|  |  |
| --- | --- |
| **Activity No. (“Skill Test”)** | |
| **Course Code: 201L** | **Program: Computer Engineering** |
| **Course Title: Data structure Analysis** | **Date Performed: 8-30-25** |
| **Section: BSCPE 2-B** | **Date Submitted: 8-30-25** |
| **Name: Gabuyo, Ivan love D.** | **Instructor:** **ENGR. Maria Rizzete Sayo** |
| 1. **Objectives** | |
| * **To demonstrate the use of arrays in manipulating and processing textual data** This activity allows the user to explore how a fullname can be represented as an array of characters and then modified by inserting symbols just like underscores at the specific position, through this, learners gains hands-on experience with how arrays function as fundamental data structure that supports efficient storage, access, and modication of data programming. * **To strengthen problem-solving and logical thinking through array traversal and conditional operations** By traversing each character in the array and applying conditions (e.g., inserting an underscore before or after a chosen letter), students practice breaking down a programming task into smaller, manageable steps. This process not only reinforces technical skills in Python but also cultivates logical reasoning, attention to detail, and an understanding of how data structures are applied to real-world problems. | |
| **2. Discussion** | |
| The program is designed to demonstrate how arrays (lists in Python) can be used to manipulate strings by inserting characters at specific points. To make the implementation reusable, the logic is placed inside a function called insert\_underscore\_after\_O(). This function begins by converting the given fullname into a list of characters, which allows each letter to be processed individually. As the list is traversed, the code checks for the first occurrence of the letter "O". When it finds it, the program appends the "O" followed immediately by an underscore ("\_") into a new list. A Boolean flag ensures that only one underscore is added, preventing multiple insertions if more than one "O" exists in the fullname.  The second part of the function focuses on traversal and output. Here, the program prints the modified array with each character labeled by an index number. Spaces are intentionally skipped so that only meaningful characters are displayed in the index sequence, making the output more concise and easier to read. By wrapping the code inside a function, it becomes flexible and organized—allowing the same logic to be applied to different names simply by calling the function with another string. This activity highlights not only the technical aspects of array manipulation and traversal but also best practices in programming, such as modularity, reusability, and clarity of output. | |
| **3. Materials and Equipment** | |
| The main equipment used for this was a computer or laptop, which served as the primary tool to write, run, and test the program. This hardware provided the processing power necessary to execute the Python code and display the output in a structured manner.  For software, the activity utilized the Python programming language, which was chosen for its simplicity and efficiency in handling lists (arrays) and traversal operations. To provide a convenient and accessible coding environment, Google Colab was used. Colab allowed the program to be executed directly in the browser without additional installation, while also enabling easy file management through Google Drive. In addition, GitHub was employed as a version control and collaboration tool. By uploading the code to a GitHub repository, the work could be documented, shared, and managed in a professional way, reflecting real-world programming practices. | |
| **4. Procedure** | |
| To begin the activity, I first defined my fullname as "GABUYO IVAN LOVE D". Since strings in Python cannot be directly manipulated at the character level, I converted the string into a list of characters. This step was important because it allowed me to handle each character separately, almost like treating them as individual building blocks that I could rearrange or add onto. I also prepared a new empty list where I planned to build the modified version of my fullname.  The next step was to write a loop to traverse the list of characters one by one. As each character was processed, I appended it into the new list. However, I also set a special condition to check if the current character was the letter "O". The idea was that as soon as the program encountered the first "O", it should insert an underscore right after it. To avoid adding multiple underscores if there were other "O"s later in the name, I used a Boolean flag. This flag started as False and turned to True after the first insertion, ensuring only one underscore was ever added.  Once I had the logic for insertion in place, I focused on how to display the results. I created a second traversal process, where the program printed out each character in the new list along with its index number. I made sure that spaces were skipped in this process so the indexes would only count actual letters and the underscore. This made the output easier to read, with each letter clearly labeled by its position, almost like a map of where everything was located in the name.  Finally, I tested the code to confirm that it worked correctly. The output displayed the fullname with a single underscore neatly placed right after the first "O", and every character showed up with a clean index. This step-by-step process not only demonstrated how arrays and loops can work together but also gave me hands-on practice with conditions, index tracking, and even small details like skipping spaces for clarity. Overall, the procedure felt like building something piece by piece, carefully adding logic until the program produced exactly what I wanted. | |
| **5. Output** | |
|  | |
| 6. Conclusion In conclusion, I was able to apply arrays in a practical way through this activity. By converting my fullname into individual characters, inserting a single underscore after the first "O", and then traversing the updated list with indexes, I clearly saw how arrays can be used to manipulate strings step by step.  This exercise also helped me realize how important small details are in programming. Using a simple flag to make sure only one underscore was added showed me how even the simplest logic can control the outcome of a program. Skipping spaces in the index display made the results cleaner and easier to read, which reminded me that clarity in output is just as important as correctness.  Overall, I achieved the objectives of the activity and learned more about how arrays and traversal work. More than that, I gained a deeper understanding of how careful planning, problem-solving, and logical thinking come together to create a functional program. | |
|  | |