lecture 4

Andrea Boskovic

2022-07-06

Control Flow

1. `if` statements

What is an if statement?

```
x <- 1

if(x > 0) {
  paste0("The value of x you put in is ", x)
}
```

```
## [1] "The value of x you put in is 1"
```

```
x <- -1

if(x > 0) {
  paste0("The value of x you put in is ", x)
}
```

2. `else` statements

What is an else statement?

In this case, our if statement didn't return anything.

```
x <- -1

if(x > 0) {
  paste0("The value of x you put in is ", x)
} else {
  paste0("the condition evaluated to false, but we have an else statement")
}
```

```
## [1] "the condition evaluated to false, but we have an else statement"
```

3. `else if` statements

```r
x <- -1

if(x > 0) {
  paste0("The value of x you put in is ", x)
} else if(x < -1) {
  paste0("hi")
} else {
  paste0("hello!")
}
```

```
## [1] "hello!"
```

Your turn: Suppose we want to check if x is greater than 5 and print out the answer. What should the condition be?

```r
x <- 6

if(x > 5) {
  print("x is greater than 5.")
} else {
  print("x is less than or equal to 5.")
}
```

```
## [1] "x is greater than 5."
```

Loops

1. `for` loops

What is it?

Allows you to iterate over a vector or a set of values until the values are exhausted.

```r
for(i in 1:10) {
  print(i)
}
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
```

```r
for(i in 1:10) {

  for(j in 1:10) {
    print(j)
  }

}
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
```

```
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
```

```
## [1] 10
```

2. `while` loops

Does the same thing as a `for` loop but in a slightly messier way

```
x <- 0
while(x < 5) {

  print(paste("x is equal to", x))
  x <- x + 1

}
```

```
## [1] "x is equal to 0"
## [1] "x is equal to 1"
## [1] "x is equal to 2"
## [1] "x is equal to 3"
## [1] "x is equal to 4"
```

String inputs

```
string_vec <- c("a", "b", "c")

for(letter in string_vec) {
  print(letter)
}
```

```
## [1] "a"
## [1] "b"
## [1] "c"
```

Nested Loops

```
for(i in 1:3) {

  for(j in 1:2) {

    print(i * j)

  }

}
```

```
## [1] 1
## [1] 2
## [1] 2
## [1] 4
## [1] 3
## [1] 6
```

Filling in Vectors using Loops

```
# the vector i want to fill
to_fill <- c()

for(i in 1:5) {

  # Goal: put each x in the to_fill vector
  x <- i + 1
  to_fill <- append(to_fill, x)

}
```

```
# the vector i want to fill
to_fill2 <- rep(NA, 5)

for(i in 1:5) {

  # filling it with i + 1
  to_fill2[i] <- i + 1

}
```

```
to_fill3 <- seq(from = 2, to = 6, by = 1)
```

Functions

- allow you to use the same code chunk repeatedly
  - if you want to do the same thing to a lot of objects, functions are good news!
- lets you generalize your workflow
- makes your code more readable
- you'll need it for assignments in this class!

```
function_name <- function(param1, param2, param3, ...) {

  # what you want your function to do
  # suppose your function calculates x based on some parameters
  # x <- _____


  return(x)

}
```

Let's say we want a function that takes in a number, squares it, and then adds 2 to it.

```
square_plus_2 <- function(val) {

  y <- (val)^2 + 2
```

```
  return(y)

}

square_plus_2(5)
```

```
## [1] 27
```

```
square_plus_2(c(1,2,3,4))
```

```
## [1]  3  6 11 18
```

```
# this doesn't work! think about why
square_plus_2("hello")
```

Improved version:

```
square_plus_2_new <- function(number) {

  # check if it's numeric
  if(is.numeric(number) == FALSE) {

    print("ERROR!!!!!")

  } else {

    # calculate and return desired value
    y <- number^2 + 2
    return(y)

  }

}

square_plus_2_new("hello")
```

```
## [1] "ERROR!!!!!"
```

```
square_plus_2_new(5)
```

```
## [1] 27
```

```
square_plus_2_new(c(1,2,3,4))
```

```
## [1]  3  6 11 18
```

Function with iteration

This is just a function with a for (or while) loop inside.

```
my_sum <- function(x, starting_sum = 0) {

  # initialize total to be 0 (aka starting_sum)
  total <- starting_sum

  # iterate through the length of x
  for(i in 1:length(x)) {

    # increase total by adding x_i onto it
    total <- total + x[i]

  }

  # return total
  return(total)

}

my_sum(x = 1:5)
```

```
## [1] 15
```

```
my_sum(x = 1:3)
```

```
## [1] 6
```