R Expressions
Introduction

Before describing some of the common programming structures in R, we need to talk about a basic concept called **Expressions**.

You’ve been using **simple expressions** so far, but we need to introduce the notion of a **compound expression**.
Simple Expressions

```r
a <- "hi"
print(2 + 2)
mean(1:10)
```
Simple Expressions

\[ a \leftarrow \text{"hi"}; \quad \text{print}(2 + 2); \quad \text{mean}(1:10) \]
Simple Expressions

```r
a <- "hi"; print(2 + 2); mean(1:10)
```

*Simple expressions, separated by semicolons, written in a single line of text*

*Although this is a perfectly valid piece of code, we don’t recommend this format because it’s hard to inspect visually.*
Simple Expressions

\[ a \leftarrow \text{"hello"} \]

\[ \text{print}(2 + 2) \]

\[ \text{mean}(1:10) \]
Compound Expressions

```r
{a <- "hello"
print(2 + 2)
mean(1:10)
}
```

*R will treat this as one “unit” or “block” of code*
Compound Expressions

```r
{ 
  a <- "hello"
  print(2 + 2)
  mean(1:10)
}
```

Although this is a perfectly valid piece of code, we never write an R expression like this (in and of itself)
So, when do we use `{...}` compound expressions?
Use of compound expressions

We use compound expressions (i.e. single expressions wrapped within braces) in programming structures like:

- Functions
- Conditionals (if-else)
- Loops (for, while)
Parenthesis, Brackets, and Braces

<table>
<thead>
<tr>
<th>( )</th>
<th>functions</th>
<th>mean(1:10)</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ ]</td>
<td>objects</td>
<td>vec[3]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>mat[2,4]</td>
</tr>
<tr>
<td>{ }</td>
<td>compound</td>
<td>a &lt;- 3</td>
</tr>
<tr>
<td></td>
<td>expressions</td>
<td>b &lt;- a^2</td>
</tr>
</tbody>
</table>
Every expression has a value!
What happens when R executes this code?

```
{  
a <- "hi"

  print(2 + 2)

  mean(1:10)

}
```
What is the value of \( x \)?

\[
x \leftarrow \{
\begin{align*}
a & \leftarrow "hi"
\end{align*}
\begin{align*}
\text{print}(2 + 2)
\end{align*}
\begin{align*}
\text{mean}(1:10)
\end{align*}
\}
\]
Repeat this mantra

Every expression in R has a value: the value of the last statement that is evaluated
Functions
Centimeters to inches

# 1 cm = 0.3937 in

x <- 10

y <- x * 0.3937

y
Centimeters to inches

# 1 cm = 0.3937 in

\[ \text{x} \leftarrow 10 \quad \text{input} \]

\[ \text{y} \leftarrow \text{x} \times 0.3937 \quad \text{processing} \]

\[ \text{y} \quad \text{output} \]
Centimeters to inches

# 1 cm = 0.3937 in

```r
x <- 10
{
  y <- x * 0.3937
  y
}
```

Wrap the **body** of the function within an R expression (i.e. within braces)
Centimeters to inches

# 1 cm = 0.3937 in

function(x) { 
  y <- x * 0.3937 
  y
}

Declare it as a function, and specify the argument(s)
Centimeters to inches

# 1 cm = 0.3937 in

\[
\text{cm2in} \leftarrow \text{function}(x) \quad \text{Assign it to an object (give it a name)}
\]

\[
\begin{align*}
\{ \\
\quad y \leftarrow x \times 0.3937 \\
\quad y \\
\}
\end{align*}
\]
# 1 cm = 0.3937 in

cm2in <- function(x) {
  y <- x * 0.3937
  return(y)
}

Reformat for readability purposes
Centimeters to inches

# 1 cm = 0.3937 in

```r
cm2in <- function(x) {
  y <- x * 0.3937
  return(y)
}
```

```r
cm2in(5)  # test it
```