

# Week-6: Code-along

Wong Zi Xin  
2023-09-14

## II. Code to edit and execute using the Code-along-6.Rmd file

### A. for loop

#### 1. Simple for loop (Slide #6)

```
# Enter code here
for (x in c(3, 6, 9)) {
  print(x)
}
```

```
## [1] 3
## [1] 6
## [1] 9
```

#### 2. for loops structure (Slide #7)

```
# Left-hand side code: for loop for passing values
for (x in 1:8) {print(x)}
```

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

```
# Right-hand side code: for loop for passing indices
for (x in 1:8)
{y <- seq(from=100,to=200,by=5)
  print(y[x])}
```

```
## [1] 100
## [1] 105
## [1] 110
## [1] 115
## [1] 120
## [1] 125
## [1] 130
## [1] 135
```

#### 3. Example: find sample means (Slide #9)

```
# Enter code here
sample_sizes <- c(5, 10, 15, 20, 25000)
sample_means <- double(length(sample_sizes))

for (i in seq_along(sample_sizes)) {
  sample_means[i] <- mean(rnorm(sample_sizes[i]))
}
sample_means
```

```
## [1] 0.542795320 -0.296663376 -0.193666693 -0.058524512 0.004633082
```

#### 4. Alternate ways to pre-allocate space (Slide #12)

```
# Example 3 for data_type=double
sample_sizes <- c(5, 10, 15, 20 ,25000)
sample_means <- rep(0, length(sample_sizes))

for (i in 1:length(sample_sizes)) {
  sample_means[i] <- mean(rnorm(sample_sizes[i]))
}
sample_means
```

```
## [1] -0.095867366 0.485352829 0.040699297 0.101259987 0.006199744
```

```
# Initialisation of data_list
data_list <- vector("list", length = 5)
sample_means <- vector("list", length(data_list))

for (i in 1:length(data_list)) {
  data_list[[i]] <- rnorm(5)
  sample_means[[i]] <- mean(data_list[[i]])
}
sample_means
```

```
## [[1]]
## [1] 0.145939
##
## [[2]]
## [1] 0.4731077
##
## [[3]]
## [1] 0.6328891
##
## [[4]]
## [1] 0.1785245
##
## [[5]]
## [1] -0.1058101
```

#### 5. Review: Vectorized operations (Slide #18)

```
# Example: bad idea!
a <- 7:11
b <- 8:12
out <- rep(0L, 5)
for (i in seq_along(a)) {
  out[i] <- a[i] + b[i]
}
out
```

```
## [1] 15 17 19 21 23
```

```
# Taking advantage of vectorization
a <- 7:11
b <- 8:12
out <- a+b
out
```

```
## [1] 15 17 19 21 23
```

### B. Functionals

#### 6. for loops vs Functionals (Slides #23 and #24)

```
# Slide 23
sample_sizes <- c(5, 10, 15, 20, 25000)
sample_summary <- function(sample_sizes, fun) {
  out <- vector("double", length(sample_sizes))
  for (i in seq_along(sample_sizes)) {
    out[i] <- fun(rnorm(sample_sizes[i]))
  }
  return(out)
}
```

```
# Slide 24
#Compute mean
sample_summary(sample_sizes,mean)
```

```
## [1] 1.01546867 0.33985299 -0.31326776 -0.31909995 0.00519357
```

```
# Compute median
sample_summary(sample_sizes,median)
```

```
## [1] -0.049534576 -0.105696292 -0.202834166 -0.034299702 0.005989598
```

```
# Compute sd
sample_summary(sample_sizes,sd)
```

```
## [1] 1.4752729 0.9206605 0.7992372 0.8070027 0.9966503
```

### C. while loop

#### 7. while loop (Slides #27)

```
# Left-hand side code: for loop
for(i in 1:5) {
  print(i)
}
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
```

```
# Right-hand side code: while loop
i <- 1
while(i <=5) {
  print(i)
  i <- i + 1
}
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
```