Assignment 14 Solutions

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

Ans: `A=A+1` evaluates to finding `A`, adding 1 to it. Then storing the value again in variable `A`. This expression makes Python to look for memory holder of 'a' twice. But `A+=1` simply means value of `A` is to incremented by 1. As memory address has to be identified once, `+=` leads to faster operation.

Q2. What is the smallest no of statements you'd have to write in most programming languages to replace the Python expr `a, b = a + b, a` ?

Ans: Minimum number of lines required to write above code in languages other Python will be 4, two for assigning initial values for variables `a` and `b`, and two for reassignment i.e. `a=a+b` and `b=a`.

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

Ans: The Most effective way to set a list of 100 integers to 0 in python is by using repition operator(`*`) or by using list comprehension.

In [1]:

```
# Method 1
list_zero=[0]*100
print(list_zero)
```

```
[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, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

In [2]:

```
# Method 2
zero_list = [0 for x in range(100)]
print(zero_list)
```

```
[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, 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.

In [3]:

```
my_list = [1,2,3]*33
print(my_list)
```

```
[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?

In [4]:

```
m = 4
n = 5

a = [[0 for x in range(n)] for x in range(m)]
print(a)
```

```
[[0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0]]
```

In [5]:

```
my_list = [[1,1],[2,2],[3,3],[4,4],[5,5]] # 2 dimensional List
for x in range(len(my_list)):
    for y in range(len(my_list[x])):
        print(my_list[x][y],end=" ")
```

```
1 1 2 2 3 3 4 4 5 5
```

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

Ans: List comprehension with string is possible.

In [6]:

```
my_list = [ele for ele in 'iNeuron']
print(my_list)
```

```
['i', 'N', 'e', 'u', 'r', 'o', 'n']
```

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

Ans: Get support with a user-written Python Programme: Start a command prompt (Windows) or terminal window (Linux/Mac). 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.

Get support with a User-written Python Program from IDLE: You can also create script files and run them in IDLE. From the Shell window menu, select **`File → New File`** . That should open an additional editing window. Type in the code to be executed. From the menu in that window, **`select File → Save or File → Save As…`** and save the file to disk. Then **`select Run → Run Module`** . The output should appear back in the interpreter

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++?

Ans: The tasks which can be performed with the functions in python are:

A function is an instance of the Object type. You can store the function in a variable. You can pass the function as a parameter to another function. You can return the function from a function. You can store them in data structures such as hash tables, lists,

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

Ans: Wrappers Around the functions are known as Decrators

Decorators allow us to wrap another function in order to extend the behavior of the wrapped function, without permanently modifying it. In Decorators, functions are taken as the argument into another function and then called inside the wrapper function.

@wrapper

def function(n):

statements(s) similar to

def function(n):

statement(s) function = wrapper(function)

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

Ans: Generator functions are a special kind of function that return a **lazy iterator**. These are objects that you can loop over like a list. However, unlike lists, lazy iterators do not store their contents in memory.

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? ¶

Ans: Generator is a written as normal function but uses **`yield`** keyword to return values instead of **`return`** keyword.

Q12. Identify at least one benefit of generators.

Ans: **`return`** statement sends a specified value back to its caller whereas **`yield`** statment can produce a sequence of values. We should use generator when we want to iterate over a sequence, but don’t want to store the entire sequence in memory.