# Week-5: Code-along

Wong Zi Xin

2023-09-10

## II. Code to edit and execute using the Code-along.Rmd file

### A. Writing a function

#### 1. Write a function to print a “Hello” message (Slide #14)

```
# Enter code here
say_hello_to <- function(name) {
  print(paste0("Hello ", name, "!"))
}
```

#### 2. Function call with different input names (Slide #15)

```
# Enter code here
say_hello_to("ZiXin")
```

```
## [1] "Hello ZiXin!"
```

```
say_hello_to("Prof")
```

```
## [1] "Hello Prof!"
```

```
say_hello_to("yingzhe")
```

```
## [1] "Hello yingzhe!"
```

#### 3. typeof primitive functions (Slide #16)

```
# Enter code here
typeof(`+`)
```

```
## [1] "builtin"
```

```
typeof(sum)
```

```
## [1] "builtin"
```

#### 4. typeof user-defined functions (Slide #17)

```
# Enter code here
typeof(say_hello_to)
```

```
## [1] "closure"
```

```
typeof(mean)
```

```
## [1] "closure"
```

#### 5. Function to calculate mean of a sample (Slide #19)

```
# Enter code here
calc_sample_mean <- function(sample_size) {
  mean(rnorm(sample_size))
}
```

#### 6. Test your function (Slide #22)

```
# With one input
calc_sample_mean(1000)
```

```
## [1] 0.02730717
```

```
# With vector input
calc_sample_mean(c(100, 300, 3000))
```

```
## [1] 0.0336569
```

#### 7. Customizing the function to suit input (Slide #23)

```
# Enter code here
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✔ dplyr     1.1.2     ✔ readr     2.1.4
## ✔ forcats   1.0.0     ✔ stringr   1.5.0
## ✔ ggplot2   3.4.3     ✔ tibble    3.2.1
## ✔ lubridate 1.9.2     ✔ tidyr     1.3.0
## ✔ purrr     1.0.2
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
sample_tibble <- tibble(sample_sizes =
                          c(100,300,3000))
sample_tibble %>%
  group_by(sample_sizes) %>%
  mutate(sample_means =
           calc_sample_mean(sample_sizes))
```

```
## # A tibble: 3 × 2
## # Groups:   sample_sizes [3]
##   sample_sizes sample_means
##          <dbl>        <dbl>
## 1          100     -0.127
## 2          300     -0.0563
## 3         3000      0.00675
```

#### 8. Setting defaults (Slide #25)

```
# First define the function
calc_sample_mean <- function(sample_size,our_mean=0,our_sd=1) {

  sample <- rnorm(sample_size,
                  mean = our_mean,
                  sd = our_sd)

  mean(sample)
}
# Call the function
calc_sample_mean(sample_size=10)
```

```
## [1] -0.3485685
```

#### 9. Different input combinations (Slide #26)

```
# Enter code here
calc_sample_mean(10, our_sd=2)
```

```
## [1] 0.1116815
```

```
calc_sample_mean(10, our_mean=6)
```

```
## [1] 5.200889
```

```
calc_sample_mean(10,6,2)
```

```
## [1] 5.076583
```

#### 10. Different input combinations (Slide #27)

```
# set error=TRUE to see the error message in the output
# Enter code here
calc_sample_mean(our_mean=5)
```

```
## Error in calc_sample_mean(our_mean = 5): argument "sample_size" is missing, with no default
```

#### 11. Some more examples (Slide #28)

```
# Enter code here
add_two <- function(x) {
  x+2
}

add_two(4)
```

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

```
add_two(-34)
```

```
## [1] -32
```

```
add_two(5.784)
```

```
## [1] 7.784
```

```
add_two <- function(x) {
  y+2
}
add_two(4)
```

```
## Error in add_two(4): object 'y' not found
```

### B. Scoping

#### 12. Multiple assignment of z (Slide #36)

```
# Enter code here
z <- 1
sprintf("The value assigned to z outside the function is %d", z)
```

```
## [1] "The value assigned to z outside the function is 1"
```

```
add_three_to_z <- function(z=2) {
  z <- 3
  return(z+3)
}
add_three_to_z()
```

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

#### 13. Multiple assignment of z (Slide #37)

```
# Enter code here
z <- 1

add_three_to_z <- function(z=2) {
  z <- 3
  return(z+3)
}
add_three_to_z(z=4)
```

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