365 DataScience Regular expressions in Python

Step 1 Most commonly used regular expressions

Regular expressions allow you to search for patterns in a text. In this way, information can be extracted very efficiently. The documentation for the Python Standard Library is an essential guide towards regular expression: https://docs.python.org/3/library/re.html

Matching characters

[aeiou] Match any vowel [^aeiou] ^ inverts selection, this matches any consonant [a-z] Match any lowercase leter from a-z \d Matches any decimal digit; this is equivalent to the class [0-9] \D Matches any non-digit character; this is equivalent to the class [^0-9] \s Matches any whitespace character; this is equivalent to the class [\t\n\r\f\v] \S Matches any non-whitespace character; this is equivalent to the class [^ \t\n\r\f\v] \w Matches any alphanumeric character; this is equivalent to the class [a-zA-Z0-9_] \W Matches any non-alphanumeric character; this is equivalent to the class [^a-zA-Z0-9_]

Repetition

? Match 0 or 1 of the preceeding group * Match 0 or more of the preceeding group + Match 1 or more of the preceeding group {n} Match exactly n of the preceeding group ^string String must begin with string string$ String must end with string

Step 2 Import the re module

```
import re
```

Step 3 Compile, match, search and group

```
# Compile the pattern.
# The 'r' in front of the quotation marks informs Python that we are us
ing a raw string.
pattern = re.compile(r'dog')

# The object returns the pattern iself and the encoding of the characte
rs.
# By default, that is Unicode.
pattern

# Tell Python what is it that you want to match against the compiled pa
ttern.
# It returns an object giving details on what kind of object that is, t
he place
# where the match occured as well as the thing that we are matching.
pattern.search('dog')
```

```
# Store the Match object in a variable called matched1.
# The match() method will only match whatever is at the start of the st
ring.
# Therefore, if the string doesn't start with 'dog', the code will thro
w an error.
matched1 = pattern.match('dog123')

# Use the group() method on that object to access what has been matched.
matched1.group()

# To search for patterns anywhere in the text, search() should be used
instead.
matched2 = pattern.search('feed the dog')
matched2.group()
```

Step 4 Create more complicated patterns

```
# With this pattern, we will search for the character 'd' followed by a
 non-digit character.
pattern = re.compile(r'd\D')

matched3 = pattern.search('dog123')
matched3.group()

# Adding a plus will output all non-digit characters that follow 'd'.
pattern = re.compile(r'd\D+')

matched3 = pattern.search('dog123')
matched3.group()

# Substituting 'D' with 'w' will match the whole string, as '\w' matche
s all
# alphanumeric characters.
pattern = re.compile(r'd\w+')

matched3 = pattern.search('dog123')
matched3.group()

# Regular expression are good for finding file names having a particula
r extension/format.
string = '''image01.jpg,image01.png, image02.jpg,my_doc.doc,my_doc2.doc
x,london_bridge.gif, another_doc.doc,my_pdf.pdf'''

# The following pattern will search for all alphanumeric characters (\w)
 followed by a dot (\.)
# and either of the extensions - jpg, png, or gif.
pattern = re.compile(r'(\w+)\.(jpg|png|gif)')

# The finditer() method will iterate through the string and, once it fi
nds something,
# it will pick up where it left off and continue the search.
```

```python
# 'm' serves as the Match object.
for m in re.finditer(pattern, string):
    print(m.group())
```