

# Usage and Refactoring Studies Of Python Regular Expressions

Carl Chapman

Iowa State University

*carlallenchapman@gmail.com*

13 April, 2016

# Overview

## 1 Regex Usage Studies

- Feature Analysis
- Behavioral Clustering
- Developer Survey

## 2 Regex Refactoring Studies

- Equivalence Model
- Community Support
- Understandability

## 3 Conclusion

- Refactoring Recommendations
- Future Work
- References

# Basic Regex Usage References Are Missing

## feature usage statistics

What features are more frequently used, and thus more important when...

building an analysis tool?

developing test regexes?

creating a toy regex language for a formal experiment?

## feature set summaries for variants and tools

What features does each language and tool support...

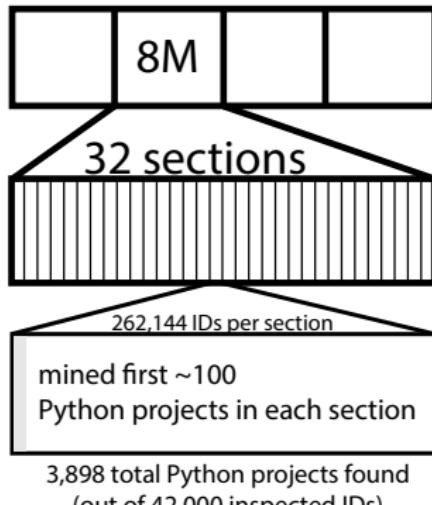
so I know how to port my regex-containing code?

so I can choose the best fitting regex variant for my needs?

so I can choose the best analysis tool?



# Project Selection



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

# Utilizations of the re module

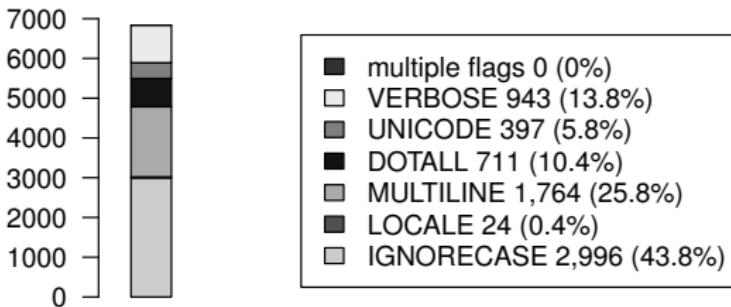
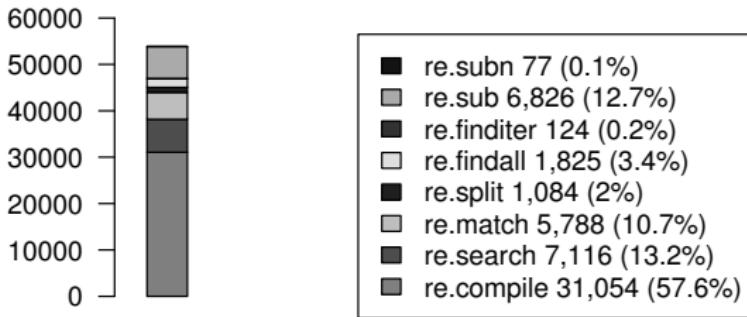
```
function          pattern          flags
r1 = re.compile("(0|-[1-9][0-9]*)$", re.MULTILINE)
```

function which function of the re module is called?

pattern string used to specify regex behavior

flags modifies the regex engine

# re module insights



# Filtering Utilizations

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

# PCRE Parsing Patterns



All Python features are recognizable by PCRE

# 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

Rank	Code	Example	% Projects	NProjects	NFiles	NPatterns	MaxTokens
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	(?<1>CasE	9.4	154	377	231	2
27	NLKB	(?<1>x)yz	8.3	137	296	94	4
28	LKB	(?<1=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

# Ranked features: 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	^	●	●	●	●	●	●	●	●	●
8	END	\$	●	●	●	●	●	●	●	●	●
9	NCCC	[~qwxzf]	●	●	●	●	●	●	●	●	●
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: 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	[^qwxrf]	●	●	●	●
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	●	●	●	●

# What Are Regexes Used For?



# How to Categorize Regex Usages

- ➊ thorough inspection of 55K utilizations
- ➋ unguided manual categorization of 4,694 regexes (in 2 or more projects), without objective basis
- ➌ cluster by syntactic similarity like Jaccard or longest substring
- ➍ formal analytical subsumption, using hampi (94%?) cannot get it to work
- ➎ formal analytical subsumption, using brics (30% or less)
- ➏ 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



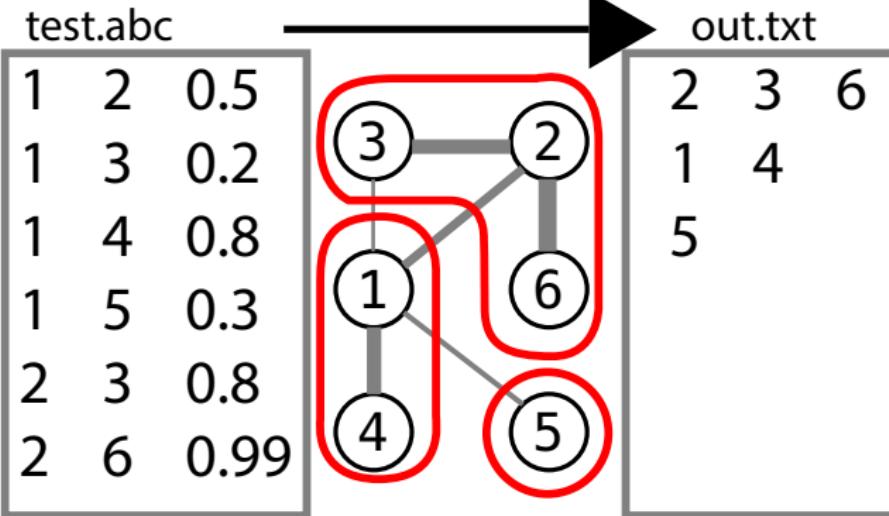
	A	B	C	D
A	1.0			
B	0.1	1.0		
C	0.75	0.8	1.0	
D	0.0	0.65	0.15	1.0

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
```



# Example Cluster

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

# Six Categories Of Clusters

**Table:** Cluster categories and sizes, ordered by number of projects containing at least one pattern in the category.

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%

# Survey Goals

- confirm or deny parts 1 and 2:...
- investigate some topics: usage freq., pain points, testing, html parsing, ephem vs pers. comparision

# Confirming PT1

- idk

# Confirming PT2

- idk

# Regex Testing

- image of regex101

# Parsing HTML

- idk, maybe some regexes that parse html

# Usage Frequency

- idk

# Ephemeral vs Persistent Users

# 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

## hard to read (7)

long ones can be hard to read

Readability. Edge cases.

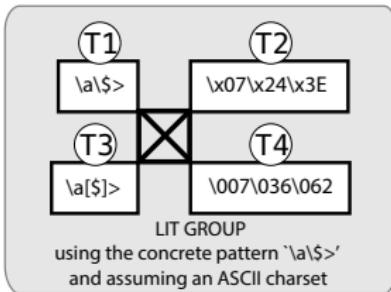
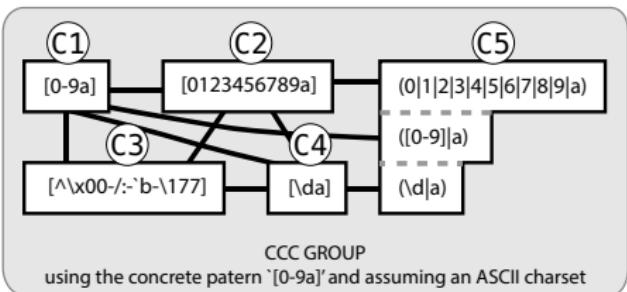
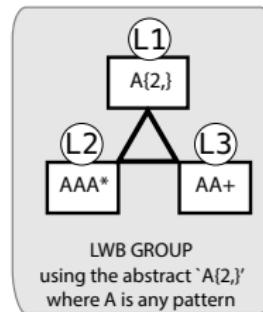
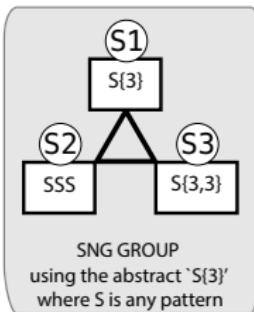
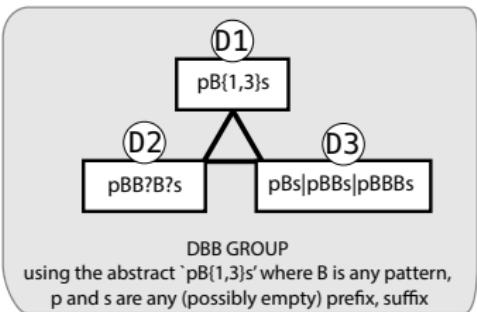
It is terrible to read (especially later after initial development)

## inconsistency across implementations (3)

Differences in implementation across languages

Some regexes work differently (or don't work) in some languages.

# Regex Equivalence Classes



# Example Equivalences

LIT : `x ≡ y`

DBB : another one

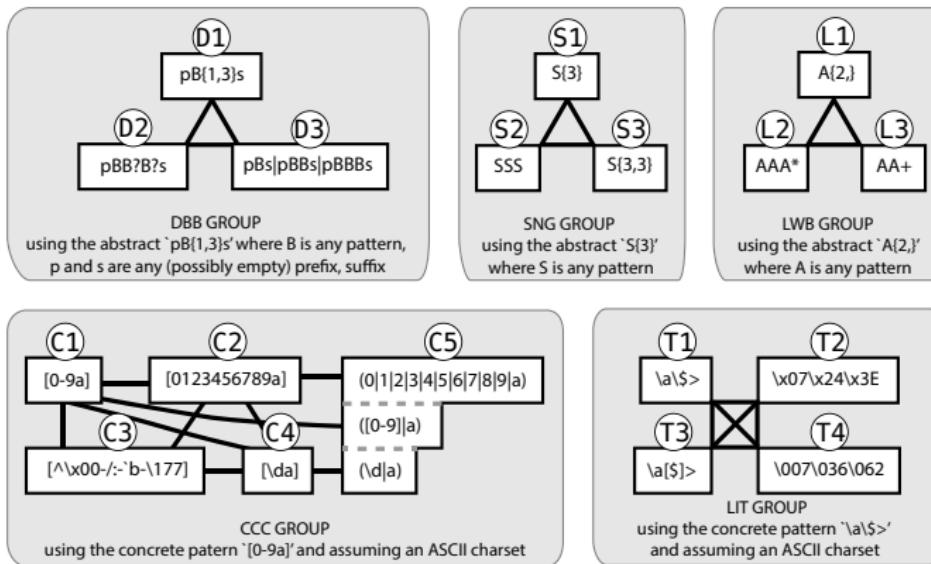
CCC : `a = b`

LWB : another one

SNG : `a = b`

# Table

# Imagetest



**Figure:** Equivalence classes with various representations of semantically equivalent representations within each class. DBB = Double-Bounded, SNG = Single Bounded, LWB = Lower Bounded, CCC = Custom Character Class and LIT = Literal

# Citation

An example of the \cite command to cite within the presentation:

This statement requires citation Smith (2012).

# Verbatim

## Example (Theorem Slide Code)

```
\begin{frame}
\frametitle{Theorem}
\begin{theorem}[Mass--energy equivalence]
$E = mc^2$
\end{theorem}
\end{frame}
```

# regex formatting test

ab\*c

# References

John Smith (2012) Title of the publication Journal Name 12(3), 45 – 678.

# Questions?