1.2.3 Basic Concepts Exercise Solutions

Exercise 1: Greeting Script in R

Question A: Construct an R script that outputs the greeting "Hello, World!" Then, alter the script to replace "World" with your own name.

Solution A:

```r
print("Hello, World!")
```

```
[1] "Hello, World!"
```

```r
print("Hello, [Your Name]!")
```

```
[1] "Hello, [Your Name]!"
```

Question B: Utilize the `cat()` function to display "Hello, World!" in the console. Observe and report any variations in the output compared to the `print()` function.

Solution B:

```r
cat("Hello, World!\n")
```

```
Hello, World!
```

Exercise 2: Storing Personal Favorites

Store your preferred number, color, and a boolean value that signifies your liking for pizza. Display these stored values.

Solution:

```r
favorite_number <- 42
favorite_color <- "blue"
likes_pizza <- TRUE
print(favorite_number)
```

```
[1] 42
```

```r
print(favorite_color)
```

```
[1] "blue"
```

```
print(likes_pizza)
```

```
[1] TRUE
```

Exercise 3: Types and Collections

Question A: Generate a vector with the numbers from 1 to 10. Determine and document the vector's type as well as the types of the elements within it.

Solution A:

```
numbers_vector <- 1:10
print(typeof(numbers_vector))
```

```
[1] "integer"
```

```
print(class(numbers_vector[1]))
```

```
[1] "integer"
```

Question B: Form a list holding a numeric value, a character string, and a boolean value. Ascertain the list's type and the types of its individual elements.

Solution B:

```
mixed_list <- list(42, "hello", TRUE)
print(typeof(mixed_list))
```

```
[1] "list"
```

```
print(sapply(mixed_list, class))
```

```
[1] "numeric"   "character" "logical"
```

Exercise 4: Checking Number Sign

Compose a script that discerns whether a given number (hard-coded in the script) is positive, negative, or zero, and relays an appropriate message.

Solution:

```
number <- -5  # Replace with any number
if (number > 0) {
  print("The number is positive.")
} else if (number < 0) {
  print("The number is negative.")
} else {
```

```
          print("The number is zero.")
        }
```

```
[1] "The number is negative."
```

Exercise 5: Iterative Loops

Question A: Write a `for` loop that computes and exhibits the squares of numbers from 1 to 10.

Solution A:

```
        for (i in 1:10) {
          print(i^2)
        }
```

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

Question B: Implement a `while` loop to effectuate a countdown from 10 to 1, displaying each countdown step.

Solution B:

```
        count <- 10
        while (count > 0) {
          print(count)
          count <- count - 1
        }
```

```
[1] 10
[1] 9
[1] 8
[1] 7
[1] 6
[1] 5
[1] 4
[1] 3
[1] 2
[1] 1
```

Exercise 6: Area Calculation Function

Craft a function named `calculate_area` that accepts a circle's radius and returns its area. Check the function's accuracy with a variety of radii.

Solution:

```
calculate_area <- function(radius) {
  return(pi * radius^2)
}

# Test the function
calculate_area(5)  # Replace with different radii
```

```
[1] 78.53982
```

Exercise 7: Vector Mathematics

Question A: Form two vectors, `a` and `b`, each with 5 random numbers. Execute and print the results of element-wise addition, subtraction, multiplication, and division.

Solution A:

```
a <- c(2, 4, 6, 8, 10)
b <- c(1, 3, 5, 7, 9)
print(a + b)
```

```
[1]  3  7 11 15 19
```

```
print(a - b)
```

```
[1] 1 1 1 1 1
```

```
print(a * b)
```

```
[1]  2 12 30 56 90
```

```
print(a / b)
```

```
[1] 2.000000 1.333333 1.200000 1.142857 1.111111
```

Question B: Use comparison operators to compare the elements of `a` and `b` and communicate the outcomes.

Solution B:

```
print(a > b)
```

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

```r
        print(a < b)
```

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

Exercise 8: Working with Factors

Institute a factor variable that delineates three kinds of fruit. Illustrate the factor variable, its levels, and transmute it into numerical values.

Solution:

```r
        fruits <- factor(c("apple", "banana", "cherry"))
        print(fruits)
```

```
[1] apple  banana cherry
Levels: apple banana cherry
```

```r
        print(levels(fruits))
```

```
[1] "apple"  "banana" "cherry"
```

```r
        print(as.numeric(fruits))
```

```
[1] 1 2 3
```

Exercise 9: Data Frame Adjustments

Forge a modest data frame that enlists student IDs, names, and scores. Accomplish these tasks:

- Select and unveil only the students' names and scores.
- Incorporate a column indicating if the student passed (score >= 50).

Solution:

```r
        students <- data.frame(
          ID = 1:5,
          Name = c("Alice", "Bob", "Charlie", "David", "Eve"),
          Score = c(76,

        49, 90, 34, 62)
        )

        # Select and print names and scores
        print(students[c("Name", "Score")])
```

```
     Name Score
1   Alice    76
2     Bob    49
```

```
3 Charlie    90
4   David    34
5     Eve    62
```

```
# Add a 'Passed' column
students$Passed <- students$Score >= 50
print(students)
```

```
  ID    Name Score Passed
1  1   Alice    76   TRUE
2  2     Bob    49  FALSE
3  3 Charlie    90   TRUE
4  4   David    34  FALSE
5  5     Eve    62   TRUE
```

Exercise 10: Interactive User Query (Optional)

Write code that asks the user to enter his name and age, then returns a message along with a mention of whether they are above the age of 18.

Solution:

```
# This task is optional as it requires interactive R environment
user_name <- readline(prompt = "Enter your name: ")
user_age <- as.numeric(readline(prompt = "Enter your age: "))

if (user_age > 18) {
  message <- paste("Hello", user_name, "- you are over 18.")
} else {
  message <- paste("Hello", user_name, "- you are not over 18.")
}

print(message)
```

Home Work

Scenario: You are a Data Scientist working for a consulting firm. One of your colleagues from the Auditing department has asked you to help them assess the financial statement of organization X.

You have been supplied with two vectors of data: monthly revenue and monthly expenses for the financial year in question. Your task is to calculate the following financial metrics:

1. **Profit for each month**: Calculate the profit for each month by subtracting expenses from revenue.
2. **Profit after tax for each month**: The tax rate is 30%. Calculate the profit after tax for each month.
3. **Profit margin for each month**: The profit margin is equal to the profit after tax divided by revenue, expressed as a percentage.

4. **Good months**: Identify the months where the profit after tax was greater than the mean profit after tax for the year.

5. **Bad months**: Identify the months where the profit after tax was less than the mean profit after tax for the year.

6. **The best month**: Identify the month where the profit after tax was maximum for the year.

7. **The worst month**: Identify the month where the profit after tax was minimum for the year.

Additional Requirements:

- All results need to be presented as vectors.
- Results for dollar values need to be calculated with $0.01 precision but presented in units of $1,000 (i.e., 1k) with no decimal points.
- Results for the profit margin ratio need to be presented in units of percentage with no decimal points.
- Note: It is okay for tax for any given month to be negative (in accounting terms, negative tax translates into a deferred tax asset).

```r
# Given data
revenue <- c(14574.49, 7606.46, 8611.41, 9175.41, 8058.65, 8105.44, 11496.28, 9766.0
expenses <- c(12051.82, 5695.07, 12319.20, 12089.72, 8658.57, 840.20, 3285.73, 5821.

# 1. Profit for each month
profit <- revenue - expenses

# 2. Profit after tax for each month (30% tax rate)
tax_rate <- 0.30
profit_after_tax <- profit * (1 - tax_rate)

# 3. Profit margin for each month
profit_margin <- (profit_after_tax / revenue) * 100

# 4. Good months (profit after tax > mean profit after tax)
mean_profit_after_tax <- mean(profit_after_tax)
good_months <- profit_after_tax > mean_profit_after_tax

# 5. Bad months (profit after tax < mean profit after tax)
bad_months <- profit_after_tax < mean_profit_after_tax

# 6. The best month (max profit after tax)
best_month <- which.max(profit_after_tax)

# 7. The worst month (min profit after tax)
worst_month <- which.min(profit_after_tax)

# Convert dollar values to units of 1,000 with no decimal points
```

```r
profit_k <- round(profit / 1000)
profit_after_tax_k <- round(profit_after_tax / 1000)

# Convert profit margin to integers (percentage with no decimal points)
profit_margin_percent <- round(profit_margin)

# Display results
profit_k
```

```
[1]  3  2 -4 -3 -1  7  8  4  3 -2  1 12
```

```r
profit_after_tax_k
```

```
[1]  2  1 -3 -2  0  5  6  3  2 -2  0  8
```

```r
profit_margin_percent
```

```
[1]  12  18 -30 -22  -5  63  50  28  23 -11   4  53
```

```r
good_months
```

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

```r
bad_months
```

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

```r
best_month
```

```
[1] 12
```

```r
worst_month
```

```
[1] 3
```