5_Control_Flow

2022-09-14

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6     v purrr   0.3.4
## v tibble  3.1.8     v dplyr   1.0.9
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

5. Control flow

5.1 Introduction

Quiz

1. What is the difference between if and ifelse()

If will only run one chunk of code if true. ifelse will run one of two options

2. In the following code, what will the value of y be if x is TRUE? What if x is FALSE? What if x is NA?

```
y <- if (x) 3
```

3, NULL, Error

3. What does switch("x", x = , y = 2, z = 3) return?

2

5.2 Choices

```
if (condition) true_action
if (condition) true_action else false_action
```

```
grade <- function(x) {
  if (x > 90) {
    "A"
  } else if (x > 80) {
    "B"
  } else if (x > 50) {
    "C"
  } else {
    "F"
  }
}
```

```
x1 <- if (TRUE) 1 else 2
x2 <- if (FALSE) 1 else 2

c(x1, x2)
```

```
## [1] 1 2
```

```
#> [1] 1 2
```

```
greet <- function(name, birthday = FALSE) {
  paste0(
    "Hi ", name,
    if (birthday) " and HAPPY BIRTHDAY"
  )
}
greet("Maria", FALSE)
```

```
## [1] "Hi Maria"
```

```
#> [1] "Hi Maria"
greet("Jaime", TRUE)
```

```
## [1] "Hi Jaime and HAPPY BIRTHDAY"
```

```
#> [1] "Hi Jaime and HAPPY BIRTHDAY"
```

5.2.1 Invalid inputs

```
if ("x") 1
#> Error in if ("x") 1: argument is not interpretable as logical
if (logical()) 1
#> Error in if (logical()) 1: argument is of length zero
if (NA) 1
#> Error in if (NA) 1: missing value where TRUE/FALSE needed
```

```
if (c(TRUE, FALSE)) 1
#> Warning in if (c(TRUE, FALSE)) 1: the condition has length > 1 and only the
#> first element will be used
#> [1] 1
```

5.2.2 Vectorised if

```
x <- 1:10
ifelse(x %% 5 == 0, "XXX", as.character(x))
```

```
##  [1] "1"   "2"   "3"   "4"   "XXX" "6"   "7"   "8"   "9"   "XXX"
```

```
#>  [1] "1"   "2"   "3"   "4"   "XXX" "6"   "7"   "8"   "9"   "XXX"
```

```
ifelse(x %% 2 == 0, "even", "odd")
```

```
##  [1] "odd"  "even" "odd"  "even" "odd"  "even" "odd"  "even" "odd"  "even"
```

```
#>  [1] "odd"  "even" "odd"  "even" "odd"  "even" "odd"  "even" "odd"  "even"
```

```
dplyr::case_when(
  x %% 35 == 0 ~ "fizz buzz",
  x %% 5 == 0 ~ "fizz",
  x %% 7 == 0 ~ "buzz",
  is.na(x) ~ "???",
  TRUE ~ as.character(x)
)
```

```
##  [1] "1"    "2"    "3"    "4"    "fizz" "6"    "buzz" "8"    "9"    "fizz"
```

```
#>  [1] "1"    "2"    "3"    "4"    "fizz" "6"    "buzz" "8"    "9"    "fizz"
```

5.2.3 switch() statement

```
x_option <- function(x) {
  if (x == "a") {
    "option 1"
  } else if (x == "b") {
    "option 2"
  } else if (x == "c") {
    "option 3"
  } else {
    stop("Invalid `x` value")
  }
}
```

```
x_option <- function(x) {
  switch(x,
    a = "option 1",
    b = "option 2",
    c = "option 3",
    stop("Invalid `x` value")
  )
}
```

```
(switch("c", a = 1, b = 2))
```

```
## NULL
```

```
#> NULL
```

```
legs <- function(x) {
  switch(x,
    cow = ,
    horse = ,
    dog = 4,
    human = ,
    chicken = 2,
    plant = 0,
    stop("Unknown input")
  )
}
legs("cow")
```

```
## [1] 4
```

```
#> [1] 4
legs("dog")
```

```
## [1] 4
```

```
#> [1] 4
```

5.2.4 Exercises

```
ifelse(TRUE, 1, "no")
```

1. What type of vector does each of the following calls to ifelse() return?

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

```
ifelse(FALSE, 1, "no")
```

```
## [1] "no"
```

```
ifelse(NA, 1, "no")
```

```
## [1] NA
```

numeric/ double, character, logical

```
x <- 1:10
if (length(x)) "not empty" else "empty"
```

2. Why does the following code work?

```
## [1] "not empty"
```

```
#> [1] "not empty"

x <- numeric()
if (length(x)) "not empty" else "empty"
```

```
## [1] "empty"
```

```
#> [1] "empty"
```

Length of the first is 10 which is true. Length of second is 0 because it is an empty numeric vector. 0 is false

5.3 Loops

```
for (item in vector) perform_action
```

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

```
## [1] 1
## [1] 2
## [1] 3
```

```
#> [1] 1
#> [1] 2
#> [1] 3
```

```
i <- 100
for (i in 1:3) {}
i
```

```
## [1] 3
```

next exits the current iteration. break exits the entire for loop.

```
for (i in 1:10) {
  if (i < 3)
    next

  print(i)

  if (i >= 5)
    break
}
```

```
## [1] 3
## [1] 4
## [1] 5
```

```
#> [1] 3
#> [1] 4
#> [1] 5
```

5.3.1 Common pitfalls

```
means <- c(1, 50, 20)
out <- vector("list", length(means))
for (i in 1:length(means)) {
  out[[i]] <- rnorm(10, means[[i]])
}
out
```

```
## [[1]]
##  [1]  1.1110444  1.6943193  1.9565292  1.2235671  0.7371116 -0.8099674
##  [7]  0.3057675  1.9729555 -0.6019747  0.5382630
##
## [[2]]
##  [1] 49.44094 48.47392 51.29961 49.77046 49.85221 48.47802 48.61829 51.90454
##  [9] 50.53050 49.73531
##
## [[3]]
##  [1] 20.40840 19.94361 19.26617 18.62162 20.53223 19.37252 20.19887 18.19023
##  [9] 19.77515 20.12776
```

```
means <- c()
out <- vector("list", length(means))
for (i in 1:length(means)) {
  out[[i]] <- rnorm(10, means[[i]])
}
#> Error in rnorm(10, means[[i]]): invalid arguments
```

```
1:length(means)
```

```
## [1] 1 2 3
```

```
#> [1] 1 0
```

```
seq_along(means)
```

```
## [1] 1 2 3
```

```
#> integer(0)

out <- vector("list", length(means))
for (i in seq_along(means)) {
  out[[i]] <- rnorm(10, means[[i]])
}
out
```

```
## [[1]]
##  [1]  2.26192101  2.01469927 -0.31047727  1.96852727  1.38923774 -0.19699640
##  [7]  0.09008051  1.05305603  1.17778329  1.47764133
##
## [[2]]
##  [1] 50.29383 49.59602 49.68607 50.23907 50.90101 50.30556 50.00092 48.09349
##  [9] 48.27088 50.13365
##
## [[3]]
##  [1] 19.78116 20.51733 19.38928 20.20405 20.48024 19.86394 18.93218 19.42029
##  [9] 19.12692 20.14940
```

```
xs <- as.Date(c("2020-01-01", "2010-01-01"))
for (x in xs) {
  print(x)
}
```

```
## [1] 18262
## [1] 14610
```

```
#> [1] 18262
#> [1] 14610
```

```
for (i in seq_along(xs)) {
  print(xs[[i]])
}
```

```
## [1] "2020-01-01"
## [1] "2010-01-01"
```

```
#> [1] "2020-01-01"
#> [1] "2010-01-01"
```

5.3.2 Related tools

while(condition) action: performs action while condition is TRUE.

repeat(action): repeats action forever (i.e. until it encounters break).

5.3.3 Exercise

```
x <- numeric()
out <- vector("list", length(x))
for (i in 1:length(x)) {
  out[i] <- x[i] ^ 2
}
out
```

1.Why does this code succeed without errors or warnings?

```
## [[1]]
## [1] NA
```

The ":" works in forward and reverse

```
xs <- c(1, 2, 3)
for (x in xs) {
  xs <- c(xs, x * 2)
}
xs
```

2. When the following code is evaluated, what can you say about the vector being iterated?

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

It gets one element added on to the end each time

```r
for (i in 1:3) {
  print(i)
  i <- i * 2
  print(i)
}
```

3. What does the following code tell you about when the index is updated?

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

Index times 3