❼❽❻

|  |  |  |
| --- | --- | --- |
| **Name: Shawaal Nadeem**  **Reg No: L1F21BSCS0781**  **Teacher Name: Sir Buzdar** |  | **Total Marks: 05 Due Date: 24/03/2024**  **AI Lab F11 Wednesday** |

***Problem No.1***

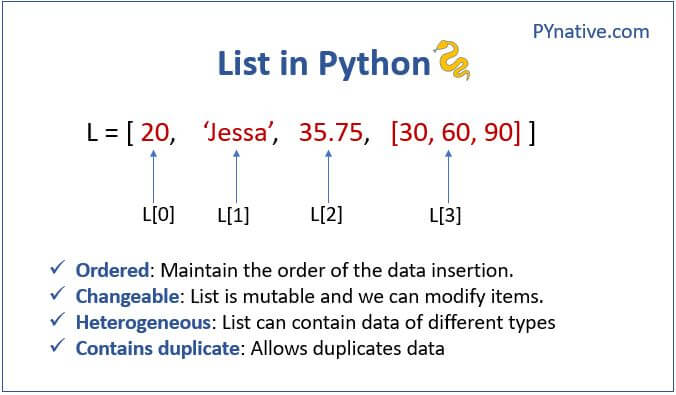
Explain the any five User-defined Data-structures form the given below in your own wording, with the help uses of internet, Chat GPT, Gemini or Copilot:

* **Arrays**: Similar to Lists, but store single type of elements
* **Stack**: Linear LIFO (Last-In-First-Out) Data structure
* **Queues**: Linear FIFO (First-In-First-Out) data structure
* **Trees**: Non-Linear data structures having a root and nodes
* **Linked Lists**: Linear data structures that are linked with pointers
* **Graphs**: Store a collection of points or nodes along with edges
* **Hash Maps**: In Python, Hash Maps are the same as Dictionaries

***Solution:***

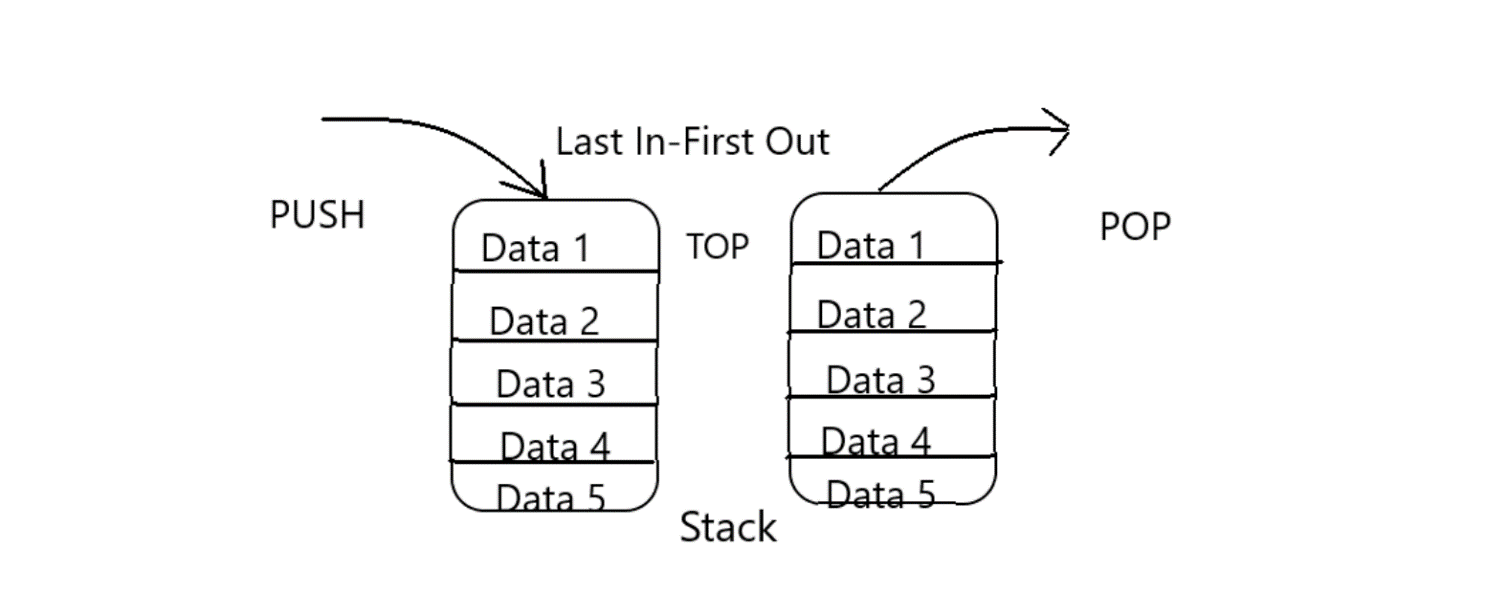
**Arrays:**

In Python, arrays aren't built-in data structures like in some other programming languages. However, you can achieve similar functionality using lists. The key thing to remember about arrays is that they hold elements of the same data type (e.g., all numbers, all strings). But in python lists we store multiple datatypes in list.



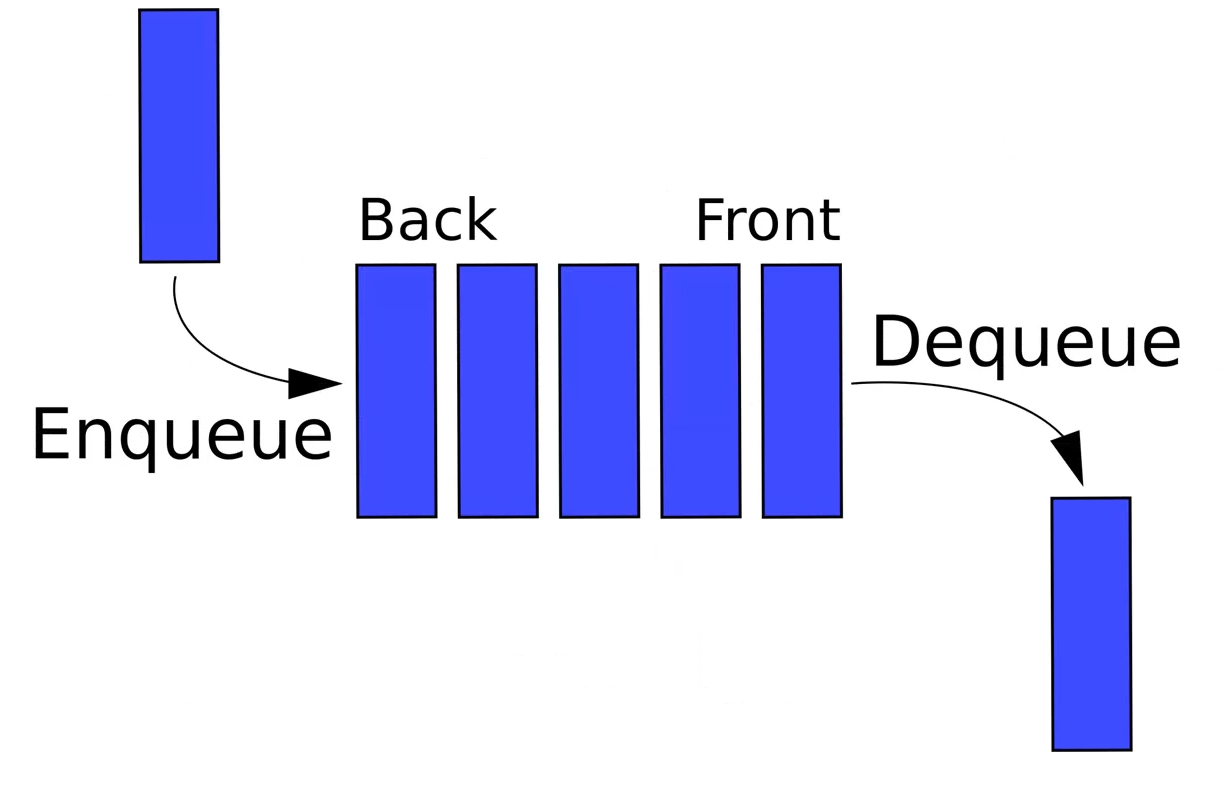
**Stack:**

Stacks are like a stack of plates: you can only add (push) and remove (pop) elements from the top. Imagine a stack of trays in a cafeteria - you can only put a new tray on top (push) and take the topmost one (pop) when you need it. Stacks are useful for keeping track of function calls (call stack) or implementing "undo/redo" functionality.



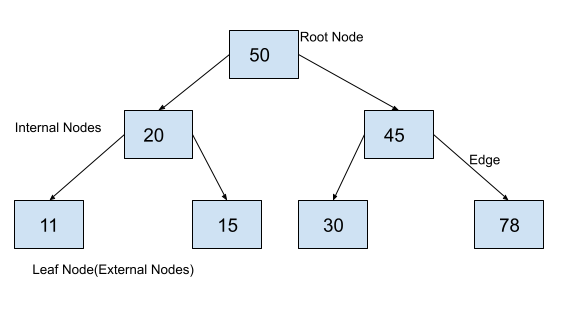
**Queues:**

Queues function like a waiting line: you add (enqueue) elements to the back and remove (dequeue) them from the front. Think of a line at a store - people join at the back (enqueue) and leave from the front (dequeue). Queues are often used for processing tasks in a specific order (e.g., printing jobs) or handling network requests.



**Trees:**

Trees are hierarchical structures that resemble an upside-down tree. They have a single root node at the top, and each node can have zero or more child nodes below it. This allows for representing relationships, like family trees, organizational charts, or file systems.



**Link List:**

Linked lists are linear data structures where elements (nodes) are not stored contiguously in memory. Each node contains data and a reference (pointer) to the next node in the list. This makes them dynamic, allowing for insertions or removals anywhere in the list without impacting the entire structure. Linked lists are useful for representing sequences where frequent modifications might occur.



**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* The End \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***