Using Algebra and Video Game Development to Motivate Program Design Among Inner-City High School Students

Marco T. Morazán
Seton Hall University

TFPIE 2020

Introduction

- Teaching programming to Inner-City HS newbies (in the USA) is tough!

- Typically
  - Taken algebra
  - Do not understand what variables are
  - Do not understand what functions are
  - Only think in terms of specific values
  - Abstraction is a major problem

Introduction

- Students can do β-reduction
  - $y = x^2 + 3x - 10$
  - $x = 2 \rightarrow y = 0$
  - Learned by rote
- They cannot, however, write their own functions
  - Never mind using a PL
- Abstraction over expressions?
  - Never required to do
  - Functions always given to them

Introduction

- How do you get students that learned HS algebra by rote to take their first steps in programing?
- How do you break them away from always thinking in terms of concrete values?
- Bottom-up approach
  - Easier to write sample expressions using concrete values
  - Abstract over expressions that look similar

Related Work

- HS Algebra
  - Start with fundamental operations: + - * /
    - Apparently, these are not functions…
  - Algebraic expressions: combination of letters and numbers (huh?)
  - Functions: relation that transforms a domain element to a range element
    - No abstraction over expressions
- Our approach
  - Start with expressions using values
  - Abstract over similar expressions to create functions
  - Write tests to show that a function computes the same value as an expression using values

Related Work

- Bootstrap
  - Introduces students to function development
  - Programming used to teach/reinforce algebra
  - Abstract over similar expressions
    - Tell students what the input and output are

- Our approach
  - Abstract over similar expressions
    - Let students discover what variables are needed
  - Write tests that show expected behavior **and** how a value is computed
    - Let students write expressions using values as they are comfortable with it

Related Work

- HtDP
  - Integrates programming with HS algebra
  - Use tables to define functions
    - Requires knowing the independent and dependent variables
    - Extract the expression

- Our approach
  - Concrete expressions with concrete values
  - Force students to think about *how* to compute a value
  - Let students discover what variables are needed by a function
  - Write tests that display behavior and *how* a value is computed

Student Background

- Upperward Bound Program
  - Federal TRIO program
  - Goal: to help prepare students for higher-education
  - Reach out to students of disadvantaged backgrounds
    - low income
    - first generation
    - disabilities
  - Focus
    - Math
    - Laboratory science
    - Composition
    - Literature
    - Foreign languages
  - Programming is offered as an addition at Seton Hall University

Student Background

- Students in the summer 2019 course
  - 15 11th graders from inner-city HS
  - Average age: 16 Range: [15..18]
  - 67% females 87% African-American
  - 93% have a laptop at home
  - 60% no programming experience
    - 40%: Java or Scratch
  - All completed HS Algebra II or higher

Course Overview

- Initially
  - Primitive types
  - Function definitions
  - Conditionals
  - Compound data of finite size
  - The Design Recipe
  - Compound data of arbitrary size
  - Functional abstraction

Write a function to compute the area of a rectangle.

What do you mean write my own function?

Why are you not giving us the functions?

Where do functions come from?

I can't write something I don't understand.

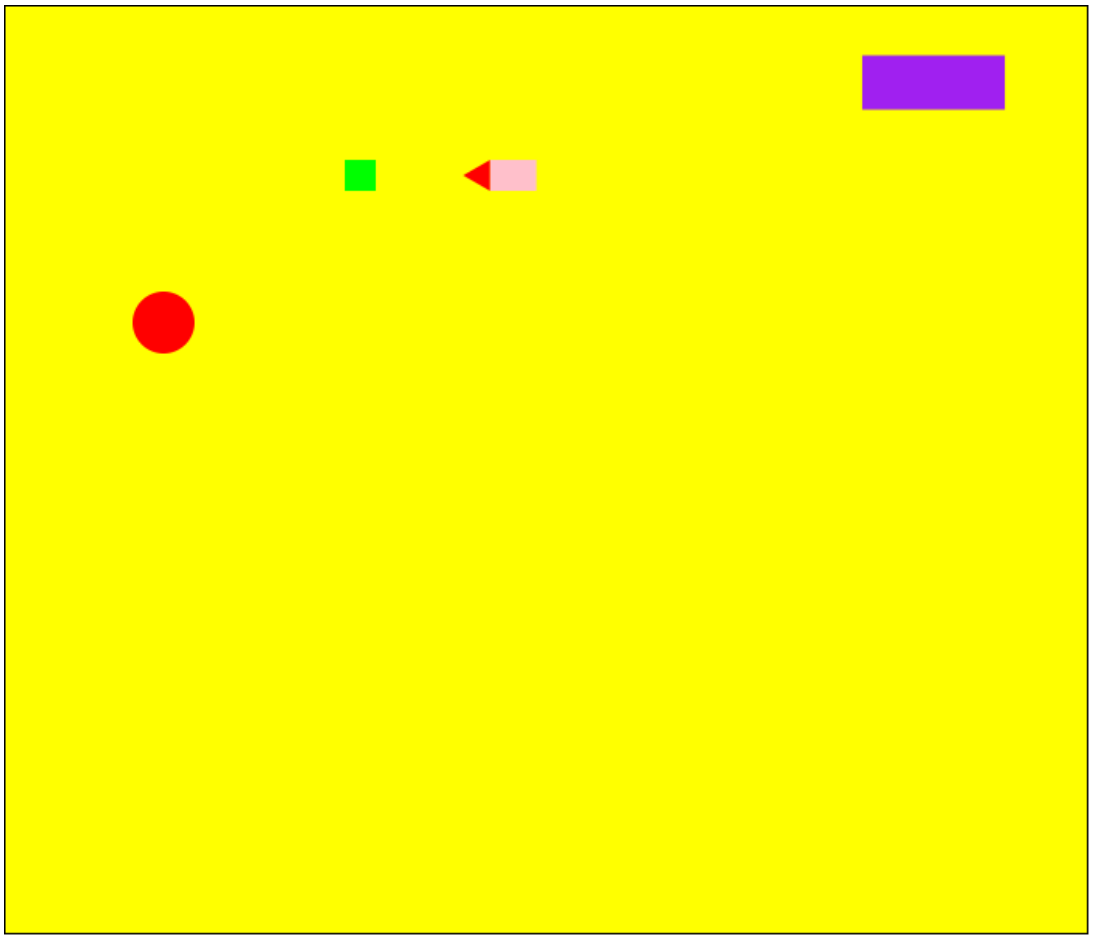
What does it mean that a function computes a value?

Course Overview

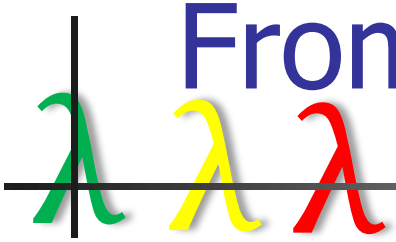
- Revised syllabus
  - Primitive types
  - Expressions with literal values
  - Abstraction over expressions
    - Function definitions
    - domain and range
  - Compound functions: Conditionals
  - The Design Recipe
  - Compound Data of Finite Size
- Goals
  - Better understanding of functions
  - Better problem solvers
  - Spark interest in programming
  - Make programming intellectually stimulating

Course Overview

- Motivation

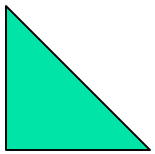


From Expressions to Functions



- Write expressions to compute the length of the hypotenuse of the following right triangles:

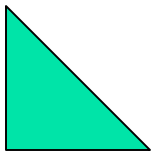
1



(sqrt (+ (sqr 1) (sqr 1)))

1

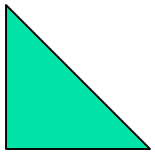
3



(sqrt (+ (sqr 3) (sqr 4)))

4

5



(sqrt (+ (sqr 5) (sqr 4)))

4

height base

Identify the differences and name them

These difference are variables

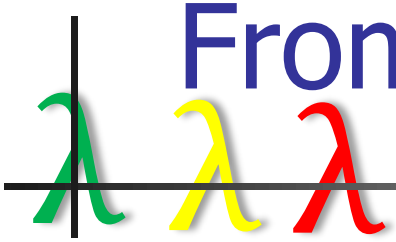
Rewrite the expressions using the vars

(sqrt (+ (sqr height) (sqr base)))

Make the variables input to a function and make the above expression its body

```
(define (hypotenue height base)
  (sqrt (+ (sqr height) (sqr base))))
```

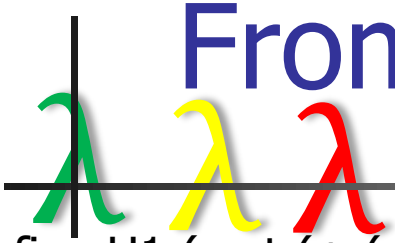
From Expressions to Functions



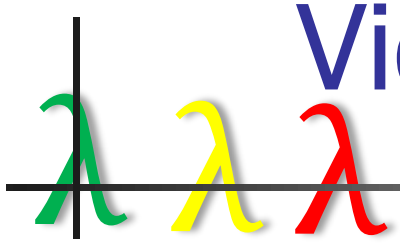
- Design Recipe

1. Define several sample expressions to compute a value
2. Identify the differences among the sample expressions
3. Give each difference a variable name
4. Identify the type of each input & the return type for the function's signature
5. Identify the purpose of the function
6. Write a function header
7. Write examples of how the function ought to work
8. Write the body of the function
9. Run the tests and redesign if necessary

From Expressions to Functions



```
(define H1 (sqrt (+ (sqr 1) (sqr 1))))
(define H2 (sqrt (+ (sqr 3) (sqr 4))))
(define H3 (sqrt (+ (sqr 5) (sqr 4))))
; illustrates how values are computed
        (check-within (hypotenuse 1 1) H1 0.1)
        (check-within (hypotenuse 3 4) H2 0.1)
        (check-within (hypotenuse 5 4) H3 0.1)
; illustrates how the function should behave
        (check-within (hypotenuse 10 10) 14.1 0.1)
        (check-within (hypotenuse 3 2) 3.6 0.1)


; ℝ>=0 ℝ>=0 → ℝ>=0
; Purpose: To compute the length of the hypotenuse of a right triangle
(define (hypotenue height base)
   (sqrt (+ (sqr height) (sqr base))))
```



Video Game Development

A world is a structure, (make-world rocket dir flevel gfuel bfuel), where

rocket is a posn

dir is either "right," "left,", "up," or "down"

flevel is $\mathbb{R}_{>=0}$

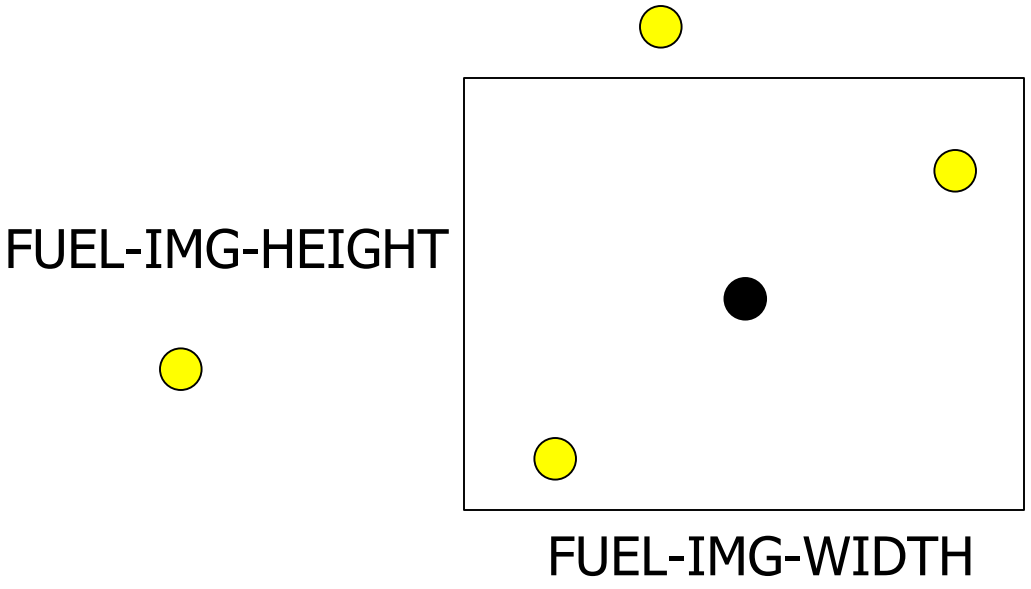
gfuel is a posn

bfuel is a posn

Video Game Development

- Relational functions
  - Similar to <, >, <=, >=
  - Return a Boolean

- Write a function to determine if a rocket has eaten a fuel

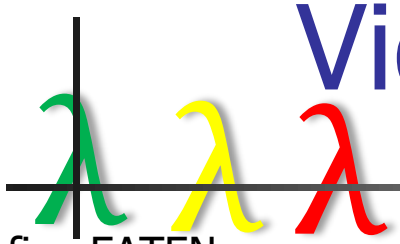


distance on x <= FUEL-IMG-WIDTH/2

and

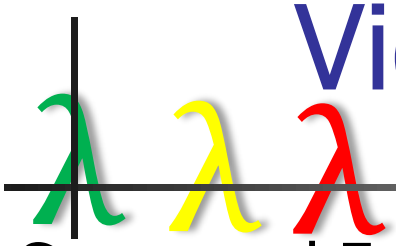
distance on y <= FUEL-IMG-HEIGHT/2

Video Game Development



```
(define EATEN
  (and (<= (distance-on-x (make-posn 100 340) (make-posn 105 335)) HALF-FUEL-IMG-WIDTH)
       (<= (distance-on-y (make-posn 100 340) (make-posn 105 335)) HALF-FUEL-IMG-HEIGHT)))
(define NOTEATEN
  (and (<= (distance-on-x (make-posn 25 10) (make-posn 500 450)) HALF-FUEL-IMG-WIDTH)
       (<= (distance-on-y (make-posn 25 10) (make-posn 500 450)) HALF-FUEL-IMG-HEIGHT)))
; Tests
(check-expect (eaten? (make-posn 100 340) (make-posn 105 335)) EATEN)
(check-expect (eaten? (make-posn 25 10) (make-posn 500 450)) NOTEATEN)
(check-expect (eaten? (make-posn 15 15) (make-posn 15 15)) #t)
(check-expect (eaten? (make-posn 250 40) (make-posn 87 21)) #f)

; rocket fuel --> Boolean
; Purpose: To determine if the given rocket has consumed the given fuel
(define (consumed? a-rocket a-fuel)
  (and (<= (distance-on-x a-rocket a-fuel) HALF-FUEL-IMG-WIDTH)
       (<= (distance-on-y a-rocket a-fuel) HALF-FUEL-IMG-HEIGHT)))
```

Video Game Development

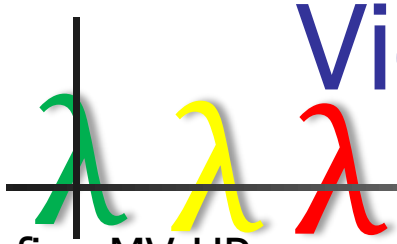


- Compound Functions
  - have more than one expression
  - used when a decision needs to be made → conditional expression

- Write a function to move a rocket in a given direction
  - class discussion → need a rocket and a direction

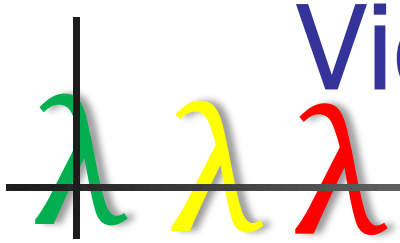
  - decision is based on the value of the given direction

Video Game Development



```
(define MV-UP
        (cond    [(string=? "up" "up") (move-rocket-up (make-posn 230 315))]
                 [(string=? "up" "down") (move-rocket-down (make-posn 230 315))]
                 [(string=? "up" "left") (move-rocket-left (make-posn 230 315))]
                 [else (move-rocket-right (make-posn 230 315))]))
(define MV-LEFT
        (cond    [(string=? "left" "up") (move-rocket-up (make-posn 10 10))]
                 [(string=? "left" "down") (move-rocket-down (make-posn 10 10))]
                 [(string=? "left" "left") (move-rocket-left (make-posn 10 10))]
                 [else (move-rocket-right (make-posn 10 10))]))

(define MV-DOWN …)

(define MV-RIGHT …)
```

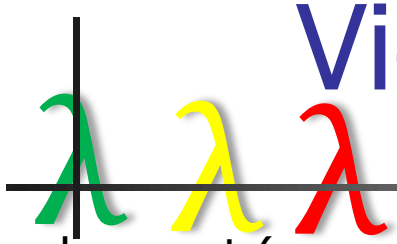
Video Game Development



```
(define MV-UP  (move-rocket-up (make-posn 230 315)))

(define MV-LEFT (move-rocket-left (make-posn 10 10)))

(define MV-DOWN ...)

(define MV-RIGHT ...)
```

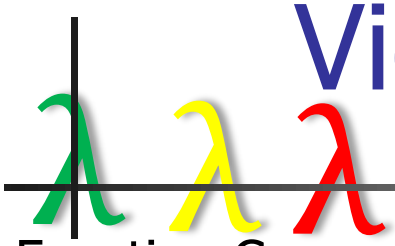
This also suggests a compound function:
Must choose among different ways to move a rocket

Video Game Development



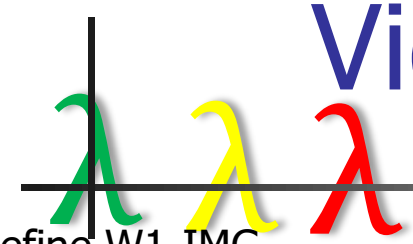
```
(check-expect (move-rocket (make-posn 230 315) "up") MV-UP)
(check-expect (move-rocket (make-posn 50 20) "down") MV-DOWN)
(check-expect (move-rocket (make-posn 98 98) "left") MV-LEFT)
(check-expect (move-rocket (make-posn 420 250) "right") MV-RIGHT)
(check-expect (move-rocket (make-posn 10 10) "right") (make-posn 15 10))
(check-expect (move-rocket (make-posn 5 70) "down") (make-posn 5 75))

; rocket direction → rocket
; Purpose: To move the given rocket in the given direction
(define (move-rocket a-rocket a-dir)
        (cond   [(string=? a-dir "up") (move-rocket-up a-rocket)]
                [(string=? a-dir "down") (move-rocket-down a-rocket)]
                [(string=? a-dir "left") (move-rocket-left a-rocket)]
                [else (move-rocket-right a-rocket)]))
```

Video Game Development



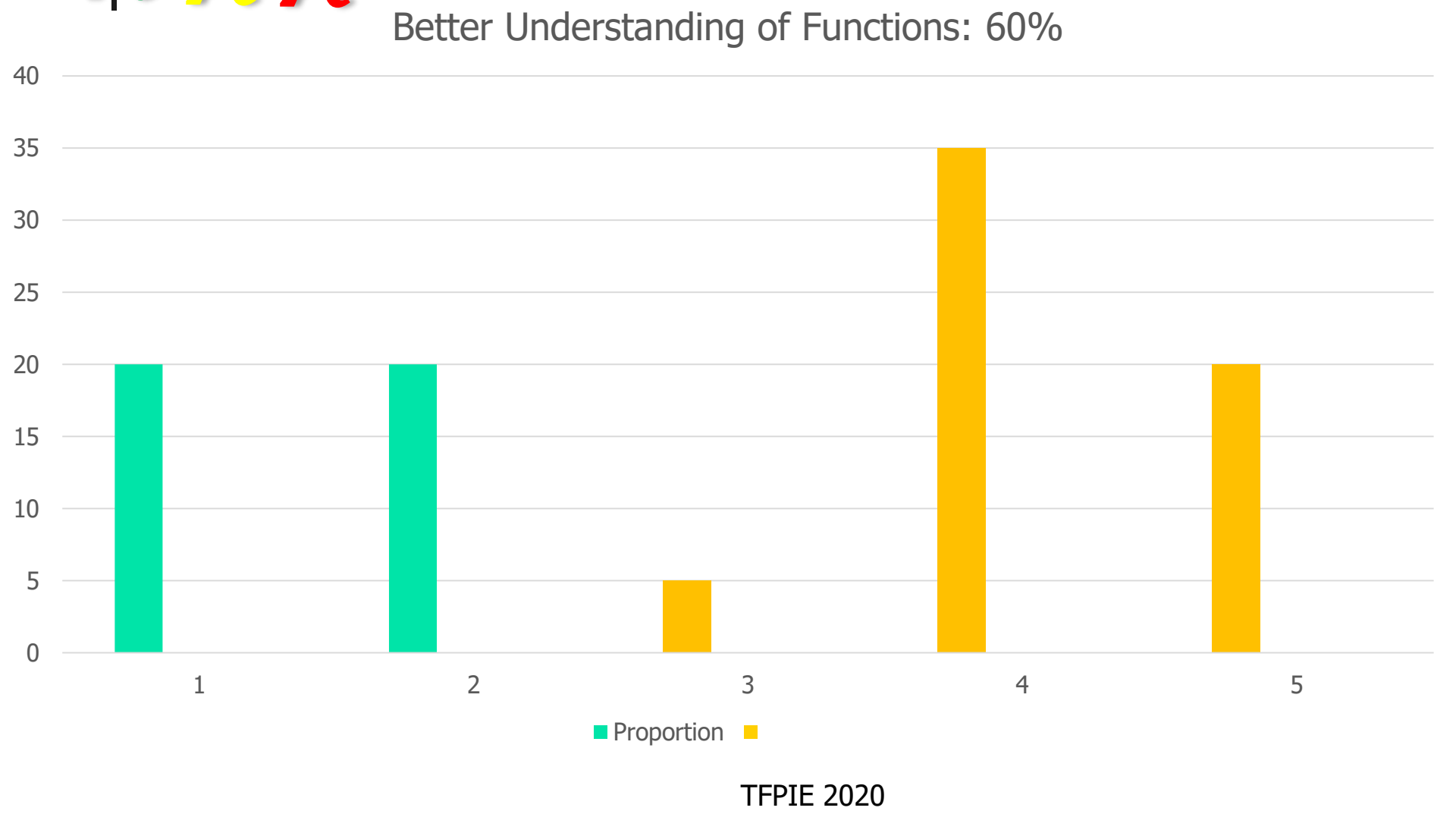
- Function Composition
  - (f ∘ g)(x) = f(g(x))
  - May require some explanation: not multiplying
  - Class discussion: output of g is the input for f
  - g computes a specialized value needed by f
- Design a function to render an image of the world
  - they already wrote functions to draw individual world components

Video Game Development

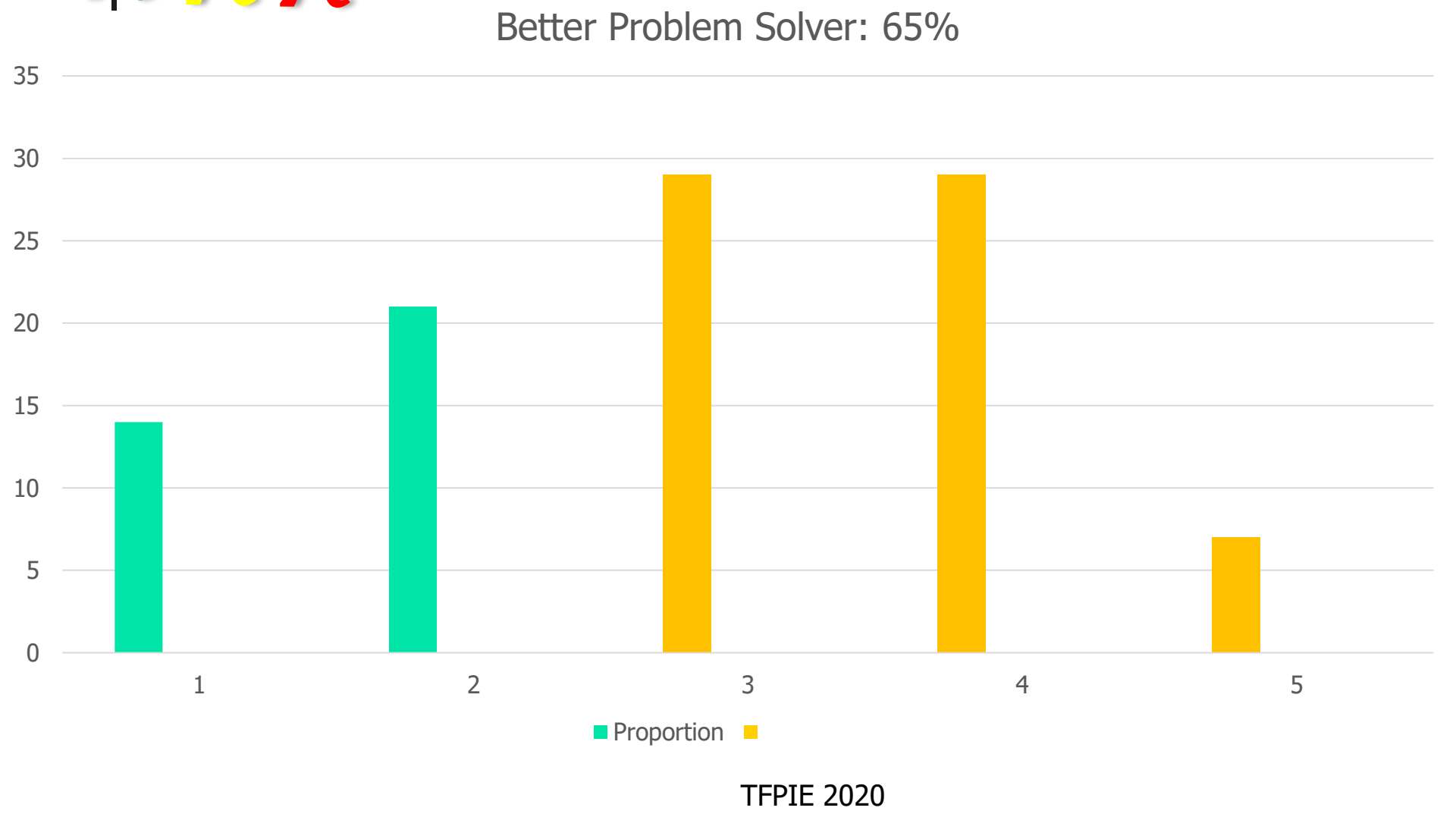


```
(define W1-IMG
  (draw-bfuel  (world-bfuel  (world-bfuel W1))
               (draw-gfuel (world-gfuel W1)
                           (draw-flevel (world-flevel W1)
                                        (draw-rocket (world-rocket W1) BACK-IMG)))))
(define W2-IMG
  (draw-bfuel (world-bfuel W2)
              (draw-gfuel (world-gfuel W2)
                          (draw-flevel (world-flevel W2)
                                       (draw-rocket (world-rocket W2) BACK-IMG)))))
(check-expect (draw-world W1) W1-IMG)
(check-expect (draw-world W2) W2-IMG)
; world → image                ; Purpose: To draw the given world in the background image
(define (draw-world a-world)
  (draw-bfuel  (world-bfuel a-world)
               (draw-gfuel (world-gfuel a-world)
                           (draw-flevel (world-flevel a-world) (draw-rocket a-world BACK-IMG)))))
```

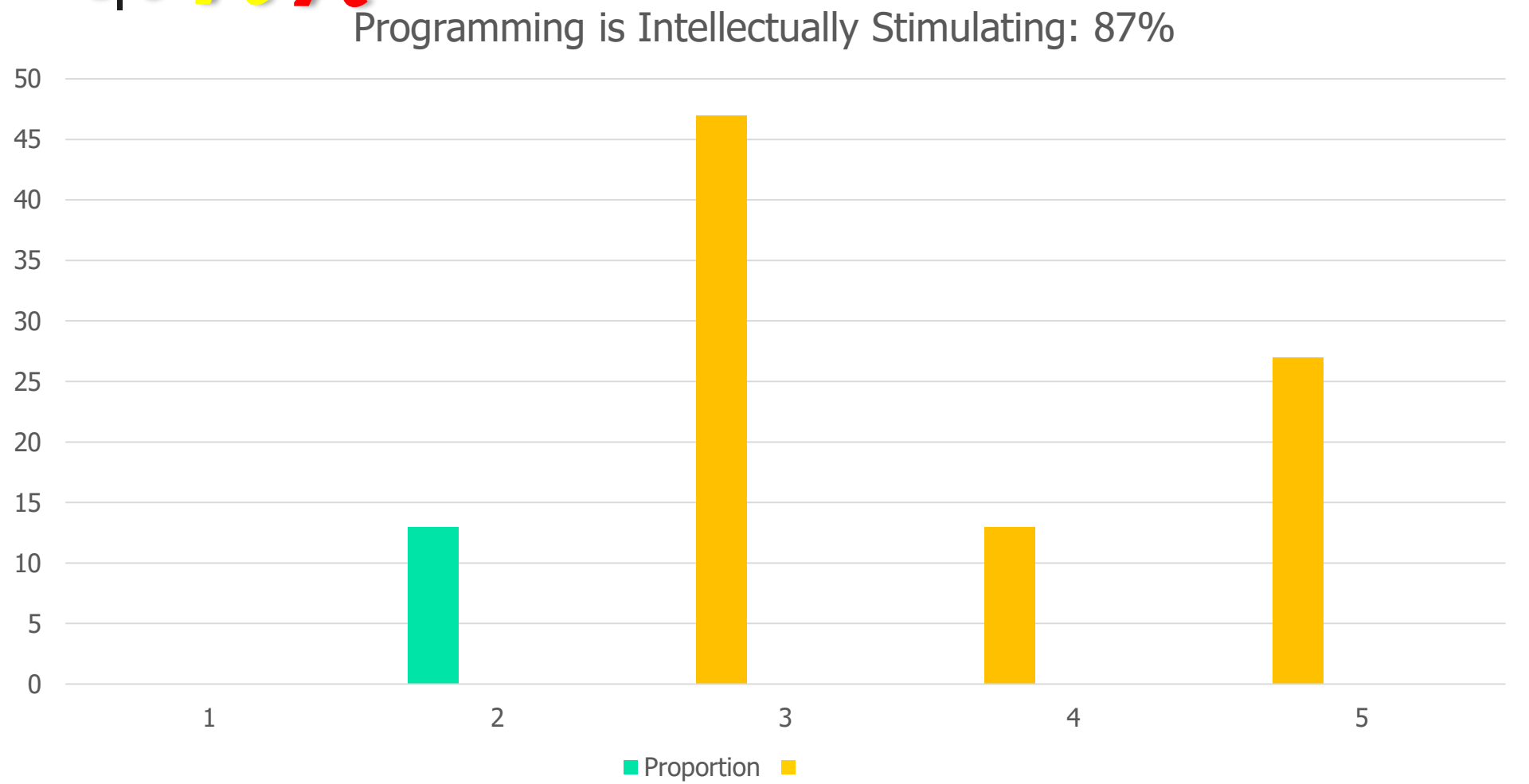
Student Feedback



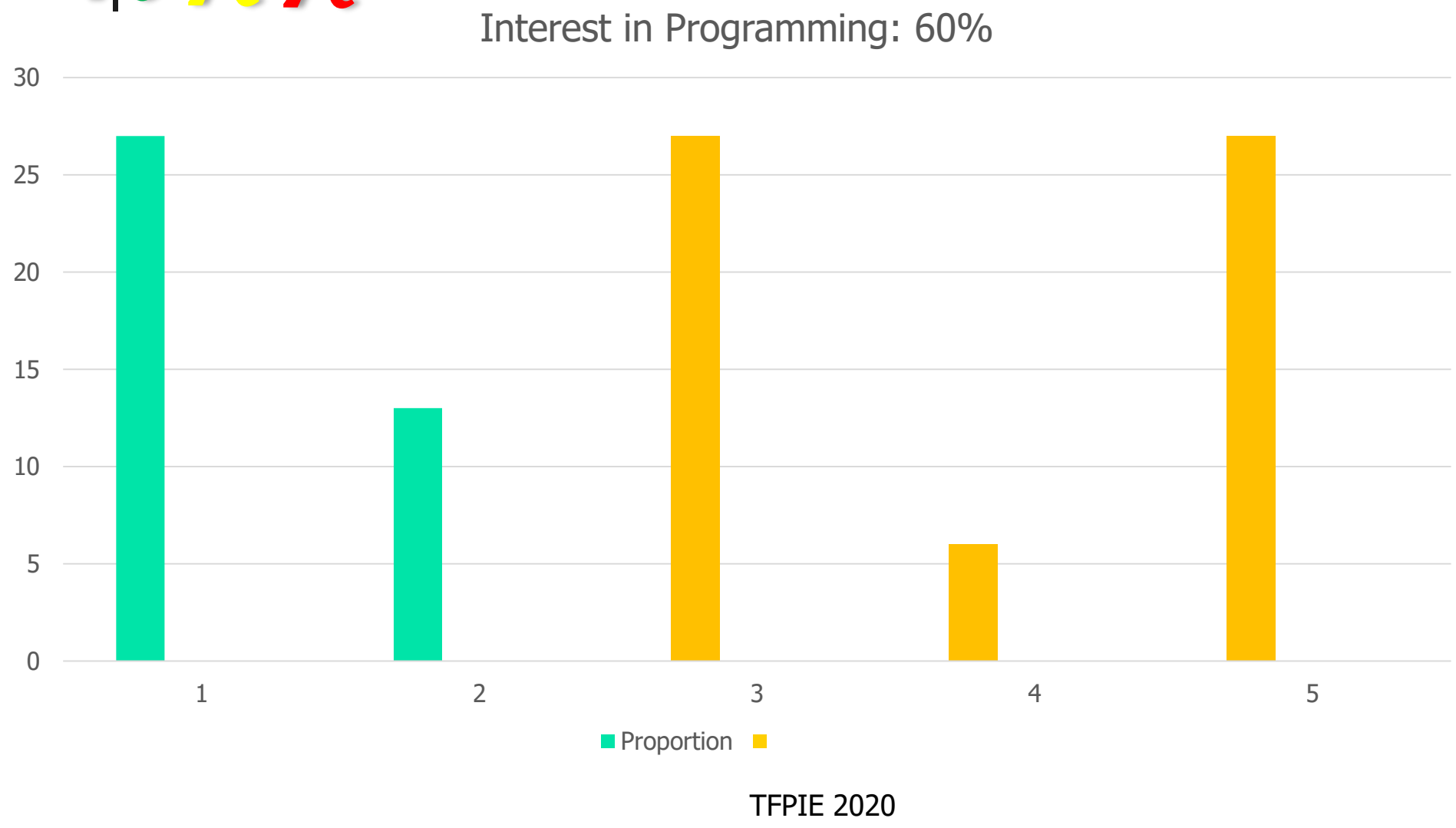
Student Feedback



Student Feedback



Student Feedback



Student Feedback

- The Complaints
  - Too much typing
    - 33% of students
    - Video game < 300 LOC including comments and tests
  - Too much work
    - Small proportion
    - Never imagined writing a video game was so much work
- Most Liked
  - Designing rockets and being creative
  - Creating the game made me think
  - Problem Solving
- Overall Feelings
  - It was great…might pursue in college
  - Interesting and liked coding in general
  - Did not like the course, because I am not at all interested in programming
  - A lot of typing, but very successful

Concluding Remarks

- Relate programming to what students have studied
- Captivate their imagination
- Make them feel that what they have studied is relevant
  - expressions with concrete values → abstraction → functions
  - relational functions
  - compound functions
  - function composition

Thanks to: Marva M. Cole, Abena A. Douglas, UBP
Sachin Mahashabde, Jeremy Y. Suero, TAs

Any Questions? ☺

- Future work
  - Introduce these techniques to CS1
  - Linchpin for poorly prepared students?