COMP364: Regular expression in Python

Jérôme Waldispühl, McGill University
Definition

A regular expression (a.k.a. Regexp) is a specific pattern that provides concise and flexible means to "match" (specify and recognize) strings of text, such as particular characters, words, or patterns of characters.

Examples:
- Words containing the word “abc”
- Words starting with a capital letter
- Words ending with a number
Syntax of Regexp

Regexp follows the syntax: <regexp 1><regexp 2>...<regexp N>
Where <regexp i> is a set of characters that can be iterated a predefined number of times.

Note: Typically, a set is represented between brackets. An undefined number of iterations is indicated by “*” (eventually null) or by “+” (strictly positive).

Examples:
• [0-9]+ // sequence of integers
• [A-Z][a-z]* // words starting with a capital letter
Syntax of regexp in Python

‘.’ : Anything
‘^’ : Start of the string
‘$’ : End of a string
‘[]’ : A set of characters
‘[^... ]’ : Negative set
‘|’ : Or
‘()’ : Match the regexp inside
‘\’ : Special characters
‘\d’ : Digit (i.e. [0-9])
‘\D’ : Non-digit (i.e. [^0-9])
‘\s’ : Whitespaces (i.e. [ \t\n\r\f\v])
‘\S’ : Non-whitespaces (i.e. [ ^\t\n\r\f\v])
‘\w’ : Alphanumeric (i.e. [a-zA-Z0-9_])
‘\W’ : Non-alphanumeric (i.e. [ ^a-zA-Z0-9_])

‘*’ : 0 or more repetitions
‘+’ : 1 or more repetitions
‘?’ : 0 or 1 repetition
‘{m,n}’ : m to n copies
Examples

What is this regexp represent?

^M([AC-GIK-MP-TVWYZ]+)$

The same but with at least 20 amino acids?

^M([AC-GIK-MP-TVWYZ]{20}) ([AC-GIK-MP-TVWYZ]+)$
Module re

**Initialization:**
Regexp module is named “re”
Start your script with `import re`

**Creation of a regexp:**
`pattern = re.compile(<regexp>)`

**Match regexp to a string:**
`result = pattern.match(<string>)`

**Shortcut:**
`result = re.match(<regexp>,<string>)`
Testing regexp

```python
if re.match(<regexp>,<string>):
    print "Regexp matches string"
```

**Exercise:**
Write a script that checks if a string (given in the command line) follows the
Capturing elements

\texttt{.group()} : returns the group of matched expressions.
Provide an argument \textit{i} if you want a specific subgroup.

```python
#!/usr/bin/python
import re

line = "cats are smarter than dogs";

matchObj = re.match( '(.*) are (.*), line)

if matchObj:
  print "matchObj.group() : ", matchObj.group()
  print "matchObj.group(1) : ", matchObj.group(1)
  print "matchObj.group(2) : ", matchObj.group(2)
else:
  print "No match!!"
```
Search vs. match

**match()** tries to match the string from the beginning, **search()** checks for a match anywhere in the string.

```python
#!/usr/bin/python
import re

line = "cats are smarter than dogs";

matchObj = re.match( 'dogs', line)
if matchObj:
    print "match --> matchObj.group() : ", matchObj.group()
else:
    print "No match!!"

matchObj = re.search( 'dogs', line)
if matchObj:
    print "search --> matchObj.group() : ", matchObj.group()
else:
    print "No match!!"
```
matches try to match the string from the beginning

```python
#!/usr/bin/python
import re

line = "cats are smarter than dogs";

matchObj = re.match( '(.* are (\d*))', line)

if matchObj:
    print "matchObj.group() : ", matchObj.group()
    print "matchObj.group(1) : ", matchObj.group(1)
    print "matchObj.group(2) : ", matchObj.group(2)
else:
    print "No match!!"
```
Applications

• Motif finding
• PROSITE
• Parse files

Problem of the assignment:
• Write a regular expression that describes a mature mRNA surrounded by START and STOP codons.
• Write a script that:
  o identify if sequence contains a mature mRNA,
  o returns the coding sequence (without START and STOP codons).