

Regular expressions for engineers

A 1 day **Hands on** training course



Description

Regular expressions are an extremely powerful tool for manipulating text and data. They are now standard features in a wide range of languages and popular tools, including Python and MySQL. Regular expressions allow you to code complex and subtle text processing that you never imagined could be automated. Once you've mastered regular expressions, they'll become an invaluable part of your toolkit. You will wonder how you ever got by without them.



Key outcomes

By the end of the course delegates will be able to:

- ✓ Use Regular Expressions.
- ✓ Troubleshoot Regular Expressions.
- ✓ Compare RE features among different versions.
- ✓ Explain how the regular expression engine works.
- ✓ Optimize REs.
- ✓ Match what you want, not what you don't want.



Training approach

This structured course uses Instructor Led Training to provide the best possible learning experience. Small class sizes ensure students benefit from our engaging and interactive style of teaching with delegates encouraged to ask questions throughout the course. Quizzes follow each major section allowing checking of learning. Hands on sessions are used throughout to allow delegates to consolidate their new skills.



Details

Who will benefit?

Anyone looking to use regular expressions.

Prerequisites

None.

Duration: 1 day

Overall rating:



Generic training



Generic training compliments product specific courses covering the complete picture of all relevant devices including the protocols "on the wire".

"Friendly environment with expert teaching that teaches the why before the how."
G.C. Fasthosts

Small class sizes



We limit our maximum class size to 8 delegates; often we have less than this. This ensures optimal interactivity between delegates and instructor.

"Excellent course. The small class size was a great benefit..."
M.B. IBM

Hands On training



The majority of our courses use hands on sessions to reinforce the theory.

"Not many courses have practice added to it. Normally just the theoretical stuff is covered."
J.W. Vodafone

Our courseware



We write our own courses; courseware does not just consist of slides and our slides are diagrams not bullet point text.

"Comprehensive materials that made the course easy to follow and will be used as a reference point."
V.B. Rockwell Collins

Customise your course



Please contact us if you would like a course to be customised to meet your specific requirements. Have the course your way.

"I was very impressed by the combination of practical and theory. Very informative. Friendly approachable environment, lots of hands on."
S.R. Qinetiq

Regular expressions for engineers

Course content

Introduction to Regular Expressions

Solving real problems, REs as a language, the filename analogy, language analogy, RE frame of mind, searching text files: egrep, egrep metacharacters, start and end of the line, character classes, matching any character with dot, alternation, ignoring differences in capitalization, word boundaries, optional items, other quantifiers: repetition, parentheses and backreferences, the great escape, expanding the foundation, linguistic diversification, the goal of a RE, more examples, RE nomenclature, improving on the status quo.

Extended introductory examples

A short introduction to Perl, matching text with regular expressions, toward a more real-world example, side effects of a successful match, Intertwined regular expression, intermission, modifying text with regular expressions, example: form letter, example: prettifying a stock price, automated editing, a small mail utility, adding commas to a number with lookahead, text-to-HTML conversion, that doubled-word thing.

Regular expression features and flavours

The regex landscape, origins of REs, care and handling of REs, Integrated handling, procedural and object-oriented handling, search-and-replace example. strings character encodings and modes, strings as REs, character-encoding issues, unicode, regex modes and match modes, common metacharacters and features, character representations, character classes and class-like constructs, anchors and other "zero-width assertions", comments and mode modifiers, grouping capturing conditionals and control.

The mechanics of expression processing

Two kinds of engines, new standards, regex engine types, from the department of redundancy department, testing the engine type, match basics, about the examples, rule 1: the match that begins earliest wins, engine pieces and parts, rule 2: the standard quantifiers are greedy, regex-directed versus text-directed, NFA engine: regex-directed, DFA engine: text-directed, first

thoughts: NFA and DFA in comparison, backtracking, two important points on backtracking, saved states, backtracking and greediness, more about greediness and backtracking, problems of greediness, multi-character "quotes", lazy quantifiers, greediness and laziness, laziness and backtracking, possessive quantifiers and atomic grouping, possessive quantifiers ?, +, *, ++ and {m,n}+, the backtracking of lookahead, is alternation greedy? taking advantage of ordered alternation, NFA DFA and posix, the longest-leftmost", posix and the longest-leftmost rule, speed and efficiency.

Practical regex techniques

Continuation lines, matching an IP address, working with filenames, matching balanced sets of parentheses, watching out for unwanted matches, matching delimited text, knowing your data and making assumptions, stripping leading and trailing whitespace, matching and HTML tag, matching an HTML link, examining an HTTP URL, validating a hostname, plucking a hostname, plucking a URL, parsing CSV files.

Crafting an efficient expression

Efficiency vs. correctness, localizing greediness, global view of backtracking, more work for POSIX NFA, work required during a non-match, being more specific, alternation can be expensive, benchmarking, know what you re measuring, benchmarking with Python, common optimisations, the mechanics of regex application, pre-application optimizations, optimizations with the transmission, optimization of the regex itself, techniques for faster expressions, common sense techniques, expose literal text, expose anchors, lazy versus greedy: be specific, split into multiple REs, mimic initial-character discrimination, use atomic grouping and possessive quantifiers, lead the engine to a match, unrolling the loop, observations, using atomic grouping and possessive quantifiers, short unrolling examples, unrolling C comments, the free flowing regex, a helping hand to guide the match, a well-guided regex is a fast regex,

