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.

Answer:

Q1. An assignment operator like += is not only for show and it can lead to faster results at runtime in certain situations. For instance, if you are concatenating strings in a loop, using the += operator to concatenate the strings can be more efficient than using the regular concatenation operator.

Q2. In most programming languages, you would need three statements to replace the Python expression a, b = a + b, a. These statements are typically as follows:

temp = a + b a = b b = temp

Q3. The most effective way to set a list of 100 integers to 0 in Python is to use a list comprehension. Here is an example:

my\_list = [0 for i in range(100)]

Q4. The most effective way to initialise a list of 99 integers that repeats the sequence 1, 2, 3 is to use the cycle function from the itertools module. Here is an example:

from itertools import cycle my\_list = [next(cycle([1, 2, 3])) for i in range(99)]

Q5. To print a multidimensional list efficiently in IDLE, you can use the pprint module. Here is an example:

from pprint import pprint my\_list = [[1, 2, 3], [4, 5, 6], [7, 8, 9]] pprint(my\_list)

Q6. Yes, it is possible to use list comprehension with a string in Python. Here is an example of how to use list comprehension to get a list of characters from a string:

my\_string = "hello" my\_list = [c for c in my\_string]

Q7. From the command line, you can get support with a user-written Python program by using the -h or --help option. For example, if your program is called myprogram.py, you can type the following command in the terminal to get help:

python myprogram.py -h

It is also possible to provide help within the program by using docstrings and the argparse module.

Yes, it is also possible to get support from inside IDLE. You can use the built-in help() function to get help on any Python object or module. For example, to get help on the range function, you can type the following command in the IDLE shell:

help(range)

Q8. In Python, functions are first-class objects, which means that they can be passed as arguments to other functions, returned as values from functions, and assigned to variables. This is not possible in languages like C or C++. In Python, you can use functions to create higher-order functions, which are functions that take other functions as arguments or return functions as values. This allows for more flexible and reusable code.

Q9. In Python, a wrapper is a function that takes another function as an argument and modifies its behavior without modifying its source code. A wrapped feature is a function that has been modified by a wrapper function. A decorator is a special kind of wrapper that uses the @ syntax to modify the behavior of a function.

Q10. If a function is a generator function in Python, it returns a generator object. A generator object is an iterator that produces a sequence of values on-the-fly as you iterate over it, rather than computing them all at once and storing them in memory.

Q11. The one improvement that must be made to a function in order for it to become a generator function is the use of the **yield** statement to produce a sequence of values.

Q12. One benefit of generators is that they can be used to generate values on-the-fly, allowing for memory-efficient processing of large data sets. This is because generators produce values lazily, only generating a new value when it is needed, rather than generating all values at once and storing them in memory. This can be particularly useful for working with large data sets or in situations where memory usage is a concern. Additionally, generators can be used to create infinite sequences of values, which is not possible with regular functions that return a fixed number of values.