Lab 3. For Loops & Strings

Review

Special ``Numbers''

```
NA  #Not Available. Missing Value

[1] NA

NaN  #Not a Number - Undefined

[1] NaN

NULL  #Just a placeholder.

NULL

Inf  #Bigger than all big

[1] Inf
```

Character Vectors

```
# I will also refer to them as StringVectors
c("String", "Elements", "Are", "Wrapped", "In", "Quotes")

[1] "String"   "Elements" "Are"      "Wrapped"  "In"       "Quotes"

# If one character element is in a vector, the remaining elements will be coerced(turned) into character elements.
var <- c("var")
# var is a variable that contains a character/string element 'var'
```

Paste & Print

```
x <- "Hello"
y <- "World!"
z <- paste(x, y, sep = "<|>")  #One string created 'Hello<|>World!
print(z)

[1] "Hello<|>World!"
```

Readline

```
x <- readline("Enter something at this prompt: ")
# creates a string vector with your input
```

as.numeric

```
c(9, "9", "nine")  # a character vector

[1] "9"    "9"    "nine"

as.numeric(c(9, "9", "nine"))  #converts elements to numeric values, if possible

Warning: NAs introduced by coercion

[1]  9  9 NA
```

for Loop

```
givenVector <- c("orange", "red", "gold")
for (eachElement in givenVector) {
    # perform these steps
}
```

Examples

A Short for Loop

1. Use a **for** loop to print the following lines by pasting together the two given vectors.

```r
# Variable to use:
mascots <- c("Skyhawk", "big red bear", "fuzzy orange", "golden eagles", "blue devil")
phrase <- "One of my mascots was a "

# Results
"One of my mascots was a Skyhawk."
"One of my mascots was a big red bear."
"One of my mascots was a fuzzy orange."
"One of my mascots was a golden eagles."
"One of my mascots was a blue devil."
```

2. Take a random sample of 10 values from the integers starting at 1 and ending at 5. Determine their product.

a. Each number should have an equal chance of being selected.

b. Numbers can be selected more than once.

Problems

3. Create a **for** loop that will sum the elements of the vector x. At the beginning of the code, you should have a variable **total <- 0**. At the end of the code, **total** should equal the total of the numbers in x. The code should be written in such a way that the values in the vector x can be replaced with another set of values, and the code will proceed to total the new values. This means that you should not explicitly type in all the values separated by a plus sign. Report back your result. Use inline code.

The total of the values is ENTER_TOTAL_HERE.

```r
# The vector x has 100 value
x <- (1:100)%/%c(3, 6, 12, 19)

total <- 0
```

4. Create a **for** loop that can be used to compute n!=n(n-1)…3x2x1. Use it to compute 10!. The code should be written in such a way that the value 10 can be replaced with another number, and the appropriate factorial will be found. This means that you should not explicitly type in all the values separated by a times sign. Report back your result. Use inline code. You should start by making a **product** variable that is set equal to one. Then update the product variable as the for loop proceeds.

I was told to compute ENTER_NUMBER factorial. I found out that it is ENTER_FACTORIAL_VALUE.

```r
product <- 1
```

Lab 4. While loops & Strings

Review

while Loop

```r
thisValue = TRUE
while (thisValue == TRUE) {
    performTheseSteps <- FALSE
    thisValue <- performTheseSteps
    # create thisValue so that it eventually becomes FALSE or insert a break statement that will kick in at a certain
    # point. Otherwise, the loop continues forever.

}
```

Examples

A Short while Loop

1. Add randomly selected elements, one at a time, from the vector 1:10 together until you have a total of at least 100. After each element is added to the total, report the value that was selected, the currect total, and the number of values selected so far.

a. Create a vector **number.selected.so.far** and set it equal to zero. It will record the number of values selected.

b. Create a vector **total.so.far** and set it equal to zero. It will record the accumulated total of the numbers that you have selected so far.

c. Create a vector **possible.values**. It will hold the integers from 1 to 10.

d. Create a **while** loop with the following properties

```
 i.    The condition to proceed with the loop checks to see if the total.so.far is less to 100.
ii.    Each iteration of the loop will:
          1. assign a randomly selected value from possible.values to a variable randomly.selected.number,
          2. add randomly.selected.number to total,
          3. increase number.selected.so.far by 1.
          4. Report the currently selected value, current total, and number of values selected so far.
```

e. Create a reporting line that indicates the number of selections made and the total. Use inline code

It took INSERT_NUMBER_OF_SELECTION selections to reach at least 100. My total was INSERT_TOTAL.

2. Simulate rolling a 6 sided fair die 50 times and average the 50 values.

Problems

Problem 3. Simulate rolling an 8 sided die 100 times and average the 100 values. The numbers on the sides of the die are 0, 1, 2, 4, 4, 6, 7, and 8.

Problem 4. Multiply randomly selected values from the sequence 0.10, 0.11, 0.12, 0.13, … 0.99, 1.00 together, one at a time, until the product is less than 0.01. After each element is multiplied to the product, report the value that was selected, the current product, and the number of values selected so far. Once it is complete, Create a reporting line that indicates the number of selections made and the total. Use inline code

It took INSERT_NUMBER_OF_SELECTION selections to reach a value less than 0.01. My final product was was INSERT_PRODUCT.