Assignment 2: Coding Basics

OVERVIEW

This exercise accompanies the lessons in Environmental Data Analytics on coding basics.

Directions

1. Rename this file `<FirstLast>_A02_CodingBasics.Rmd` (replacing `<FirstLast>` with your first and last name).
2. Change "Student Name" on line 3 (above) with your name.
3. Work through the steps, **creating code and output** that fulfill each instruction.
4. Be sure to **answer the questions** in this assignment document.
5. When you have completed the assignment, **Knit** the text and code into a single PDF file.
6. After Knitting, submit the completed exercise (PDF file) to Sakai.

Basics, Part 1

1. Generate a sequence of numbers from one to 30, increasing by threes. Assign this sequence a name.

2. Compute the mean and median of this sequence.

3. Ask R to determine whether the mean is greater than the median.

4. Insert comments in your code to describe what you are doing.

```
#1. #give the sequence a name using the <- symbols and then call the console to print the sequence with
three_sequence <- seq(1,30,3)
three_sequence
```

```
##  [1]  1  4  7 10 13 16 19 22 25 28
```

```
#2. #call the mean and median functions using the name of the sequence I created above
mean_three_sequence <- mean(three_sequence)
median_three_sequence <- median(three_sequence)
mean_three_sequence
```

```
## [1] 14.5
```

```
median_three_sequence
```

```
## [1] 14.5
```

```
#3. #this statement will print TRUE or FALSE in the console depending on the relation between the two v
mean(three_sequence) > median(three_sequence)
```

```
## [1] FALSE
```

Basics, Part 2

5. Create a series of vectors, each with four components, consisting of (a) names of students, (b) test scores out of a total 100 points, and (c) whether or not they have passed the test (TRUE or FALSE) with a passing grade of 50.
6. Label each vector with a comment on what type of vector it is.
7. Combine each of the vectors into a data frame. Assign the data frame an informative name.
8. Label the columns of your data frame with informative titles.

```
#5.
c('Kelly', 'Sarah', 'Meghan', 'Emma')
```

```
## [1] "Kelly"  "Sarah"  "Meghan" "Emma"
```

```
c(89, 98, 93, 40)
```

```
## [1] 89 98 93 40
```

```
c('TRUE', 'TRUE', 'TRUE', 'FALSE')
```

```
## [1] "TRUE"  "TRUE"  "TRUE"  "FALSE"
```

```
#6.
names <- c('Kelly', 'Sarah', 'Meghan', 'Emma') #this is a character vector
scores <- c(89, 98, 93, 40) #this is a numeric vector
pass <- c('TRUE', 'TRUE', 'TRUE', 'FALSE') #this is a character vector
class(names)
```

```
## [1] "character"
```

```
class(scores)
```

```
## [1] "numeric"
```

```
class(pass)
```

```
## [1] "character"
```

```r
#7.
student_passes <- as.data.frame(cbind(names, scores, pass))

#8.
colnames(student_passes) <- c('Student Name', 'Score Received', 'Pass or Fail')
student_passes
```

```
##   Student Name Score Received Pass or Fail
## 1        Kelly              89         TRUE
## 2        Sarah              98         TRUE
## 3       Meghan              93         TRUE
## 4         Emma              40        FALSE
```

9. QUESTION: How is this data frame different from a matrix?

   Answer: This data frame is different from a matrix because it contains data of different types. A matrix is classified by having all of the same data type so that matrix multiplication and other linear algebra operations can be performed. Data frames do not need a uniform data type, as we see in this data frame, which is made up of character vectors and numeric vectors.

10. Create a function with an if/else statement. Your function should take a **vector** of test scores and print (not return) whether a given test score is a passing grade of 50 or above (TRUE or FALSE). You will need to choose either the `if` and `else` statements or the `ifelse` statement.

11. Apply your function to the vector with test scores that you created in number 5.

```r
#10.
check_scores <- function(scores) {
  result <- ifelse(scores >= 50, TRUE, FALSE)
  print(result)
}

#11.
check_scores(scores)
```

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

12. QUESTION: Which option of `if` and `else` vs. `ifelse` worked? Why?

    Answer: I used ifelse and it worked because I loaded in my arguments according to the formula (logical expression, value if TRUE, value if FALSE). My function takes a score and tests if it is equal to or greater than 50. If it is, the output is TRUE. If it is not, the output is FALSE.