Report on Today's Learning

Today was my fifth day of internship at Surfboard Payments, I learned about logical concepts in the morning and Python how run and what is python threads later in the day.

In the morning, I studied implications, biconditionals, conditionals, negations, conjunctions, and disjunctions. These concepts are essential in logic and help in understanding how statements are connected. An implication is a statement that expresses a conditional relationship, meaning if one thing happens, another thing follows. A biconditional is a statement that is true in both directions, meaning two conditions depend on each other. Conditionals describe cause-and-effect relationships, and negations represent the opposite of a given statement. Conjunctions combine two statements using "and," meaning both statements must be true for the whole statement to be true. Disjunctions combine two statements using "or," meaning at least one of the statements must be true for the whole statement to be true. I also learned how these concepts are connected and how logical relationships work in mathematics and programming. Understanding these logical relationships is important because they form the foundation of reasoning, problem-solving, and decision-making.

Later in the day, I explored how Python reads and executes code. Python is an interpreted language, meaning it reads and executes code line by line. If there is an error, Python immediately stops and shows an error message. These errors can be syntax errors, which occur due to incorrect code structure, or runtime errors, which happen while the program is running. Syntax errors occur when the code is written incorrectly and cannot be understood by the Python interpreter. Runtime errors, on the other hand, happen while the program is running due to unexpected conditions, such as dividing by zero or accessing an undefined variable. Learning how Python processes code and handles errors is crucial in debugging and writing error-free programs.

I also learned about threads in Python. A thread is like a worker that performs a task inside a program. By default, Python runs in a single-threaded mode, meaning tasks happen one after another in a sequence. This is called single-threading. In single-threading, one worker completes one task before moving to the next, which can be slow if a program has tasks that require waiting, such as downloading files or reading data from a database. In a single-threaded

program, every task must wait for the previous task to complete before starting.

Multi-threading allows multiple tasks to run at the same time, making programs more efficient. For example, in a restaurant, one worker takes orders, another cooks, and another serves food. This helps things get done faster. Python has a threading module to use multi-threading. But Python has a rule called the Global Interpreter Lock (GIL), which allows only one thread to run Python code at a time. So, multi-threading is good for tasks that involve waiting, like network requests, but not for tasks that require a lot of processing, like complex calculations. Because of the GIL, even if multiple threads are created, they do not run independently. Instead, they take turns. This means Python multi-threading is better for waiting tasks rather than for high-computation tasks.

today's learning both logical concepts and Python execution processes. I now understand how implications, biconditionals, conditionals, negations, conjunctions, and disjunctions work and their connectivity in logical reasoning. Additionally, I gained knowledge about Python threads, the difference between single-threading and multi-threading, and how Python handles multi-threading with the GIL. This knowledge will help me improve my programming skills and logical thinking in future projects. Learning these concepts has given me a better understanding of how logic is applied in programming and how Python executes tasks efficiently. Practicing these concepts through real-world examples will help me develop better problem-solving skills and improve my ability to write efficient code.