CMPU 101 §02 · Computer Science I

Evaluating Functions and Conditionals

7 September 2022

Assignment 1 due 11:59pm on Wednesday

Assignment 2 comes out on Thursday

Where are we?

We've been using Pyret to write expressions that use:

Data, including

numbers (`0`, `-10`, `0.4`),

strings (`""`, `"hi"`, `"111"`), and

images (`circle(2, "solid", "red")`).

Which we modify or combine using operators or functions like **`+`**, **`string-append`**, and **`above`**.

Distinguishing types of data helps to catch mistakes.

If you try to give

a string to `/` or

a number to **`overlay`**,

we want Pyret to catch the problem right early,
giving a helpful error message.

We've seen that we can create more complicated programs by composing function calls, e.g.,

```
1 + (2 / 3)
```

or

```
string-append("hello ",
  string-append("Pyret ", "world!"))
```

And we can give a name to the result of an expression, e.g.,

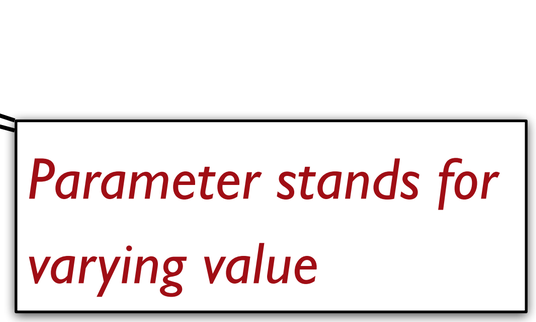
```
total = 2 + 3
```

Defining functions

Remember functions from middle-school math:

Given $f(x) = \cos(x) + 2$

$f(0) = 1 + 2 = 3$



Pyret functions work the same way:

```
fun f(x): num-cos(x) + 2 end

f(0)
→ num-cos(0) + 2
→ 1 + 2
→ 3
```

Function definitions in Pyret have this form:

```
fun ⟨function-name⟩ (⟨arg-name⟩, ...):
  ⟨expression⟩
end
```

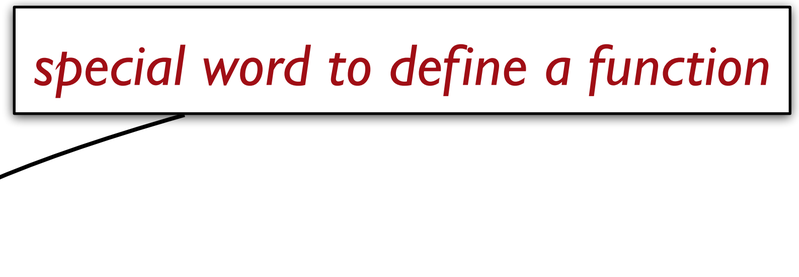
Example

Mary Berry needs to know how many cakes to bake for her cake shop.

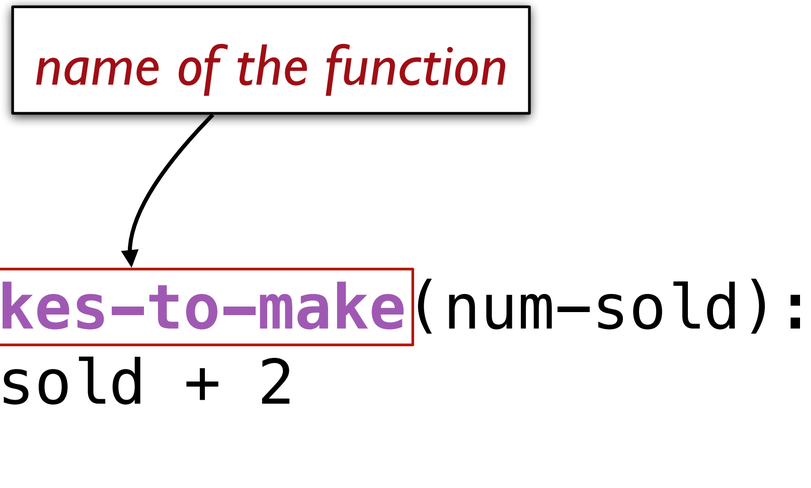
To avoid running out or having too many, she likes to bake two cakes more than the number she sold the previous day.

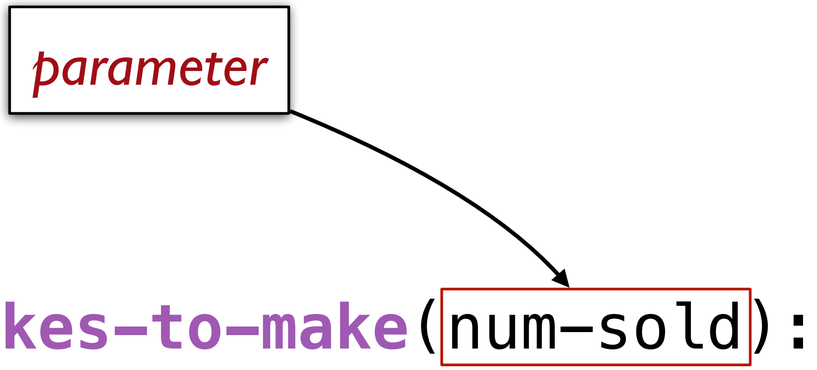
E.g., if Mary sells eight cakes on Monday, she makes ten cakes on Tuesday.

Let’s write some code to help Mary.



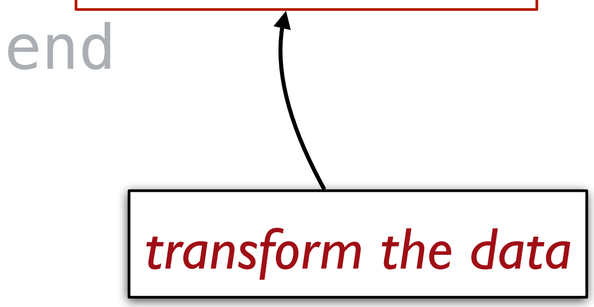

```
fun cakes-to-make(num-sold):
  num-sold + 2
end
```


name of the function
fun cakes-to-make(num-sold):
num-sold + 2
end

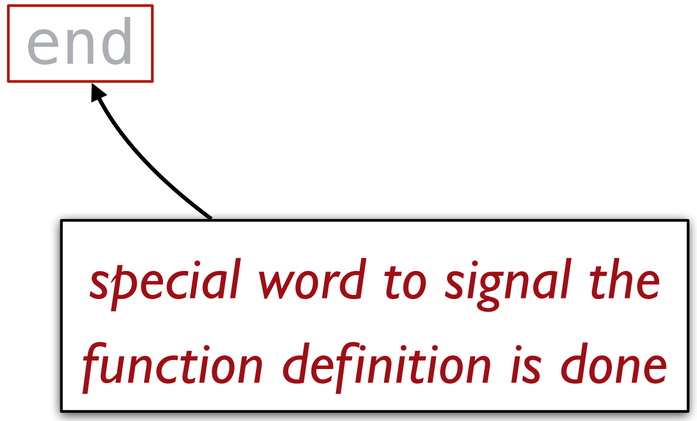



```
fun cakes-to-make(num-sold):
  num-sold + 2
end
```

```
fun cakes-to-make(num-sold):
  num-sold + 2
end
```



```
fun cakes-to-make(num-sold):
  num-sold + 2
end
```

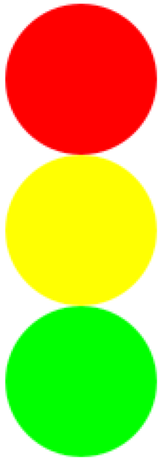


Functional abstraction

```
# Draw a traffic light
above(  circle(40, "solid", "red"),
  above(circle(40, "solid", "yellow"),
        circle(40, "solid", "green"))
```

Unchanging

Varying



```
# Draw a traffic light
above(circle(40, "solid", "red"),
  above(circle(40, "solid", "yellow"),
    circle(40, "solid", "green")))

# Can be changed to
fun bulb(color):
  circle(40, "solid", color)
end

above(bulb("red"),
  above(bulb("yellow"),
    bulb("green")))
```

```
fun bulb(color):
  circle(40, "solid", color)
end

fun traffic-light():
  above(bulb("red"),
    above(bulb("yellow"),
      bulb("green")))
end
```

Example

For Mary's cake shop, we want to determine the price of each cake based on the cost of the ingredients and the time to prepare it.

The price is twice the cost of the ingredients plus 1/4 of the preparation time in minutes.

Chocolate cake
Ingredients: $10
Preparation time: 20 minutes

```
choc-cake-price = (2 * 10) + (0.25 * 20)
```

Cheesecake
Ingredients: $15
Preparation time: 36 minutes

```
cheesecake-price = (2 * 15) + (0.25 * 36)
```

We use functions to avoid repetitive code when we need to perform the same operations on different values.

```
choc-cake-price = (2 * 10) + (0.25 * 20)
```

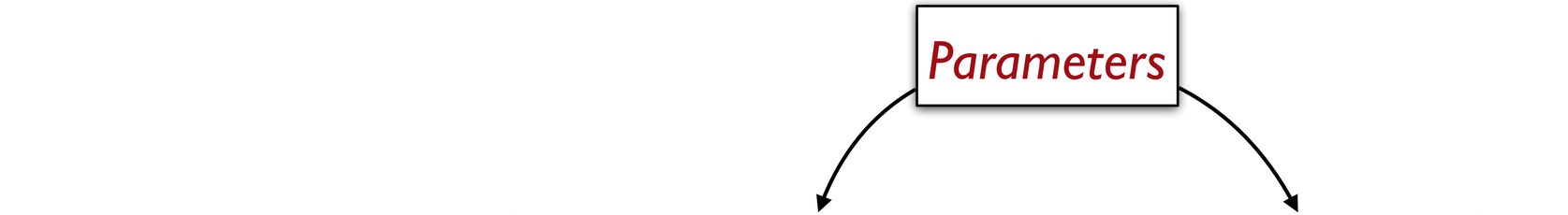
```
cheesecake-price = (2 * 15) + (0.25 * 36)
```

```
(2 * ingredients-cost) + (0.25 * prep-time)
```

We use functions to avoid repetitive code when we need to perform the same operations on different values.

```
choc-cake-price = (2 * 10) + (0.25 * 20)
```

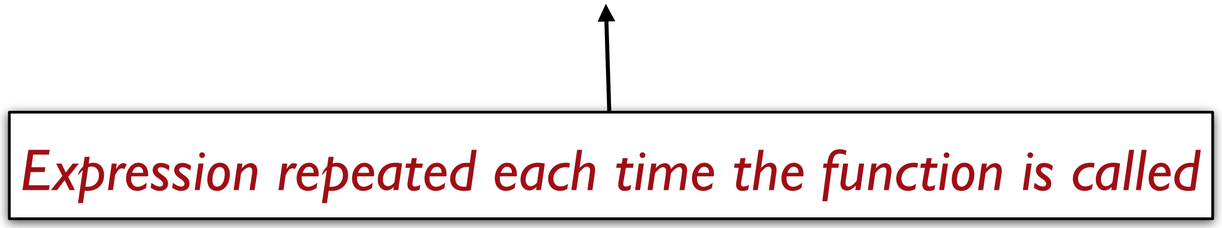
```
cheesecake-price = (2 * 15) + (0.25 * 36)
```




```
fun cake-price(ingredients-cost, prep-time):
  (2 * ingredients-cost) + (0.25 * prep-time)
end
```

The parameters are the values passed into the function that it needs to know for each operation.

```
fun cake-price(ingredients-cost, prep-time):
  (2 * ingredients-cost) + (0.25 * prep-time)
end
```



```
fun cake-price(ingredients-cost, prep-time):
  (2 * ingredients-cost) + (0.25 * prep-time)
end
```

To calculate the price of chocolate cake or cheesecake, you simply call your function and pass in the relevant values:

```
# Price of chocolate cake
cake-price(10, 20)

# Price of cheesecake
cake-price(15, 36)
```

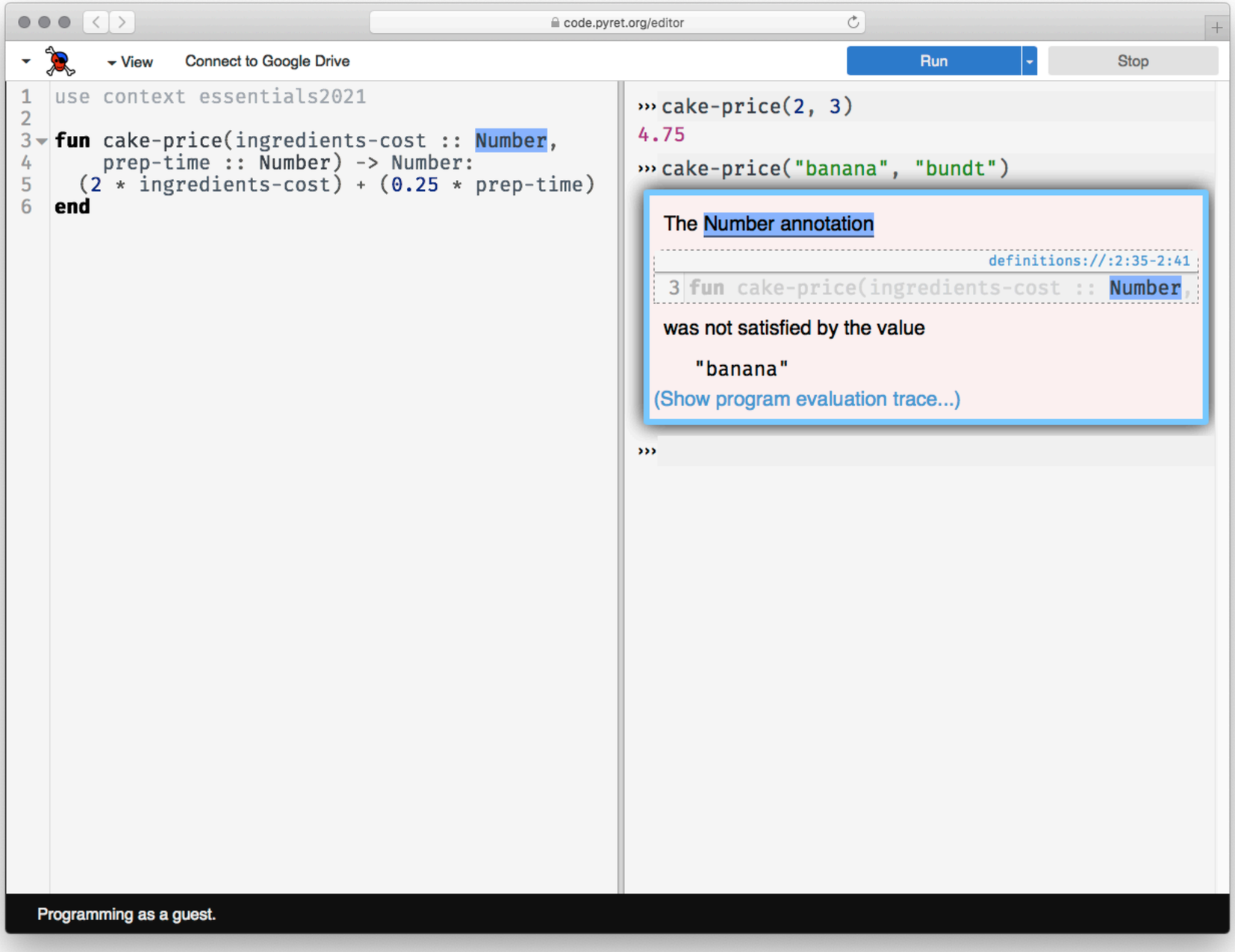
Improving our function definitions

```
fun cake-price(ingredients-cost :: Number,
    prep-time :: Number):
  (2 * ingredients-cost) + (0.25 * prep-time)
end
```

We specify the type of each parameter so that Pyret will check that we pass in the right kind of values, just like for built-in operations like **+** *and* ***above****.*

```
fun cake-price(ingredients-cost :: Number,
    prep-time :: Number) -> Number:
  (2 * ingredients-cost) + (0.25 * prep-time)
end
```

And we can specify the type of value the function returns.


code.pyret.org/editor
View
Connect to Google Drive
Run
Stop
1 use context essentials2021
2
3 fun cake-price(ingredients-cost :: Number,
4 prep-time :: Number) -> Number:
5 (2 * ingredients-cost) + (0.25 * prep-time)
6 end
››› cake-price(2, 3)
4.75
››› cake-price("banana", "bundt")
The Number annotation
definitions://:2:35-2:41
3 fun cake-price(ingredients-cost :: Number,
was not satisfied by the value
"banana"
(Show program evaluation trace...)
›››
Programming as a guest.

```
fun cake-price(ingredients-cost :: Number,
    prep-time :: Number) -> Number:
  doc: "Calculate price of cake based on
ingredient cost and prep time"
  (2 * ingredients-cost) + (0.25 * prep-time)
end
```

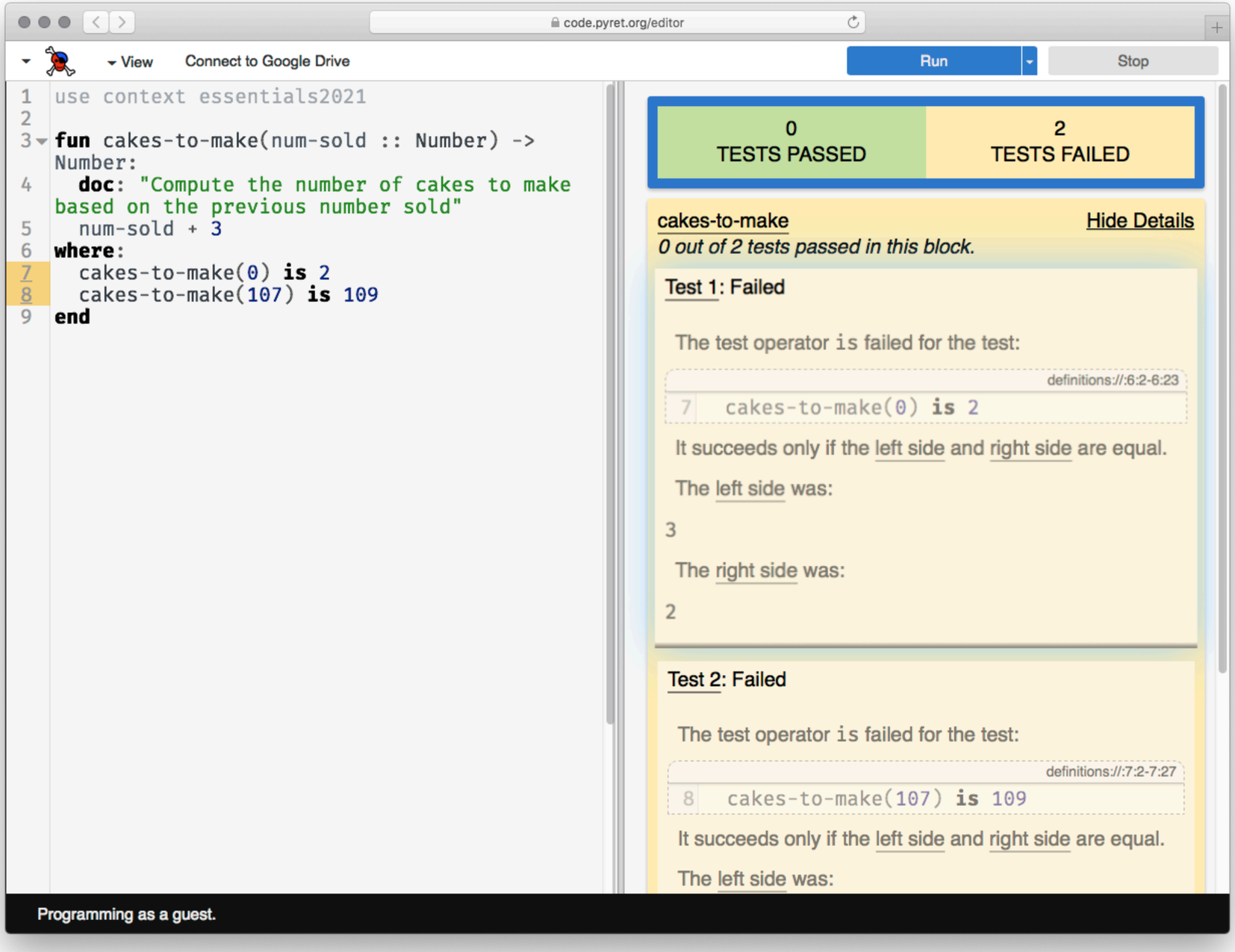
Additionally, a docstring explains what the function does.

“Programs must be written for people to read, and only incidentally for machines to execute.”

Hal Abelson & Gerald Sussman with Julie Sussman, *Structure and Interpretation of Computer Programs*, 1979

```
fun cakes-to-make(num-sold :: Number) -> Number:
  doc: "Compute the number of cakes to make based on
the previous number sold"
  num-sold + 2
end
```

```
fun cakes-to-make(num-sold :: Number) -> Number:
  doc: "Compute the number of cakes to make based on
the previous number sold"
  num-sold + 2
where:
  cakes-to-make(0) is 2
  cakes-to-make(107) is 109
end
```


code.pyret.org/editor
View
Connect to Google Drive
Run
Stop
use context essentials2021
fun cakes-to-make(num-sold :: Number) -> Number:
doc: "Compute the number of cakes to make based on the previous number sold"
num-sold + 3
where:
cakes-to-make(0) is 2
cakes-to-make(107) is 109
end
0
TESTS PASSED
2
TESTS FAILED
cakes-to-make
Hide Details
0 out of 2 tests passed in this block.
Test 1: Failed
The test operator is failed for the test:
definitions://:6:2-6:23
7 cakes-to-make(0) is 2
It succeeds only if the left side and right side are equal.
The left side was:
3
The right side was:
2
Test 2: Failed
The test operator is failed for the test:
definitions://:7:2-7:27
8 cakes-to-make(107) is 109
It succeeds only if the left side and right side are equal.
The left side was:
Programming as a guest.

www.amazon.com/s?k=jane+austen&rh=n%3A283155&dc&qid=1599505617&rnid=29411

amazon | Books | jane austen | Hello, Sign in Account & Lists | Returns & Orders | Try Prime | 0 Cart

Hello Select your address | Best Sellers | Customer Service | New Releases | Today's Deals | Find a Gift | Whole Foods | Gift Cards | Free Shipping | Registry | Sell | Coupons | AmazonBasics | #FoundItOnAmazon | Shopper Toolkit

Books | Advanced Search | New Releases | Best Sellers & More | Children's Books | Textbooks | Textbook Rentals | Best Books of the Month

1-16 of over 20,000 results for **Books** : **"jane austen"** | Sort by: Featured

Eligible for Free Shipping
- [ ] Free Shipping by Amazon

All customers get FREE Shipping on orders over $25 shipped by Amazon

Kindle Unlimited
- [ ] Kindle Unlimited Eligible

Department
- Any Department
  - **Books**
    - Literature & Fiction
    - Romance

Avg. Customer Review
- ★★★★☆ & Up
- ★★★☆☆ & Up
- ★★☆☆☆ & Up
- ★☆☆☆☆ & Up

Book Series
- [ ] Penguin Classics
- [ ] The Cousins
- [ ] The Isle of Destiny Series
- [ ] Pemberley Estates
- [ ] Kiss the Wallflower
- [ ] Regency Love
- [ ] The Rules of Love
- [ ] The Smoky Hills Academy
- [ ] Ladies of Bath
- [ ] The Heirloom Collection

See more

Deals
- [ ] Today's Deals

New Releases
- Last 30 days
- Last 90 days
- Coming Soon

Book Format
- Paperback
- Hardcover
- Kindle Edition
- Large Print
- Audible Audiobook
- Audio CD
- Board Book

Author
- Jane Austen

The Jane Austen Collection: Deluxe 6-Volume Slipcase Edition (Arcturus Collector's Classics)

by Jane Austen | Sep 15, 2016

★★★★½ 274

Hardcover

$39.49 ~~$59.95~~

prime Get it as soon as **Wed, Sep 9**
FREE Shipping by Amazon

More Buying Choices
$37.60 (32 used & new offers)

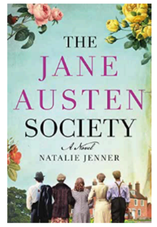
Kindle

$0.49 ~~$12.99~~

Available instantly

Paperback

Other format: Audio CD



The Jane Austen Society: A Novel

by Natalie Jenner | May 26, 2020

★★★★½ 781

Hardcover

$16.19 ~~$26.99~~

Get it as soon as **Wed, Sep 9**
FREE Shipping on your first order shipped by Amazon

More Buying Choices
$13.49 (44 used & new offers)

Kindle

$13.99 ~~$26.99~~

Available instantly

Audible Audiobook

$0.00 ~~$27.99~~

Free with Audible trial

Other formats: Audio CD , Paperback

Jane Was Here: An Illustrated Guide to Jane Austen's England

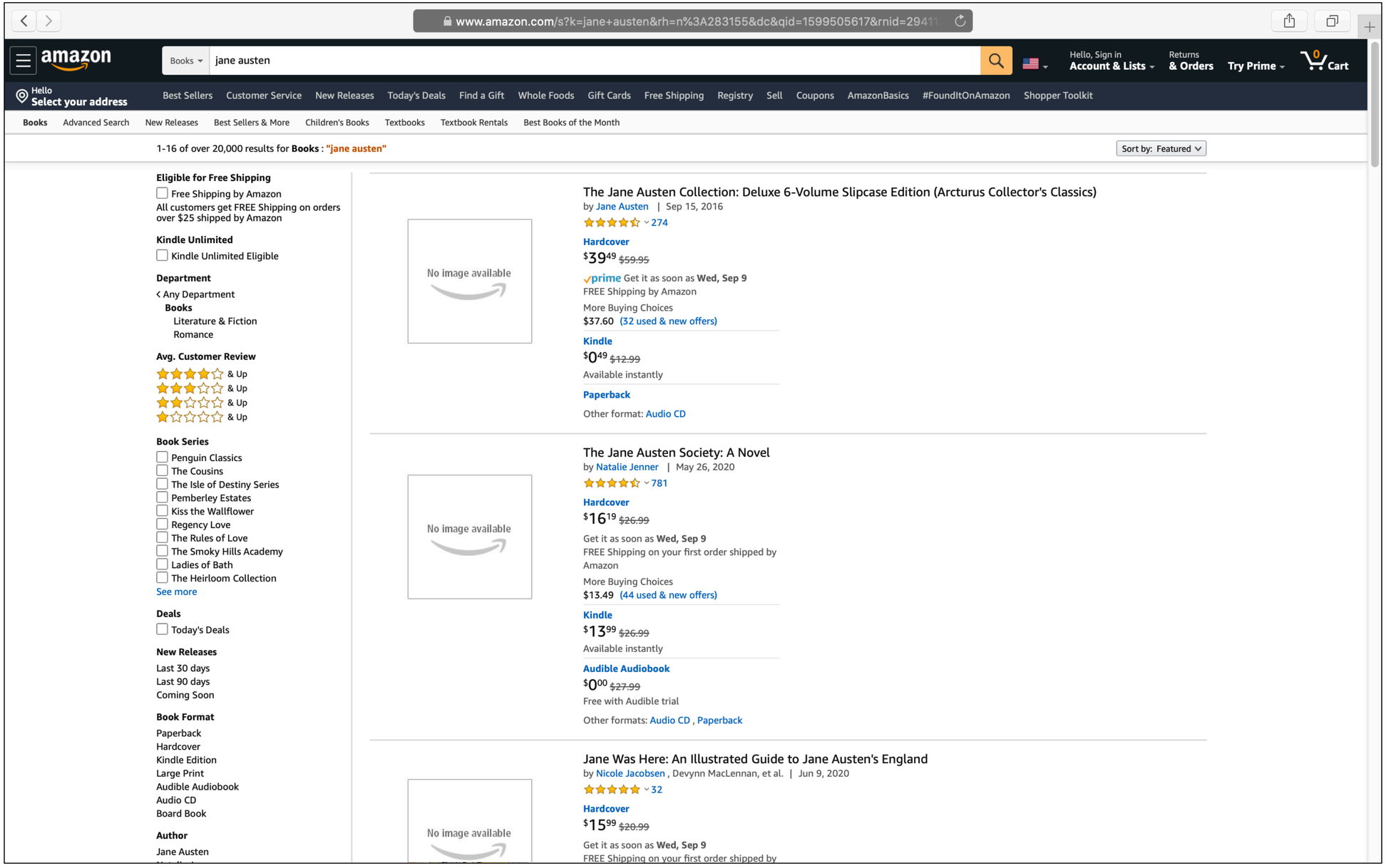
by Nicole Jacobsen , Devynn MacLennan, et al. | Jun 9, 2020

★★★★★ 32

Hardcover

$15.99 ~~$20.99~~

Get it as soon as **Wed, Sep 9**
FREE Shipping on your first order shipped by


www.amazon.com/s?k=jane+austen&rh=n%3A283155&dc&qid=1599505617&rnid=29411
amazon
Books
jane austen
Hello, Sign in
Account & Lists
Returns
& Orders
Try Prime
Cart
Hello
Select your address
Best Sellers
Customer Service
New Releases
Today's Deals
Find a Gift
Whole Foods
Gift Cards
Free Shipping
Registry
Sell
Coupons
AmazonBasics
#FoundItOnAmazon
Shopper Toolkit
Books
Advanced Search
New Releases
Best Sellers & More
Children's Books
Textbooks
Textbook Rentals
Best Books of the Month
1-16 of over 20,000 results for Books : "jane austen"
Sort by: Featured
Eligible for Free Shipping
Free Shipping by Amazon
All customers get FREE Shipping on orders over $25 shipped by Amazon
Kindle Unlimited
Kindle Unlimited Eligible
Department
Any Department
Books
Literature & Fiction
Romance
Avg. Customer Review
& Up
& Up
& Up
& Up
Book Series
Penguin Classics
The Cousins
The Isle of Destiny Series
Pemberley Estates
Kiss the Wallflower
Regency Love
The Rules of Love
The Smoky Hills Academy
Ladies of Bath
The Heirloom Collection
See more
Deals
Today's Deals
New Releases
Last 30 days
Last 90 days
Coming Soon
Book Format
Paperback
Hardcover
Kindle Edition
Large Print
Audible Audiobook
Audio CD
Board Book
Author
Jane Austen
No image available
The Jane Austen Collection: Deluxe 6-Volume Slipcase Edition (Arcturus Collector's Classics)
by Jane Austen | Sep 15, 2016
274
Hardcover
$39.49 $59.95
prime Get it as soon as Wed, Sep 9
FREE Shipping by Amazon
More Buying Choices
$37.60 (32 used & new offers)
Kindle
$0.49 $12.99
Available instantly
Paperback
Other format: Audio CD
No image available
The Jane Austen Society: A Novel
by Natalie Jenner | May 26, 2020
781
Hardcover
$16.19 $26.99
Get it as soon as Wed, Sep 9
FREE Shipping on your first order shipped by Amazon
More Buying Choices
$13.49 (44 used & new offers)
Kindle
$13.99 $26.99
Available instantly
Audible Audiobook
$0.00 $27.99
Free with Audible trial
Other formats: Audio CD , Paperback
No image available
Jane Was Here: An Illustrated Guide to Jane Austen's England
by Nicole Jacobsen , Devynn MacLennan, et al. | Jun 9, 2020
32
Hardcover
$15.99 $20.99
Get it as soon as Wed, Sep 9
FREE Shipping on your first order shipped by

```
fun rectangle-area(r):
  image-height(r) * image-width(r)
end
```

```
fun rectangle-area(r :: Image) -> Number:
  doc: "Return the rectangular area of the image"
  image-height(r) * image-width(r)
where:
  rectangle-area(rectangle(0, 0, "solid", "black"))
    is 0
  rectangle-area(rectangle(2, 3, "outline", "blue"))
    is 6
end
```

Booleans and `if` expressions

```
true
false
```

To combine Boolean values, we can use **and**:

⟨*expression 1*⟩ `and` ⟨*expression 2*⟩

and **`or`**:

⟨*expression 1*⟩ `or` ⟨*expression 2*⟩

Evaluation of **`and`** stops – is "short-circuited" – as soon as one of the expressions being combined evaluates to `false`.

Evaluation of **`or`** stops as soon as one of the expressions evaluates to `true`.

```
>>> true and false
false
>>> true or false
true
>>> (1 < 2) and (2 > 3)
false
>>> (1 <= 0) or (1 == 1)
true
```

To change an expression that evaluates to `true` to be `false` or vice versa, use **`not`**:

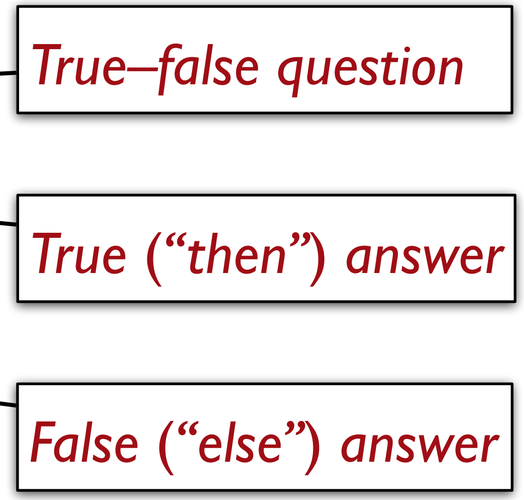
```
>>> not(1 == 0)
true
```

```
i1 = rectangle(10, 20, "solid", "red")
i2 = rectangle(20, 10, "solid", "blue")

image-width(i1) < image-width(i2)
```

```
rect = rectangle(10, 20, "solid", "red")

if image-width(rect) < image-height(rect):
  "tall"
else:
  "wide"
end
```

To form an **`if`** expression:

```
if ⟨expression⟩:
   ⟨expression⟩
else:
   ⟨expression⟩
end
```



Evaluation rule for `if` expressions

1 If the question expression is not a value, evaluate it, and replace with value.

2 If the question is `true`, replace entire **`if`** expression with true answer expression.

3 If the question is `false`, replace entire **`if`** expression with false answer expression.

4 If the question is a value other than true or false, so produce an error.

```
rect = rectangle(10, 20, "solid", "red")

if image-width(rect) < image-height(rect):
  "tall"
else:
  "wide"
end
```

What if, instead of producing a Boolean to say if an image is tall or not, we classify them as "tall", "square", or "wide"?

```
rect = rectangle(10, 20, "solid", "red")

if image-width(rect) < image-height(rect):
  "tall"
else if image-width(rect) == image-height(rect):
  "square"
else:
  "wide"
end
```

```
rect = rectangle(10, 20, "solid", "red")

fun image-type(img :: Image) -> String:
  doc: "Classify an image as tall, square, or wide"
  if image-width(img) < image-height(img):
    "tall"
  else if image-width(img) == image-height(img):
    "square"
  else:
    "wide"
  end
where:
  image-type(rect) is "tall"
end
```

```
rect = rectangle(10, 20, "solid", "red")

fun image-type(img :: Image) -> String:
  doc: "Classify an image as tall, square, or wide"
  if image-width(img) < image-height(img):
    "tall"
  else if image-width(img) == image-height(img):
    "square"
  else:
    "wide"
  end
where:
  image-type(rect) is "tall"
  image-type(rectangle(10, 10, "solid", "blue")) is "square"
  image-type(rectangle(20, 10, "solid", "blue")) is "wide"
end
```

Acknowledgments

This class incorporates material from:

Kathi Fisler, Brown University

Gregor Kiczales, University of British Columbia

Jonathan Gordon, Vassar College