Master of Science in
Genomics and Bioinformatics

Lecture 9-2
Regular Expressions

GNBF5010

Instructor: Jessie Y. Huang

Regular Expression in general

What is Regular Expression

- A **Regular Expression**, or **RegEx** is a sequence of characters that defines a **search pattern**.
- Typically used to **find** a sequence of characters within a string, so you can extract and manipulate them.
- All modern languages have packages for RegEx.

- Example:
  abc+ matches a string that has ab followed by one or more c, like abccc

Regular Expression: Basic matching

- Each of these symbols matches a single character

`.`	Any character
`\d`	A digit character (0123456789)
`\D`	A non-digit character
`\w`	A word character (letters, digits, and _)
`\W`	A non-word character
`\s`	A whitespace character (`␣`, `\t`, `\r`, `\n`)
`\S`	A non-whitespace character
`␣`	A space
`\t`	A tab
`\n`	A new line character

Regular Expression: Quantifiers

`X*`	0 or more repetitions of X
`X+`	1 or more repetitions of X
`X?`	0 or 1 instance of X
`X{m}`	Exactly m instances of X
`X{m,n}`	Between m and n (inclusive) instances of X

- A quantifier by default just applies to its preceding character. We can use (…) to specify the explicit quantifier "scope".

Example: ab+ matches ab, abb, abbb, abbbb …
`(ab)+` matches ab, abab, ababab …

- Quantifiers are by default greedy in regex. We can use "?" to make it lazy.

Example: greedy: `^.*b` aabaaba
lazy: `^.*?b` aabaaba

Regular Expression: Character classes

- A character class [...] matches any of the characters in the class.
  Example: `[aeiou]` matches any vowels.

- Use ^ to specify the complement set.
  Example: [^aeiou] matches any non-vowels.

- Use - to specify a range of letters or digits.
  Example: [a-f] is equivalent to [abcdef]
  [0-9a-f] is equivalent to [0123456789abcdef]

- Note that a character class [...] matches a single character
  Example: `[abc][123]` matches `a2`, but not `ab2`

Regular Expression: Boundaries

- Used to "anchor" your pattern to some edge, but don't match any characters

^	Matches at the beginning of the line or string
$	Matches at the end of the line or string
\b	A word boundary, i.e. any edge between a `\w` and `\W`
\B	A non-word boundary

Example:

`\bcat\b` has a match in "`the cat in the hat`",
but not in "`locate`"

Regular Expression: Disjunction

`(X|Y)`	matches X or Y

Example:

`\b(cat|dog)s\b` matches cats or dogs, but not catdogs

Regular Expression: Special characters

- `{}[]()^$|.*+?\` and `-` inside a character class [...] have special meaning in regex, so must be “escaped” with a “\” to match the character themselves.
- Example:
  `\.` matches the period `.`
  `\\` matches the backslash `\`

Example

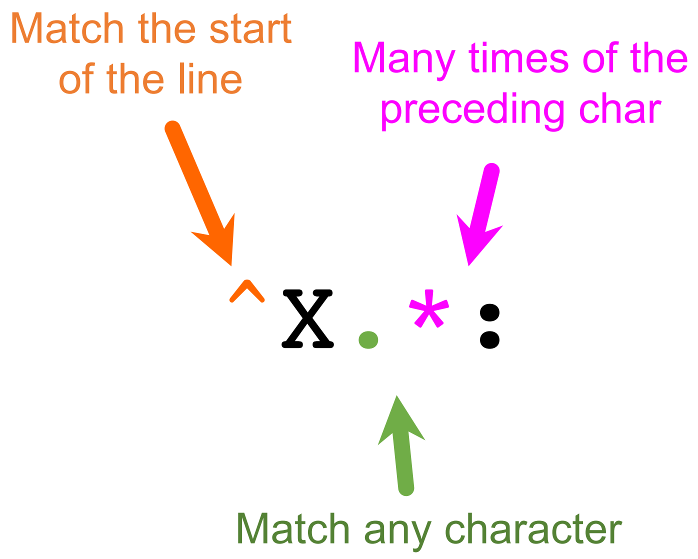
- Given a text file , write a regular expression that will match the following lines in the file.

```
...
X-Sieve: CMU Sieve 2.3
X-DSPAM-Result: Innocent
X-DSPAM-Confidence: 0.8475
X-Content-Type-Message-Body: text/plain
...
```

Example (continued)

- Given a text file , write a regular expression that will match the following lines in the file.

```
...
X-Sieve: CMU Sieve 2.3
X-DSPAM-Result: Innocent
X-DSPAM-Confidence: 0.8475
X-Content-Type-Message-Body: text/plain
...
```



Example (continued)

- Given a text file , write a regular expression that will match the following lines in the file.

```
...
X-Sieve: CMU Sieve 2.3
X-DSPAM-Result: Innocent
X-DSPAM-Confidence: 0.8475
X-Content-Type-Message-Body: text/plain
...
```

Match the start of the line

One or more non-space characters

```
r'^X-\S+:'
```

Match any non-whitespace character

(Fine-tuning your match)

Python Regular Expression

Regular Expression in Python

In python, regular expressions are usually notated with raw strings.

```
import re

str1 = "date is 28/11/2020"
re.split(r'[\s/]', str1)     # ['date', 'is', '28', '11', '2020']

str2 = "my activestate platform account is now active"
re.findall(r'ac..ve', str2) # ['active', 'active']
```

[Warning to Perl addicts] *Only use regex if there is no other way. Don't forget the python string methods and data structures.*

re module functions

- The re module enables the functionality of an regular expression (pattern A below). It also features a number of popular functions.

`re.findall(A, B)`	Returns a list of all matches of pattern A in string B
`re.search(A, B)`	Returns a Match object for the first occurrence of pattern A in string B; returns None if no match
`re.split(A, B)`	Splits string B into a list using pattern A as the delimiter; returns the list
`re.sub(A, B, C)`	Replaces occurrences of pattern A with string B in string C; returns the modified copy of C

- A Match object stores properties about a match, e.g. match_obj.span() returns a tuple of the start, and end position of the match; match_obj.group() returns the part of the string where there was a match

Examples

```
import re

txt1 = "12 dogs,11 cats, 1 egg"
x1 = re.findall(r'\d+', txt1)
print(x1)      # ['12', '11', '1']

txt2 = "This... is a test, short and sweet, of split()."
x2 = re.split(r'\W+', txt2)
print(x2)      # ['This', 'is', 'a', 'test', 'short',
               #      'and', 'sweet', 'of', 'split', '']

txt3 = "blue socks and red shoes"
txt3c = re.sub(r'(blue|white|red)', 'black', txt3)
print(txt3c)   # "black socks and black shoes“

txt4 = "We just received $10.00 for cookies."
x4 = re.findall(r'\$[0-9.]+', txt4)
print(x4)      # ["$10.00"]
```

Exercise

Write the regular expressions to match the strings with the following patterns. Use re.search() to test your expressions.

a) Has ‘q’ or ‘D’
b) Has ‘*th’ in it
c) Starts with ‘q’ or ‘D’
d) Has a substring where the first letter is ‘L’, ‘I’, ‘V’ or ‘M’, the second letter is ‘F’ or ‘Y’, and the third, forth and fifth letters are ’PWM’. For example, ‘`AVVYPWMIL`‘ will be a match.

References

- Chapter 11 of *Python for Everybody* (www.py4e.com)
- Chapter 21 of *A Primer for Computational Biology*

- http://web.mit.edu/hackl/www/lab/turkshop/slides/regex-cheatsheet.pdf
- https://www.w3schools.com/python/python_regex.asp
- https://docs.python.org/3/howto/regex.html

WHENEVER I LEARN A NEW SKILL I CONCOCT ELABORATE FANTASY SCENARIOS WHERE IT LETS ME SAVE THE DAY.
OH NO! THE KILLER MUST HAVE FOLLOWED HER ON VACATION!
BUT TO FIND THEM WE'D HAVE TO SEARCH THROUGH 200 MB OF EMAILS LOOKING FOR SOMETHING FORMATTED LIKE AN ADDRESS!
IT'S HOPELESS!
EVERYBODY STAND BACK.
I KNOW REGULAR EXPRESSIONS.
PERL!
TAP TAP

http://xkcd.com/208/