Introduction
OOOO

RQ1: Survey
OOOOOOO

RQ3: Repository
OOOOOOOOOO

RQ4: Similarity
OOOOOOOO

Conclusion
OOO

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?

Why regular expressions?

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

Why regular expressions?

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

Introduction
○○○○

RQ1: Survey
○○○○○○○

RQ3: Repository
○○○○○○○○○○○

RQ4: Similarity
○○○○○○○○

Conclusion
○○○

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?

- We wanted to write a tool to support regex creation.

Why regular expressions?

- We wanted to write a tool to support regex creation.
- But...

Why regular expressions?

- We wanted to write a tool to support regex creation.
- But...

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

Introduction
○●○○

RQ1: Survey
○○○○○○○

RQ3: Repository
○○○○○○○○○○

RQ4: Similarity
○○○○○○○○

Conclusion
○○○

Why regular expressions?

- We wanted to write a tool to support regex creation.
- But...

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

- and...

Why regular expressions?

- We wanted to write a tool to support regex creation.
- But...

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

- and...

Regex feature usage references are missing!

Introduction
○○●○

RQ1: Survey
○○○○○○○

RQ3: Repository
○○○○○○○○○○

RQ4: Similarity
○○○○○○○○

Conclusion
○○○

Research goals

Explore regex

1. Context (developer survey)
2. Features (repository analysis)
3. Use cases (similarity analysis)

Regular expressions: The basics

- $(ab*c|yz*)\$$
  - ✓ abbbbbbc
  - ✓ y
  - ✓ abcy

Regular expressions: The basics

- $(ab*c|yz*)\$$

  ✓ abbbbbbc

  ✓ y

  ✓ abcy

  ✗ abcccc

  ✗ yxw

Regular expressions: The basics

- $\mathtt{(ab*c|yz*)\$}$
  - ✓ abbbbbbc
  - ✓ y
  - ✓ abcy

  - ✗ abcccc
  - ✗ yxw

- $\mathtt{(ab*c|yz*)}$

Introduction
○○○●

RQ1: Survey
○○○○○○○

RQ3: Repository
○○○○○○○○○○

RQ4: Similarity
○○○○○○○○

Conclusion
○○○

Regular expressions: The basics

- $(ab*c|yz*)\$$

  ✓ abbbbbbc

  ✓ y

  ✓ abcy

  ✗ abccc

  ✗ yxw

- $(ab*c|yz*)$

  ✓ abbbbbbc

  ✓ y

  ✓ abcy

  ✓ abcccc

  ✓ yxw

Introduction
oooo

RQ1: Survey
●oooooo

RQ3: Repository
oooooooooo

RQ4: Similarity
ooooooo

Conclusion
ooo

Part 1

RQ1

In what contexts do professional developers use regular expressions?

Survey context

- 18 professional developers
- 9 years average development experience
- Small mobile payment management company
- 30 questions in a Google form

Introduction
oooo

RQ1: Survey
ooo●oooo

RQ3: Repository
ooooooooooo

RQ4: Similarity
oooooooo

Conclusion
ooo

How often and where do developers use regexes?

- 50% – at least once per week

How often and where do developers use regexes?

- 50% – at least once per week
- Most often within command line and text editor tools, then general purpose and scripting programming languages
- Database queries using regexes were rare

Introduction
oooo

RQ1: Survey
oooo●ooo

RQ3: Repository
ooooooooooo

RQ4: Similarity
oooooooo

Conclusion
ooo

Common regex activities

How often do you use regexes for…

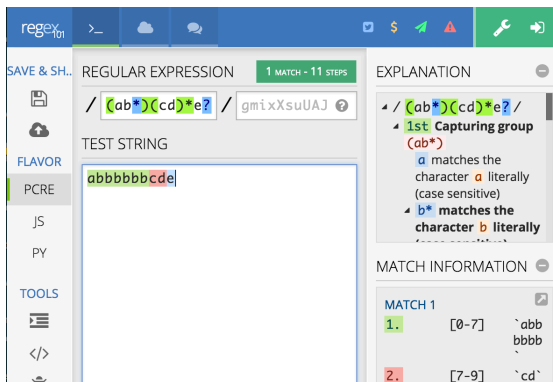| Activity	Frequency
Locating content within a file or files	4.4
Capturing parts of strings	4.3
Parsing user input	4.0

Key: 6 = very frequently, 5 = frequently, 4 = occasionally, 3 = rarely, 2 = very rarely, 1 = never

Testing regular expressions

Developers test regular expressions <u>less often</u> than other code.

Introduction
oooo

RQ1: Survey
ooooo●oo

RQ3: Repository
oooooooooo

RQ4: Similarity
oooooooo

Conclusion
ooo

Testing regular expressions

Developers test regular expressions <u>less often</u> than other code.

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

Introduction
oooo

RQ1: Survey
ooooooo●o

RQ3: Repository
oooooooooo

RQ4: Similarity
ooooooooo

Conclusion
ooo

Pain points

hard to compose (11 = 61%)

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

...trickiness to getting the expression right

Introduction
oooo

RQ1: Survey
ooooooo●o

RQ3: Repository
ooooooooooo

RQ4: Similarity
oooooooo

Conclusion
ooo

Pain points

hard to compose (11 = 61%)

...very difficult to write them since I've never read up on them.
...trickiness to getting the expression right

hard to read (7 = 39%)

long ones can be hard to read
Readability. Edge cases.
It is terrible to read (especially later after initial development)

Introduction
oooo

RQ1: Survey
ooooooeo

RQ3: Repository
ooooooooooo

RQ4: Similarity
oooooooo

Conclusion
ooo

Pain points

hard to compose (11 = 61%)

...very difficult to write them since I've never read up on them.
...trickiness to getting the expression right

hard to read (7 = 39%)

long ones can be hard to read
Readability. Edge cases.
It is terrible to read (especially later after initial development)

inconsistency across implementations (3 = 17%)

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

Notable Observations

- Regexes are composed fairly frequently by developers
- Testing regexes is less common than testing other code
- Developers find regexes hard to read and write

Introduction
0000

RQ1: Survey
0000000●

RQ3: Repository
0000000000

RQ4: Similarity
00000000

Conclusion
000

Notable Observations

- Regexes are composed fairly frequently by developers
- Testing regexes is less common than testing other code
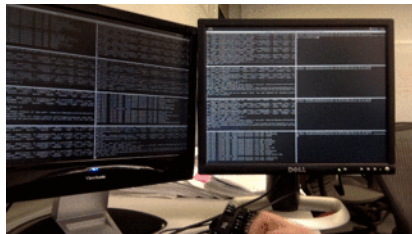- Developers find regexes hard to read and write

but....

Introduction
0000

RQ1: Survey
0000000●

RQ3: Repository
0000000000

RQ4: Similarity
00000000

Conclusion
000

Notable Observations

- Regexes are composed fairly frequently by developers
- Testing regexes is less common than testing other code
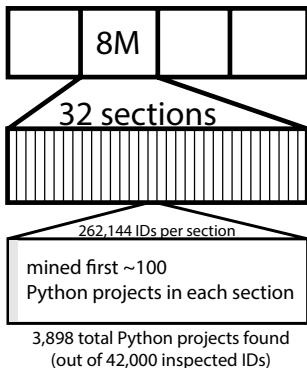- Developers find regexes hard to read and write

but....

How do developers <u>really</u> use regexes?

Introduction
oooo

RQ1: Survey
ooooooo

RQ3: Repository
●ooooooooo

RQ4: Similarity
oooooooo

Conclusion
ooo

Part 2

RQ3

Which regular expression language features are most commonly used in Python?

Project selection with 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
(42%) contained one or more
regex utilization.

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

Filtering utilizations and patterns

53,894 unique utilizations observed.

12.7% use behavioral flags

6.5% were non-static patterns

43,525 utilizations remain

Introduction
○○○○

RQ1: Survey
○○○○○○○

RQ3: Repository
○○○●○○○○○○

RQ4: Similarity
○○○○○○○○

Conclusion
○○○

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

114 had various errors

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

114 had various errors

13,597 usable patterns from 1,645 projects remain for analysis

PCRE parsing patterns

`^m+(f(z)*)+`	0	1	2	2	1	0

`(ab*c|yz*)$`	1	2	0	1	0	1

OR   KLE   ADD   CG   STR   END

Feature statistics - Top 8

Rank	Code	Example	% Projects	% Patterns
1	ADD	`z+`	73.2	44.1
2	CG	`(caught)`	72.6	52.4
3	KLE	`.*`	66.8	44.3
4	CCC	`[aeiou]`	62.4	32.9
5	ANY	`.`	61.1	34.3
6	RNG	`[a-z]`	51.6	19.3
7	STR	`^`	51.4	26.2
8	END	`$`	50.3	23.3

Regex research tools

- Remember that we wanted to write a tool to support regex creation?

Regex research tools

- Remember that we wanted to write a tool to support regex creation?
- But regex feature usage references were missing (not anymore!).

Regex research tools

- Remember that we wanted to write a tool to support regex creation?
- But regex feature usage references were missing (not anymore!).
- So,

Regex research tools

- Remember that we wanted to write a tool to support regex creation?
- But regex feature usage references were missing (not anymore!).
- So,

How well do past and present regex research tools meet the needs of developers? (Hampi, Rex, RE2, brics)

Which features are supported by analysis tools?

Rank	Code	Example	Brics	Hampi	Rex	RE2
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	RE2
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`	●	●	●	●

Introduction
oooo

RQ1: Survey
ooooooo

RQ3: Repository
ooooooooo●o

RQ4: Similarity
ooooooooo

Conclusion
ooo

Survey vs. Repository

How often do you use....

Group	Code	Survey	Repo Rank
endpoint anchors	(STR, END)	4.4	7, 8
capture groups	(CG)	4.2	2
word boundaries	(WNW)	3.5	22
lazy repetition	(LZY)	2.9	15

Key: 6 = very frequently, 5 = frequently, 4 = occasionally,
3 = rarely, 2 = very rarely, 1 = never

Introduction
oooo

RQ1: Survey
ooooooo

RQ3: Repository
oooooooooo●

RQ4: Similarity
ooooooooo

Conclusion
ooo

Notable Observations

- Regexes are common, but not everywhere (42% of projects, 32 utilizations per project)
- Current regex research tools cover the most common features

Notable Observations

- Regexes are common, but not everywhere (42% of projects, 32 utilizations per project)
- Current regex research tools cover the most common features

but....

Notable Observations

- Regexes are common, but not everywhere (42% of projects, 32 utilizations per project)
- Current regex research tools cover the most common features

but....

What are the regexes doing?

Part 3

RQ4

How behaviorally similar are regexes across projects?

How to find common behaviors?

1. ~~thorough inspection of 53K utilizations~~

How to find common behaviors?

1. ~~thorough inspection of 53K utilizations~~
2. ~~cluster by syntactic similarity like Jaccard or longest substring~~

How to find common behaviors?

1. ~~thorough inspection of 53K utilizations~~
2. ~~cluster by syntactic similarity like Jaccard or longest substring~~
3. ~~formal analytical subsumption, no sufficient tools at the moment~~

Introduction
oooo

RQ1: Survey
ooooooo

RQ3: Repository
oooooooooo

RQ4: Similarity
o●oooooo

Conclusion
ooo

How to find common behaviors?

1. ~~thorough inspection of 53K utilizations~~
2. ~~cluster by syntactic similarity like Jaccard or longest substring~~
3. ~~formal analytical subsumption, no sufficient tools at the moment~~
4. Chosen technique: cluster by behavioral similarity using Rex

Example

A `(ab*c|yz*)$`

- abbbbbbbc
- y
- abcy
- pac
- abcyzzz

B `(ab*c|yz*)`

- y
- abc
- abcy
- abcccc
- yxw

Example

A | `(ab*c|yz*)$`

- abbbbbbbc
- y
- abcy
- pac
- abcyzzz

A matches 3/5 =
60% of B's strings

B | `(ab*c|yz*)`

- y
- abc
- abcy
- abcccc
- yxw

Example

A `(ab*c|yz*)$`

- abbbbbbbc
- y
- abcy
- pac
- abcyzzz

A matches $3/5 =$ 60% of B's strings

B `(ab*c|yz*)`

- y
- abc
- abcy
- abcccc
- yxw

B matches $5/5 =$ 100% of A's strings

Example

A `(ab*c|yz*)$`
- abbbbbbbc
- y
- abcy
- pac
- abcyzzz

B `(ab*c|yz*)`
- y
- abc
- abcy
- abcccc
- yxw

A matches $3/5 =$ 60% of B's strings

B matches $5/5 =$ 100% of A's strings

A and B are 80% similar

Similarity Matrix → MCL

Rex generates
400 strings for each regex.
Average scores to
half-matrix for MCL

Scope

- 3,582 (26%) of patterns appeared in multiple projects
- 711 unsupported by Rex

Scope

- 3,582 (26%) of patterns appeared in multiple projects
- 711 unsupported by Rex

- 2,871 patterns analyzed from 722 (44%) of the projects

Clustering Results

From 2,871 distinct regexes:
$\rightarrow$ 186 clusters with size $\geq$ 2
$\rightarrow$ 2,042 unclustered regexes

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:
$\rightarrow$ 186 clusters with size $\geq$ 2
$\rightarrow$ 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]+)`

Notable Observations

- Finding a specific character is quite common, 25% of projects (in contrast with survey)

- Regexes are often used to parse source code!

Notable Observations

- Finding a specific character is quite common, 25% of projects (in contrast with survey)
- Regexes are often used to parse source code!

but....

Notable Observations

- Finding a specific character is quite common, 25% of projects (in contrast with survey)
- Regexes are often used to parse source code!

but....

- Similarity metric is approximate
- Metric is perhaps <u>too</u> sensitive to differences in literals

Introduction
oooo

RQ1: Survey
ooooooo

RQ3: Repository
ooooooooooo

RQ4: Similarity
ooooooooo

Conclusion
●oo

Opportunities for future work!

Opportunities for future work!

Better Similarity Metrics

Our similarity metrics are empirical, can we do it analytically?

Opportunities for future work!

Better Similarity Metrics

Our similarity metrics are empirical, can we do it analytically?

Migration Support for Developers

Supported regex features are different among languages.

Opportunities for future work!

Better Similarity Metrics

Our similarity metrics are empirical, can we do it analytically?

Migration Support for Developers

Supported regex features are different among languages.

Identifying Best Practices?

Could impact regex education and improve comprehension.

Recap

- Regexes are (sort of) everywhere! (42% of Python projects studied, but more frequently in text editors and IDEs)

Recap

- Regexes are (sort of) everywhere! (42% of Python projects studied, but more frequently in text editors and IDEs)
- Everyone writes regexes! (50% of deves write them weekly)

Recap

- Regexes are (sort of) everywhere! (42% of Python projects studied, but more frequently in text editors and IDEs)
- Everyone writes regexes! (50% of deves write them weekly)
- Regexes are hard to read/write! (this is a pain point)

Recap

- Regexes are (sort of) everywhere! (42% of Python projects studied, but more frequently in text editors and IDEs)
- Everyone writes regexes! (50% of deves write them weekly)
- Regexes are hard to read/write! (this is a pain point)

also...

- Current tools support most of the most common features
- Regexes are often used for parsing/validating source code
- Many opportunities for future work!

Questions?
Katie Stolee – ktstolee@ncsu.edu

(psst! Graduate students! I'm hiring!)