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

It saving three keystrokes including spaces ,Yes it will it will give faster results.

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=a+b

b=a

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

my\_list = []

for i in range(100):

my\_list.append(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?

markers = [(97,64),(45,84)]

result = ''.join(f'&markers={pair}' for pair in markers)

return result

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

Every Python installation comes with an **Integrated Development and Learning Environment**, which you’ll see shortened to IDLE or even IDE. These are a class of applications that help you write code more efficiently

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++?

A python **callable()** function in Python is something that can be called. This built-in function checks and returns True if the object passed appears to be callable, otherwise False.

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

Wrappers around the functions are also knows as decorators which are a very powerful and useful tool in Python since it allows programmers to modify the behavior of function or class. Decorators allow us to wrap another function in order to extend the behavior of the wrapped function, without permanently modifying it.

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

a generator is a function that returns **an object (iterator)** which we can iterate over (one value at a time).

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?

It is fairly simple to create a generator in Python. It is as easy as defining a normal function, but with a yield statement instead of a return statement.

Q12. Identify at least one benefit of generators.

* Generator functions allow you to declare a function that behaves like an iterator.
* They allow programmers to make an iterator in a fast, easy, and clean way.