FIT5196 DATA WRANGLING

Week 3

Regular Expressions

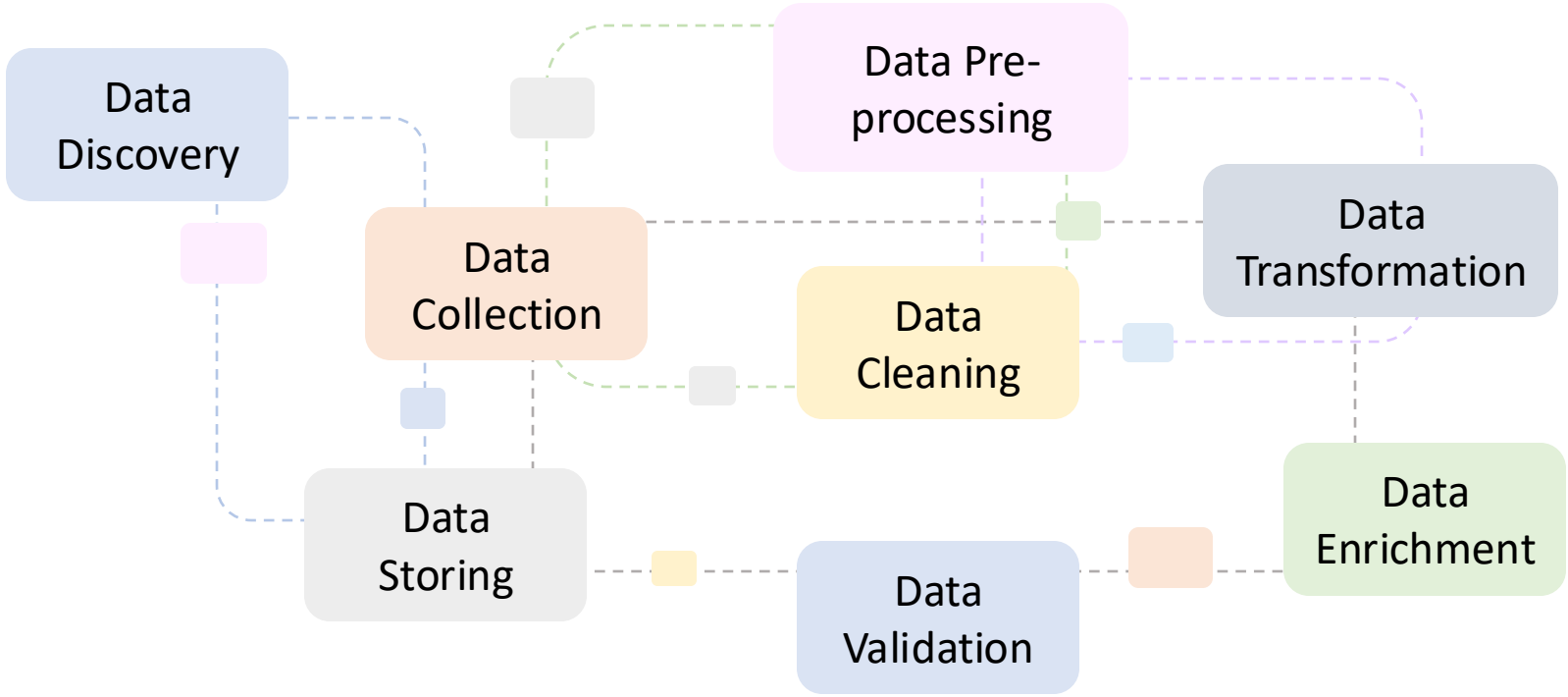
By Jackie Rong

Faculty of Information Technology

Monash University

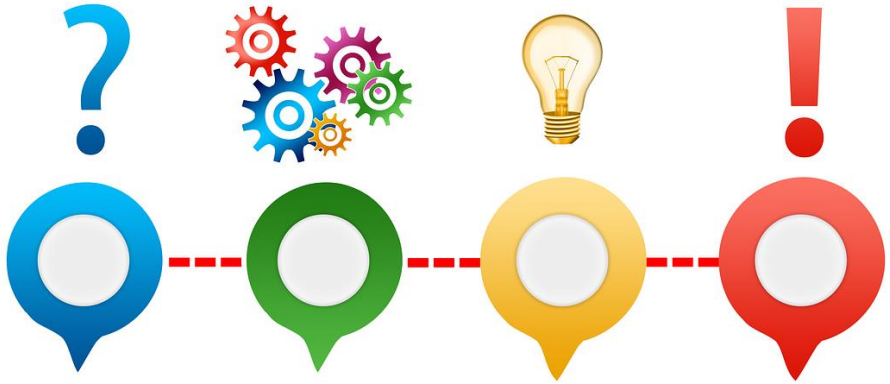
Data Wrangling Tasks

- **Data Wrangling** is the process of acquiring, cleaning, structuring, and enriching raw data into a format that is directly usable for analysis.



Outline

- What are Regular Expressions and Why?
- Regular Expression Syntax
  - Character Sets
  - Repetition
  - Grouping
  - Raw String in Python



MONASH University

Regular Expressions

- A **regular expression** is a set of symbols that describes a text pattern.
- Why regular expressions?
  - Regular expressions are useful in finding, replacing and extracting information from text, such as log files, HTML/XML files, and other documents
    - Search a document for colour or neighbour with or without 'u'
    - Convert a tab-delimited file to a comma-delimited file
    - Find duplicate words in a text
    - Search and replace "Bob" and "Bobby" with "Robert"
  - Regular expressions are useful in verifying whether input fits into the text pattern, such as verifying
    - phone numbers: Does a phone number have the right number of digits?
    - emails: Is an email address in a valid format?
    - date: Is a date in the right format? Does the month exceed 12?

Example – Validate Emails

r"(^[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$)"[1]

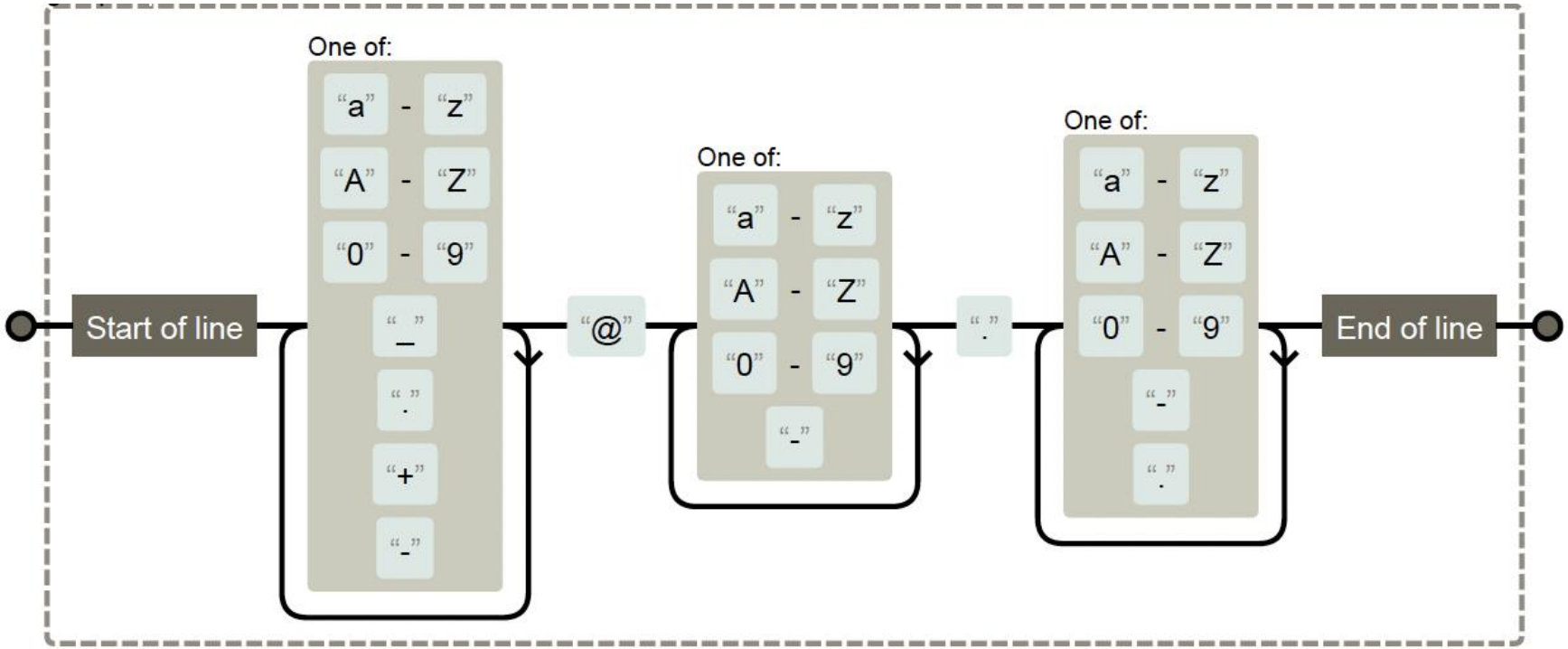
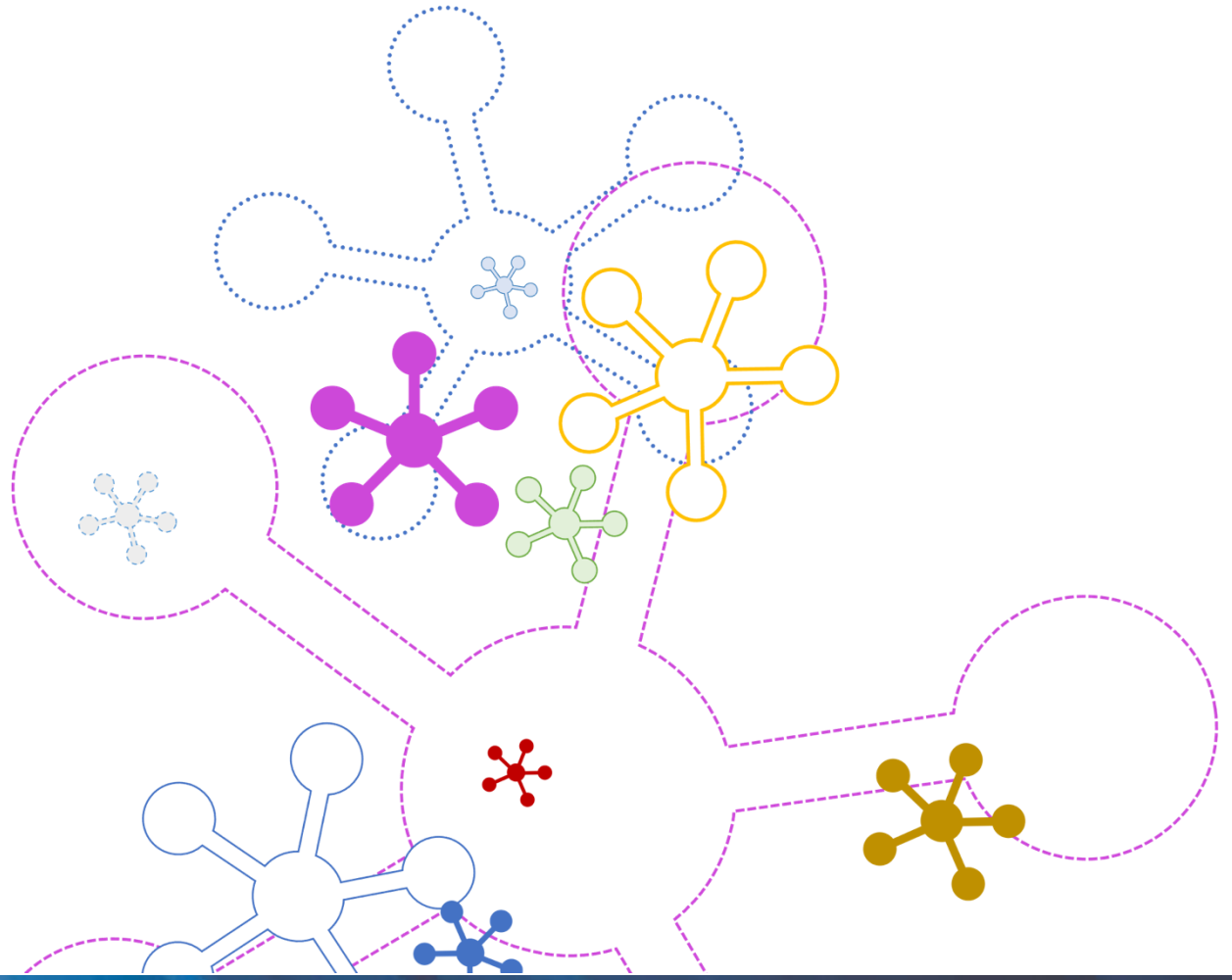


Figure: Figure generated by https://regexper.com/

[1]http://emailregex.com/

Regular Expression Syntax

- Character Sets
- Repetition
- Grouping
- Raw String in Python



Character Sets

- The most obvious feature of regular expressions is matching strings with one or more literal characters, called string literals.
  - Everything is essentially a character in regular expressions.
    - cat matches “cat”
    - cat matches the first three characters of “cattle” and “catfish”
  - It is similar to searching in text editing program.
  - Matching is case-sensitive:
    - cat does not match “Cat”.
  - How does regular expression engine work?

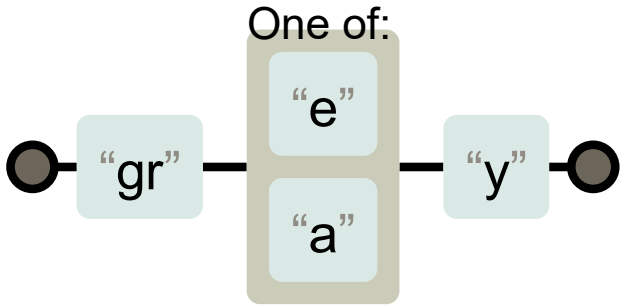
Character Sets […]

Assume that we are going to match the following two words:

grey gray

- What should the regular expression be?

  […] indicate a set of characters
  - Matches any one of several characters in the set, but only one
  - The order of characters does not matter.
  - The regular expression is **gr[ea]y**
  - **gr[ea]y** does not match grAy, graay, or graey.



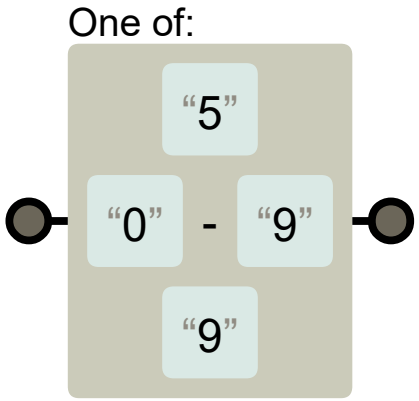
Character Ranges [a-zA-Z] and [0-9]

Assume that we are going to match Victorian car plate numbers, for example

XRA 000, 1AA 1AA

Note the letters can be from A to Z, and the numbers can be from 0 to 9. What should the regular expression be?

- Character ranges can be indicated by giving two characters and separating them by a '-'.
  - Example:
    - [0-9]
    - [a-z] or [A-Z]
- Caution
  - [50-99] is not all numbers from 50 to 99, it is the same as [0-9].



Character Ranges [a-zA-Z] and [0-9]

Assume that we are going to match victory car plate numbers, for example

XRA 000, 1AA 1AA

Note the letters can be from A to Z, and the numbers can be from 0 to 9.

What the regular expression should be?

[A-Z0-9][A-Z][A-Z]\s[0-9][A-Z0-9][A-Z0-9]

Negative Character Sets [^…]

Assume that we are going write a regular expression that present only the live animals from the following text

hog dog bog

what is the regular expression?

[^…]: If the first character of the set is ^, all the characters that are not in the set will be matched.

- [^b]og matches “hog" and “dog", but not “bog".
- Caution:
  - Does see[^mn] match “see”?
  - Does see[^mn] match “see ”?

Try the regular expression in Pythex (http://pythex.org/)!

Metacharacters Inside Character Sets [.+]

Assume that we are going to match the following two strings:

var(9), var[0]

Now, we need to match **()** and **[ ]**, how can we do that?

- Metacharacters inside character sets are already escaped. In other words, they lose their special meaning inside sets.
  - Example:
    - h[ai.u]t matches “hat”, “h.t”, but not “hot”
- Exceptions
  - **]**, **-**, **^** and **** that do need to be escaped.
    - h[ai.u]t → h[ai]u]t?

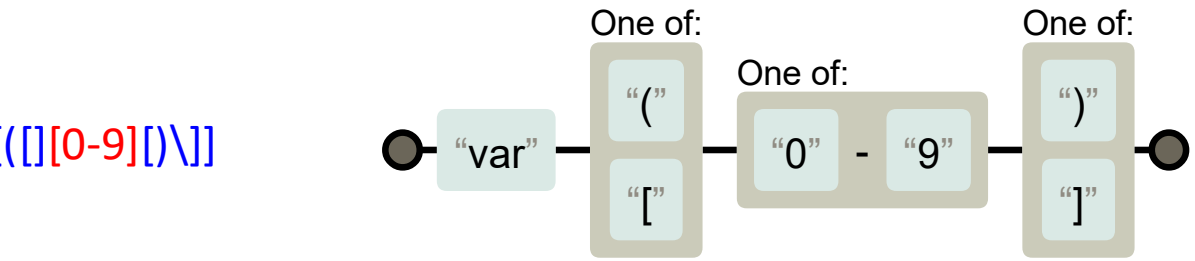
Meta-characters Inside Character Sets [.+]

Assume that we are going to match the following two strings:

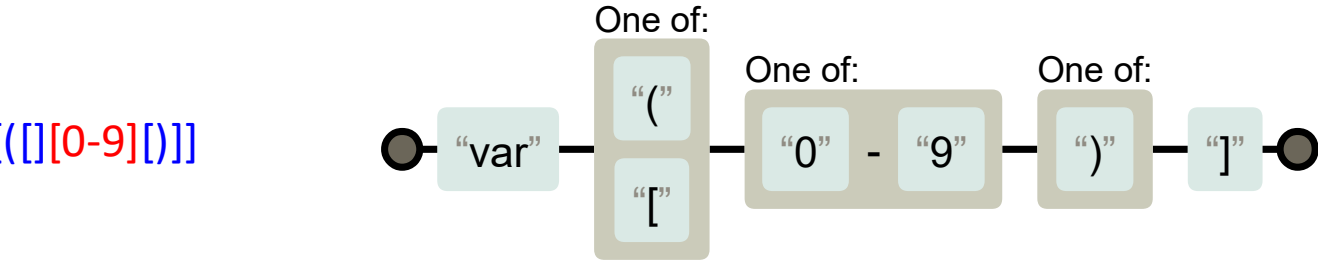
var(9), var[0]

Now, we need to match **()** and **[ ]**, how can we do that?

var[([][0-9][)\]]



var[([][0-9][)]]



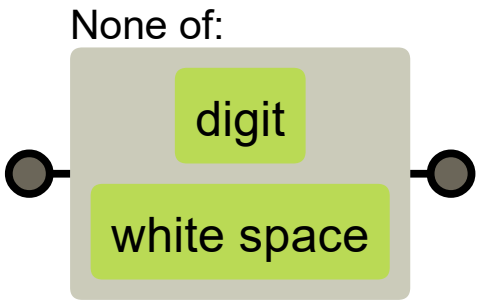
Shorthand Character Sets

Shorthand	Meaning	Equivalent
\d	Matches any decimal digits from 0 to 9	[0-9]
\w	Matches any word character	[a-zA-Z0-9_]
\s	Matches any white space character	[\t\n\r]
\D	Matches any non-digit character	[^0-9]
\W	Matches any non-alphanumeric character	[^a-zA-Z0-9_]
\S	Matches any non-whitespace character	[^ \t\n\r]

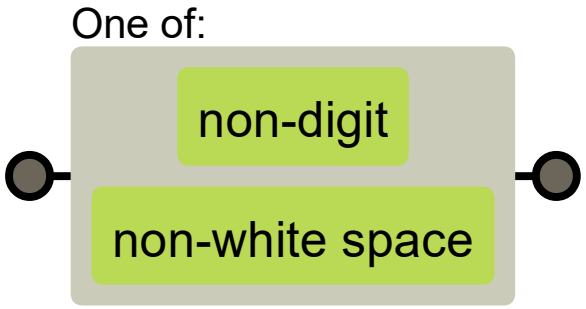
- \d\d\d\d matches four-digit numbers, such as “2016”, but not text.
- \w\w\w matches three-word characters, such as “abc”, “123” and “d_b”
- \w\w\s\w matches “ab c” but not “a bc”.
- [\w]-[\w] matches two characters separated by a hyphen.
- ^\d] is the same as [\D]

Shorthand Character Sets

- Caution:
  - Is [^\d\s] the same as [\D\S]?
    - [^\d\s]: Not digit OR space character



    - [\D\S]: EITHER NOT digit OR NOT space character



Repetition Expression

- Repetition meta-characters

Meta-character	Meaning
*	Match 0 or more repetitions of the preceding regex
+	Match 1 or more repetitions of the preceding regex
?	Match 0 or 1 repetition of the preceding regex

Examples: Assume we are going to match the following words

oops ooops ooooops oooooops

but not

ops

which regular expression(s) should we use?

A. oo*ps
B. ooo*ps
C. oo+ps
D. oo?ps

Repetition Expression

- Repetition meta-characters

Meta-character	Meaning
*	Match 0 or more repetitions of the preceding regex
+	Match 1 or more repetitions of the preceding regex
?	Match 0 or 1 repetition of the preceding regex

Examples: Assume we are going to match the following words

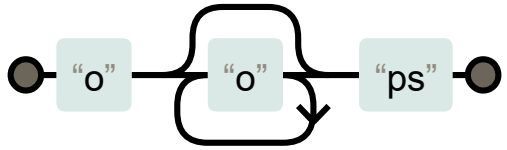
oops ooops oooooops ooooooops

but not

ops

which regular expression(s) should we use?

A. oo*ps
B. ooo*ps
C. oo+ps
D. oo?ps



Repetition Expression

- Repetition meta-characters

Meta-character	Meaning
*	Match 0 or more repetitions of the preceding regex
+	Match 1 or more repetitions of the preceding regex
?	Match 0 or 1 repetition of the preceding regex

Examples: Assume we are going to match the following words

oops ooops oooooops ooooooops

but not

ops

which regular expression(s) should we use?

A. oo*ps
B. ooo*ps
C. oo+ps
D. oo?ps

Repetition Expression

- Repetition meta-characters

Meta-character	Meaning
*	Match 0 or more repetitions of the preceding regex
+	Match 1 or more repetitions of the preceding regex
?	Match 0 or 1 repetition of the preceding regex

Examples: Assume we are going to match the following words

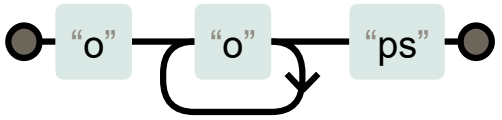
oops ooops oooooops ooooooops

but not

ops

which regular expression(s) should we use?

A. oo*ps
B. ooo*ps
C. oo+ps
D. oo?ps



Repetition Expression

- Repetition meta-characters

Meta-character	Meaning
*	Match 0 or more repetitions of the preceding regex
+	Match 1 or more repetitions of the preceding regex
?	Match 0 or 1 repetition of the preceding regex

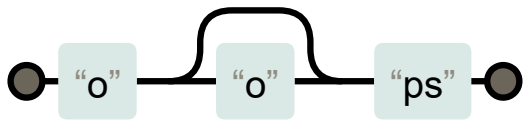
Examples: Assume we are going to match the following words

oops ooops oooooops ooooooops

but not ops

which regular expression(s) should we use?

A. oo*ps
B. ooo*ps
C. oo+ps
D. oo?ps



Try the regular expression in Pythex!

Repetition Expression

- Quantified repetitions
- {m, n}: matches exactly from m to n repetitions of the preceding regular expression.
  - m (min) and n (max) are positive numbers
  - m must be always be included, can be 0
  - n is optional
- Three syntax
  - \d{2} matches numbers with exactly 2 digits.
  - \d{2, 4} matches numbers with 2 to 4 digits.
  - \d{2, } matches numbers with at least 2 digits (n is infinite).

Try the “oops” example in Pythex, but with {m, n}

Repetition Expression

Suppose we are going to match the following

report_2016_09	assignment_2016_9
budget_16_08	assignment_08_7

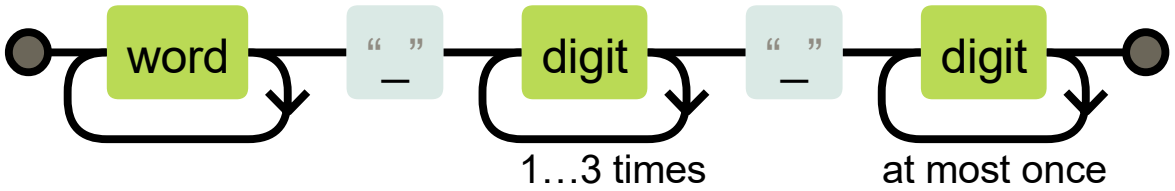
but not

report_201609_39	assignment_6_9000
budget_2345678_08	assignment_000999_7

what is the regular expression?

- (

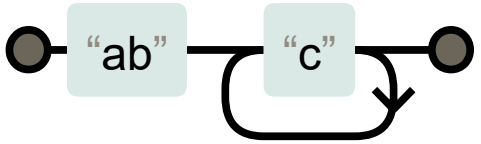
\w+_\d{2,4}_\d{1,2}



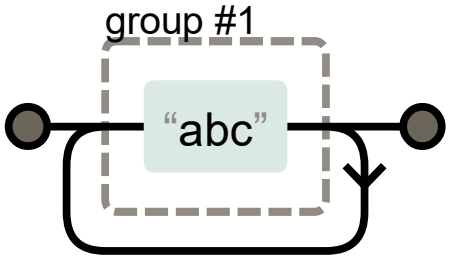
Grouping (…)

- (…) matches whatever regular expression is inside the parentheses and indicates the start and end of a group.
  - Apply repetition operators to a group of regular expressions
  - Makes regular expressions easier to read
  - Capture groups for use in matching, replacing and extraction, i.e., the contents of a group can be retrieved.
  - Cannot be used insides a character set.
- For example,

abc+ matches abc, abcc, abcccc

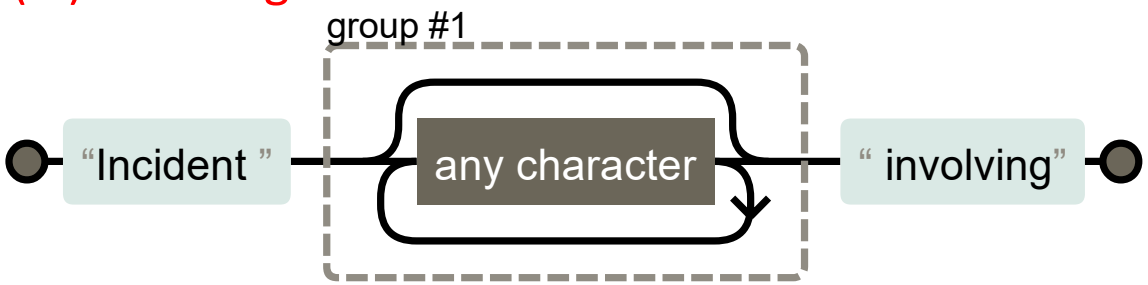


(abc)+ matches abc, abcabc, abcabcabc



Grouping (…)

- (…) matches whatever regular expression is inside the parentheses and indicates the start and end of a group.
  - Apply repetition operators to a group of regular expressions
  - Makes regular expressions easier to read
  - Capture groups for use in matching, replacing and extraction, i.e., the contents of a group can be retrieved.
  - Cannot be used insides a character set.
- For example,
  - "Incident American Airlines Flight 11 involving a Boeing 767-223ER in 2001"
  - Regular expression: (.*) involving



Try it with python script from https://regex101.com/ !!!

Alternation |

- "|" is an OR operator
  - A|B will match any string that matches either A or B
  - Ordered: leftmost expression gets precedence.
  - Multiple patterns can be daisy-chained.
  - Group alternation expressions to keep them distinct.
- Examples:
  - apple|orange matches "apple" and "orange"
  - (apple|orange) juice matches "apple juice" and "orange juice"
  - w(ei|ie)rd matches both "weird" and "wierd".

The Backslash Plague \

- The back slash \indicates special forms or to allow special characters to be used without invoking their special meaning.

Character	Stage
\section	Text string to be matched
\\section	Escaped backslash for re.compile()
\\\\section	Escaped backslashes for a Python string literal

- So, to match a literal backslash, one has to write '\\\\' as the regular expression string
- Can we simplify the expression?

Raw String r”…”

- Raw String suppress actual meaning of escape characters, and do not treat the backslash as a special character at all.

Regular Python String Literal	Raw String
”\\\\section”	r”\\section”
”\\w+\\s+”	r”\w+\s+”

- Regular expressions will often be written in Python code using this raw string notation.

Summary & To-do List

- Please download and read materials provided on Moodle.
- Review content learnt from Week 3.

- Assessments
  - Make your group selection for Assessment 1
  - Read Assessment 1 specification
  - Read the tasks and start to allocate the work.

- Next week: Exploratory Data Analysis (EDA)