SECTION 14: ADVANCED PYTHON MODULES, 2 hours 23 minutes, 13 sections

9/13 Python Regular Expressions Part Three

- **-> syntax for regular expressions**
  - -> the or operator
  - -> searching for patterns
  - -> either there is a match or not
  - **-> we can use the pipe operator (|), to search for 'or'**
  - -> wild card operators
  - -> re.findall()
  - -> it returns all of the matches which are correct
  - -> things which come before the c in this case are returned
  - -> starts with and ends with
  - -> re.findall()
  - -> to find everything that starts with a number
  - -> we can also search if the text itself starts with a number

- **-> we can also use exclusion -> to leave out characters**
  - -> he's defined a variable (phrase)
  - -> we want to exclude digits / numbers
  - -> pattern = r'[^\d]'
  - -> then re.findall(pattern, phrase)
  - -> if we want to get the words back together, then we can use a + sign on the end of this
  - -> to get rid of punctuation from a sentence
  - **-> re.findall(r'[^!.?]+, test_phrase) <- exclude all of these symbols in this phrase (in this case punctuation)**
  - -> re.findall(r'[^!.?]+',test_phrase) <- we can also remove the spaces, which will return a list of all the words

- **-> we can also use ' '.join(clean) to return all the words joined together again**
  - -> we need to break down regular expression pattern codes

- **-> square brackets**
  - -> these are for grouping things together
  - -> certain things are for inclusion / exclusion
  - -> we can group them for inclusion / exclusion
  - -> he has defined a string in a variable
  - -> the string is long
  - -> we want words which have a hyphen or words in the middle of them
  - -> there is no longer punctuation there
  - -> we are removing the spaces and returning a list of words
  - -> this is done using .join(clean)
  - -> breaking down regular expression pattern codes
  - -> we can also break down square brackets / braces
  - -> we can group these via inclusion / exclusion
  - -> in this example, he's defined a variable which has a string stored as its value
  - -> we are getting rid of words which have a - hyphen in the middle of them

```
In [80]: text = 'Only find the hypen-words in this sentence. But you do not kno

In [81]: pattern = r'[\w]+'

In [82]: re.findall(pattern,text)

Out[82]: ['Only',
          'find',
          'the',
          'hypen',
          'words',
          'in',
          'this',
```

```
]: pattern = r'[\w]+-[\w\d]+'
```

- -> he is going this in this example with r'
- -> + <- occurs one or more times
- -> we are checking for a pattern
- -> this returns groups of alpha numerics
- -> this returns the hyphenated words

```
In [87]: text = 'Hello, would you like some catfish?'
         texttwo = "Hello, would you like to take a catnap?"
         textthree = "Hello, have you seen this caterpillar?"

In [90]: re.search(r'cat(fish|nap|claw)',textthree)
```

- -> you can also separate groups using a brace notation (square brackets)
- -> we have three different strings and each are stored in different variables
- -> to find the different options, we can group together multiples
- -> this finds and returns different matches
- -> it's asking for matches which are in all three of the strings
- -> there is documentation for regu.ar expressions in the Python official documentation
- -> we have character identifiers and quantifiers