Exploring Regular Expression Usage and Context in Python

Carl Chapman, Kathryn T. Stolee*

Iowa State University, North Carolina State University

carlallenchapman@gmail.com, ktstolee@ncsu.edu

19 July, 2016

Why Regular Expressions?

- Regexes are everywhere! (we think...)

Why Regular Expressions?

- Regexes are everywhere! (we think...)
- Everyone writes regexes! (we think...)

Why Regular Expressions?

- Regexes are everywhere! (we think...)
- Everyone writes regexes! (we think...)
- Regexes are hard to read/write! (again, we think...)

Why Regular Expressions?

- Regexes are everywhere! (we think...)
- Everyone writes regexes! (we think...)
- Regexes are hard to read/write! (again, we think...)
- So, we wanted to write a tool to support regex creation.

Why Regular Expressions?

- Regexes are everywhere! (we think...)
- Everyone writes regexes! (we think...)
- Regexes are hard to read/write! (again, we think...)
- So, we wanted to write a tool to support regex creation.
- But...

Why Regular Expressions?

- Regexes are everywhere! (we think...)
- Everyone writes regexes! (we think...)
- Regexes are hard to read/write! (again, we think...)
- So, we wanted to write a tool to support regex creation.
- But...

Regex feature usage references are missing!

Why Regular Expressions?

- Regexes are everywhere! (we think...)
- Everyone writes regexes! (we think...)
- Regexes are hard to read/write! (again, we think...)
- So, we wanted to write a tool to support regex creation.
- But...

Regex feature usage references are missing!

- and...

Why Regular Expressions?

- Regexes are everywhere! (we think...)
- Everyone writes regexes! (we think...)
- Regexes are hard to read/write! (again, we think...)
- So, we wanted to write a tool to support regex creation.
- But...

Regex feature usage references are missing!

- and...

We don't know how/when/why developers use regexes!

Research Goals

1. RQ1: In what contexts do professional developers use regular expressions?
2. RQ2: How is the re module used in Python projects?
3. RQ3: Which regular expression language features are most commonly used in Python?
4. RQ4: How behaviorally similar are regexes across projects?

Research Goals

1. RQ1: In what contexts do professional developers use regular expressions?
2. RQ2: How is the re module used in Python projects?
3. RQ3: Which regular expression language features are most commonly used in Python?
4. RQ4: How behaviorally similar are regexes across projects?

What is a Regular Expression?

example

In Python: Utilizations of the re module

function pattern flags

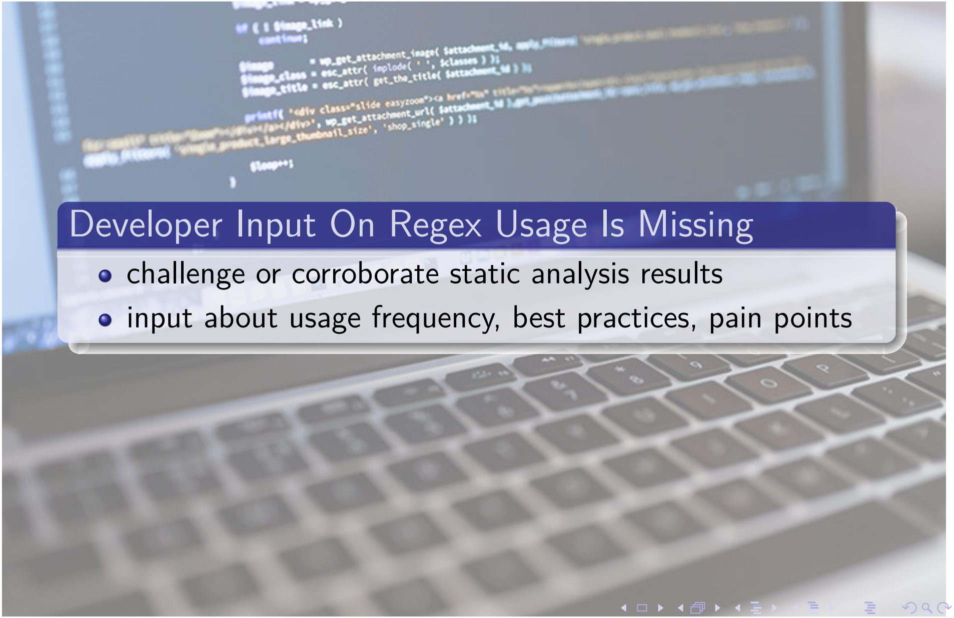
```
r1 = re.compile("(0|-?[1-9][0-9]*)$", re.MULTILINE)
```

function which function of the re module is called?

pattern string used to specify regex behavior

flags modifies the regex engine

How Do Developers Say They Use Regexes?



Developer Input On Regex Usage Is Missing

- challenge or corroborate static analysis results
- input about usage frequency, best practices, pain points

Feature Usage Is Consistent With Analysis

	CG	**STR or END**	**LZY**	**WNW**	**look-arounds**
very frequently	2	1	0	1	0
frequently	4	9	2	3	1
occasionally	9	5	6	6	2
rarely	2	2	2	2	5
very rarely	1	1	4	6	7
never	0	0	4	0	3
avg	5.8	6.1	4	4.8	3.5

Ranked Order: CG (2), STR/END (7,8), LZY (15), WNW (22), look-arounds (21, 25, 27, 28)

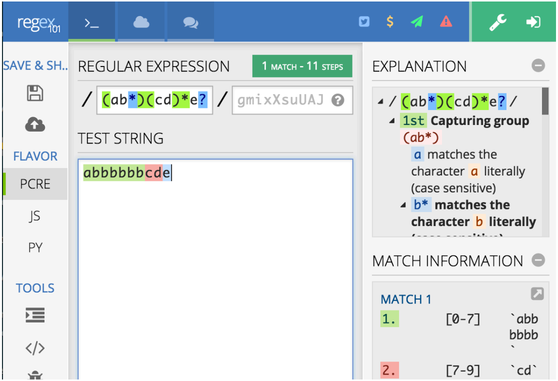
Task Frequencies are Mostly Consistent With Behavioral Categories

	Capturing	**Counting Lines**	**Counting All**	**Finding**	**Filtering**	**Single Char**	**Parse User Input**	**Parse Generated**	**Other**
v. freq	1	1	1	3	0	0	2	2	0
freq.	9	2	3	7	1	0	5	1	1
occ.	3	5	4	3	8	1	5	4	0
rarely	5	3	3	4	2	3	3	3	0
v. rarely	0	3	4	1	5	5	3	5	1
never	0	4	3	0	2	9	0	3	16
avg	3.3	2.0	2.2	3.4	2.1	**0.8**	3	2.1	0.3

Developers said they did not frequently search for a single character.

Regex Testing

	Always	V. Freq	Freq.	Occ.	Rarely	V. Rarely	Never
test code	4	**7**	**5**	1	0	0	1
test regex	3	4	**5**	**5**	1	1	0



1 MATCH - 11 STEPS

/ (ab*)(cd)*e? / gmixXsuUAJ

abbbbbbcde

/ (ab*)(cd)*e? /
1st Capturing group (ab*)
a matches the character a literally (case sensitive)
b* matches the character b literally

MATCH 1
1. [0-7] `abbbbbb`
2. [7-9] `cd`

50% say they use testing tools like www.regex101.com

Usage Frequency - By Technical Environment

Heaviest regex use is in command line tools and text editors.

Language/Environment	**0**	**1-5**	**6-10**	**11-20**	**21-50**	**51+**
General (e.g., Java)	1	6	5	3	1	2
Scripting (e.g., Perl)	5	4	3	3	2	1
Query (e.g., SQL)	15	2	0	0	1	0
Command line (e.g., grep)	2	5	3	2	0	6
Text editor (e.g., IntelliJ)	2	5	0	5	1	5

Ephemeral vs Persistent Users

Task	Persistence Freq.	Ephemeral Freq.	Difference
Counting substrings that match a pattern	3	1.7	1.2
Parsing user input	3.6	2.7	0.9
Capturing parts of strings	3.8	3.1	0.7
Parsing generated text	2.4	1.9	0.5
Locating content within a file or files	3.6	3.2	0.4
Filtering collections (lists, tables, etc.)	2.2	1.9	0.3
Counting lines that match a pattern	1.8	2.1	-0.3

Code	Persistent Freq.	Ephemeral Freq.	Difference
LKA, NLKA, LKB, NLKB	3.2	2.2	1.0
LZY	3	2.8	0.2
STR, END	4.4	4.4	0
CG	4.2	4.2	0
WNW	3.4	3.5	-0.1

Pain Points

hard to compose (11)

...very difficult to write them since I‘ve never read up on them.

...trickiness to getting the expression right

inconsistency across implementations (3)

Differences in implementation across languages

Some regexes work differently (or don‘t work) in some languages.

hard to read (7)

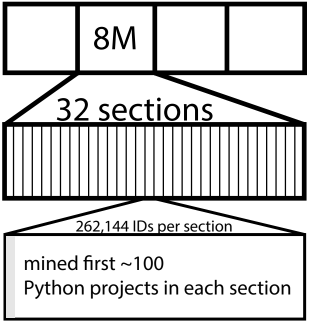
long ones can be hard to read

Readability. Edge cases.

It is terrible to read (especially later after initial development)

Project Selection

Find Python projects using the GitHub API.



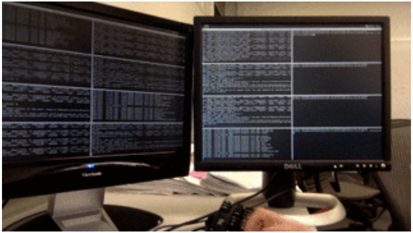
8M

32 sections

262,144 IDs per section

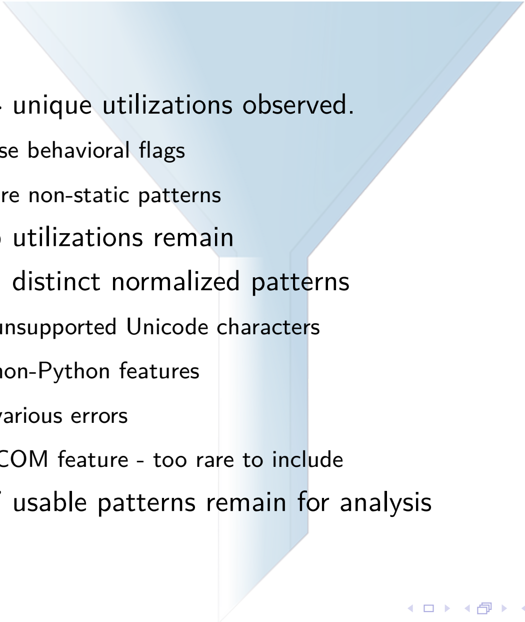
mined first ~100
Python projects in each section

3,898 total Python projects found
(out of 42,000 inspected IDs)



Out of 3,898 pseudo-randomly selected Python projects, 1,645 contained one or more utilization.

Filtering Utilizations And Patterns



53,894 unique utilizations observed.

12.7% use behavioral flags

6.5% were non-static patterns

43,525 utilizations remain

13,711 distinct normalized patterns

73 had unsupported Unicode characters

17 had non-Python features

22 had various errors

2 had ECOM feature - too rare to include

13,597 usable patterns remain for analysis

Feature Statistics

Rank	Code	Example	% Projects	NProjects	NFiles	NPatterns	MaxTokens
1	ADD	z+	73.2	1,204	9,165	6,003	30
2	CG	(caught)	72.6	1,194	9,559	7,130	17
3	KLE	.*	66.8	1,099	8,163	6,017	50
4	CCC	[aeiou]	62.4	1,026	7,648	4,468	42
5	ANY	.	61.1	1,005	6,277	4,657	60
6	RNG	[a-z]	51.6	848	5,092	2,631	50
7	STR	^	51.4	846	5,458	3,563	12
8	END	$	50.3	827	5,393	3,169	12
9	NCCC	[^qwxf]	47.2	776	3,947	1,935	15
10	WSP	\s	46.3	762	4,704	2,846	32
11	OR	a|b	43	708	3,926	2,102	15
12	DEC	\d	42.1	692	4,198	2,297	24
13	WRD	\w	39.5	650	2,952	1,430	13
14	QST	z?	39.2	645	3,707	1,871	35
15	LZY	z+?	36.8	605	2,221	1,300	12
16	NCG	a(?:b)c	24.6	404	1,709	791	28
17	PNG	(?P<name>x)	21.5	354	1,475	915	16
18	SNG	z{8}	20.7	340	1,267	581	17
19	NWSP	\S	16.4	270	776	484	10
20	DBB	z{3,8}	14.5	238	647	367	11
21	NLKA	a(?!yz)	11.1	183	489	131	3
22	WNW	\b	10.1	166	438	248	36
23	NWRD	\W	10	165	305	94	6
24	LWB	z{15,}	9.6	158	281	91	3
25	LKA	a(?=bc)	9.6	158	358	112	4
26	OPT	(?i)CasE	9.4	154	377	231	2
27	NLKB	(?<!x)yz	8.3	137	296	94	4
28	LKB	(?<=a)bc	7.3	120	255	80	4
29	ENDZ	\Z	5.5	90	149	89	1
30	BKR	\1	5.1	84	129	60	4
31	NDEC	\D	3.5	58	92	36	6
32	BKRN	(P?=name)	1.7	28	44	17	2
33	VWSP	\v	0.9	15	16	13	2
34	NWNW	\B	0.7	11	11	4	2

Feature Statistics - Top 8

Rank	Code	Example	% Projects	NProjects	NFiles	NPatterns	MaxTokens
1	ADD	z+	73.2	1,204	9,165	6,003	30
2	CG	(caught)	72.6	1,194	9,559	7,130	17
3	KLE	.*	66.8	1,099	8,163	6,017	50
4	CCC	[aeiou]	62.4	1,026	7,648	4,468	42
5	ANY	.	61.1	1,005	6,277	4,657	60
6	RNG	[a-z]	51.6	848	5,092	2,631	50
7	STR	^	51.4	846	5,458	3,563	12
8	END	$	50.3	827	5,393	3,169	12

What Features Are Missing In Other Languages?

Rank	Code	Example	Python	Perl	.Net	Ruby	Java	RE2	JavaScript	POSIX ERE
1	ADD	z+	●	●	●	●	●	●	●	●
2	CG	(caught)	●	●	●	●	●	●	●	●
3	KLE	.*	●	●	●	●	●	●	●	●
4	CCC	[aeiou]	●	●	●	●	●	●	●	●
5	ANY	.	●	●	●	●	●	●	●	●
6	RNG	[a-z]	●	●	●	●	●	●	●	●
7	STR	ˆ	●	●	●	●	●	●	●	●
8	END	$	●	●	●	●	●	●	●	●
9	NCCC	[ˆqwxf]	●	●	●	●	●	●	●	●
10	WSP	\s	●	●	●	●	●	●	●	🔴
11	OR	a|b	●	●	●	●	●	●	●	●
12	DEC	\d	●	●	●	●	●	●	●	🔴
13	WRD	\w	●	●	●	●	●	●	●	🔴
14	QST	z?	●	●	●	●	●	●	●	●
15	LZY	z+?	●	●	●	●	●	●	●	🔴
16	NCG	a(?:b)c	●	●	●	●	●	●	●	🔴
17	PNG	(?P<name>x)	●	●	○	○	○	●	○	○

Rank	Code	Example	Python	Perl	.Net	Ruby	Java	RE2	JavaScript	POSIX ERE
18	SNG	z{8}	●	●	●	●	●	●	●	●
19	NWSP	\S	●	●	●	●	●	●	●	🔴
20	DBB	z{3,8}	●	●	●	●	●	●	●	●
21	NLKA	a(?!yz)	●	●	●	●	●	🔴	●	🔴
22	WNW	\b	●	●	●	●	●	●	●	🔴
23	NWRD	\W	●	●	●	●	●	●	●	🔴
24	LWB	z{15,}	●	●	●	●	●	●	●	●
25	LKA	a(?=bc)	●	●	●	●	●	🔴	●	🔴
26	OPT	(?i)CasE	●	●	●	●	●	●	🔴	🔴
27	NLKB	(?<!x)yz	●	●	●	●	●	🔴	🔴	🔴
28	LKB	(?<=a)bc	●	●	●	●	●	🔴	🔴	🔴
29	ENDZ	\Z	●	○	○	○	○	○	○	○
30	BKR	\1	●	●	●	●	●	🔴	●	●
31	NDEC	\D	●	●	●	●	●	●	●	🔴
32	BKRN	(P?=name)	●	●	○	○	○	○	○	○
33	VWSP	\v	●	●	●	○	●	●	●	●
34	NWNW	\B	●	●	●	●	●	●	●	○

Ranked features: Languages - Notable Missing Features

Rank	Code	Example	Python	Perl	.Net	Ruby	Java	RE2	JavaScript	POSIX ERE
21	NLKA	`a(?!yz)`	●	●	●	●	●	🔴	●	🔴
22	WNW	`\b`	●	●	●	●	●	●	●	🔴
23	NWRD	`\W`	●	●	●	●	●	●	●	🔴
24	LWB	`z{15,}`	●	●	●	●	●	●	●	●
25	LKA	`a(?=bc)`	●	●	●	●	●	🔴	●	🔴
26	OPT	`(?i)CasE`	●	●	●	●	●	●	🔴	🔴
27	NLKB	`(?<!x)yz`	●	●	●	●	●	🔴	🔴	🔴
28	LKB	`(?<=a)bc`	●	●	●	●	●	🔴	🔴	🔴
29	ENDZ	`\Z`	●	○	○	○	○	○	○	○
30	BKR	`\1`	●	●	●	●	●	🔴	●	●
31	NDEC	`\D`	●	●	●	●	●	●	●	🔴

What Features Are Not Supported By Analysis Tools?

Rank	Code	Example	Brics	Hampi	Rex	Automata.Z3
1	ADD	z+	●	●	●	●
2	CG	(caught)	●	●	●	●
3	KLE	.*	●	●	●	●
4	CCC	[aeiou]	●	●	●	●
5	ANY	.	●	●	●	🔴
6	RNG	[a-z]	●	●	●	●
7	STR	^	🔴	●	●	●
8	END	$	🔴	●	●	🔴
9	NCCC	[^qwxf]	●	●	●	🔴
10	WSP	\s	🔴	●	●	●
11	OR	a|b	●	●	●	●
12	DEC	\d	🔴	●	●	●
13	WRD	\w	🔴	●	●	●
14	QST	z?	●	●	●	●
15	LZY	z+?	🔴	●	🔴	🔴
16	NCG	a(?:b)c	🔴	●	🔴	🔴
17	PNG	(?P<name>x)	○	●	○	○

Rank	Code	Example	Brics	Hampi	Rex	Automata.Z3
18	SNG	z{8}	●	●	●	●
19	NWSP	\S	🔴	●	●	🔴
20	DBB	z{3,8}	●	●	●	●
21	NLKA	a(?!yz)	🔴	🔴	🔴	🔴
22	WNW	\b	🔴	🔴	🔴	🔴
23	NWRD	\W	🔴	●	●	🔴
24	LWB	z{15,}	●	●	●	🔴
25	LKA	a(?=bc)	🔴	🔴	🔴	🔴
26	OPT	(?i)CasE	🔴	●	🔴	🔴
27	NLKB	(?<!x)yz	🔴	🔴	🔴	🔴
28	LKB	(?<=a)bc	🔴	🔴	🔴	🔴
29	ENDZ	\Z	○	○	○	●
30	BKR	\1	🔴	🔴	🔴	🔴
31	NDEC	\D	🔴	●	●	🔴
32	BKRN	\g<name>	○	●	○	○
33	VWSP	\v	○	○	●	○
34	NWNW	\B	○	○	○	○

Comparison Of Language Feature Sets

Code	Example	Python	Perl	.Net	Ruby	Java	RE2	JavaScript	POSIX ERE
ADD	z+	●	●	●	●	●	●	●	●
CG	(caught)	●	●	●	●	●	●	●	●
KLE	.*	●	●	●	●	●	●	●	●
CCC	[aeiou]	●	●	●	●	●	●	●	●
ANY	.	●	●	●	●	●	●	●	●
RNG	[a-z]	●	●	●	●	●	●	●	●
STR	^	●	●	●	●	●	●	●	●
END	$	●	●	●	●	●	●	●	●
NCCC	[^qwxf]	●	●	●	●	●	●	●	●
WSP	\s	●	●	●	●	●	●	●	○
OR	a|b	●	●	●	●	●	●	●	●
DEC	\d	●	●	●	●	●	●	●	○
WRD	\w	●	●	●	●	●	●	●	○
QST	z?	●	●	●	●	●	●	●	●
LZY	z+?	●	●	●	●	●	●	●	○
NCG	a(?:b)c	●	●	●	●	●	●	●	○
PNG	(?P<name>x)	●	●	○	○	○	●	○	○
SNG	z{8}	●	●	●	●	●	●	●	●
NWSP	\S	●	●	●	●	●	●	●	○
DBB	z{3,8}	●	●	●	●	●	●	●	●
NLKA	a(?!yz)	●	●	●	●	●	○	●	○
WNW	\b	●	●	●	●	●	●	●	○
NWRD	\W	●	●	●	●	●	●	●	○
LWB	z{15,}	●	●	●	●	●	●	●	●
LKA	a(?=bc)	●	●	●	●	●	○	●	○
OPT	(?i)CasE	●	●	●	●	●	●	○	○
NLKB	(?<!x)yz	●	●	●	●	●	○	○	○
LKB	(?<=a)bc	●	●	●	●	●	○	○	○
ENDZ	\Z	●	○	○	○	○	○	○	○
BKR	\1	●	●	●	●	●	○	●	●
NDEC	\D	●	●	●	●	●	●	●	○
BKRN	(P?=name)	●	●	○	○	○	○	○	○
VWSP	\v	●	●	●	○	●	●	●	●
NWNW	\B	●	●	●	●	●	●	●	○

Code	Example	Python	Perl	.Net	Ruby	Java	RE2	JavaScript	POSIX ERE
RCUN	(?n)	○	●	○	○	○	○	○	○
RCUZ	(?R)	○	●	○	○	○	○	○	○
GPLS	\g{+1}	○	●	○	○	○	○	○	○
GBRK	\g{name}	○	●	○	○	○	○	○	○
GSUB	\g<name>	●	●	○	●	○	○	○	○
KBRK	\k<name>	○	●	●	●	●	○	○	○
IFC	(?(cond)X)	○	●	●	○	○	○	○	○
IFEC	(?(cnd)X|else)	○	●	●	○	○	○	○	○
ECOD	(?{code})	○	●	○	○	○	○	○	○
ECOM	(?#comment)	●	●	●	●	○	○	○	○
PRV	\G	○	●	●	●	●	○	○	○
LHX	\uFFFF	○	●	●	●	●	○	●	○
POSS	a?+	○	●	○	●	●	○	○	○
NNCG	(?<name>X)	○	●	●	●	●	○	○	○
MOD	(?i)z(?-i)z	○	●	●	●	●	●	○	○
ATOM	(?>X)	○	●	●	●	●	○	○	○
CCCI	[a-z&&[^f]]	○	○	○	●	●	○	○	○
STRA	\A	●	●	●	●	●	●	○	○
LNLZ	\Z	○	●	●	●	●	●	○	○
FINL	\z	○	●	●	●	●	●	○	○
QUOT	\Q...\E	○	●	○	○	●	●	○	○
JAVM	\p{javaMirrored}	○	○	○	○	●	○	○	○
UNI	\pL	○	●	○	○	●	●	○	○
NUNI	\PS	○	●	○	○	●	●	○	○
OPTG	(?flags:re)	○	●	●	●	●	●	○	○
EREQ	[[=o=]]	○	○	○	○	○	○	○	●
PXCC	[:alpha:]	○	●	○	●	○	●	●	●
TRIV	[^]	○	○	○	○	○	○	●	○
CCSB	[a-f-[c]]	○	○	●	○	○	○	○	○
VLKB	(?<=ab.+)	○	○	●	○	○	○	○	○
BAL	(?<close-open>)	○	○	●	○	○	○	○	○
NCND	(?(<n>)X|else)	○	●	●	●	○	○	○	○
BRES	(?|(A)|(B))	○	○	○	○	○	○	○	○
QNG	(?'name're)	○	○	●	●	○	○	○	○

What Are Regexes Used For?

Non-Anecdotal Knowledge About Usage Is Missing

- task categories
- behavioral categories

How to Categorize Regex Usages

1. ~~thorough inspection of 53K utilizations~~
2. ~~unguided manual categorization of 4,694 regexes (in 2 or more projects), without objective basis~~
3. ~~cluster by syntactic similarity like Jaccard or longest substring~~
4. ~~formal analytical subsumption, using hampi (94%?) cannot get it to work~~
5. ~~formal analytical subsumption, using brics (30% or less)~~
6. Chosen technique: cluster by behavioral similarity using Rex (61%)

Measuring Behavioral Similarity

```
Pattern A matches 100/100 of A's strings
Pattern B matches  90/100 of A's strings
Pattern A matches  50/100 of B's strings
Pattern B matches 100/100 of B's strings
```

	A	B
A	1.0	0.9
B	0.5	1.0

	A	B	C	D
A	1.0	0.0	0.9	0.0
B	0.2	1.0	0.8	0.7
C	0.6	0.8	1.0	0.2
D	0.0	0.6	0.1	1.0

→

	A	B	C	D
A	1.0			
B	0.1	1.0		
C	0.75	0.8	1.0	
D	0.0	0.65	0.15	1.0

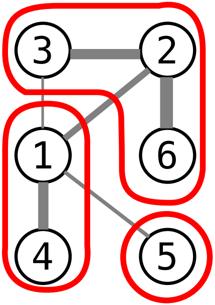
Rex (?) generates
400 strings for each regex.
Convert scores to half-matrix
to make *.abc file for mcl.

MCL example

```
>mcl test.abc --abc -o out.txt
```

test.abc

```
1   2   0.5
1   3   0.2
1   4   0.8
1   5   0.3
2   3   0.8
2   6   0.99
```



out.txt

```
2   3   6
1   4
5
```

mcl works by alternating between expansion and inflation (?)

Clustering Results

Example Cluster

Index	Pattern	NProjects	Index	Pattern	NProjects
1	`\s*([^: ]*)\s*:(.*)`	9	7	`[:]`	6
2	`:+`	8	8	`([^:]+):(.*)`	6
3	`(:)`	8	9	`\s*:\s*`	4
4	`(:+)`	8	10	`\:`	2
5	`(:)(:*)`	8	11	`^([^:]*):[^:]*$`	2
6	`^([^:]*): *(.*)`	8	12	`^[^:]*:([^:]*)$`	2

From 2,871 distinct regexes

186 clusters where size $\geq$ 2

2,042 unclustered regexes

Six Categories Of Clusters

Category	Clusters	Patterns	Projects	% Projects
Multi Matches	21	237	295	40%
Specific Char	17	103	184	25%
Anchored Patterns	20	85	141	19%
Two or More Chars	16	40	120	16%
Content of Parens	10	46	111	15%
Code Search	15	27	92	13%

Multi Matches `(\s)`, `,|;`

Specific Char `:+`, `}`, `%`

Anchored Patterns `^[-_A-Za-z0-9]+$`

Two Or More Chars `@[a-z]+`

Content of Parens `<(.+)>`, `<[^>]*?>`

Code Search `.*rlen=([0-9]+)`

Questions?