Strings and Patterns

Checking for substrings

Can use `in` to check if a string is in another string

```
string="TREEBRANCH IS brown"

if "own" in string:
    print("own is in there")

if "tree" in string:
    print("tree is in there")
else:
    print("tree is not in there")
```

Check if a string starts with another

```
files = ["image001.jpg","IMG_981.JPG","fav.gif","news.txt"]
for filename in files:
    if filename.startswith("image"):
        print(filename,"stars with image")
    if filename.endswith("gif") or filename.lower().endswith(".jpg"):
        print(filename,"is an image type")
```

Patterns and Regular Expressions

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))
```

Match parts of strings with more complicated construction

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

Biological examples

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 Regular expressions

Replacement options

The `re.sub()` function allow replacement

`re.sub(pattern, repl, string, count=0, flags=0)`

To replace all instances of 'cat' with 'dog'

```
#!/usr/bin/env python3
import re
message="The cat curled up on the couch for a catnap"
newmsg = re.sub(r'cat',r'dog',message)
print(message)
print(newmsg)
# only replace first instance
newmsg = re.sub(r'cat',r'dog',message,1)
print(newmsg)
```

Now with a pattern

```
import itertools, sys, re, os
Chr8="http://sgd-archive.yeastgenome.org/sequence/S288C_reference/chromosomes/fasta/chr08.fs

PREsite=r'TGA[AT]AC'
REPLACE='PREPRE'
Chr8File="chr08.fsa"
if not os.path.exists(Chr8File):
    os.system("curl -O {}".format(Chr8))

# define what a header looks like in FASTA format
def isheader(line):
    return line[0] == '>'

def aspairs(f):
    seq_id = ''
    sequence = ''
    for header,group in itertools.groupby(f, isheader):
        if header:
            line = next(group)
            seq_id = line[1:].split()[0]
        else:
            sequence = ''.join(line.strip() for line in group)
            yield seq_id, sequence

with open(Chr8File,"rt") as fh:
    seqs = aspairs(fh)
    for seqinfo in seqs:
        seqstr = seqinfo[1].lower()
        newseq=re.sub(PREsite,REPLACE,seqstr,flags=re.IGNORECASE)
        print(newseq)
```

Demonstrating matches from random DNA

```
#!/usr/bin/env python3
#Python code to demonstrate pattern matching

# import the regular expression library
import re
import random
random.seed(11012) # initialize the starting seed - we will all have basically same result

# a random DNA sequence generator
def rand_DNA (length):
    rand_DNA=""
    bases = ['A', 'C', 'G', 'T' ]
    base_ct = len(bases)
    for n in range(length):
        rand_DNA += bases[random.randint(0,base_ct-1)]

    return rand_DNA


# lets initialize a pattern we want to match
# let's use the PRE motif which is a binding site for
# a transcription factor
# based on this paper:
#

EcoRI   = "GAATTC"
Bsu15I  = "ATCGAT"
```

```
Bsu36I  = "CCT[ACGT]AGG"
BsuRI   = "GGCC"
EcoRII  = "CC[AT]GG"

RestrictionEnzymes = [EcoRI, Bsu15I, Bsu36I,
                      BsuRI, EcoRII]

# Now let's search for this element in DNA sequence

DNA = rand_DNA(100000)
#print DNA
for RE in RestrictionEnzymes:
    pattern = re.compile(RE)
    match = pattern.search(DNA)
    count = pattern.findall(DNA)
    print(RE,"matches", len(count), "sites")
#    while match:
#        print match.group(0), match.start(), match.end()
#        match = pattern.search(DNA,match.end()+1)

    print( "//")
```