Regular expressions

Michelle Torres

August 22, 2017

REGULAR EXPRESSIONS: WHAT, WHY AND HOW?

- Regular expressions are useful to extract information from text.
- Set of “rules” to identify or match a particular sequence of characters.
- Most text in ASCII: letters, digits, punctuation and symbols (but unicode can also be used)
- In Python, mainly through library `re`.

THE ABC'S (AND 123S) OF REGEXPR

- ABC: just letters
- Numbers: the character "\d" can be used in place of any digit from 0 to 9
- Wildcard: . [to actually look for a period, use"\."]
  - `"cat."`, `"896."`, `"?=+."`, `"abc1"`

INCLUDING CHARACTERS

- The "." sometimes is too powerful!
- Use brackets []. The instruction will only match a single character/letter inside the bracket and nothing else.
  - Match ONLY the first 3 words: `"can", "man", "fan", "dan", "ran", "pan"`

INCLUDING CHARACTERS

- The "." sometimes is too powerful!
- Use brackets []. The instruction will only match a single character/letter inside the bracket and nothing else.
  - Match ONLY the first 3 words: `"can", "man", "fan", "dan", "ran", "pan"` ⇒ [cmf]an

EXCLUDING CHARACTERS

- To exclude chunks, use a hat (^)
- `[^abc]`: do not include if there's an a. b or c.
- Filter the first two words: `"hog", "dog", "bog"`

EXCLUDING CHARACTERS

- To exclude chunks, use a hat (^)
- `[^abc]`: do not include if there's an a. b or c.
- Filter the first two words: `"hog", "dog", "bog"` ⇒ [^b]og

SOME TRICKS!

- Ranges: [0-6], [a-z]
- Lower vs. Upper: [a-zA-Z]
- Shortcut for characters in English: \w = [a-zA-Z0-9_]
  - Filter the first 3 words: `"Ana","Bob","Cpc", "aax", "bby", "ccz"`

SOME TRICKS!

- Ranges: [0-6], [a-z]
- Lower vs. Upper: [a-zA-Z]
- Shortcut for characters in English: \w = [a-zA-Z0-9_]
  - Filter the first 3 words: `"Ana","Bob","Cpc", "aax", "bby", "ccz"` ⇒ [A-Z]\w\w
- Repetitions: a{3} = match the character a 3 times, .{2,6} = between 2 and 6 of any character.
  - Match ONLY the first two words: `wazzzzzzup, wazzup, wazup` ⇒ waz{2,}up
- *Kleene* star represents either 0 or more or 1 or more of the character that it follows: \d* =any number of digits, \d+ = at least one digit.
  - Match the first three words: `"aaaabcc","aabbbbc", "aacc", "a"`

SOME TRICKS!

- Ranges: [0-6], [a-z]
- Lower vs. Upper: [a-zA-Z]
- Shortcut for characters in English: \w = [a-zA-Z0-9_]
  - Filter the first 3 words: `"Ana","Bob","Cpc", "aax", "bby", "ccz"` ⇒ [A-Z]\w\w
- Repetitions: a{3} = match the character a 3 times, .{2,6} = between 2 and 6 of any character.
  - Match ONLY the first two words: `wazzzzzzup, wazzup, wazup` ⇒ waz{2,}up
- *Kleene* star represents either 0 or more or 1 or more of the character that it follows: \d* =any number of digits, \d+ = at least one digit.
  - Match the first three words: `"aaaabcc","aabbbbc", "aacc", "a"` ⇒ a{2}.*

MORE TRICKS...

- Optionality: ab?c = match either “ac” or ”abc” because “b’ is optional.
  - Match the first 3 words: `1 file found?, 2 files found?, 13 files found?, No files found.`

MORE TRICKS...

- Optionality: ab?c = match either “ac” or ”abc” because “b’ is optional.
  - Match the first 3 words: `1 file found?, 2 files found?, 13 files found?, No files found.` ⇒ \d+ files? found\?
- White space(s): \r,\n,\t,“ ” → Use \s!
- Whitespace characters are just like any other character and the special metacharacters like the star and the plus can be used as well.
  - Match the first 3 words from: `1. abc, 2. abc, 3. abc, 4.abc`

MORE TRICKS...

- Optionality: ab?c = match either “ac” or ”abc” because “b’ is optional.
  - Match the first 3 words: `1 file found?, 2 files found?, 13 files found?, No files found.` ⇒ \d+ files? found\?
- White space(s): \r,\n,\t,“ ” → Use \s!
- Whitespace characters are just like any other character and the special metacharacters like the star and the plus can be used as well.
  - Match the first 3 words from: `1.  abc, 2.  abc, 3. abc, 4.abc` ⇒ \d\.\s+[a-z]{3}
- \S: any non-space character

LOCATION IN A WORD

- Start: ^
  - Extract “Mission” but only from sentences that begin with that word $\Rightarrow$ ^Mission
- End: $, `word`$
- Grouping: ()
  - Extract the file name without the extension
    `file_record_transcript.pdf`

LOCATION IN A WORD

- Start: ^
  - Extract “Mission” but only from sentences that begin with that word $\Rightarrow$ ^Mission
- End: $, `word`$
- Grouping: ()
  - Extract the file name without the extension `file_record_transcript.pdf` $\Rightarrow$ ^(file\w+)
- Nested groups: extract multiple layers of information.
  - Get the full date and year of `"Jan 1987", "MAy 1969", "Aug 2011"`

LOCATION IN A WORD

- Start: ^
  - Extract “Mission” but only from sentences that begin with that word ⇒ ^Mission
- End: $, `word`$
- Grouping: ()
  - Extract the file name without the extension `file_record_transcript.pdf` ⇒ ^(file\w+)
- Nested groups: extract multiple layers of information.
  - Get the full date and year of `"Jan 1987", "MAy 1969", "Aug 2011"` ⇒ (\w{3}\s(\d+))
- Conditionals: `I love (cats|dogs)`
- Metacharacters: \d, \w, \s, \D, \W, \S

PYTHON IMPLEMENTATION

- Use *raw* strings instead of regular Python strings. Raw strings begin with a special prefix (r) and signal Python not to interpret backslashes and special metacharacters in the string, allowing you to pass them through directly to the regular expression engine $\Rightarrow$ a pattern like "\n\w" will not be interpreted and can be written as r"\n\w" instead of "\\n\\w" as in other languages, which is much easier to read.

PYTHON FUNCTIONS

matchObject = re.search(pattern, input_str, flags=0)

matchList = re.findall(pattern, input_str, flags=0)

matchList = re.finditer(pattern, input_str, flags=0)

replacedString = re.sub(pattern, replacement_pattern, input_str, count, flags=0)