**Q1. Is an assignment operator like += only for show? Is it possible that it would lead to faster results at the runtime?**

No, the assignment operator += is not just for show, and it can have a practical purpose in programming. It is not about achieving faster results at runtime but rather providing a more concise and expressive way to update the value of a variable.

The += operator combines the addition (+) and assignment (=) operations into a single operation. It adds the value on the right-hand side of the operator to the existing value of the variable on the left-hand side and then assigns the result back to the variable.

**Q2. What is the smallest number of statements you'd have to write in most programming languages to replace the Python expression a, b = a + b, a?**

Minimum number of lines required to write above code in languages other Python will be 4, two for assigning initial values for variables a and b, and two for reassignment i.e. a=a+b and b=a.

**Q3. In Python, what is the most effective way to set a list of 100 integers to 0?**

my\_list = [0] \* 100

[0] \* 100 creates a list containing 100 zeros. The \* operator performs list replication, duplicating the [0] element 100 times. This approach is efficient because it avoids iterating over the list and setting each element to 0 individually.

**Q4. What is the most effective way to initialise a list of 99 integers that repeats the sequence 1, 2, 3? S If necessary, show step-by-step instructions on how to accomplish this.**

my\_list = [1,2,3]\*33

print(my\_list)

**Q5. If you're using IDLE to run a Python application, explain how to print a multidimensional list as efficiently?**

my\_list = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]

for sublist in my\_list:

for element in sublist:

print(element, end=" ")

print()

**Q6. Is it possible to use list comprehension with a string? If so, how can you go about doing it?**

Yes, it is possible to use list comprehension with a string.

my\_string = "Hello, World!"

new\_list = [character for character in my\_string]

print(new\_list)

**Q7. From the command line, how do you get support with a user-written Python programme? Is this possible from inside IDLE?**

**Get support with a user-written Python Programme**: Start a command prompt (Windows) or terminal window (Linux/Mac). If the current working directory is the same as the location in which you saved the file, you can simply specify the filename as a command-line argument to the Python interpreter.

**Get support with a User-written Python Program from IDLE:** You can also create script files and run them in IDLE. From the Shell window menu, select File → New File. That should open an additional editing window. Type in the code to be executed. From the menu in that window, select File → Save or File → Save As… and save the file to disk. Then select Run → Run Module. The output should appear back in the interpreter.

**Q8. Functions are said to be “first-class objects” in Python but not in most other languages, such as C++ or Java. What can you do in Python with a function (callable object) that you can't do in C or C++?**

The tasks which can be performed with the functions in python are:

A function is an instance of the Object type.

You can store the function in a variable.

You can pass the function as a parameter to another function.

You can return the function from a function.

You can store them in data structures such as hash tables, lists,

**Q9. How do you distinguish between a wrapper, a wrapped feature, and a decorator?**

A wrapper is a general term referring to a function or class that wraps around another feature to modify or extend its behaviour. The wrapped feature is the original function or class being wrapped. A decorator is a specific type of wrapper in Python used to modify the behaviour of functions or methods, defined using the @ syntax.

**Q10. If a function is a generator function, what does it return?**

Generator functions are a special kind of function that return a lazy iterator. These are objects that you can loop over like a list. However, unlike lists, lazy iterators do not store their contents in memory.

**Q11. What is the one improvement that must be made to a function in order for it to become a generator function in the Python language?**

Generator is a written as normal function but uses yield keyword to return values instead of return keyword.

**Q12. Identify at least one benefit of generators.**

return statement sends a specified value back to its caller whereas yield statment can produce a sequence of values. We should use generator when we want to iterate over a sequence, but don’t want to store the entire sequence in memory.