COMP364: PROSITE & Regexp

Jérôme Waldispühl, McGill University

Capturing elements

.group() : returns the group of matched expressions.

Provide an argument *i* if you want a specific subgroup.

```
#!/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!!"
```

Capturing elements

(?P<name>...) : the substring matched by the group is accessible within the rest of the regular expression via the symbolic group name name.

Example:

(?P<id>[a-zA-Z_]\w*) can be referenced as .group('id').

Search vs. match

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

```
#!/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!!"
```

.match()

match() tries to match the string from the beginning

```
#!/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!!"
```

PROSITE

PROSITE is a protein database. It consists of entries describing the protein families, domains and functional sites as well as **amino acid patterns**, signatures, and profiles in them.

PROSITE patterns are regular expressions used to characterize functional sites and perform database searches.

Pattern syntax

- IUPAC one-letter codes for the amino acids,
- ‘x’ represents any amino acid,
- ‘[…]’ is a set of accepted amino acids,
- ‘{…}’ is a set of non-accepted amino acids,
- ‘-’ separate amino acids in the pattern,
- ‘(k)’ indicates a repetition (k times),
- ‘<‘ and ‘>’ represent the beginning and end of the sequence.

Example:

< A-x-[ST](2)-x(0,1)-V-{C}

This pattern, which must be in the N-terminal of the sequence (`<'), is translated as: Ala-any-[Ser or Thr]-[Ser or Thr]-(any or none)-Val-(anything but Cys)

Executing a command

Solution 1: Use the function system() from the module os

Example: os.system('ls -l') calls the command ls from the script. N.B. The output is not captured and instead printed in the terminal as usual.

Solution 2: Use the subprocess module.

Example: subprocess.call(['ls', '-l']) Does the same as above.

subprocess module

subprocess.check_call(...) : Same as call but raise an Error if failed.

subprocess.check_output(...) : Run command with arguments and return its output as a byte string.

Example:

```
> o = subprocess.check_output(['ls','-l'])
> print o
total 9656\ndrwx------+  95 jeromew  staff    3230 23 Jan 12:03 Desktop\n ...
```