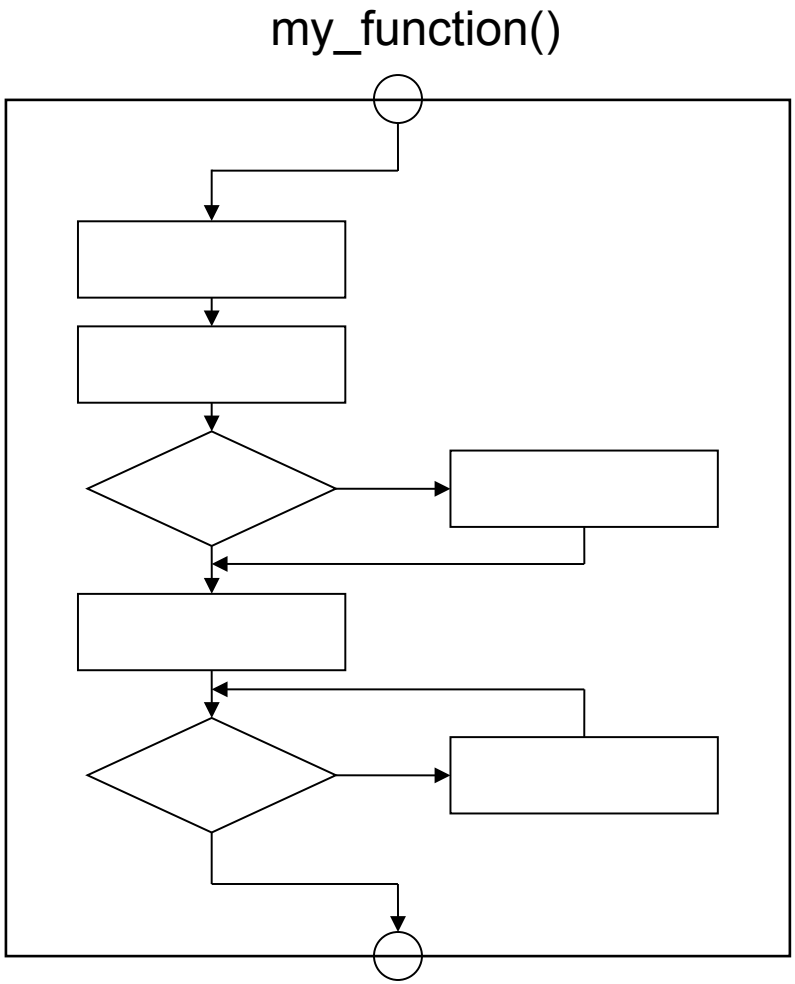
Programming: functions

- A function encapsulates an algorithm
- Functions break a program down into modules
- Modularized programs are easier to write, debug, maintain, and modify
- Functions make algorithms easier to reuse



Making a function in C

```
type function_name(type argument, …) {
    expression;
    return variable_name;
}

double diff_two_nums(double x, double z) {
    double y = x - z;
    return y;
}
```

Objects declared in the arguments or in the function can only be seen inside the function. These are called local variables. Concept: scope.

C

Make a function

```
function_name <- function(arguments) {
    expression
    return(object)
}
```

Break out the code for a single simulation into a separate function

Use the function in the for loop

C

Scope in C

- See examples in c_functions.md
- Variables are local or global depending on where they are declared

Making a function in R

- ?"function" – only the bare bones

```
function_name <- function(arguments) {
    expression
    return(object)
}
```

Class style

explicit return

indent (4 spaces)

closing brace aligns with first letter of function name

R

Making a function in R

```
function_name <- function(arguments) {
    expression
    return(object)
}

diff_two_nums <- function(x, z) {
    y <- x - z
    return(y)
}
```

R

Making a function in R

```
function_name <- function(arguments) {
    expression
    return(object)
}

diff_two_nums <- function(x, z) {
    y <- x - z
    return(y)
}
```

Objects listed in the arguments or defined in the function can only be seen inside the function. These are called local variables. Concept: scope.

R

Scope in R

- See examples in functions.R
- Good programming practice: avoid global variables
  - Define local variables by including in argument list or initializing within the function
  - Global variables in R make programs harder to maintain and debug since they are not explicitly declared

Make a function

```
function_name <- function(arguments) {
    expression
    return(object)
}
```

Exercise:
Make a function to calculate the linear model given the model parameters and a vector of x data. In other words, turn the following into a function:

```
y <- b_0 + b_1 * x
```

Use vectorized operations

R

Make a function

```
function_name <- function(arguments) {
    expression
    return(object)
}
```

Exercise:
Make a function to calculate the linear model given the model parameters and a vector of x data. In other words, turn the following into a function:

```
y <- b_0 + b_1 * x
```

Solution:

```
linmod <- function(b_0, b_1, x) {
  y <- b_0 + b_1 * x
  return(y)
}
```

Use vectorized operations

R