looking for something in this format**

    - **-> the re library**
        - **regular expressions library**
            - **-> specialised pattern strings and searching for patterns in text**

        - **they exist in special syntax formats / patterns / types**
            - -> **you have to spell the regular expressions in a specific way -> e.g if we are looking for a phone number**

        - **regex pattern -> a regular expression pattern**
            - -> r"(/d/d/d)-/d/d/d-/d/d/d"
            - -> this is an example regex (regular expression) pattern
            - -> r" is saying "don't treat this like a regular string"
            - -> \d <- digit
            - -> looking for three digits in a row
            - -> we know they are going to be in a certain format
            - -> constructing general regular expression patterns
            - -> these can also use quantifiers
                - r"(\d{3}-\d{3}-\d{4}"
            - -> how to use the regular expression library to focus on sections within text

        - **-> this lecture**
            - how to use the regex library and syntax

    - **-> in the project .ipynb file**
        - how to search for basic patterns
            - text = " their number is 07743382957"
            - 'number' in text
                - this is a boolean asking true or false -> is this in the text
            - **import re <- import the regular expression module**
            - **pattern = 'phone'**
            - **re.search(pattern, text) <- this returns that it is a match object**
                - **a match object -> if there was a match to the phone and where the index location reports back to**
                - **-> search for this thing in the text (the thing is a 'regular expression' -> something repeating which we are searching for)**

- -> pattern = 'NOT IN TEXT'
- **re.search(pattern,text) <- we are searching for text in the string called pattern**
- **-> match = re.search(pattern,text) <- in the regular expressions libary search for the pattern**

- **match.span()**
  - match = re.search(pattern,text)
  - match.span() <- this returns the index location of the span
  - match.start()
  - match.end() <- this returns the index of the match

- **-> text = 'my phone once, my phone twice'**
  - **<u>match = re.search('phone', text) <- search for the regular expression phone in the string stored in the variable called text (this returns back one result called a span)</u>**
  - **then match -> returns the span (the item we are searching for in text starts on the 3rd index and ends on the 8th)**
  - **<u>-> the find all function (this returns all the results)</u>**
    - **<u>matches = re.findall('phone',text) <- use the regular expression module to find all of the instances (not just one which is what search does) where 'phone' is in the variable called text which contains strings</u>**
    - **them printing matches returns all of the items in the search which match**
    - -> matches -> this is a list of all the different things which came up in the search
    - -> you can len(matches) for the number of elements in the array which matched it

- **-> for match in re.finditer('phone',text):**
  - **for match in re.finditer('phone',text): <- this iterates through the text and returns each match object which is found**
    - print(match) <- these are the match objects
    - -> you could match.span() <- this returns back the index from the start to end)
    - -> then to print out the entire object, it's
      - **match.group() <- this returns back the text.match which you were looking for**

  - **-> summary**
    - **search <- pass in the pattern and the text**
    - **findall <- pattern and text, this returns the list of matches**
    - **<u>finditer <- a combination of the two (returning back match objects for the pattern in the text and can call methods off of the match object)</u>**