

# Regular Expressions

# Regular Expressions

- `regexp(str,pattern,options)`

```
>> str = 'bat cat can car COAT court cut ct  
CAT-scan';
```

```
>> regexpi(str, 'c[aeiou]+t')
```

```
ans =
```

```
    5    17    28    35
```

```
>> regexpi(str, 'c[aeiou]+t', 'match')
```

```
ans =
```

```
'cat'  'COAT'  'cut'  'CAT'
```

# Regular Expressions: capture

```
>> str = 'bat cat can car COAT court cut ct  
CAT-scan';
```

```
>> C=regexp(str, 'c([aeiou]+)t', 'tokens')
```

```
C =
```

```
 {1x1 cell} {1x1 cell} {1x1 cell} {1x1 cell}
```

```
>> [C{:}]
```

```
ans =
```

```
'a' 'OA' 'u' 'A'
```

# Regular Expressions

---

.	any character
[abc]	any of the a b c characters
[^abc]	none of the a b c characters
[a-c]	any character in the range a to c.
\s	any white-space. [ \f\n\r\t\v]
\S	any non-white-space. [^ \f\n\r\t\v]
\w	any alphanumeric or underscore. [a-zA-Z_0-9]
\W	any non-alphanumeric or underscore.
\d	any digit. [0-9]
\D	any non-digit.
\	escape a special character. e.g., \. will match a period, \[ will match an opening square bracket.

---

# Regular Expressions

```
>> str = 'The rain in Spain falls mainly on the plain.';
```

```
>> regexp(str, '..ain', 'match')
```

```
ans =
```

```
'rain' 'Spain' 'main' 'plain'
```

```
>> regexp(str, '[rpm]ain', 'match')
```

```
ans =
```

```
'rain' 'pain' 'main'
```

```
>> regexp(str, '[A-Z]\w*', 'match')
```

```
ans =
```

```
'The' 'Spain'
```

# Repetition

```
>> str = 'ggle gogle google googole goooogle';
```

```
>> regexp(str, 'go?gle', 'match')  
'ggle' 'gogle'
```

```
>> regexp(str, 'go+gle', 'match')  
'gogle' 'google' 'googole' 'goooogle'
```

```
>> regexp(str, 'go*gle', 'match')  
'ggle' 'gogle' 'google' 'googole' 'goooogle'
```

```
>> regexp(str, 'go{2,}gle', 'match')  
'google' 'gooogle' 'goooogle'
```

```
>> regexp(str, 'go{2,3}gle', 'match')  
'google' 'gooogle'
```

?	zero or one
+	one or more
*	zero or more
{m,n}	at least m, at most n
{n,}	n or more
{n}	exactly n

# Exercise

```
>> str = 'After 1st one comes the 2nd item.  
After 3rd comes thee 4th. 2+10.5 is not  
13.5.';
```

- Extract all words that come after a "the" or "thee".
- Extract the last words of the sentences. (A sentence ends with a period and a word must contain at least one letter.)
- Extract all numbers.

# Grouping

- Non-capture grouping: `(?:expr)`  
>> `str = 'Marge lets Norah see Sharon's telegram';`  
>> `regexp(str, '[^aeiou][aeiou]{2,}', 'match')`  
`ans =`  
`'see'`  
  
>> `regexp(str, '(?:[^aeiou][aeiou]){2,}', 'match')`  
`ans =`  
`'Nora'    'haro'    'tele'`



# Lookaround

- Look ahead
  - `(?=expr)`
  - `(?!expr)`
- Look behind
  - `(?<=expr)`
  - `(?<!expr)`

```
>> regexp('telegraph television telephone', '(tele)\w*', 'match')
ans =
    'telegraph'    'television'    'telephone'
```

```
>> regexp('telegraph television telephone', '(?<=tele)\w*', 'match')
ans =
    'graph'    'vision'    'phone'
```

# Alternative match - expr1|expr2

```
>> str='10 kilometer. 1 km.';
```

```
>> regexp(str, '\d+  
(kilometer|kilometers|km)', 'match')
```

```
ans =
```

```
    '10 kilometer'    '1 km'
```

```
>> str='black velvet';
```

```
>> regexp(str, 'bl(ack|ue)', 'match')
```

```
ans =
```

```
    'black'
```

# Exercise

```
>> str='They have a black cat, a gray elephant, a yellow snake, a brown pigeon, a gray wolf, and a white bear at the zoo.'
```

- Extract all of the black or gray animals that they have at the zoo.

# Positional Operators

>> str='abc bac ade cba'

'a\w+': abc, ac, ade

'^a\w+': abc

'\<a\w+': abc, ade

'\w+a\$': cba

Operator	Usage
^expr	beginning of the input string.
expr\$	end of the input string.
\<expr	beginning of a word.
expr\>	end of a word.

# Backreferencing Tokens: \N

```
>> regexp('one=one, one=two, two=two  
two=three', '(\w+)=\1', 'match')
```

```
ans =
```

```
'one=one'    'two=two'
```

# Named tokens: (?<expr>)

```
>> str='John Smith; John Brown';  
>> a = regexp(str, '(?<firstname>[A-Z][a-z]+) (?<lastname>[A-Z][a-z]+)', 'names')
```

```
a =
```

```
1x2 struct array with fields:
```

```
    firstname  
    lastname
```

```
>> a(1)
```

```
ans =
```

```
    firstname: 'John'  
    lastname: 'Smith'
```

# Named Tokens

```
>> str1 = '134 Main Street, Boulder, CO, 14923';
```

```
>> regexp(str1,[ ...  
'(?<adrs>\d+\s\S+\s(Road|Street|Avenue|Drive)), ' ...  
'(?<city>[A-Z][a-z]+), ' ...  
'(?<state>[A-Z]{2}), ' ...  
'(?<zip>\d{5})' ] , 'names')
```

```
ans =
```

```
adrs: '134 Main Street'  
city: 'Boulder'  
state: 'CO'  
zip: '14923'
```

# Tokens in replacement: \$N

```
>> str='John Smith; John Brown';
```

```
>> regexprep(str, ...
```

```
'([A-Z][a-z]+) ([A-Z][a-z]+)', '$2, $1')
```

```
ans =
```

```
Smith, John; Brown, John
```

```
>> regexprep(str, ...
```

```
'(?<firstname>[A-Z][a-z]+) (?<lastname>[A-Z][a-z]+)', '$<lastname>, $<firstname>')
```



# Dynamic Regular Expressions: \${cmd}

- Append all words with their lengths:  
  
>> str = 'ggle gogle google gooogle goooogle';  
>> regexprep(str, '(\w+)', ...  
'\$1:\${num2str(length(\$1))}')  
  
ans =  
ggle:4 gogle:5 google:6 gooogle:7 goooogle:8

# Dynamic Expressions

- Randomize all but first and last letter of each word:

```
>> s='One flew over the cuckoo''s nest.';
```

```
>> regexprep(s, '(?<=\w)\w{2,}(?=\w)',  
'${$0(randperm(length($0)))}')
```

# Exercise: Cesar cipher

- The following cipher was generated by replacing each letter with the next letter in the alphabet. Use a `regexprprep` command to decipher it.

```
>> str='uif rvjdl cspxo gpy kvnqt pwfs uif  
eph';
```

# Exercise: Typoglycemia

- *Create a typoglycemic version of a given string.*
- *According to a researcher (sic) at Cambridge University, it doesn't matter in what order the letters in a word are, the only important thing is that the first and last letter be at the right place. The rest can be a total mess and you can still read it without problem. This is because the human mind does not read every letter by itself but the word as a whole.*
- *Aoccdrnig to a rscheearch at Cmabrigde Uinervtisy, it deosn't mttar in waht oredr the ltteers in a wrod are, the olny iprmoetnt tihng is taht the frist and lsat ltteer be at the rghit pclae. The rset can be a toatl mses and you can sitll raed it wouthit porbelm. Tihs is bcuseae the huamn mnid deos not raed ervey lteter by istlef, but the wrod as a wlohe.*