**21083192**

**Hyunah Noh**

**Q.5 Write a report reflecting on what you think is the most significant activity within the unit and what it contributed to your learning regarding an ISYS5002 activity or topic. As an appendix to the report, please include all your weekly journals and discussion board posts.**

**Appendix**

**[Week01 Reflective journal]**

The emergence of COVID-19 in the Fourth Industrial Revolution era is rapidly changing the world, and we are all dependent on technology in our daily lives. Despite the fact that I have no background or knowledge in computers or programming, I believe that learning programming not only allows me to understand how a computer thinks but also helps me understand the steps to solving problems in my life.

The first week's lecture began with a distinction between hardware, which is a tangible component, and software, which controls a computer system so that users can use it. And in The Google Colab, we completed some commands that allow me to write and execute Python in my browser. By looking at a simple example and typing it, I was able to see how the code was executed. Although it is a high-level language that humans can write and understand, I discovered that simple errors such as spaces in the wrong places and missing punctuation render commands useless. I'm looking forward to taking this course in this era and believe that it will be my new challenge to improve my abstract thinking and problem-solving skills.

**[Week02 Reflective journal]**

In this week's class, we learned about the problem-solving methodology that consists of 6 steps and what a computer does. To understand what a computer does, we need to know how to store values in a variable, which is a symbolic name to hold data in computer memory, and how a computer operates this stored data. Python has five data types: int, float, str, bool, and list, and data types help to distinguish different types of values.

As a result, programmers should use these types correctly to avoid wasting computer memory. I was able to solve the problem and understand the example better after listening to the lecture and practising it in the next lab session. We learned the functions by applying three of the many Python functions to examples (input, output, and type). The most interesting aspect for me was that when we input the number into a variable, the Python interpreter reads it as a string. So, in order to calculate the numbers received as input functions, we must convert the string type to the int type. This problem stands out because I had to think about it for about ten minutes because it wasn't solved during the laboratory. We've only learned the variables and some functions that form the foundation of Python. As I learn more, I believe we'll be able to create a programme one day.

**[Week03 Reflective journal]**

There are things in writing code that we have to decide which block you need to perform or distinguish between to be executed. In this week of our computer laboratory, we've learned the If-else statement in Python to execute each part of a given condition when it's true.

Simply put, if a condition is true, then the body statement is executed and otherwise won't. Also, if there are several conditions to write down, use el-if statement. To understand this statement well, we applied it to calculate tax outcomes by changing income. Surprising things to me, except that when I tried to write code directly without pseudocode before, it wasn't easy to adapt programming language even if I thought it was a simple calculation. This moment reminds me why we should write pseudocode before; it is an important step in an algorithm. And next, as a random function, a built-in module, we stored random numbers on the list. In Python, there are a lot of convenient functions that we can add to use. Also, using a count-controlled for loop, we can contact each value of this list. For loop has a header specifying the iteration and body to execute per iteration. We can declare this loop to execute as much code as we want on the body. So, I guess we're going to use especially these two functions more.

**[Week04 Reflective journal]**

This week, we learned about Python's built-in functions, which support programming. Since almost functions are already configured in Python, we may call the functions whenever we need to. However, built-in functions cannot meet all requirements in our needs. The logic must be written in the function we constructed if we can't use built-in functions to execute and obtain some results. It begins with "def," the name of the function, a parameter if we require input, and then writes a statement on the following line. However, we must make a return at the end in order to receive the value determined by logic.

Using a while loop, we created a list of ten random even/odd numbers during the lab session. Additionally, the tax payable example that was practised in the prior weeks was used as a function. Although it was simple to implement as a using function, it was challenging to connect and make it perform flow. I found the answer by conducting a Google search, but I felt still had to experiment on my own to learn.

**[Week05 Reflective journal]**

Prior to this week, we had to create logic to check if the value format is appropriate using an if-else statement. As we've learned input validation, our programming was getting short and convenient with functions in Python. An isdigit() method returns true or false after checking the value's digits, which means we don't need to check the characters of values. And when errors occur, Python stops and shows us the message. In order to handle the error as we call it with no stop, there is the try-except statement in python. Normally, we write logic that we want to let test in the try block, and we can handle such error messages and distinguish error types in except blocks. We made logic to keep asking until getting the valid input by using the While and for loop statement and had to write down all the conditions to check the value.

But, PyInputPlus suggests many functions used to take various types of input. By using this module, we don't need to anticipate which types of value income; this module will keep asking the user till getting valid input. There are so many installed modules and functions too useful, so we use them properly. The next step is related to data. Almost all programs interact while saving data from users and processing data in the program. We open a File such as text,csv.. that stores data, save data using the write function and, of course, read data by print function. We must close a file after processing is complete.

Even though I used the graph function I had previously learnt to graph the data I had imported, I still believe there is a lot of room for improvement and practice when it comes to importing and processing particular types of data.

**[Week06 Reflective journal]**

This week, I've learned how to create modules and debug them. The former is customising or creating the module, as I understand. After creating, we need to download it on another page and import the name of the files to use modules. e.g., If I create functions with the name "calculate", I need to write down "import. Calculate" on the top of the page. And after that, we can use the functions like this "calculate.function\_name". The second thing starts by distinguishing between testing and debugging. Verifying and certifying that software or an application is bug-free is the process of testing. Debugging is the process of repairing a software issue. It is the process of locating, examining, and eliminating mistakes. This activity starts when the programme doesn't work as it should and ends after the issue has been fixed and the software has been tested successfully. Breakpoints are checkpoints that stop your scripts from running at certain lines. You can use the pauses to look at your experience and fix bugs, keep an eye on variables, and look at the call stack. If you put a breakpoint in a script at a certain line, the script stops before running that line. With the assert keyword, you can check if a condition in your code returns True. If it doesn't, the programme will throw an Assertion Error if the condition isn't true. This part was really hard to understand compared to what I've learned before. Especially, I didn't know about the use of the assert keyword because it was similar to If-statement to me. The example I learned during the laboratory is fairly hard to understand so I will practice more.

**Reflective Report**

In the last 12 weeks, we have had expanded learning of Programming with Python. It started from the basic concept of Programming to writing a program by doing practical approaching in the lab sessions. Firstly, we’ve learned the basics of concepts about computers such as hard/software, Input/Output, Pseudocode and Algorithms by doing simple practice. It was important and fundamental to assimilate computational thinking. And next for four weeks, through the practice of writing the code, I was able to understand the fragmentary concepts in more detail and depth. Also, by using Python language, I could learn the grammar and diverse modules that Python has. This process makes me recognize that writing down the code was an effective method more than reading and watching to understand the concepts and I could learn why following the problem-solving methodology that consists of 6 steps is important to write code effectively. After this, learning was expanded to using interfaces and databases, managing files, and drawing graphs based on the data. With these convenient modules, I can learn to use them easily and quickly. These experiences made me feel real programming. But I was surprised that there are a lot of smart functions and modules and it requires searching practice on Google to use proper modules. I briefed and summarized the 12 weeks as short as possible, and the most significant activity within this unit was. This course is designed to teach students with little or no prior programming skills and computational principles like me. Before I started the unit, I was worried if I could become familiar with the new field of programming which is a very professional and new field. Therefore, the theory-oriented lecturing time and the practice time immediately following it was very helpful in adapting and understanding programming. Starting with Python's grammatical concept, I was able to objectively feel which part I was good or bad at by applying various examples step by step. Features such as database and visualization that I learned during the last four or five years were able feel like I was working on a project and it was a visually interesting experience. Through this unit, I can gain confidence in their ability to apply programming tools and approaches to a wide range of challenges. Of course, this is not the end, and I need to study more about what I lacked such as algorithm and object-oriented, but it was beneficial to know what good programming is and what computational thinking is. And through this unit, was a time to think in detail about how to finish my master’s course well and what field I will work in. I believe that it was my best challenge to improve my computational thinking and problem-solving skills. Also, I can say that I have discipline knowledge and a range of other skills and capabilities from this learning.