Exercises for Recap Session 1

2024-04-18

Exercise 1: Basic object types I

1. Create a vector containing the numbers 2, 5, 2.4 and 11.

```
ex1_vec <- c(2, 5, 2.4, 11)
```

2. Replace the second element with 5.9.

```
ex1_vec[2] <- 5.9
ex1_vec
```

```
[1]  2.0  5.9  2.4 11.0
```

3. Add the elements 3 and 1 to the beginning, and the elements "8.0" and "9.2" to the end of the vector.

```
va_1 <- c(3, 1)
va_2 <- c("8.0", "9.2")
ex1_vec_extended <- c(va_1, ex1_vec, va_2)
ex1_vec_extended
```

```
[1] "3"   "1"   "2"   "5.9" "2.4" "11"  "8.0" "9.2"
```

4. Create a vector with the numbers from -8 to 9 (step size: 0.5)

```
ex1_vec_4 <- seq(-8, 9, by = 0.5)
ex1_vec_4
```

```
 [1] -8.0 -7.5 -7.0 -6.5 -6.0 -5.5 -5.0 -4.5 -4.0 -3.5 -3.0 -2.5 -2.0 -1.5 -1.0
[16] -0.5  0.0  0.5  1.0  1.5  2.0  2.5  3.0  3.5  4.0  4.5  5.0  5.5  6.0  6.5
[31]  7.0  7.5  8.0  8.5  9.0
```

5. Compute the square root of each element of the first vector using vectorisation.

```r
sqrt(ex1_vec_4)
```

```
Warning in sqrt(ex1_vec_4): NaNs produced
```

```
 [1]       NaN       NaN       NaN       NaN       NaN       NaN       NaN
 [8]       NaN       NaN       NaN       NaN       NaN       NaN       NaN
[15]       NaN       NaN 0.0000000 0.7071068 1.0000000 1.2247449 1.4142136
[22] 1.5811388 1.7320508 1.8708287 2.0000000 2.1213203 2.2360680 2.3452079
[29] 2.4494897 2.5495098 2.6457513 2.7386128 2.8284271 2.9154759 3.0000000
```

6. Create a character vector containing then strings `"Number_1"` to `"Number_5"`. Use suitable helper functions to create this vector quickly.

```r
ex1_char_vec <- paste0("Number_", seq(1, 5))
ex1_char_vec
```

```
[1] "Number_1" "Number_2" "Number_3" "Number_4" "Number_5"
```

Exercise 2: Basic object types II

Consider the following vector:

```r
ex_2_vec <- c(1, "2", FALSE)
```

1. What is the type of this vector? Why?
2. What happens if you coerce this vector into type integer? Why?
3. What does `sum(is.na(x))` tell you about a vector `x`? What is happening here?
4. Is it a good idea to use `as.integer()` on double characters to round them to the next integer? Why (not)? What other ways are there to do the rounding?

Exercise 3: Define a function

Create functions that take a vector as input and returns:

1. The last value.
2. Every element except the last value and any missing values.
3. Only even numbers.

   Hint: Use the operation `x %% y` to get the remainder from diving `x` by `y`, the so called 'modulo y'. For even numbers, the modulo 2 is zero.

Apply your function to the following example vector:

```r
ex_3_vec <- c(1, -8, 99, 3, NA, 3, -0.5)
```

Exercise 4: Lists

1. Create a list that contains three elements called `'a'`, `'b'` and `'c'`. The first element should correspond to a double vector with the elements `1.5`, `-2.9` and `99`. The second element should correspond to a character vector with the elments `'Hello'`, `'3'`, and `'EUF'`. The third element should contain three times the entry `FALSE`.
2. Transform this list into a `data.frame` and a `tibble`. Then apply `str()` to get information about the respective structure. How do the results differ?

Exercise 5: Data frames and the study semester distribution at EUF

The package `DataScienceExercises` contains a data set called `EUFstudentsemesters`, which contains information about the distribution of study semesters of enrolled students at the EUF in 2021. You can shortcut the data set as follows:

```r
euf_semesters <- DataScienceExercises::EUFstudentsemesters
```

1. What happens if you extract the column with study semesters as a vector and transform it into a `double`?
2. What is the average study semester of those students being in their 8th or earlier semester?
3. How many students are in their 9th or higher study semester?

4. What does `typeof(euf_semesters)` return and why?