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

Answer =

No it is not only for show.  
The += operator is not only more concise, but it can also be faster than the first code snippet. This is because some compilers and interpreters can optimize the += operator, resulting in faster code execution.

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?

Answer =

In most programming languages, you would need at least two statements 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?

Answer =

l = [0] \* 100  
print(l)

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 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.

Answer =

l1 = [1,2,3]  
count = int(101/len(l1))

l = l1 \* count   
print(l)

[1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3]

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

Answer =

by simply writing the name of the variable in a cell, we can print multidimensional list

l = [[1,2,3,4,5,6],['A','B', 'C','D'],[5,6,7,8]]  
print(l)

[[1, 2, 3, 4, 5, 6], ['A', 'B', 'C', 'D'], [5, 6, 7, 8]]

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

Answer =

List comprehension with a string is indeed possible in Python.

By repeatedly iterating through an already-existing iterable object and applying a certain operation or condition, list comprehension is a condensed method of producing a new list. Each character in a string is treated as a separate member of the iterable in this example.

C = 'CHINMAY PATIL'  
print([i\*2 for i in C])

['CC', 'HH', 'II', 'NN', 'MM', 'AA', 'YY', ' ', 'PP', 'AA', 'TT', 'II', 'LL']

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

Answer =

Step for command line:

1. Open command prompt.

2. Go to the directory containing python program file(.py)

3. Type "python file\_name.py", press enter to run the file.

Steps for python IDLE:

1. Open python IDLE.

2. Click on file>open>file\_name.py

3. Press F5 key to run the program.

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++?

Answer =

In Python, functions are like any other object, such as an int or a list. That means that you can use functions as arguments to other functions, store functions as dictionary values, or return a function from another function. This leads to many powerful ways to use functions.

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

Answer =

A wrapper, a wrapped feature, and a decorator are all same. They can be thought of as functions which modify the functionality of another function. They help to make your code shorter and more "Pythonic"

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

Answer =

If a function is a generator function, it does not return any values, instead, it return an iterator object, which we can iterate through to get values.

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?

Answer =

We can use "yield" keyword instead of "return" to make function a generator.

In Python, a generator function is defined with the yield keyword, which functions similarly to the return keyword. Instead of returning one result and ending, a function that uses the yield keyword transforms into a generator function that may produce a series of values on the fly.

def even\_nos(n):  
 for i in range(n):  
 if i % 2 == 0:  
 yield i  
for i in even\_nos(20):  
 print(i)

0

2

4

6

8

10

12

14

16

18

Q12. Identify at least one benefit of generators.

Answer =

While processing or iterating through a sequence of values in classical programming, lists or arrays are frequently used to store the values. Unfortunately, these data structures can easily become memory-intensive and slow down the computer when dealing with lengthy data sequences.

On the other hand, generators produce values instantly and don't need to keep the complete sequence in memory at once. This means that, especially when working with big data sets or endless sequences, generators are far more memory-efficient than conventional data structures like lists or arrays.