In lecture we've been using the Interactions Pane. It evaluates one expression at a time, immediately after you enter it.

I'm typing this whole document in the Definitions Pane.

DrRacket doesn't evaluate the expressions in the Definitions Pane until you ask it to, by clicking the "Run" button (or using the "Run" menu item, or the corresponding short-cut key).

When you "run" the document, DrRacket evaluates all the expressions one at a time, in order, but not pausing for interaction in-between, just showing the value of each expression. It still uses the Interactions Pane as the place to show the value of each expression.

A semi-colon introduces a "comment": something for the human reader, that DrRacket skips when you run the document. Everything from a semi-colon up to the end of the line containing the semi-colon is skipped.

Here's a simple expression, just an image pasted in:

Let's cut it up into the three pieces suggested by the lines in the image, just for fun (to practice programming).

To get access to all the image functions we've been using, we need to say: (require picturing-programs)

Now, let's see how tall the image is, in pixels (dots).

I ran the document so far, and looked at the Interactions Pane to find out the result is 160.
The rest of the document was produce by that process (I ran part of it, added more expressions based on the result, ran it again, etc).

The place to first cut looks about half-way down: 80 pixels.
There's a function crop-top that takes an image and number of pixels.
I hadn't memorized whether it:
1. Crops the picture to produce a top part, or crops off/away a top part to produce a bottom part.
2. Expects the image and number of pixels in that order.

Try it with the number first and image second: read the error message, it's very informative and helpful.

Here's the result, pasted in from the Interactions Pane:

An image itself is an expression, producing itself, so now if you run this document you'll actually see the image twice in the Interactions Pane.

That cropped away the top half, but not quite enough. Playing around it turns out that removing 84 pixels looks good.

The result:
To calculate 3/5 of 450 we multiply:

\[
\frac{3}{5} \times 450 = \text{result}
\]

The result is 76. Using crop-bottom:

\[
\text{crop-bottom}
\]

I put that expression on two lines, but it didn't have to be.

Notice that DrRacket indents the second argument when you press return,

emphasizing it's "inside" the expression.

"Words" (including single images and single numbers) and clauses (compound
expressions, which are parenthesized and have the verb/function/operation
first) can have more than one space between them and continue across multiple
lines. But please use single spacing within lines as in English.

When writing an expression across multiple lines, DrRacket indents automatically
when you press enter/return. Leave it that way. If you mess up the indentation
after use the menu item Racket / Reindent All to have DrRacket fix it.

The result of the above expression is:

\[
\text{image-width}
\]

It's width in pixels is:

\[
\text{image-width}
\]

The result is 450.

The left part appears to be about 3/5 of the image.

To calculate 3/5 of 450 we multiply:

\[
\left(\frac{3}{5}\right) \times 450 = \text{result}
\]

The result is 270, so:
; Playing around, 275 gets a better result:

; Result:

Leprechaun

; For the left side, let's demonstrate that where a value is expected,
; any expression producing that value can be used:

; Result:

; For more fun programming, let's put the pieces back together,
; with the TOP TWO SWAPPED:
Result:

There's a fun side-effect of making that re-arranged image:
- to experience The Vanishing Leprechaun.
- Count the number of leprechauns.
- Now count the number of them in the original image.
- Freaky.
- We'll talk later in the course about problem-solving and try to apply some general techniques to this puzzle.