**FLOW OF FUNCTION EXCEUTION:**

The steps for the flow of functions execution are:

* **Function Call :** Execution begins when the function is called by a higher-level function or the program's entry point.
* **Stack Frame Creation**: A new stack frame, also known as an activation record or call frame, is created on the call stack to store information related to the function call. This stack frame contains space for function parameters, local variables, and the return address.
* **Parameter Passing:** If the function has parameters, the arguments supplied during the function call are copied into the stack frame for use within the function.
* **Local Variable Allocation:** Local variables declared within the function are allocated space in the stack frame. The initial values of these variables are set if they have explicit initializations.
* **Function Body Execution:** The function's body is executed sequentially, starting from the first statement. This includes performing operations, control flow (if statements, loops, etc.), and other logic defined within the function.
* **Return Statement:** If the function has a return statement, the result of the function is computed and stored in a designated location (e.g., a CPU register or memory). The return value may be void (no return value), a simple data type (e.g., int, string), or a reference type (e.g., an object).
* **Local Variable Cleanup:** Local variables go out of scope and are deallocated as the function execution reaches its end.
* **Return Address Restoration**: The return address, which points to the instruction to execute after the function call, is restored from the stack frame.
* **Stack Frame Removal:** The current stack frame is popped from the call stack. This effectively unwinds the function call, returning control to the caller function.
* **Return Value Passing:** If the function has a return value, it is passed back to the calling function.

The return value can be stored in a variable, used in expressions, or ignored if the function returns void.

* **Continued Execution:** Control returns to the calling function, which continues executing from where it left off. The return value from the called function can be used in subsequent operations or expressions.
* **Function Cleanup:** Any resources allocated within the function, like dynamically allocated memory, should be properly released (e.g., through Dispose for disposable objects).