Solutions: A First Guide to Programming in R

2022-01-20

Contents

2 The Basics

Exercise 2.1. In the following code box, create three new variables named 'name', 'age', and 'birthday' (in a MM/DD/YYYY format), containing your info. What data types are these? When you're done, run the code to make sure you've declared your variables correctly.

Exercise 2.2. Use the `max` and `min` functions to find the largest value of `list_of_numbers` (see below), then store those two values in two different variables. Then find the product of the max and min, save it to another variable, and print it using the `print` function (note that both `max`, `min`, and `print` can all take in one parameter).

Exercise 2.3. Write a function called `quad_form` that accepts three parameters (`a, b, c`) and return the positive solution of the quadratic formula (so return $x = \frac{-b+\sqrt{b^2-4ac}}{2a}$). A few hints:

- The square root command is `sqrt()` (alternatively, you can raise something to the 0.5 power with the carat).
- Order of operations **does** work as you've been taught in school (fun fact: multiplication has higher precedence than division in R). However, you need to be explicit when writing multiplication, i.e. you need to use the asterisk to show that you are multiplying two numbers together.
- Be sure to explicitly return the number you calculate! R will actually return the output of the last line that you write so you don't need to type `return`, but it is very good practice to do so.

```
quad_form = function(a, b, c){
 x = (-b + sqrt(b^2 - 4*a*c))/(2*a)
 return(x)
}

# test if this works
quad_form(1, 4, 4)
```

```
## [1] -2
```

Exercise 2.4. Predict the output of this program! To check your answer, just copy the code and run it.

```
print("Green eggs and ham.")
```

```
## [1] "Green eggs and ham."
```

```
if(1 > 2 || "red" == "blue"){
  print("I do not like green eggs and ham.")
}
```

```
if("red" == "green" || 2 > 1){
  print("I do not like them, Sam-I-Am.")
}
```

```
## [1] "I do not like them, Sam-I-Am."
```

```
if(!(3 != 3 || "pass" == "fail")){
  print("I do so like green eggs and ham.")
}
```

```
## [1] "I do so like green eggs and ham."
```

Exercise 2.5. Print out 10, 9, 8, ..., 1 in descending order. Can you think of more than one way to do it?

```
# with a for loop, with some math; this does it line by line
# don't do this in practice
for(i in 1:10){
  print(10 - i + 1)
}
```

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

```
# the "R" way to do it
print(10:1)
```

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

```
# using the seq function
print(seq(10, 1))
```

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

Exercise 2.6. (nested for loops) Print out all the possible sums from rolling two dice, with repetition (i.e. 1+1, 1+2, etc.). You should see 36 values.

Exercise 2.7. Add random numbers to `x` until `x` is greater than 10, printing `x` after every addition. Use the function `runif(1)` to generate one random number at a time.

Exercise 2.8. Given the following 10×10 matrix, calculate the mean across each row and down each column. Store these store values two separate numeric vectors without using a for loop.

```
mean_mat = matrix(1:100, nrow = 10, ncol = 10)  # data frame of values

# the solution is very unexciting
row_means = rowMeans(mean_mat)
col_means = colMeans(mean_mat)
```

5 Vectors, Vectorized Operators, and Boolean Indexing

Exercise 5.1. Given the following numeric data, remove the values that contain NA values.

```
numeric_values = c(1, 2, NA, 3, NA, NA, NA, 4)
numeric_values = numeric_values[!is.na(numeric_values)]
numeric_values
```

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

Exercise 5.2. Given the same numeric data, convert the NA values to -1. (Hint: you can do boolean indexing with assignment.)

```
numeric_values = c(1, 2, NA, 3, NA, NA, NA, 4)
numeric_values[is.na(numeric_values)] = -1
numeric_values
```

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

Exercise 5.3. Given the same numeric data, remove the NA values. Then create a new vector where all values smaller than the mean are replaced with `"small"`; otherwise, they are replaced with `"large"`.

```
numeric_values = c(1, 2, NA, 3, NA, NA, NA, 4)
numeric_values = numeric_values[!is.na(numeric_values)]

mean_value = mean(numeric_values)

numeric_values = ifelse(numeric_values < mean_value, "small", "large")
numeric_values
```

```
## [1] "small" "small" "large" "large"
```