

# Functional Programming in Racket

Peter Campora

ULL

August 25, 2019

# Programming Language Hot Takes



# Programming Language Hot Takes



There are numerous programming languages and even more opinions about them.

# Programming Language Hot Takes



There are numerous programming languages and even more opinions about them.  
So, why so many languages and how do they differ?

# Programming Language Hot Takes



There are numerous programming languages and even more opinions about them.

So, why so many languages and how do they differ?

And why is *functional programming* becoming such a buzzword?

# Why Functional Programming?

```
var total = 0

def addToTotal(x: Int): Int = {
    total += x
    total
}

assert(addToTotal(1) == addToTotal(1))
```

NOPE

# Why Functional Programming?

```
var total = 0

def addToTotal(x: Int): Int = {
    total += x
    total
}

assert(addToTotal(1) == addToTotal(1))
```

NOPE

- The function above isn't *referentially transparent*

# Why Functional Programming?

```
var total = 0

def addToTotal(x: Int): Int = {
    total += x
    total
}

assert(addToTotal(1) == addToTotal(1))
```

NOPE

- The function above isn't *referentially transparent*
- Which means it really isn't even a function in the *mathematical* sense.

# Why Functional Programming?

```
var total = 0

def addToTotal(x: Int): Int = {
    total += x
    total
}

assert(addToTotal(1) == addToTotal(1))
```

NOPE

- The function above isn't *referentially transparent*
- Which means it really isn't even a function in the *mathematical* sense.
- This makes it hard to create reproducible behavior or prove code correctness

# Why Functional Programming?

```
var total = 0

def addToTotal(x: Int): Int = {
    total += x
    total
}

assert(addToTotal(1) == addToTotal(1))
```

NOPE

- The function above isn't *referentially transparent*
- Which means it really isn't even a function in the *mathematical* sense.
- This makes it hard to create reproducible behavior or prove code correctness
- Concurrent programming is made difficult since order of evaluation matters.

# Why Functional Programming?

```
var total = 0

def addToTotal(x: Int): Int = {
    total += x
    total
}

assert(addToTotal(1) == addToTotal(1))
```

NOPE

- The function above isn't *referentially transparent*
- Which means it really isn't even a function in the *mathematical* sense.
- This makes it hard to create reproducible behavior or prove code correctness
- Concurrent programming is made difficult since order of evaluation matters.
- Functional idioms creep into imperative languages. Lambda and streams were added to Java, and JavaScript code becomes increasingly functional.

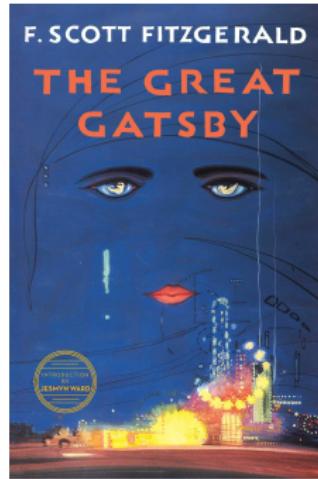
# What Things are Beautiful?



# What Things are Beautiful?



# What Things are Beautiful?



# Can Code be Beautiful?

Write a program for summing the squares of numbers from 1 to n.

# Can Code be Beautiful?

Write a program for summing the squares of numbers from 1 to n.

```
public class SumSquares
{
    public static void main(final int[] nums)
    {
        int sum = 0;
        for (int i : nums)
            sum += i * i;
        System.out.println("The sum of the squares is:
→   " + sum);
    }
}
```

# An Equivalent Program

Let's get our first look at a functional program written in Racket.

# An Equivalent Program

Let's get our first look at a functional program written in Racket.

```
;Number -> Number
```

```
(define (square x) (* x x))
```

```
;Number List -> Number List
```

```
(define (sum-of-squares lst)
  (sum (map square lst)))
```

# An Equivalent Program

Let's get our first look at a functional program written in Racket.

```
;Number -> Number
(define (square x) (* x x))
```

```
;Number List -> Number List
(define (sum-of-squares lst)
  (sum (map square lst)))
```

Don't worry about the meaning and syntax of the program, yet.  
Given time, the parens will become beautiful!

# What *Is* Code Beauty?

First let's hear from you!

## *What Is Code Beauty?*

First let's hear from you! Let's consider some other views.

- Utilitarian Program Beauty

## *What Is Code Beauty?*

First let's hear from you! Let's consider some other views.

- Utilitarian Program Beauty
  - The programming language that maximizes the utility of writing my program is the best. Minimize bugs, execution time, and time until production.

## *What Is Code Beauty?*

First let's hear from you! Let's consider some other views.

- Utilitarian Program Beauty

- The programming language that maximizes the utility of writing my program is the best. Minimize bugs, execution time, and time until production.
- Critique: Hard to measure which features objectively help the process of writing programs.

# *What Is Code Beauty?*

First let's hear from you! Let's consider some other views.

- Utilitarian Program Beauty
  - The programming language that maximizes the utility of writing my program is the best. Minimize bugs, execution time, and time until production.
  - Critique: Hard to measure which features objectively help the process of writing programs.
- Egoist Program Beauty

# *What Is Code Beauty?*

First let's hear from you! Let's consider some other views.

- Utilitarian Program Beauty
  - The programming language that maximizes the utility of writing my program is the best. Minimize bugs, execution time, and time until production.
  - Critique: Hard to measure which features objectively help the process of writing programs.
- Egoist Program Beauty
  - The programming language that allows me to write clever, expressive programs is best.

# *What Is Code Beauty?*

First let's hear from you! Let's consider some other views.

- Utilitarian Program Beauty
  - The programming language that maximizes the utility of writing my program is the best. Minimize bugs, execution time, and time until production.
  - Critique: Hard to measure which features objectively help the process of writing programs.
- Egoist Program Beauty
  - The programming language that allows me to write clever, expressive programs is best.
  - Critique: Engineering is largely a social endeavor. Other people need to be able to read your programs and understand abstractions.

# *What Is Code Beauty?*

First let's hear from you! Let's consider some other views.

- Utilitarian Program Beauty
  - The programming language that maximizes the utility of writing my program is the best. Minimize bugs, execution time, and time until production.
  - Critique: Hard to measure which features objectively help the process of writing programs.
- Egoist Program Beauty
  - The programming language that allows me to write clever, expressive programs is best.
  - Critique: Engineering is largely a social endeavor. Other people need to be able to read your programs and understand abstractions.
- Aesthetic Program Beauty

# *What Is Code Beauty?*

First let's hear from you! Let's consider some other views.

- Utilitarian Program Beauty
  - The programming language that maximizes the utility of writing my program is the best. Minimize bugs, execution time, and time until production.
  - Critique: Hard to measure which features objectively help the process of writing programs.
- Egoist Program Beauty
  - The programming language that allows me to write clever, expressive programs is best.
  - Critique: Engineering is largely a social endeavor. Other people need to be able to read your programs and understand abstractions.
- Aesthetic Program Beauty
  - Beauty is subjective according to appreciation of the idioms of some paradigm (i.e. cubism in art).

# What *Is* Code Beauty?

First let's hear from you! Let's consider some other views.

- Utilitarian Program Beauty
  - The programming language that maximizes the utility of writing my program is the best. Minimize bugs, execution time, and time until production.
  - Critique: Hard to measure which features objectively help the process of writing programs.
- Egoist Program Beauty
  - The programming language that allows me to write clever, expressive programs is best.
  - Critique: Engineering is largely a social endeavor. Other people need to be able to read your programs and understand abstractions.
- Aesthetic Program Beauty
  - Beauty is subjective according to appreciation of the idioms of some paradigm (i.e. cubism in art).
  - Critique: Disconnected from the empirical or mathematical viewpoints we typically employ to measure the value of scientific contributions.

# Looks Like We Don't Know



## So, Why Ask?

- Universities are a place for the creation and acquisition of knowledge.

## So, Why Ask?

- Universities are a place for the creation and acquisition of knowledge.
- We should put our opinions under scrutiny.



## So, Why Ask?

- Universities are a place for the creation and acquisition of knowledge.
- We should put our opinions under scrutiny.
- Computer scientists are critical thinkers.



## So, Why Ask?

- Universities are a place for the creation and acquisition of knowledge.
- We should put our opinions under scrutiny.
- Computer scientists are critical thinkers.
- Computer science continues on even though many of the first major results were about undecidability.



# How Do We Write Programs?



# How Do We Write Programs?



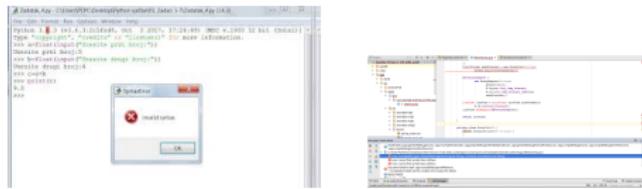
Let's hear from you.

# Is Programming Hard?

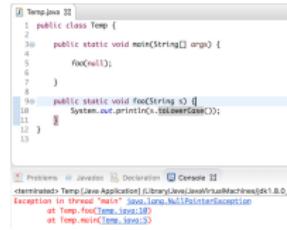
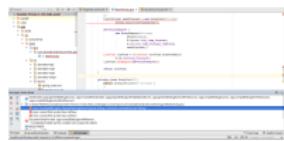
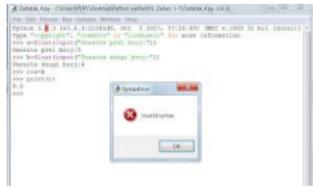
```
file:///C:/Users/Py/Downloads/test.py (1,0)
line 1: from Tkinter import *
^
SyntaxError: invalid syntax
Python 2.7 (r27:66117, Oct 2 2009, 17:24:48) [GCC 4.4.1 20100505 (Debian 4.4.1-18)] on linux2
Type "help", "copyright" or "credits" for more information.
>>> exit(1)
```



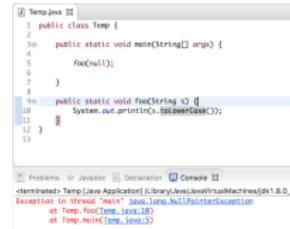
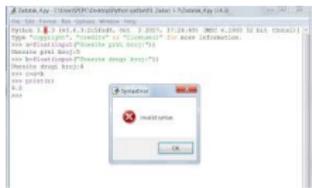
# Is Programming Hard?



# Is Programming Hard?

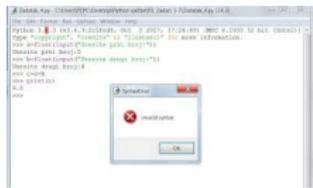


# Is Programming Hard?

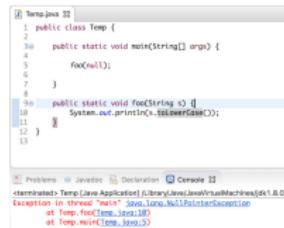


```
1 FibonacciPracitce>java DispFibonacci
2 Displa...
3 Exception in thread "main" java.lang.StackOverflowError
4 at Fibonacci.ca...
5 at Fibonacci.ca...
6 at Fibonacci.ca...
7 at Fibonacci.ca...
8 at Fibonacci.ca...
9 at Fibonacci.ca...
10 at Fibonacci.ca...
11 at Fibonacci.ca...
12 at Fibonacci.ca...
13 at Fibonacci.ca...
14 at Fibonacci.ca...
15 at Fibonacci.ca...
16 at Fibonacci.ca...
17 at Fibonacci.ca...
18 at Fibonacci.ca...
19 at Fibonacci.ca...
20 at Fibonacci.ca...
21 at Fibonacci.ca...
22 at Fibonacci.ca...
23 at Fibonacci.ca...
24 at Fibonacci.ca...
25 at Fibonacci.ca...
26 at Fibonacci.ca...
27 at Fibonacci.ca...
28 at Fibonacci.ca...
29 at Fibonacci.ca...
30 at Fibonacci.ca...
31 at Fibonacci.ca...
32 at Fibonacci.ca...
33 at Fibonacci.ca...
34 at Fibonacci.ca...
35 at Fibonacci.ca...
36 at Fibonacci.ca...
37 at Fibonacci.ca...
38 at Fibonacci.ca...
39 at Fibonacci.ca...
40 at Fibonacci.ca...
41 at Fibonacci.ca...
42 at Fibonacci.ca...
43 at Fibonacci.ca...
44 at Fibonacci.ca...
45 at Fibonacci.ca...
46 at Fibonacci.ca...
47 at Fibonacci.ca...
48 at Fibonacci.ca...
49 at Fibonacci.ca...
50 at Fibonacci.ca...
51 at Fibonacci.ca...
52 at Fibonacci.ca...
53 at Fibonacci.ca...
54 at Fibonacci.ca...
55 at Fibonacci.ca...
56 at Fibonacci.ca...
57 at Fibonacci.ca...
58 at Fibonacci.ca...
59 at Fibonacci.ca...
60 at Fibonacci.ca...
61 at Fibonacci.ca...
62 at Fibonacci.ca...
63 at Fibonacci.ca...
64 at Fibonacci.ca...
65 at Fibonacci.ca...
66 at Fibonacci.ca...
67 at Fibonacci.ca...
68 at Fibonacci.ca...
69 at Fibonacci.ca...
70 at Fibonacci.ca...
71 at Fibonacci.ca...
72 at Fibonacci.ca...
73 at Fibonacci.ca...
74 at Fibonacci.ca...
75 at Fibonacci.ca...
76 at Fibonacci.ca...
77 at Fibonacci.ca...
78 at Fibonacci.ca...
79 at Fibonacci.ca...
80 at Fibonacci.ca...
81 at Fibonacci.ca...
82 at Fibonacci.ca...
83 at Fibonacci.ca...
84 at Fibonacci.ca...
85 at Fibonacci.ca...
86 at Fibonacci.ca...
87 at Fibonacci.ca...
88 at Fibonacci.ca...
89 at Fibonacci.ca...
90 at Fibonacci.ca...
91 at Fibonacci.ca...
92 at Fibonacci.ca...
93 at Fibonacci.ca...
94 at Fibonacci.ca...
95 at Fibonacci.ca...
96 at Fibonacci.ca...
97 at Fibonacci.ca...
98 at Fibonacci.ca...
99 at Fibonacci.ca...
100 at Fibonacci.ca...
```

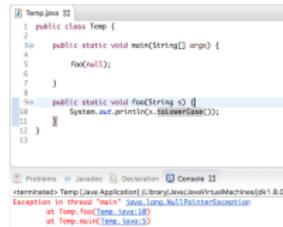
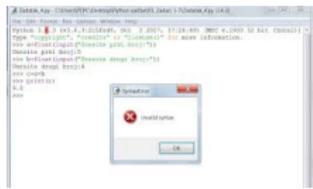
# Is Programming Hard?



```
user@host:/tmp$ segFault$ cat segFault.c
void main() {
    char *str = "Hello, world!";
    *str = 'A';
}
user@host:/tmp$ segFault$ gcc segFault.c -o segFault
user@host:/tmp$ ./segFault
Segmentation fault
user@host:/tmp$ segFault$
```



# Is Programming Hard?



```
user@host:/tmp$ segfault$ cat segfault.c
void main() {
    char *str = "Hello, world!";
    *str = 'A';
}
user@host:/tmp$ segfault$ gco segfault.c -o segfault
user@host:/tmp$ segfault
Segmentation fault
user@host:/tmp$ segfault$
```



# How Programming Is Taught



# How Programming Is Taught



- Programming courses typically start by motivating new features with concrete programs.

# How Programming Is Taught



- Programming courses typically start by motivating new features with concrete programs.
- Given enough examples, students begin to match problem descriptions to using features.

# How Programming Is Taught



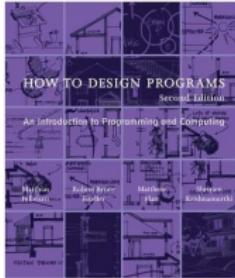
- Programming courses typically start by motivating new features with concrete programs.
- Given enough examples, students begin to match problem descriptions to using features.
- From there, programming becomes gluing together recognized patterns.

# How Programming Is Taught

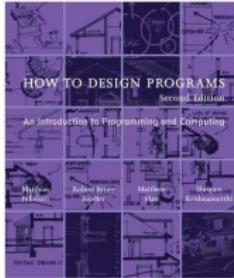


- Programming courses typically start by motivating new features with concrete programs.
- Given enough examples, students begin to match problem descriptions to using features.
- From there, programming becomes gluing together recognized patterns.
- **There has to be a better way!**

# The Better Way

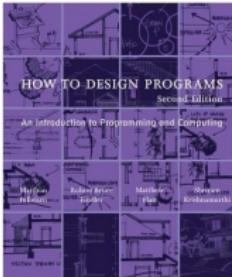


# The Better Way



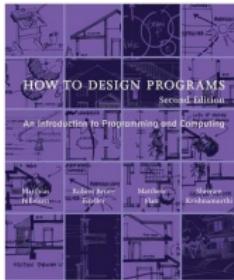
- This book (and course) will teach systematic program design.

# The Better Way



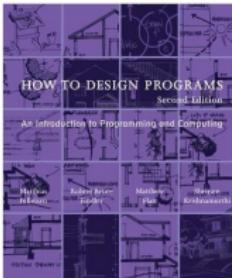
- This book (and course) will teach systematic program design.
- It presents recipes for designing two different types of programs.

# The Better Way



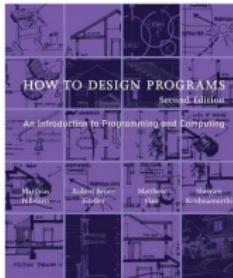
- This book (and course) will teach systematic program design.
- It presents recipes for designing two different types of programs.
  - Big bang (event driven) programs

# The Better Way



- This book (and course) will teach systematic program design.
- It presents recipes for designing two different types of programs.
  - Big bang (event driven) programs
  - Batch programs (effectful scripts)

# The Better Way



- This book (and course) will teach systematic program design.
- It presents recipes for designing two different types of programs.
  - Big bang (event driven) programs
  - Batch programs (effectful scripts)
- Isn't this a course on functional programming?

# Learning Functional Programming from the Ground Up



# Learning Functional Programming from the Ground Up



- Why How to Design Programs?

# Learning Functional Programming from the Ground Up

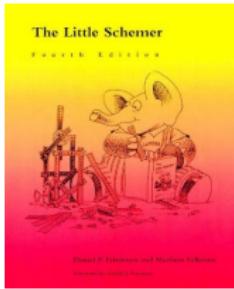


- Why How to Design Programs?
- When I first tried to learn functional programming, I failed.

# Learning Functional Programming from the Ground Up



- Why How to Design Programs?
- When I first tried to learn functional programming, I failed.

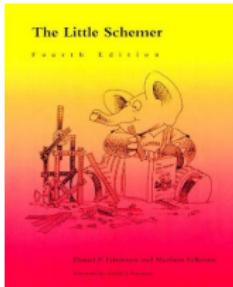


-

# Learning Functional Programming from the Ground Up

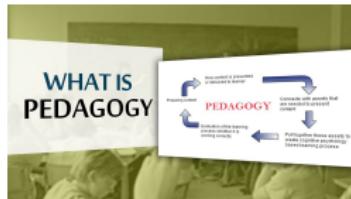


- Why How to Design Programs?
- When I first tried to learn functional programming, I failed.



- We will learn functional programming and systematic design, together!

# Why Racket and Not X?



# Why Racket and Not X?



- Teaching functional programming in an imperative-first language is awkward

# Why Racket and Not X?



- Teaching functional programming in an imperative-first language is awkward
- Other functional languages have additional complexity

# Why Racket and Not X?



- Teaching functional programming in an imperative-first language is awkward
- Other functional languages have additional complexity
- It's focused on being a good language for education

# Course Structure

Grades will come from:

- Programming assignments

# Course Structure

Grades will come from:

- Programming assignments
- 4 or so projects

# Course Structure

Grades will come from:

- Programming assignments
- 4 or so projects
- An easy test and final team project

# Course Structure

Grades will come from:

- Programming assignments
- 4 or so projects
- An easy test and final team project
- *Try not to stress about grades and don't cheat*

# Course Structure

Grades will come from:

- Programming assignments
- 4 or so projects
- An easy test and final team project
- *Try not to stress about grades and don't cheat*
- I don't want to use this:

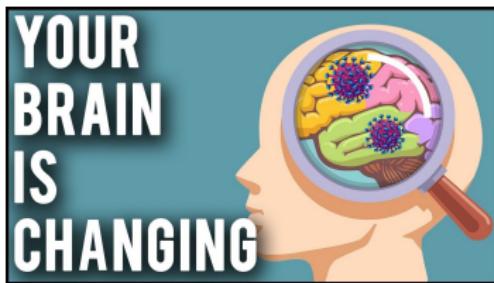
*Moss*

---

**A System for Detecting Software Similarity**

---

# The Main Things I Want



# What do you want?