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

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?

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

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

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

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

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++?

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

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

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?

Q12. Identify at least one benefit of generators.

A1. The += operator is not just for show and can lead to faster results at runtime. When using +=, the object on the left-hand side is modified in place, rather than creating a new object. This can be faster than creating a new object each time.

A2. In most programming languages, you would need at least three statements to replace the Python expression a, b = a + b, a: a temporary variable to store the sum of a and b, an

assignment statement to set a to b, and another assignment statement to set b to the temporary variable.

A3. The most effective way to set a list of 100 integers to 0 in Python is to use a list comprehension:

python
```
my_list = [0 for _ in range(100)]
```

A4. The most effective way to initialize a list of 99 integers that repeats the sequence 1, 2, 3 is to use a list comprehension with the modulo operator:

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

A5. To print a multidimensional list efficiently in IDLE, you can use the pprint module, which can print lists in a more readable format:

lua
```
import pprint
my_list = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
pprint.pprint(my_list)
```

A6. Yes, it is possible to use list comprehension with a string. For example, you can use list comprehension to create a list of the ASCII values of each character in a string:

java
```
my_string = "Hello, world!"
my_list = [ord(char) for char in my_string]
```

A7. To get support with a user-written Python program from the command line, you can use the Python help() function or the pydoc module. From inside IDLE, you can use the same help() function or access the help documentation through the Help menu.

A8. In Python, functions are first-class objects, which means they can be passed as arguments to other functions, returned as values from functions, and assigned to variables. This allows for powerful functional programming techniques that are not possible in C or C++.

A9. A wrapper is a function or class that provides a layer of abstraction around another function or class. A wrapped feature is the original function or class that the wrapper is wrapping around. A decorator is a special type of wrapper that adds functionality to a function or class without modifying its code.

A10. If a function is a generator function, it returns a generator object, which can be used to iterate over a sequence of values. The values are generated on the fly as the generator is iterated over, rather than being generated all at once and stored in memory.

A11. The one improvement that must be made to a function in order for it to become a generator function in Python is to use the yield keyword instead of return to generate a sequence of values.

A12. One benefit of generators is that they allow for efficient memory usage, as they generate values on the fly rather than generating and storing all values in memory at once. This can be particularly useful for working with large datasets or infinite sequences. Additionally, generators can simplify code by allowing for a more declarative style of programming.