Regular Expressions: Pattern matching

Pattern matching

Sometime we want to match not a single equality but a pattern. Mostly this is used for text processing.

https://docs.python.org/3/library/re.html

Regular expressions (RE) are used to match a string. It is a test to see if a string matches a pattern.

Simple usage

```
import re
RESULT = re.search(PATTERN,QUERYSTRING)
if RESULT:
    # WE HAD A MATCH
else:
    # WE DID NOT HAVE A MATCH
```

```
import re
m = re.search("bow","elbow")
if m:
    print("matched bow")
else:
    print("did not match bow")
```

Regular expressions and matching

Matches pattern to string There are several components to the match.

- All the alpha numeric characters match themselves
- Upper and lowercase are respected
- Special characters to match extra patterns
  - \d matches numeric (0-9)
  - \D matches NOT numeric not(0-9)
  - \s matches white space
  - \S matches NOT white space
  - [A-Z] - ranges, all letters A-Z
  - . - matches anything

```
re.search('\d bird', '8 birds') # true
re.search('\d bird', '1 bird') # true
re.search('\d bird', 'A bird') # false

re.search('[123] bird', '1 bird')  # true
re.search('[0-3] bird', '4 birds') # false

re.search('\d bird', '4 Birds') # false
re.search('\d [Bb]ird', '4 Birds') # true
```

Modifiers

Additionally the RE grammar allows repetitions

-
  -
    - match one or more times
-
  -
    - match zero or more times
- ? - match 0 or 1 time

```
re.search('\d birds?','8 birds') # true
re.search('\d birds?','1 bird') # true

re.search('A+B','AAAAAAB') # true
re.search('A+B','AB') # true
re.search('A+B','B') # false

re.search('A*B','AAAAAAB') # true
re.search('A*B','AB') # true
re.search('A*B','B') # true
```

Grouping patterns and Capture

Use Parentheses to group patterns and further repeat. Items in the parentheses that are captured can be retrieved and used.

```
import re
m = re.search("((AB)+)C","ABABABCDED")
if m:
    print("Group 0",m.group(0))
    print("Group 1",m.group(1))
    print("Group 2",m.group(2))
```

Context of pattern

- ^ - matches beginning of string
- $ - matches end of string

```
re.search('\d bird', '8 birds') # true
re.search('\d bird$', '8 birds') # false
re.search('^\d bird', '8 birds') # true
re.search('^\d bird', '10 birds') # false
```

pattern searching

If you want to find more than one occurance, or count the number occurance you can use `search` or `findall` options

```
start =0
m = re.search(pattern, string, start)
while( m ):
     #  process this match
     start = m.end()+1
     m = re.search(pattern,string,start)
```

Speeding up

Python REs have an option called `compile` which will (potentially) improve speed of pattern matching

```
pattern = re.compile("AACA")
matches = pattern.search(DNA)
if match:
    print(match.group(0))
`
```

Practical example

Restriction Enzymes

```
EcoRI   = "GAATTC"
EcoRII  = "CC[AT]GG"

RestrictionEnzymes = [EcoRI, EcoRII]
DNA = "ACAGACGAGAGAATTCGGTAGAT"
for RE in RestrictionEnzymes:
   pattern = re.compile(RE)
   match = pattern.search(DNA)
   count = pattern.findall(DNA)
   print(RE,"matches", len(count), "sites")

 print("//")
```

More examples

See https://github.com/biodataprog/code_templates/tree/master/Regexp