

4-2015

# Developing Beginner-Friendly Programming Error Messages

Aaron D. Lemmon

Emma Sax

Paul A. Schliep

Follow this and additional works at: [http://digitalcommons.morris.umn.edu/urs\\_2015](http://digitalcommons.morris.umn.edu/urs_2015)

 Part of the [Computer Sciences Commons](#)

---

## Recommended Citation

Lemmon, Aaron D.; Sax, Emma; and Schliep, Paul A., "Developing Beginner-Friendly Programming Error Messages" (2015).  
*Undergraduate Research Symposium 2015*. Book 5.  
[http://digitalcommons.morris.umn.edu/urs\\_2015/5](http://digitalcommons.morris.umn.edu/urs_2015/5)

This Book is brought to you for free and open access by the Undergraduate Research Symposium at University of Minnesota Morris Digital Well. It has been accepted for inclusion in Undergraduate Research Symposium 2015 by an authorized administrator of University of Minnesota Morris Digital Well. For more information, please contact [skulann@morris.umn.edu](mailto:skulann@morris.umn.edu).



## UNIVERSITY OF MINNESOTA MORRIS

### Basic Background

This project began in 2012 to transition an introductory computer science course (currently taught in the Racket language) to Clojure, a new programming language released in 2007. This project requires new tools to be created in order to teach computer science to novice programmers.

#### Specific Tools Our Project Needs

- A programming environment
- Improved error handling
- Tools for testing code
- Tools for managing code

### Our Goals & Open Source Development

#### Our Goals

- Explore creating the programming environment
- Improve error handling
- Rewrite simple functions to improve usability for new users
- Document common use cases for our system
- Incorporate useful hints and enhanced error messages to assist students

Our project is open source, which means that anyone can view, modify, and distribute the code.

### Introduction to Clojure

- Clojure is a newer programming language similar to Racket
- Offers integration with other languages like Java
- Offers better parallel processing
- Clojure is used more often in the business world than Racket
- Limitations - unintuitive error messages

### Error Handling

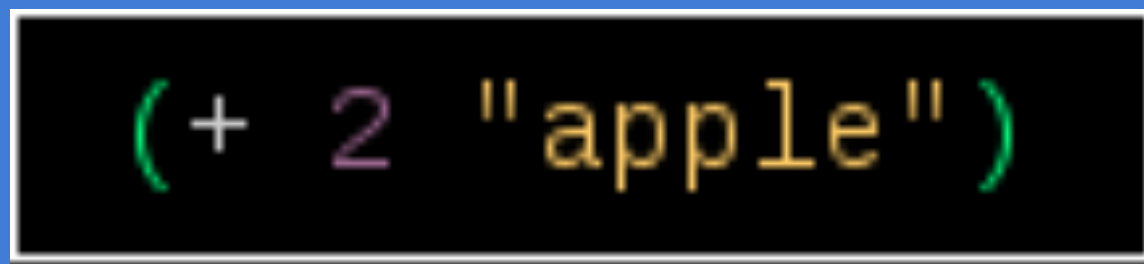
An error in computer programming is an expression or statement that cannot be understood or handled by a computer.

#### Good Error Handling Includes:

- A way for users to find where their mistakes occurred
- An explanation for why their mistake occurred in simple terms
- Examples of causes that commonly result in the error
- Hints to assist with the correction of the mistakes

The top picture shows a standard Clojure error message.

The bottom picture shows our modified error message that includes more details about the error, as well as a sequence of functions for tracing where the error occurred.



```
Exception in thread "main" java.lang.AssertionError: Assert failed: (check-if-numbers? "+" args 1)
    at corefns.corefns$PLUS_.doInvoke(corefns.clj:174)
    at clojure.lang.RestFn.invoke(RestFn.java:421)
    at intro.core$main.doInvoke(core.clj:663)
    at clojure.lang.RestFn.invoke(RestFn.java:397)
    at clojure.lang.Var.invoke(Var.java:375)
    at user$eval6267.invoke(NO_SOURCE_FILE:1)
    at clojure.lang.Compiler.eval(Compiler.java:6703)
    at clojure.lang.Compiler.eval(Compiler.java:6693)
    at clojure.lang.Compiler.eval(Compiler.java:6666)
    at clojure.core$eval.invoke(core.clj:2927)
    at clojure.main$eval_opt.invoke(main.clj:288)
    at clojure.main$initialize.invoke(main.clj:307)
```

```
ERROR: in function + second argument "apple" must be a number but is a string
corefns.corefns/+ (corefns.clj line 174)
intro.core/-main (core.clj line 663)
clojure.core/eval (core.clj line 2927)
```

### User Scenarios

- A user scenario is a hypothetical narrative detailing a common student workflow
- User scenarios allow us to explore the common outcomes of student mistakes

#### Example of a user scenario:

Erroneous code fragment:

```
print(Hello World)
```

Clojure error message:

```
CompilerException java.lang.RuntimeException: Unable to resolve symbol: Hello in this context
```

Our modified error message:

```
ERROR: Compilation error: name Hello is undefined
```

Student edit:

```
print("Hello World")
```

Clojure error message:

```
ClassCastException java.lang.String cannot be cast to clojure.lang.IFn
```

Our modified error message:

```
ERROR: Attempted to use a string, but a function was expected.
```

Corrected code:

```
(print "Hello World")
```

### Hints

#### How Hints are Helpful

- Errors can have several underlying causes that the error message may not convey
- Having multiple hints per error can offer various suggestions for resolving issues
- Hints provide a more human interpretation of the mistake
- Hints are meant to guide the student in finding a solution to the error

#### Example of a hint:

Erroneous code fragment:

```
print(Hello World)
```

Our modified error message:

```
ERROR: Compilation error: name Hello is undefined
```

Hint:

It looks like Clojure is expecting that Hello is something named in your program. If you want Hello and any following words to be plain text, try surrounding them with double quotes. If Hello is referring to something named in your program, make sure it is spelled correctly.

### Future Work

#### Error Handling

- Wrap up development of runtime and compilation error handling
- Add contextual information on errors for students
- Document more user scenarios
- Develop more hints

#### Usability Studies

- Test usability of our software with a group of beginner students
- Gather feedback to determine which areas of our program needs improvement
- Integrate feedback to enhance our system's usability

### More Information

Aaron Lemmon: lemmon031@morris.umn.edu  
Emma Sax: saxxx027@morris.umn.edu  
Paul Schliep: schli202@morris.umn.edu

<http://github.com/Clojure-Intro-Course>



### References

Marceau, G., Fidler, K., and Krishnamurthi, S. *Measuring the effectiveness of error messages designed for novice programmers*. N.p.: Proceedings of the 42nd ACM Technical Symposium on Computer Science, 2012

Fogus, Michael. *The Joy of Clojure: Thinking the Clojure Way*. N.p.: Manning Publications, 2011. Print.

Emerick, Chas, Brian Carper, and Christophe Grande. *Clojure Programming*. N.p.: O'Reilly Media, 2012. Print.

<http://clojuredocs.org/>

<http://jayfields.com/expectations/index.html>