Week-5: Code-along

Chen Zi Xin

2023-09-09

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)

```
x <- "Hello"
print(x)
```

```
## [1] "Hello"
```

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

```
name <- "Fred"
print(paste0("Hello",name,"!"))
```

```
## [1] "HelloFred!"
```

```
name <- "Candice"
print(paste0("Hello",name,"!"))
```

```
## [1] "HelloCandice!"
```

3. typeof primitive functions (Slide #16)

```
typeof(`+`)
```

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

```
typeof(sum)
```

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

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

```
typeof(TRUE)
```

```
## [1] "logical"
```

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

```
mean(rnorm(100))
```

```
## [1] -0.1284413
```

6. Test your function (Slide #22)

```
mean_sample <- function()
{mean(rnorm(n))}
```

```
calc_sample_mean <-(c(100, 300, 3000))
```

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

```
sample_tibble <- tibble(sample_sizes= c(100,300,3000))
```

8. Setting defaults (Slide #25)

```
calc_sample_mean <- function(sample_size,
our_mean=0,
our_sd=1) {
sample <- rnorm(sample_size,
mean = our_mean,
sd = our_sd)
mean(sample)
}
```

9. Different input combinations (Slide #26)

```
calc_sample_mean(10, our_sd = 2)
```

```
## [1] -0.8755187
```

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

```
## [1] 6.084426
```

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

```
## [1] 4.445199
```

10. Different input combinations (Slide #27)

```
calc_sample_mean(our_mean = 5)
```

```
## Error in calc_sample_mean(our_mean = 5): argument "sample_size" is missing, with n
o default
```

11. Some more examples (Slide #28)

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

add_two(332.4)
```

```
## [1] 334.4
```

B. Scoping

12. Multiple assignment of z (Slide #36)

```
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"
```

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

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

13. Multiple assignment of z (Slide #37)

```
z <- 1
foo <- function(z = 2) {
z <- 3
return(z+3)
}
foo(z = 4)
```

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

```
sprintf("The final value of z after reassigning it to a different value inside the fu
nction is %d",z)
```

```
## [1] "The final value of z after reassigning it to a different value inside the fun
ction is 1"
```