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

The plus-equals operator += provides a convenient way to add a value to an existing variable and assign the new value back to the same variable. In the case where the variable and the value are strings, this operator performs string concatenation instead of addition.

There hardly is a difference in the work python performs for either statement:

Code:

>>> import dis

>>> def inplace\_add():

... a = 0

... a += 1

...

>>> def add\_and\_assign():

... a = 0

... a = a + 1

...

>>> dis.dis(inplace\_add)

2 0 LOAD\_CONST 1 (0)

3 STORE\_FAST 0 (a)

3 6 LOAD\_FAST 0 (a)

9 LOAD\_CONST 2 (1)

12 INPLACE\_ADD

13 STORE\_FAST 0 (a)

16 LOAD\_CONST 0 (None)

19 RETURN\_VALUE

>>> dis.dis(add\_and\_assign)

2 0 LOAD\_CONST 1 (0)

3 STORE\_FAST 0 (a)

3 6 LOAD\_FAST 0 (a)

9 LOAD\_CONST 2 (1)

12 BINARY\_ADD

13 STORE\_FAST 0 (a)

16 LOAD\_CONST 0 (None)

19 RETURN\_VALUE

The difference is a INPLACE\_ADD versus a BINARY\_ADD.

The resulting timings are too close to call which one would be faster:

>>> import timeit

>>> timeit.timeit('inplace\_add', 'from \_\_main\_\_ import inplace\_add', number=10000000)

0.32667088508605957

>>> timeit.timeit('add\_and\_assign', 'from \_\_main\_\_ import add\_and\_assign', number=10000000)

0.34172606468200684

So, in python, the difference is negligible.

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?

We definitely have to write more than one line of code in most programming languages to replicate the Python expression a, b = a + b, a. The following code is an example for C++ language:

Code:

#include <iostream>

using namespace std;

int main() {

int a, b;

a = 7;

b = 2;

// printing the sum of a and b

cout << "b= a + b = " << (a + b) << endl;

return 0;

}

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

Using list comprehensions:

It consists of square brackets containing an expression followed by a for clause and further followed by an optional if clause. The expression can be any type of object that we want to put on the list. Since we are initializing the list with zeros, our expression will just be 0.

arr = [0 for i in range(100)]

Using the \* operator:

The \* operator can be used as [object]\*n where n is the no of elements in the array.

arr = [0]\*100

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.

tot\_ele= int(input('enter the total number of elements in the list: ')) # to enter total elements in list

in\_seq= input('enter the sequence with comma as separation: ') # to enter the repeating sequence

seq= list(in\_seq.split(',') ) #splitting the sequence and storing it as list

times= tot\_ele//len(seq) # finding multiplication factor

new\_seq= seq \* times # creating the new sequence by list multiplication

out\_seq= [int(x) for x in new\_seq] #converting list elements from strings to integers, list comprehension

print(f'the required list is: {out\_seq}') # printing the final list

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

Just run the following code in IDLE:

a = [[2, 4, 6, 8, 10], [3, 6, 9, 12, 15], [4, 8, 12, 16, 20]]

for record in a:

print(record)

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

List comprehension in Python is an easy and compact syntax for creating a list from a string or another list. It is a very concise way to create a new list by performing an operation on each item in the existing list. List comprehension is considerably faster than processing a list using the for loop.

Code:

names = ['Steve', 'Bill', 'Ram', 'Mohan', 'Abdul']

names2 = [s for s in names if 'a' in s]

print(names2)

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

Using whatever editor you’ve chosen, create a script file called hello.py containing the following:

print("Hello, World!")

Now save the file, keeping track of the directory or folder you chose to save into.

Start a command prompt or terminal window. If the current working directory is the same as the location in which you saved the file, you can simply specify the filename as a command-line argument to the Python interpreter: python hello.py

Most Python installations contain a rudimentary IDE called IDLE. The name ostensibly stands for Integrated Development and Learning Environment. The same can be done from the IDLE also.

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 short, it means there are no restrictions on the object's use. It's the same as any other object.

A first class object is an entity that can be dynamically created, destroyed, passed to a function, returned as a value, and have all the rights as other variables in the programming language have.

Depending on the language, this can imply:

* being expressible as an anonymous literal value
* being storable in variables
* being storable in data structures
* having an intrinsic identity (independent of any given name)
* being comparable for equality with other entities
* being passable as a parameter to a procedure/function
* being returnable as the result of a procedure/function
* being constructible at runtime
* being printable
* being readable
* being transmissible among distributed processes
* being storable outside running processes

In C++ functions themselves are not first class objects, however:

* You can override the '()' operator making it possible to have an object function, which is first class.
* Function pointers are first class.
* boost bind, lambda and function do offer first class functions

In C++, classes are not first class objects but instances of those classes are. In Python both the classes and the objects are first class objects.

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.

When decorating an object, there is an attribute named \_\_wrapped\_\_ that holds the reference to the original one.

So now if we use this, we can ac­cess it di­rect­ly with­out hav­ing to re­sort to the old quirks.

wraps() is a decorator that is applied to the wrapper function of a decorator. It updates the wrapper function to look like wrapped function by copying attributes such as \_\_name\_\_, \_\_doc\_\_ (the docstring), etc.

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

In Python, a generator is a function that returns an iterator that produces a sequence of values when iterated over. Generators are useful when we want to produce a large sequence of values, but we don't want to store all of them in memory at once.

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?

The return statement must be replace by the yield statement.

In Python, similar to defining a normal function, we can define a generator function using the def keyword, but instead of the return statement we use the yield statement.

def generator\_name(arg):

# statements

yield something

Here, the yield keyword is used to produce a value from the generator.

Q12. Identify at least one benefit of generators.

Advantages of Python generators:

* Easier to build iterators using generators.
* They are memory efficient since they produce one item at a time. Read the proof.
* They can represent an infinite stream of data