TYPE-B:LIST MANIPULATIONS:CH-11

1) What is the difference between following two expressions, if lst is given as [1, 3, 5]

(i) lst * 3 and lst *= 3
(ii) lst + 3 and lst += [3]

sol:
i) lst * 3 vs lst *= 3
lst * 3 → creates a new list, original lst unchanged.
lst *= 3 → modifies the original list, repeating it 3 times.

(ii) lst + 3 vs lst += [3]
lst + 3 → ✗ error, cannot add a number directly to a list
lst += [3] → adds 3 to the list, modifies the original list.

2) Given two lists:
L1 = ["this", 'is', 'a', 'List'], L2 = ["this", ["is", "another"], "List"]

Which of the following expressions will cause an error and why?

1. L1 == L2
2. L1.upper( )
3. L1[3].upper( )
4. L2.upper( )
5. L2[1].upper( )
6. L2[1][1].upper( )

sol:

Expression	Works / Error	Reason
L1 == L2	Works ✓	List comparison is allowed
L1.upper()	Error ✗	Lists don't have upper()
L1[3].upper()	Works ✓	L1[3] is a string
L2.upper()	Error ✗	Lists don't have upper()
L2[1].upper()	Error ✗	L2[1] is a list
L2[1][1].upper()	Works ✓	L2[1][1] is a string

3) From the previous question, give output of expressions that do not result in error.

sol:

Expression	Output	Notes
L1 == L2	False	Compares lists element by element

L1[3].upper()	'LIST'	L1[3] is string "List"
L2[1][1].upper()	'ANOTHER'	L2[1][1] is string "another"
L1.upper()	Error ✗	Lists do not have `.upper()`
L2.upper()	Error ✗	Lists do not have `.upper()`
L2[1].upper()	Error ✗	L2[1] is a list, not a string

4) Given a list L1 = [3, 4.5, 12, 25.7, [2, 1, 0, 5], 88]

1. Which list slice will return [12, 25.7, [2, 1, 0, 5]]?
2. Which expression will return [2, 1, 0, 5]?
3. Which list slice will return [[2, 1, 0, 5]]?
4. Which list slice will return [4.5, 25.7, 88]?

sol:

1. [12, 25.7, [2, 1, 0, 5]] → L1[2:5]
2. [2, 1, 0, 5] → L1[4]
3. [[2, 1, 0, 5]] → L1[4:5]
4. [4.5, 25.7, 88] → L1[1:6:2]

5) Given a list L1 = [3, 4.5, 12, 25.7, [2, 1, 0, 5], 88], which function can change the list to:

1. [3, 4.5, 12, 25.7, 88]
2. [3, 4.5, 12, 25.7]
3. [[2, 1, 0, 5], 88]

sol:

[3, 4.5, 12, 25.7, 88] → L1.pop(4) or del L1[4]
[3, 4.5, 12, 25.7] → L1.pop(5) (after previous step) or del L1[4:]
[[2, 1, 0, 5], 88] → L1[0:4] = [] or del L1[0:4]

6) What will the following code result in?

```
L1 = [1, 3, 5, 7, 9]
print (L1 == L1.reverse( ) )
print (L1)
```

sol:

```
False
[9, 7, 5, 3, 1]
```

7) Predict the output:

```
my_list= [ 'p', 'r', 'o', 'b', 'l' , 'e', 'm']
my_list[2:3] = []
print(my_list)
my_list[2:5] = []
print(my_list)
```

sol:

['p', 'r', 'b', 'l', 'e', 'm']
['p', 'r', 'm']

8) Predict the output:

```
List1 = [13, 18, 11, 16, 13, 18, 13]
print(List1.index(18))
print(List1.count(18))
List1.append(List1.count(13))
print(List1)
```

sol:
1
2
[13, 18, 11, 16, 13, 18, 13, 3]

9) Predict the output:

```
Odd = [1,3,5]
print( (Odd +[2, 4, 6])[4] )
print( (Odd +[12, 14, 16])[4] - (Odd +[2, 4, 6])[4] )
```

sol:
4
10

10) Predict the output:

```
a, b, c = [1,2], [1, 2], [1, 2]
print(a == b)
print (a is b)
```

sol:
True
False

11)(a) L1, L2 = [2, 4] , [2, 4]

```
L3 = L2
L2[1] = 5
print(L3)
```

sol:
[2, 5]

11)(b) L1, L2 = [2, 4] , [2, 4]

```
L3 = list(L2)
L2[1] = 5
print(L3)
```

sol:

[2, 4]

12) Find the errors:

1. L1 = [1, 11, 21, 31]
2. L2 = L1 + 2
3. L3 = L1 * 2
4. Idx = L1.index(45)

sol:

1. L1 = [1, 11, 21, 31] → Correct, no error.
2. L2 = L1 + 2 → Error; cannot add a list and an integer. Correct: L2 = L1 + [2].
3. L3 = L1 * 2 → Correct, repeats the list.
4. Idx = L1.index(45) → Error; 45 is not in the list. Correct: use a value that exists, e.g., Idx = L1.index(21).

13)(a) Find the errors:

```
L1 = [1, 11, 21, 31]
An = L1.remove(41)
```

sol:

- L1 = [1, 11, 21, 31] → correct, no error
- An = L1.remove(41) → error, 41 is not in the list
- remove() method does not return a value, so assigning it to An will make An = None

13)(b) Find the errors:

```
L1 = [1, 11, 21, 31]
An = L1.remove(31)
print(An + 2)
```

sol:

- L1 = [1, 11, 21, 31] → correct, no error
- An = L1.remove(31) → remove() does not return a value, it modifies the list in place, so An becomes None
- print(An + 2) → error, cannot add None and an integer

14)(a) Find the errors:

```
L1 = [3, 4, 5]
L2 = L1 * 3
print(L1 * 3.0)
print(L2)
```

sol:

The error is in print(L1 * 3.0) because you cannot multiply a list by a float. List repetition only works with an integer, so you should use something like L1 * 3.

14)(b) Find the errors:

```
L1 = [3, 3, 8, 1, 3, 0, '1', '0', '2', 'e', 'w', 'e', 'r']
print(L1[: :-1])
print(L1[-1:-2:-3])
print(L1[-1:-2:-3:-4])
```

sol:
The error is in L1[-1:-2:-3:-4] because a slice can have only one step; having two steps causes a syntax error.

15) What will be the output of following code?

```
x = ['3', '2', '5']
y = ''
while x:
   y = y + x[-1]
   x = x[:len(x) - 1]
print(y)
print(x)
print(type(x), type(y))
```

sol:
523
[]
<class 'list'> <class 'str'>

16) Complete the code to create a list of every integer between 0 and 100, inclusive, named nums1 using Python, sorted in increasing order.

sol:
nums1 = list(range(101))

17) Let nums2 and nums3 be two non-empty lists. Write a Python command that will append the last element of nums3 to the end of nums2.

sol:
nums2.append(nums3[-1])

18) Consider the following code and predict the result of the following statements.

```
bieber = ['om', 'nom', 'nom']
counts = [1, 2, 3]
nums = counts
nums.append(4)
```

1. counts is nums
2. counts is add([1, 2], [3, 4])

sol:

1. counts is nums → True
2. counts is add([1,2],[3,4]) → Error / False

19) What is the output of the following code?

```
numbers = list(range(0, 51, 4))
results = []
for number in numbers:
   if not number % 3:
      results.append(number)
print(results)
```

sol:

[0, 12, 24, 36, 48]

20) Following code prints the given list in ascending order. Modify the code so that the elements are printed in the reverse order of the result produced by the given code.

```
numbers = list(range(0, 51, 4))
i = 0
while i < len(numbers):
   print(numbers[i] , end = " ")
   i += 3
# gives output as : 0 12 24 36 48
```

sol:

```
numbers = list(range(0, 51, 4))
i = len(numbers) - 1    # start from the last index
while i >= 0:
   print(numbers[i], end=" ")
   i -= 3
```