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

The += assignment operator is not just for show; it can lead to faster results at runtime for certain data types. For mutable objects like lists, += can modify the object in place, which can be more efficient than creating a new object.

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?

In most programming languages, you would need three statements to replace the Python expression a, b = a + b, a:

temp = a + b

a = temp

b = a

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

my\_list = [0] \* 100

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 = [(i % 3) + 1 for i in range(99)]

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

To print a multidimensional list efficiently in IDLE, you can use a loop to iterate through the rows and columns, and then use the print() function to display each element.

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

Yes, list comprehension can be used with a string.

my\_string = "Hello"

char\_list = [char for char in my\_string]

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

From the command line, you can get support for a user-written Python program by using the --help option, like python my\_program.py --help. This is not directly available from inside IDLE, but you can access documentation and help resources using the help() function or by typing commands in the interactive shell.

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++?

In Python, functions are first-class objects, which means you can:

* Assign functions to variables.
* Pass functions as arguments to other functions.
* Return functions from other functions.
* Store functions in data structures (like lists or dictionaries).

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

* A "wrapper" is a function that adds behavior to another function or callable object.
* A "wrapped feature" is the original function or object that is being extended or modified by the wrapper.
* A "decorator" is a special kind of wrapper that is applied using the @decorator\_name syntax before a function definition.

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

If a function is a generator function, it returns a generator object when called. A generator object is an iterator that generates values lazily and allows you to iterate over a sequence of values without loading them all into memory at once.

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?

To make a function a generator function in Python, you need to use the yield keyword instead of return to yield values one at a time during each iteration.

Q12. Identify at least one benefit of generators.

One benefit of generators is that they allow for lazy evaluation and memory efficiency. They generate values on-the-fly and only when needed, which can be very useful when dealing with large datasets or sequences. This reduces memory consumption and improves performance compared to creating and storing the entire sequence in memory.