

# Spiral: splitters for identifiers in source code files

Michael Hucka<sup>1</sup>

<sup>1</sup> Department of Computing and Mathematical Sciences, California Institute of Technology, Pasadena, CA 91125, USA

DOI: [00.00000/joss.00000](https://doi.org/10.0000/joss.00000)

## Software

- [Review](#) ↗
- [Repository](#) ↗
- [Archive](#) ↗

Submitted: 00 January 0000

Published: 00 January 0000

## Licence

Authors of JOSS papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC-BY](#)).

## Summary

*Spiral* is a Python 3 package that implements numerous identifier splitting algorithms. *Identifier splitting* (also known as *identifier name tokenization*) is the task of breaking apart program identifier strings such as `getInt` or `readUTF8stream` into component tokens: `[get, int]` and `[read, UTF8, stream]`. The need for splitting identifiers arises in a variety of contexts, including natural language processing (NLP) methods applied to source code analysis and program comprehension.

*Spiral* is easy to use. Here are some examples of calling the Ronin splitter algorithm on inputs that would challenge simpler splitters. The following Python code,

```
from spiral import ronin
for s in [ 'mStartCData', 'nonnegativedecimaltype', 'getUtf8Octets',
          'savefileas', 'nbrOfbugs' ]:
    print(ronin.split(s))
```

produces the following output:

```
['m', 'Start', 'C', 'Data']
['nonnegative', 'decimal', 'type']
['get', 'Utf8', 'Octets']
['save', 'file', 'as']
['nbr', 'Of', 'bugs']
```

*Spiral* also includes a command-line program named `spiral`; it will split strings provided on the command line or in a file, and is useful for experimenting with *Spiral*.

## The need for sophisticated splitting algorithms

Splitting identifiers is deceptively difficult and remains a research problem for which no perfect solution exists today. Even in cases where the input consists of identifiers that strictly follow conventions such as camel case, ambiguities can arise. For example, to split `J2SEProjectTypeProfiler` into `[J2SE, Project, Type, Profiler]` requires the reader to recognize `J2SE` as a unit. The task of splitting identifiers is made more difficult when there are no case transitions or other obvious boundaries in an identifier.

*Spiral* provides some several basic naive splitting algorithms, such as a straightforward camel-case splitter, as well as more elaborate heuristic splitters, including a novel algorithm we call *Ronin*. *Ronin* uses a variety of heuristic rules, English dictionaries, and tables of token frequencies obtained from mining source code repositories. It includes a default table of term frequencies derived from an analysis of over 46,000 randomly selected software projects in GitHub that contained at least one Python source code file.

## Splitters available in Spiral

The following table lists the splitters implemented in Spiral at this time:

Splitter name	Operation
<code>delimiter_split</code>	split only at characters <code>\$ ~ _ . : / @</code>
<code>digit_split</code>	split only at digits
<code>pure_camelcase_split</code>	split at forward camel case transitions (lower to upper case)
<code>safe_simple_split</code>	split at hard delimiter characters and forward camel case only; won't split strings that don't follow strict camel case
<code>simple_split</code>	split at hard delimiter characters and forward camel case, even if a string doesn't follow strict camel case conventions
<code>elementary_split</code>	split by hard delimiters, forward camel case, and digits
<code>heuristic_split</code>	split by hard delimiters, forward camel case, and digits, but recognize special cases such as <code>utf8</code> , <code>sha256</code> , etc.
<i>Samurai</i>	frequency-based approach published in the literature
<i>Ronin</i>	frequency-based approach originally based on Samurai

The name “Ronin” is a play on the use of the name “Samurai” (Enslin et al. 2009) for their identifier splitting algorithm. The core loop of Ronin is based on Samurai, but substantially modified and extended. A goal for Ronin was to produce a splitter that had good performance using only a global table of token frequencies, without the need for an additional table of frequencies mined from the source code currently being analyzed. This makes Ronin usable even without preprocessing a code base to extract token frequencies.

The name *Spiral* is a loose acronym based on “*SPlitters for IdentifieRs: A Library*”.

## Acknowledgments

This material is based upon work supported by the [National Science Foundation](#) under Grant Number 1533792. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

## References

Enslin, Eric, Emily Hill, Lori Pollock, and K. Vijay-Shanker. 2009. “Mining Source Code to Automatically Split Identifiers for Software Analysis.” In *Proceedings of the 6th IEEE International Working Conference on Mining Software Repositories (Msr'09)*, 71–80. IEEE Press. <https://doi.org/10.1109/MSR.2009.5069482>.