Source Code Feature Extraction:

User’s Guide

In this document we give an overview of the patterns included in the Source Code Feature Extraction library. Patterns are designed to be run using the -wholefile option and follow the following naming convention: <language-identifier>.<pattern>

The output structure of the pattern is given in the following way:

b.line\_comments

\- b.line\_comment

\- b.line\_comment\_context

\- b.line\_comment\_body

\- b.line\_comment\_text

Items on the same indentation level correspond to children in the tree structured output. Items that are indented further are children of the item above it. \- is used to show the tree structure, and optional patterns are indicated with \?.

**Bash**

Language Identifier - b

**Inline Comments**

Matches in line comments starting with a '#'. Also

captures the code preceding the comment on the same line.

Run: rosie -wholefile b.line\_comments <bash-source-file>

Structure: b.line\_comments

\- b.line\_comment

\- b.line\_comment\_context

\- b.line\_comment\_body

\- b.line\_comment\_text

**String Literals**

Currently matches for typical bash string and character literals.

Does allow for escaped single and double quotes

Run: rosie -wholefile b.strings <bash-source-file>

Structure: b.strings

\- b.string

**Functions**

Pattern to match function definitions. Does not

capture function bodies.

Run: rosie -wholefile b.functions <bash-source-file>

Structure: b.functions

\- b.function

\- b.function\_name

**C**

Language Identifier - c

**String Literals**

Currently matches for typical c string and character literals.

Does allow for escaped single and double quotes

A "aliased" version is provided to allow for suppressed

output in other patterns

Run: rosie -wholefile c.strings <c-source-file>

Structure: c.strings

\- c.string

c.characters

\- c.character

**Inline Comments**

Matches in line comments starting with a '//'. Also

captures the code preceding the comment on the same line.

Run: rosie -wholefile c.line\_comments <c-source-file>

Structure: c.line\_comments

\- c.line\_comment

\- c.line\_comment\_context

\- c.line\_comment\_body

\- c.line\_comment\_text

**Block Comments**

Matches block comments. Block comments in c

start with /\* and end with \*/

Run: rosie -wholefile c.block\_comments <c-source-file>

Structure: c.block\_comments

\- c.block\_comment

\- c.block\_comment\_body

**Dependencies**

Matches dependencies declared with "include"

Run: rosie -wholefile c.dependencies <c-source-file>

Structure: c.dependencies

\- c.dependency

**Functions**

Pattern to match function definitions. This does not

capture the functions body.

Run: rosie -wholefile c.functions <c-source-file>

Structure: c.functions

\- c.function

\- c.function\_call

\- c.static

\- c.return\_type

\- c.pointer

\- c.function\_name

\- c.parameters

\- c.single\_param

**Structs**

Pattern to match struct definitions. Does not

capture struct bodies.

Run: rosie -wholefile c.structs <c-source-file>

Structure: c.structs

\- c.struct

\- c.struct\_name

**C++**

language Identifier - cpp

**String Literals**

Currently matches for typical c string and character literals.

Does allow for escaped single and double quotes

A "aliased" version is provided to allow for suppressed output

in other patterns

Run: rosie -wholefile cpp.strings <cpp-source-file>

Structure: cpp.strings

\- cpp.string

cpp.characters

\- cpp.character

NOTE cpp.strings and cpp.characters are defined AFTER comments to allow for

commented strings to be ignored.

**Inline Comments**

Matches in line comments starting with a '//'. Also

captures the code preceding the comment on the same line.

Run: rosie -wholefile cpp.line\_comments <cpp.source-file>

Structure: cpp.line\_comments

\- cpp.line\_comment

\- cpp.line\_comment\_context

\- cpp.line\_comment\_body

\- cpp.line\_comment\_text

**Block Comments**

Matches block comments. Block comments in c

start with /\* and end with \*/

Run: rosie -wholefile cpp.block\_comments <cpp.source-file>

Structure: cpp.block\_comments

\- cpp.block\_comment

\- cpp.block\_comment\_body

**Functions**

Pattern to match function definitions. This does not

capture the functions body. This is currently the same syntatic

definition of a C++ function.

New features of C++ have not yet been accounted for.

Run: rosie -wholefile cpp.functions <cpp-source-file>

Structure: cpp.functions

\- cpp.function

\- cpp.static

\- cpp.return\_type

\- cpp.pointer

\- cpp.function\_name

\- cpp.parameters

\- cpp.single\_param

**Structs/Class**

Pattern to match struct definitions. Does not

capture struct bodies.

Run: rosie -wholefile [cpp.structs | cpp.classes] <c++-source-file>

Structure: cpp.structs | cpp.classes

\- cpp.struct | cpp.class

\- cpp.struct\_name | cpp.class\_name

**C#**

Language Identifier - cs

**Inline Comments**

Matches in line comments starting with a '//'. Also

captures the code preceeding the comment on the same line.

Run: rosie -wholefile cs.line\_comments <C#-source-file>

Structure: cs.line\_comments

\- cs.line\_comment

\- cs.line\_comment\_context

\- cs.line\_comment\_body

\- cs.line\_comment\_text

**Block Comments**

Matches block comments starting with "/\*"

and ending with "\*/".

Run: rosie -wholefile cs.block\_comments <ruby-source-file>

Structure: rb.block\_comments

\- rb.block\_comment

\- rb.comment\_body

**String Literals**

Currently matches for typical C# string and character literals.

Does allow for escaped single and double quotes

Run: rosie -wholefile cs.strings <C#-source-file>

Structure: cs.strings

\- cs.string

**Dependencies**

Matches the C# dependencies in the file declared

with the "using" keyword.

Run: rosie -wholefile cs.dependencies <C#-source-file>

Structure: cs.dependencies

\- cs.dependency

\~ cs.alias\_statement

\- cs.alias

\- cs.type

\- cs.type

**Functions**

Pattern to match all function definitions in a file.

Does not match function bodies and constructors.

Run: rosie -wholefile cs.functions <C#-source-file>

Structure: cs.functions

\- cs.funcdef

\? cs.accessmod

\? cs.mod

\- cs.returntype

\- cs.methodname

\? cs.paramlist

\- cs.param

\? cs.parammod

\- cs.paramtype

\- cs.paramname

\? cs.defaultval

**Class Defs**

Pattern to match all class definitions in a file.

Does not match class bodies.

Run: rosie -wholefile cs.classes <C#-source-file>

Structure: cs.classes

\- cs.classdef

\- cs.classname

\- cs.parentclasslist

\- cs.parentclass

**Go**

Language Identifier - go

**Inline Comments**

Matches in line comments starting with a '//'. Also

captures the code preceding the comment on the same line.

Run: rosie -wholefile go.line\_comments <go-source-file>

Structure: go.line\_comments

\- go.line\_comment

\- go.line\_comment\_context

\- go.line\_comment\_body

\- go.line\_comment\_text

**Block Comments**

Matches block comments. Block comments in go

start with /\* and end with \*/

Run: rosie -wholefile go.block\_comments <go-source-file>

Structure: go.block\_comments

\- go.block\_comment

\- go.block\_comment\_body

**String Literals**

Currently matches for typical Go string and character literals.

Does allow for escaped single and double quotes

Run: rosie -wholefile go.strings <Go-source-file>

Structure: go.strings

\- go.string

**Dependencies**

Matches dependencies declared with "import" and

packages listed with "package".

Run: rosie -wholefile go.dependencies <go-source-file>

Structure: go.dependencies

\- go.package

\- go.package\_text

\- go.dependency

\- go.dependencies\_text

\- go.dependency\_factor

\- go.import\_list

**Functions**

Pattern to match function definitions. Does not

capture function bodies or constructors.

Run: rosie -wholefile go.functions <go-source-file>

Structure: go.functions

\- go.function

\- go.function\_name

\- go.parameters

\- go.single\_param

\- go.return

\- go.single\_return

**Structs**

Pattern to match struct definitions. Does not

capture struct bodies.

Run: rosie -wholefile go.structs <go-source-file>

Structure: go.structs

\- go.struct

\- go.struct\_name

**Java**

Language Identifier - java

**String Literals**

Currently matches for typical java string and character literals.

Does allow for escaped single and double quotes

A "aliased" version is provided to allow for suppressed output in other patterns

Run: rosie -wholefile java.strings <java-source-file>

Structure: java.strings

\- java.string

java.characters

\- java.character

NOTE java.strings and java.characters are defined

AFTER inline comments to allow forcommented strings to be ignored.

**Inline Comments**

Matches for inline java comments starting with "//".

Output provides the context of the comment i.e. the

text listed on the same line,

Values escaped within string literals are ignored.

Run: rosie -wholefile java.line\_comments <java-source-file>

Structure: java.line\_comments

\- java.line\_comment

\- java.line\_comment\_context

\- java.line\_comment\_body

\- java.line\_comment\_text

**Block Comments**

Matches for java block comments starting with "/\*" and ending with "\*/".

Output simply provides the contents of the comment.

Run: rosie -wholefile java.block\_comments <java-source-file>

Structure: java.block\_comments

\- java.block\_comment

\- java.block\_comment\_body

**Dependencies**

Matches dependencies declared with "import" and

packages listed with "package".

Run: rosie -wholefile java.dependencies <java-source-file>

Structure: java.dependencies

\- java.package

\- java.package\_text

\- java.dependency

\- java.dependencies\_text

**Functions**

Pattern to match function definitions. Does not

capture function bodies or constructors.

Run: rosie -wholefile java.functions <java-source-file>

Structure: java.functions

\- java.function

\- java.access\_sp

\- java.return\_type

\- java.function\_name

\- java.parameters

\- java.single\_param

\- java.exceptions

\- java.single\_excep

**Classes**

Pattern to match class definitions. Does not

capture class bodies.

Run: rosie -wholefile java.classes <java-source-file>

Structure: java.classes

\- java.class

\- java.access\_sp

\- java.class\_name

\- java.hierarcy

\- java.parent\_name

\- java.implementation

\- java.interface\_name

**Javascript**

Language Identifier - js

**String Literals**

Catches strings in the file any text inside the charactesrs " x "

Also catches \" to continue matching the string if " is escaped

Run : rosie -wholefile "js.strings" <javascript-source-file>

Structure : js.strings

\- js.strings\_text

**Inline Comments**

Matches in line comments starting with a '//'. Also

captures the code preceding the comment on the same line.

Run: rosie -wholefile "js.line\_comments" <javascript-source-file>

Structure: js.line\_comments

\- js.line\_comment

\- js.line\_comment\_context

\- js.line\_comment\_body

\- js.line\_comment\_text

**Block Comments**

Matches in line comments starting with a '/\*' '\*/'. Also

captures the code preceding the comment on the same line.

Run: rosie -wholefile "js.block\_comments" <javascript-source-file>

Structure: js.block\_comments

\- js.block\_comment

\- js.block\_comment\_pre

\- js.block\_comment\_body

**Dependencies**

Matches all the import statements present in the source file

Run: rosie -wholefile "js.line\_comments" <javascript-source-file>

Structure: js.dependencies

\- js.dependencies\_pre

\- js.dependency

\- js.dependencies\_lonemember

\- js.member

\- js.member\_text

\- js.dependencies\_multimember

\- js.multi\_members

\- js.member

\- js.member\_text

\- js.dependencies\_module

\- js.module\_text

**Function Definitions**

Catches function calls present in the file

Run: rosie -wholefile "js.line\_comments" <javascript-source-file>

Structure: js.functions

\- js.function\_call

\- js.function

\- js.function\_name

\- js.parameters

\- js.multip\_param

\- js.single\_param

**Classes**

Matches class definitions. This DOES not match any information within

the body of the class, currently this is not possible within rosie.

Run: rosie -wholefile js.classes <javascript-source-file>

Structure: js.classes

\- js.class\_definition

\- js.class

\- js.class\_name

\- js.parameters

\- js.multip\_param

\- js.single\_param

\- js.extends

**Python**

Language Identifier - py

**String Literals**

Currently matches for typical c string and character literals.

Does allow for escaped single and double quotes

Run: rosie -wholefile py.strings <c-source-file>

Structure: py.strings

\- py.string

**Inline Comments**

Matches in line comments starting with a '//'. Also

captures the code preceeding the comment on the same line.

Run: rosie -wholefile cs.line\_comments <C#-source-file>

Structure: cs.line\_comments

\- cs.line\_comment

\- cs.line\_comment\_context

\- cs.line\_comment\_body

\- cs.line\_comment\_text

**Dependencies**

Matches Python import statements. This should include the complete definition of a Python import aka from, import, as, etc.

Run: rosie -wholefile cs.dependencies <python-source-file>

Structure: cs.dependencies

\- cs.dependency

\- py.import\_all

\- py.parent\_module

\- py.import\_from\_parent

\- py.parent\_module

\- py.import\_module

\- py.import\_module

\- py.module

\- py.module\_alias

**Functions**

Matches python function definition. This DOES not match any information within the body of the function, currently this is not possible within rosie.

Run: rosie -wholefile cs.functions <python-source-file>

Structure: cs.functions

\- cs.function

\- py.function\_name

\- py.paraeters

**Classes**

Matches python class definitions. This DOES not match any information within

the body of the class, currently this is not possible within rosie.

Run: rosie -wholefile cs.functions <python-source-file>

Structure: cs.classes

\- cs.class

\- py.class\_name

\- py.parents -

**R**

Language Identifier - r

**Inline Comments**

Matches in line comments starting with a '#'. Also

captures the code preceding the comment on the same line.

Run: rosie -wholefile r.line\_comments <r-source-file>

Structure: r.line\_comments

\- r.line\_comment

\- r.line\_comment\_context

\- r.line\_comment\_body

\- r.line\_comment\_text

**String Literals**

Currently matches for typical R string and character literals.

Does allow for escaped single and double quotes

Run: rosie -wholefile cs.strings <R-source-file>

Structure: r.strings

\- r.string

Dependencies

Matches dependencies declared with "source" and "library"

Run: rosie -wholefile r.dependencies <r-source-file>

Structure: r.dependencies

\- r.library

\- r.dependencies\_text

\- r.source

\- r.dependencies\_text

Functions

Pattern to match function definitions. Does not

capture function bodies.

Run: rosie -wholefile r.functions <r-source-file>

Structure: r.functions functions in a file

\- r.function\_name

\- r.parameters

\- r.single\_param

**Ruby**

Language Identifier - rb

**Block Comments**

Matches block comments. Block comments in ruby

start with a "=begin" on a new line and end with

a "=end" on a new line. Both "=begin" and "=end"

must appear first on a line.

Run: rosie -wholefile rb.block\_comments <ruby-source-file>

Structure: rb.block\_comments

\- rb.block\_comment

\- rb.comment\_body

**Inline Comments**

Matches in line comments starting with a '#'. Also

captures the code preceeding the comment on the same line.

Run: rosie -wholefile rb.line\_comments <ruby-source-file>

Structure: rb.line\_comments

\- rb.line\_comment

\- rb.line\_comment\_context

\- rb.line\_comment\_body

\- rb.line\_comment\_text

**String Literals**

Currently matches for typical ruby string and character literals.

Does allow for escaped single and double quotes

Run: rosie -wholefile rb.strings <ruby-source-file>

Structure: rb.strings

\- rb.string

**Dependencies**

Matches dependencies declared with "load", "require", or

"require\_relative".

Run: rosie -wholefile rb.dependencies <ruby-source-file>

Structure: rb.dependencies

\- rb.dependency

\- rb.module

**Functions**

Pattern to match function definitions. Does not

capture function bodies.

Run: rosie -wholefile rb.functions <ruby-source-file>

Structure: rb.functions

\- rb.function

\- rb.funcname

\- rb.paramlist

\- rb.paramdef

\- rb.param

\- rb.paramval

\- rb.param

**Class Defs**

Pattern to match all class definitions.

Does not match class bodies.

Run: rosie -wholefile rb.classes <ruby-source-file>

Structure: rb.classes

\- rb.class

\- rb.classname

\- rb.superclass

**Visual Basic**

Language Identifier - vb

Inline Comments

Matches in line comments starting with a "'". Also

captures the code preceeding the comment on the same line.

Run: rosie -wholefile vb.line\_comments <vb-source-file>

Structure: vb.line\_comments

\- vb.line\_comment

\- vb.line\_comment\_context

\- vb.line\_comment\_body

\- vb.line\_comment\_text

String Literals

Currently matches for typical vb string literals.

Does allow for escaped single and double quotes

Run: rosie -wholefile vb.strings <vb-source-file>

Structure: vb.strings

\- vb.string

Dependencies

Matches dependencies declared with the "Imports"

key word.

Run: rosie -wholefile vb.dependencies <vb-source-file>

Structure: rb.dependencies

\- rb.dependency

\- vb.dependency\_alias

\- vb.alias\_name

\- vb.dependency\_name

**Functions**

Matches functions, including the function body.

Run: rosie -wholefile vb.functions <vb-source-file>

Structure: vb.functions

\- vb.function

\- vb.funcdef

\- vb.funcname

\- vb.paramlist

\- vb.paramstmnt

\- vb.param

\- vb.datatype

\- vb.datatype

\- vb.funcbody

**Class Defs**

Matches all classes in a file including the class body.

Run: rosie -wholefile vb.classes <vb-source-file>

Structure: vb.classes

\- vb.class

\- vb.classdef

\- vb.classname

\- vb.typelist

\- vb.typenames

\- vb.typename

\- vb.constraintlist

\- vb.constraint

\- vb.parentclass

\- vb.interfacenames

\- vb.interfacename

\- vb.classbody