EX2_loops.R

wanida

2023-11-09

```
###################################Loops#######################################

for(i in 1:10) { # Head of for-loop, i is in particular vector

  x1 <- i^2      # Code block
  print(x1)      # Print results
}
```

```
## [1] 1
## [1] 4
## [1] 9
## [1] 16
## [1] 25
## [1] 36
## [1] 49
## [1] 64
## [1] 81
## [1] 100
```

```
x2 <- c("Samsung", "Apple", "Meta", "Google", "Microsoft")  # Create character vector

for(i in x2) {     # Loop over character vector, i in x2 vector

  print(paste("The name", i, "consists of", nchar(i), "characters."))
}
```

```
## [1] "The name Samsung consists of 7 characters."
## [1] "The name Apple consists of 5 characters."
## [1] "The name Meta consists of 4 characters."
## [1] "The name Google consists of 6 characters."
## [1] "The name Microsoft consists of 9 characters."
```

```
# appending to a vector
x3 <- numeric()
for(i in 1:10) {    # Head of for-loop
  x3 <- c(x3, i^2)  # Code block, adding value in x3 in R format using "," (or .append in Py3)
}
print(x3)
```

```
##  [1]   1   4   9  16  25  36  49  64  81 100
```

```r
# nested for loop (bad bad bad
# --> need to using hashing to eliminate complexity)
x4 <- character()                                          # Create empty data object
loop_work <- 0
for(i in 1:5) {                                            # Head of first for-loop
  for(j in 1:5) {                                          # Head of nested for-loop
    loop_work <- loop_work + 1            # Creating a counter to count each generation of loop
    x4 <- c(x4, paste(LETTERS[i], letters[j], sep = "_"))  # Code block, iterating 25 steps!
          #LETTERS, letters is a function here
  }
}

### A better way using hashing --> this is on almost every tech interview
library(hash)
```

```
## hash-2.2.6.3 provided by Decision Patterns
```

```r
h <- hash()                # Making hash environment/object
x5 <- c()                  # Create an empty vector
hash_work <- 0             # counter starting with 0

#Creating a hash map
#First "loop" create a Big and litter letters in h environment
for(i in 1:5){
  hash_work <- hash_work + 1
  h[LETTERS[i]] <- letters[1:5]   #Creating a key with big letter with length of i as 1-5
                                  #and adding value = letter 1-5 to hash table h
}                           # h is a hash env with big letter as key and little letter as value
                            # Whole thing is called Hash map

for(j in 1:length(h)){              # Iterate with j with length of hash environment
  hash_work <- hash_work + 1
  x5 <- c(x5, paste(names(h)[j], h[[names(h)[j]]] ,sep='_'))
  #Creating a vector called x5, returning key in h at j (from 1 through length of h) with [ ]
  # and value of h key with [[ ]], or can be "h[[ LETTER[j] ]]", with a separator
}
print(hash_work)
```

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

```r
print(loop_work)
```

```
## [1] 25
```

```r
#make sure we just made the same two vectors
all(x4 == x5)
```

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

```r
# for loop with break statement
for(i in 1:10) {                    # Head of for-loop
  x6 <- i^2                         # Code block
  print(x6)                         # Print results
  if(i >= 5) {                      # Conditionally stop for-loop
    break                           # Using break-statement
  }
}
```

```
## [1] 1
## [1] 4
## [1] 9
## [1] 16
## [1] 25
```

```r
# for loop with next statement (skip)
for(i in 1:10) {                      # Head of for-loop
  if(i %in% c(1, 3, 5, 7, 9)) {       # Conditionally skip iteration, when i is odd number
    next                              # Using next-statement to skip those i odd numbers
  }
  x7 <- i^2                           # Code block, after skipping, square not skipping i
  print(x7)                           # Print results
}
```

```
## [1] 4
## [1] 16
## [1] 36
## [1] 64
## [1] 100
```

```r
# iterating over a dataframe
iris_new1 <- iris
for(i in 1:ncol(iris_new1)) {                          # Head of for-loop
  if(grepl("Width", colnames(iris_new1)[i])) {         # Logical condition, pattern = "Width", data = coln
    iris_new1[ , i] <- iris_new1[ , i] + 1000          # Code block, add 1000 to df, iterating in colume
  }
}
head(iris_new1)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1      1003.5          1.4      1000.2  setosa
## 2          4.9      1003.0          1.4      1000.2  setosa
## 3          4.7      1003.2          1.3      1000.2  setosa
## 4          4.6      1003.1          1.5      1000.2  setosa
## 5          5.0      1003.6          1.4      1000.2  setosa
## 6          5.4      1003.9          1.7      1000.4  setosa
```

```r
###################################While Loops#####################################
#If your condition is not met, the loop will go forever

i <- 1            # set the initial value
while (i < 6) {   # Head of while loop + test condition
```

```
  print(i)        # Code block
  i = i+1         # Code block (make sure you add 1 or the condition will not be met!)
}
```

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

```
typeof(i)
```

```
## [1] "double"
```

```
####################################Apply#####################################
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

```
# row sums for the 1st 5 rows and 1st 4 columns of IRIS, MARGIN 1 is row
apply(iris[1:5,1:4],MARGIN=1,FUN=sum)
```

```
##    1    2    3    4    5
## 10.2  9.5  9.4  9.4 10.2
```

```
# col means for all rows of the 1st 4 columns of IRIS, MARGIN 2 is column
apply(iris[,1:4],MARGIN=2,FUN=mean)
```

```
## Sepal.Length  Sepal.Width Petal.Length  Petal.Width
##     5.843333     3.057333     3.758000     1.199333
```

```
# Custom function for apply where "square" is a function.
square <- function(x){
  x^2
}

# row & col custom function for the 1st 5 rows and 1st 4 columns of IRIS,
# MARGIN = c(1,2) to do function to all elements in both cols and rows
apply(iris[1:5,1:4],MARGIN=c(1,2),FUN=square)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1        26.01       12.25         1.96        0.04
## 2        24.01        9.00         1.96        0.04
## 3        22.09       10.24         1.69        0.04
## 4        21.16        9.61         2.25        0.04
## 5        25.00       12.96         1.96        0.04
```

```r
iris[1:5,1:4]
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1          5.1         3.5          1.4         0.2
## 2          4.9         3.0          1.4         0.2
## 3          4.7         3.2          1.3         0.2
## 4          4.6         3.1          1.5         0.2
## 5          5.0         3.6          1.4         0.2
```

```r
1.4^2
```

```
## [1] 1.96
```