

Regular expressions

- Regular expressions are a very powerful tool to do string matching and processing
- Allows you to do things like:
 - Tell me if a string starts with a lowercase letter, then is followed by 2 numbers and ends with "ing" or "ion"
 - Replace all occurrences of one or more spaces with a single space
 - □ Split up a string based on whitespace or periods or commas or ...
 - Give me all parts of the string where a digit is proceeded by a letter and then the '#' sign

Regular expressions: literals

- We can put any string in a regular expression

 /test/
 - matches any string that has "test" in it
 - /this class/
 - matches any string that has "this class" in it
 - /Test/
 - case sensitive: matches any string that has "Test" in it

Regular expressions: character classes

- A set of characters to match:
 put in brackets: []
 - [abc] matches a single character a or b or c
- □ For example:
- /[Tt]est/
 - matches any string with "Test" or "test" in it
- Can use to represent ranges
 - [a-z] is equivalent to [abcdefghijklmnopqrstuvwxyz]
 - [A-D] is equivalent to [ABCD]
 - [0-9] is equivalent to [0123456789]

Regular expressions: character classes

$\hfill\square$ For example:

- /[0-9][0-9][0-9][0-9]/
 matches any four digits, e.g. a year
- Can also specify a set NOT to match
- ^ means all character EXCEPT those specified
- [^Aa] all characters except 'a'
- [^0-9] all characters except numbers
- [^A-Z] not an upper case letter

Regular expressions: character classes

Meta-characters

- w word character (a-zA-Z_0-9)
- $\blacksquare \setminus W$ non word-character (i.e. everything else)
- □ \d digit (0-9)
- \s whitespace character (space, tab, endline, ...)
- . matches any character

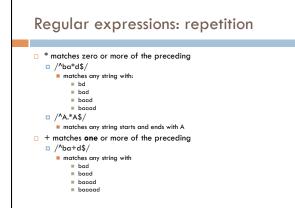
For example

- □ /19\d\d/
- would match any 4 digits starting with 19
- □ /\s/
- \blacksquare matches anything with a whitespace (space, tab, etc) \blacksquare /\S/ or /[^\s]/
- matches anything with at least one non-space character

Regular expressions:

beginning and end

- \square ^ marks the beginning of the line
- \$ marks the end of the line
- /test/
- test can occur anywhere
 /^test/
- must start with test
- /test\$/
- must end with test
- /^test\$/
- must be exactly test





```
? zero or 1 occurrence of the preceding
/fights?/
```

matches any string with "fight" or "fights" in it

- □ {n,m} matches n to m inclusive
- □ /ba{3,4}d/
- matches any string with
 - baaad
 - baaaad

Regular expressions: repetition revisited

- What if we wanted to match:
- This is very interesting
- This is very very interesting
- This is very very very interesting
- Would /This is very+ interesting/ work?
 No... + only corresponds to the 'y'
 - /This is (very)+interesting/

Regular expressions: disjunction

- | has the lowest precedence and can be used
 /cats|dogs/
 - matches:
 - cats
 - dogsdoes NOT match:
 - = catsogs
 - /^l like (cats | dogs)\$/
 - matches:
 - I like cats
 I like dogs

Some examples

- All strings that start with a capital letter
- IP addresses
- 255.255.122.122
- Matching a decimal number
- □ All strings that end in ing
- $\hfill\square$ All strings that end in ing or ed
- All strings that begin and end with the same character

Some examples

- All strings that start with a capital letter
 /^[A-Z]/
- IP addresses
- /\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}/
- Matching a decimal number
- □ /[-+]?[0-9]*\.?[0-9]+/
 □ All strings that end in ing
- □ /ing\$/
- All strings that end in ing or ed
 /(ing|ed)\$/

Regular expressions: memory

- All strings that begin and end with the same character
- Requires us to know what we matched already
- □ ()
 - used for precedence
 - also records a matched grouping, which can be referenced later
- □ /^(.).*\1\$/
 - all strings that begin and end with the same character

Regular expression: memory

- \square /She likes (\w+) and he likes $\backslash 1/$
- $\hfill\square$ We can use multiple matches $\hfill\blacksquare$ /She likes (\w+) and (\w+) and he also likes \1 and \2/

Regular expression search

```
□ <string> =~ /regex/
```

string_var> =~ /regex/

returns the index of the first occurrence if there is a match nil if it does not match

```
>> "this is a test" =~ /is/
=> 2
>> "this is a test" =~ /blah/
=> nil
```

```
>> "this is a test" =~ /^.*(is).*1/
```

```
 \begin{array}{l} \Rightarrow & \text{This is a test} = -2 \\ \Rightarrow & 0 \\ \Rightarrow & x = \text{"this is a test"} \\ \Rightarrow & \text{this is a test"} \\ \Rightarrow & x = - /^{*}.*(\text{is}).* \setminus 1 / \\ \Rightarrow & 0 \end{array}
```

Regular expressions: substitution

- We can also substitute matches
 - □ sub returns a new string with the substitution. only substitutes first occurrence
 - sub! ALSO modifies the current string
 - gsub substitutes ALL occurrences of the pattern, but does not modify
 - gsub! ALSO modifies current string

Regular expression subsitution

>> x = "test" => "test"

- >> x.sub(/t/, "e") => "eest"
- >> x => "test"
- >> x.sub!(/t/, "e") => "eest"
- >> x => "eest"
- >> x = "test"
- => "test"
- >> x.gsub(/t/, "e") => "eese"