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

8/13 Python Regular Expressions Part Two

- -> how to build the special regular expression identifying pattern codes
- -> previous video -> doing a pattern search using the regular expression library
- -> building out patterns with identifier syntax
- -> converting the items in a table into regular expressions
- -> regular strings for pattern searching

- **character identifiers**
  - -> there is a table of these
  - -> a table which explains what each of the character identifiers are
  - -> we want to convert these into characters
  - -> \d <- digits; this is a placeholder for digits
  - **-> we're searching for something, a pattern in the text**
  - -> alpha numerics include underscores
  - -> we can also look for whitespace
  - -> \D <- a non digit
  - **-> backslashes are used for this**
  - -> in this case, to search for a telephone number
  - -> we can also search for plurals

Regular Expressions Patterns

Character Identifiers

Character	Description	Example Pattern Code	Example Match
\d	A digit	file_\d\d	file_25
\w	Alphanumeric	\w-\w\w\w	A-b_1
\s	White space	a\sb\sc	a b c
\D	A non digit	\D\D\D	ABC
\W	Non-alphanumeric	\W\W\W\W\W	*-+=)
\S	Non-whitespace	\S\S\S\S	Yoyo

Character	Description	Example Pattern Code	Exam
+	Occurs one or more times	Version \w-\w+	Vers
{3}	Occurs exactly 3 times	\D{3}	
{2,4}	Occurs 2 to 4 times	\d{2,4}	
{3,}	Occurs 3 or more	\w{3,}	an
*	Occurs zero or more times	A*B*C*	
?	Once or none	plurals?	

```
In [ ]:
```

- **quantifiers**
  - **repetition of the same characters**
    - -> there are tables of this
    - -> we don't want to have to write the character identifiers multiple times
    - -> + <- the character identifier occurs more than one times
    - -> * <- if it occurs 0 or more times
      - -> does it occur 0 or more times?
    - -> AAACC <- for example
    - -> ? <- once or none (plurals)
    - -> he's done example of a regex for a phone number, with dashes
    - -> we are asking for three digits - three digits - four digits
    - -> the second task is to find phone numbers, and extract their area codes
    - -> we can use groups for any regular tasks that have grouping expressions
    - -> e.g results.group()
    - -> we can call them by the group position

Quantifiers

Character	Description	Example Pattern Code	Example Match
+	Occurs one or more times	Version \w-\w+	Version A-b1_1
{3}	Occurs exactly 3 times	\D{3}	abc
{2,4}	Occurs 2 to 4 times	\d{2,4}	123
{3,}	Occurs 3 or more	\w{3,}	anycharacters
*	Occurs zero or more times	A*B*	AAACC
?	Once or none	plurals?	plural

```
In [33]: phone = re.search(r'\d{3}-\d{3}-\d{4}',text)

In [34]: phone
Out[34]: <_sre.SRE_Match object; span=(19, 31), match='408-555-7777'>

In [35]: phone_pattern = re.compile(r'(\d{3})-(\d{3})-(\d{4})')

In [36]: results = re.search(phone_pattern,text)

In [37]: results.group()
Out[37]: '408-555-7777'

In [41]: results.group(4)
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-41-278b3d462a98> in <module>()
----> 1 results.group(4)

IndexError: no such group
```

  - **if we want the first group, for example - results.group(1)**

        - -> this only returns back the first group
        - -> then asking it for groups outside of what we have will return an error message
        - -> we can extract parts of the expression, and at the same time return a match
        - -> additional regular expression syntax <- wildcards and pipe operators
        - -> the groupings can be called individually

- -> quantifiers have special characters
- -> we are then looking at whilecard syntax / starts / ends with
- -> we can use quantifiers
- -> tacking them on
- -> we can take subsections of the entire pattern
- -> separating each of the groups with parenthesis
- -> we can 'grab' subgroups