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

It is not just for show. The += operator can lead to faster results at runtime, especially for mutable objects like lists. It performs in-place addition, modifying the existing object, which can be more memory-efficient.

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?

a, b = a + b, a # Equivalent statements in some languages (e.g., JavaScript):

# [a, b] = [a + b, a]; or (without destructuring) a = a + b; b = a - b; a = a - b;

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

my\_list = [0] \* 100 # Creates a list of 100 integers, all set to 0

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.

repeating\_sequence = [1, 2, 3]

my\_list = repeating\_sequence \* 33 # Repeats the sequence 33 times to get a list of 99 integers

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

matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]

for row in matrix:

print(row)

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

string\_variable = "hello"

char\_list = [char for char in string\_variable]

print(char\_list) # Output: ['h', 'e', 'l', 'l', 'o']

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

You can use the command 'python -h' for help or 'python script.py -h' for help on a specific script.

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, allowing them to be assigned to variables, passed as arguments to other functions, and returned from other functions.

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

Wrapper: A general term for an object or function that wraps around another.

Wrapped Feature: The original object or function being wrapped.

Decorator: A specific kind of wrapper in Python, often used with the @decorator syntax.

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

A generator function returns a generator object, which is an iterator.

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, use the 'yield' statement instead of 'return' to produce a series of values.

Q12. Identify at least one benefit of generators.

Generators provide lazy evaluation, producing values on-demand, saving memory and improving performance.