Defining functions (in Racket)

Our definitions $f(x) = x^2$, $g(x, y) = x + y$ become

\[
\begin{align*}
(\text{define} \ (f \ x) \ (\text{sqr} \ x)) \\
(\text{define} \ (g \ x \ y) \ (+ \ x \ y))
\end{align*}
\]

define is a **special form** in Racket (i.e. it looks like a Racket function, but not all of its arguments are evaluated).

It **binds** (i.e. associates or pairs up) a name to an expression (which uses the parameters that follow the name).
Defining functions (in Racket)

A function definition in Racket consists of:

1. a name for the function,

2. a list of parameters,

3. a single body expression which is an expression.

The body expression typically uses the parameters together with other built-in and user-defined functions.

As pointed out in the previous few slides, a function definition in Racket has the same components (in a slightly different format) as function definition in mathematics.
Applying user-defined functions in Racket

An application of a user-defined function substitutes arguments for the corresponding parameters throughout the definition’s expression.

\[
(\text{define } (g \ x \ y) \ ( + \ x \ y))
\]

The substitution for \((g \ 3 \ 5)\) would be \(( + \ 3 \ 5)\).
Applying user-defined functions in Racket

When faced with choices of substitutions, we use the same rules defined earlier: (1) apply functions only when all arguments are values; when you have a choice, take the leftmost one.

\[
\begin{align*}
(g (g 1 3) (f 3)) & \quad \Rightarrow (g (+ 1 3) (f 3)) \\
& \quad \Rightarrow (g 4 (f 3)) \\
& \quad \Rightarrow (g 4 (sqr 3)) \\
& \quad \Rightarrow (g 4 9) \\
& \quad \Rightarrow (+ 4 9) \\
& \quad \Rightarrow 13
\end{align*}
\]

\[
\begin{align*}
g(g(1, 3), f(3)) & \quad = g(1 + 3, f(3)) \\
& \quad = g(4, f(3)) \\
& \quad = g(4, 3^2) \\
& \quad = g(4, 9) \\
& \quad = 4 + 9 \\
& \quad = 13
\end{align*}
\]
Applying user-defined functions in Racket

Each parameter name has meaning only within the body of its function. I.e. *the two uses of* \( x \) *are independent.*

\[
(\text{define } (f \ x \ y) \ (+ \ x \ y))
\]
\[
(\text{define } (g \ x \ z) \ (* \ x \ z))
\]

Additionally, the following two function definitions define the *same* function:

\[
(\text{define } (f \ x \ y) \ (+ \ x \ y))
\]
\[
(\text{define } (f \ a \ b) \ (+ \ a \ b))
\]
Defining constants

The definitions $k = 3, p = k^2$ become

(define k 3)
(define p (sqr k))

The effect of (define k 3) is to bind the name $k$ to the value 3.

In (define p (sqr k)), the expression (sqr k) is first evaluated to give 9, and then $p$ is bound to that value.
Advantages of constants

• Can give *meaningful names* to useful values (e.g. *interest-rate*, *passing-grade*, and *starting-salary*).

• *Reduces errors* when such values need to be changed

• Makes programs *easier to understand*

Constants can be used in any expression, including the body of function definitions

Sometimes (incorrectly) called variables, but their values cannot be changed (until CS 136)
Scope
The **scope** of an identifier is *where it has effect within the program.*

- The smallest enclosing scope has priority
- Can’t duplicate identifiers within the same scope

```
(define x 3)
(define (f x y)
    (— x y))
(define (g a b)
    (+ a b x))
(+ 1 x)
```

Racket Error: f: this name was defined...
Programming in DrRacket

The definitions window:

• Can *save* and restore your work to/from a file

• Can *accumulate definitions* and expressions

• Run button *loads contents* into Interactions window

• Provides a Stepper to let one *evaluate expressions step-by-step*

• Features: error highlighting, subexpression highlighting, syntax checking

• Can check the scope of a constant...
DrRacket can show what expression each identifier is bound to.
Programs in Racket

A Racket program is a sequence of definitions and expressions.

To run a program, the expressions are evaluated, using substitution, to produce values.

Expressions may also make use of special forms (e.g. define), which look like functions, but don’t necessarily evaluate all their arguments.
Goals of this module

You should understand the basic syntax of Racket, how to form expressions properly, and what DrRacket might do when given an expression causing an error.

You should be comfortable with these terms: function, parameter, application, argument, constant, expression.

You should be able to define and use simple arithmetic functions.

You should understand the purposes and uses of the Definitions and Interactions windows in DrRacket.
Module 02 Summary

Some of the Components of Racket

1. **Values** are numbers or other mathematical objects. [7]
   Examples: 5, \( \frac{4}{9} \) (a rational number), \( \pi \).

2. **Expressions** combine values with operators and functions. [7]
   Examples: \( 5 + 2 \), \( \sin(2\pi) \), \( \frac{\sqrt{2}}{100\pi} \).

3. **Functions** generalize similar expressions. [8]

4. **Function definitions** consist of a name, parameters and an algebraic expression, e.g. (define (f x y) (+ x y)). [9]
Module 02 Summary

How to Use a Function

5. An **application** of a function supplies **arguments** for the **parameters**, which are substituted into the algebraic expression. [10]

6. Functions are evaluated by **substitution**. [11]
   (a) Functions are applied to **values**. [13]
   (b) When there is a **choice** of possible substitutions, always take the **leftmost choice**. [13]
   (c) **Result**: This substitution process ensures there is no ambiguity in the meaning of a program.
Module 02 Summary
Parenthesis, Yields and Special Forms

7. Do not include extra parentheses in an expression. [17]
8. The yields symbol, ⇒, indicates a step in the substitution process. [18]
9. A special form looks like a Racket function but not all of its arguments are evaluated. [24]
10. The special form define binds (i.e. associates) a function name with a function body. [24]
Module 02 Summary

Parameters, Scope and Constants

11. Each parameter name has meaning only within the body of its function. [28]

12. The special form define can also be used to define constants. [29]

13. Constant names can be arbitrary, so pick meaningful ones. [30]

14. An identifier’s scope is where it has effect. [31]
   (a) The smallest enclosing scope has priority. [31]
   (b) Can’t duplicate identifiers within the same scope. [31]
Module 02 Summary

A Racket Program

15. DrRacket has a **Definitions Window** to store definitions and an **Interactions Window**. [32]

16. A Racket **program** consists of a sequence of definitions and expressions. [34]

17. To run a program, the expressions are evaluated, using substitution, to produce values. [34]