Call by Reference & Call by value

Call by value

- In call by value method of parameter passing, the values of actual parameter are copied to the function's formal parameters.
- There are two copies of parameters stored in different memory locations.
- One is the original copy & the other is the function copy.
- Any changes made inside functions are not reflected in the actual parameters of the caller.
- In call by values, we can not alter the values of actual variables through function calls.
- This method is preferred when we have to pass some small values that should not change.
- Call by value is considered safer as original data is preserved.

Call by Reference

- In call by reference method of parameter passing, the address of the actual parameters is passed to the function as the formal parameters. In C, we use pointers to achieve call by reference.
- Both the actual & formal parameters refer to the same locations.
- Any changes made inside the function are actually reflected in the actual parameters of the caller.
- This method is preferred when we have to pass a large amount of data to the function.
- Call by reference is risky as it allows direct modification in original data.

Expt. No.

Call by Value & Call by Reference

→ These two ways are generally differentiated by the type of values passed to them as parameters.

→ The parameters passed to the function are called actual parameters

What are Actual & Formal Parameter in C

Actual parameters are the values that are passed to a function when it is called. They are also known as arguments. These values can be constants, variables, expressions, or even other function calls.

In C, actual parameters are enclosed in parantheses & separated by comma.

for eg: int result = add(2,3);

2, 3 are the actual parameters.

Formal Parameters in C:

formal parameters are the variable declared in the function header that are used to receive the values of the actual parameters passed during function calls. formal parameters are used to define the function signature.

Teacher's Signature _____

Comparison parameter	Python 2	Python 3
Year of Release	Python 2 was released in the year 2000.	Python 3 was released in the year 2008.
"Print" Keyword	In Python 2, print is considered to be a statement & not a function.	In Python 3, print is considered to be a function & not a statement.
Storage of Strings	In Python 2, strings are stored as ASCII by default.	In Python 3, strings are stored as UNICODE by default.
Division of Integers	On the division of two integers, we get an integral value in Python 2. For instance, 7/2 yields 3 in Python 2.	On the division of two integers, we get a floating point value in Python 3. For instance, 7/2 yields 3.5 in Python 3.
Exceptions	In Python 2, exceptions are enclosed in notations.	In Python 3, exceptions are enclosed in parantheses.
Variable leakage	The values of global variables do change in Python 2 if they are used inside a for-loop.	The value of variables never changes in Python 3.
Iteration.	In Python 2, the xrange() functions has been defined for iterations	In Python 3, the new range() function was introduced to perform iterations.

Ease of Syntax	Python 2 has more complicated Syntax than Python 3.	Python 3 has an easier Syntax compared to Python 2.
Libraries	A lot of libraries of Python 2 are not forward compatible.	A lot of libraries are created in Python 3 to be strictly used with Python 3.
Usage in today's times	Python 2 is no longer in use since 2020.	Python 3 is more popular than Python 2 & is still in use in today's times.
Backward compatibility	Python 2 codes can be ported to Python 3 with a lot of effort.	Python 3 is not backward compatible with Python 2.
Application	Python 2 was mostly used to become a DevOps Engineer. It is no longer in use after 2020.	Python 3 is used in a lot of fields like Software Engineering, Data Science, etc.