Name(s) of all authors: Ryu Sonoda, Divyansh Singh

Lab Journal Question 5

3/4/2022

Written/online sources used: Hehn, J., & Mendez, D. (n.d.). Combining Design Thinking and Software Requirements Engineering to create Human-centered Software-intensive Systems.

Help obtained (Acknowledgments): None

"We confirm that the above list of sources is complete AND that we have not talked to anyone else about the solution to this problem."

1. What does it mean *"Functions are for humans and computers"*. Why is that? What should you consider when writing a function?

   What does it mean *"Functions are for humans and computers"?*
   Although computers execute actual functions, they should be understood by both humans and computers. They should be defined keeping in mind both the usability for humans (as in, they should be simple, accessible, and encapsulate data well) and the limitations for computers (in terms of inputs provided to and outputs expected from them).

   Why is that?
   Because it makes it easier for other people to understand what the code is doing. Additionally, it improves maintainability of the code.

   What should you consider when writing a function?
   - The name of a function should be short and meaningful (ideally verb).
   - If the name is long, we can use snake_case or camelCase (Be consistent).
   - If functions are similar, use consistent names.
   - Avoid overwriting the existing functions.

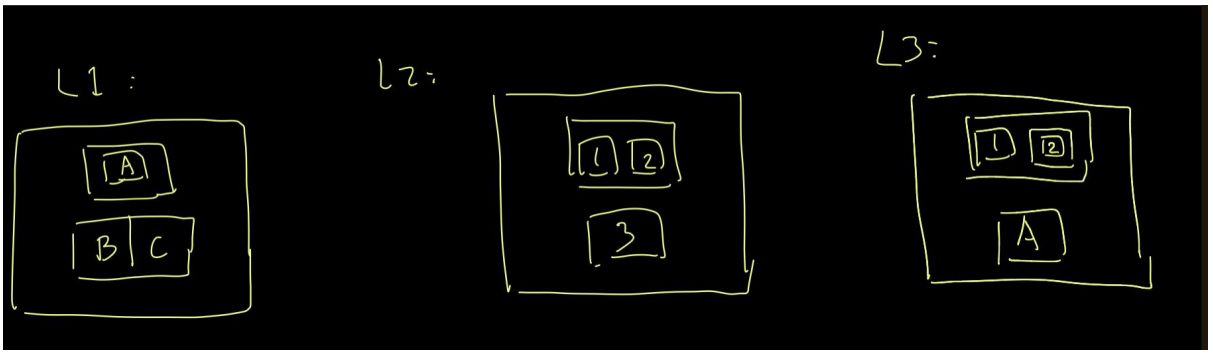
2. What does # do? What should you consider when using it?
   # allows us to add comments. In the comment, we should explain why we are writing the code rather than what it does or how it does.

3. What are the two key properties of a vector, complement your explanation providing your own examples (code).
   The first property is type. The elements in a vector must have the same type.
   Ex)
   vector_char = c('a', 'b', 'c')
   typeof(vector_char)
   result : `"character"`

The second property is length. Length indicates the number of elements in the vector.
Ex)
vector_char = c('a', 'b', 'c')
typeof(vector_char)
result : 3

4. Read Compare two numeric vectors. What does near() do?
   Near() allows us to safely compare two floating point numbers, as it has a built in tolerance for values that are close enough. Such comparisons are not possible with "==" as floating point numbers often involve rounding offs, thereby making comparisons that involve square roots and squares of even similar numbers unreliable.

5. Draw the visual representation of the following lists:
   - `L1 <- list(list('A'), c('B', 'C'))`
   - `L2 <- list(list(1, 2), 3)`
   - `L3 <- list(list(1, list(2)), 'A')`



6. Summarize chapter 21. What is this chapter about?
   The chapter 21 discusses how to write a clean and good quality code by using iteration. Using iteration can reduce the duplication of the code, which minimizes the mistakes while maximizing the maintainability. It talks about the three components of loops, namely,the output, the sequence and the body.
   The output - Sufficient space must be allocated for the output before the loop starts
   The sequence - This basically decides the list of values of which we need to iterate.
   The body - This is the block of code that decides the functions that need to be performed within the loop.

   Utilizing a loop is one way to achieve the goal but there are other several useful functions in R such as map() and walk(). By calling map(), we can apply a certain function to every element in a vector. Additionally, safely() allows us to check

where the error occurred when using map(). If the side effect is more important than the return value, we can use walk() instead of map().

7. What is Design Thinking? Why is it used in Software Development?

Design Thinking is a mindset and often non-technical approach to penetrating the problem space from the perspective of the user and delivering non-technical throw-away prototypes that allow for a better understanding of the problem early on. Design Thinking is a sentient innovation that uses the developer's tools to integrate people's needs, technological possibilities, and business success requirements.

The reason why it is used in software development is as the software development process got more complex, system and domain boundaries became ambiguous and human factors became more important. As a result, software development requires more problem-centric ways of thinking than technology-centric approaches. Since Design Thinking brings both problem-centric and human-centered ways of thinking, it is used in software development.

Hehn, J., & Mendez, D. (n.d.). Combining Design Thinking and Software Requirements Engineering to create Human-centered Software-intensive Systems.