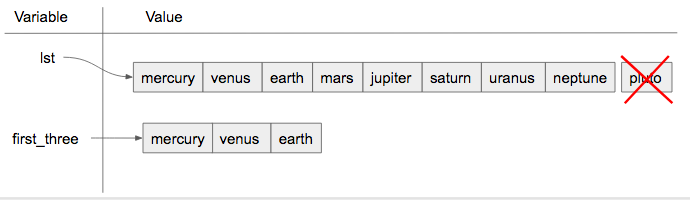
Questions

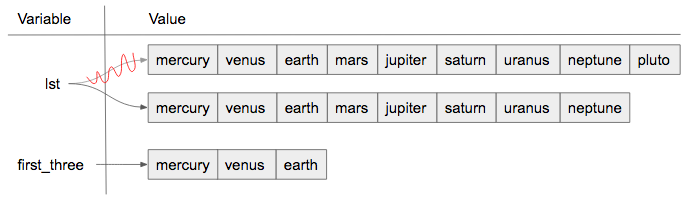
seqmut-1-1: Which of these is a correct reference diagram following the execution of the following code?

lst = ['mercury', 'venus', 'earth', 'mars', 'jupiter', 'saturn', 'uranus', 'neptune', 'pluto']

lst.remove('pluto')

first\_three = lst[:3]





Top of Form

A. I.  
B. II.  
C. Neither is the correct reference diagram.  
Check meCompare me

Bottom of Form

✔️ Yes, when we are using the remove method, we are just editing the existing list, not making a new copy.

**Not yet graded**

seqmut-1-4: What will be the value of a after the following code has executed?

a = ["holiday", "celebrate!"]

quiet = a

quiet.append("company")

The value of a will be : ["holiday", "celebrate!", "company"]



seqmut-1-5: Could aliasing cause potential confusion in this problem?

b = ['q', 'u', 'i']

z = b

b[1] = 'i'

z.remove('i')

print(z)

Top of Form

A. yes  
B. no  
Check meCompare me

Bottom of Form

✔️ Yes, b and z reference the same list and changes are made using both aliases.

seqmut-1-13: Given that we want to accumulate the total sum of a list of numbers, which of the following accumulator patterns would be appropriate?

nums = [4, 5, 2, 93, 3, 5]

s = 0

for n in nums:

s = s + 1

nums = [4, 5, 2, 93, 3, 5]

s = 0

for n in nums:

s = n + n

nums = [4, 5, 2, 93, 3, 5]

s = 0

for n in nums:

s = s + n

Top of Form

A. I.  
B. II.  
C. III.  
D. none of the above would be appropriate for the problem.

Bottom of Form

seqmut-1-14: Given that we want to accumulate the total number of strings in the list, which of the following accumulator patterns would be appropriate?

lst = ['plan', 'answer', 5, 9.29, 'order, items', [4]]

s = 0

for n in lst:

s = s + n

lst = ['plan', 'answer', 5, 9.29, 'order, items', [4]]

for item in lst:

s = 0

if type(item) == type("string"):

s = s + 1

lst = ['plan', 'answer', 5, 9.29, 'order, items', [4]]

s = ""

for n in lst:

s = s + n

lst = ['plan', 'answer', 5, 9.29, 'order, items', [4]]

s = 0

for item in lst:

if type(item) == type("string"):

s = s + 1

Top of Form

A. 1.  
B. 2.  
C. 3.  
D. 4.  
E. none of the above would be appropriate for the problem.

Ans:D Bottom of Form

seqmut-1-15: Which of these are good names for an accumulator variable? Select as many as apply.

Top of Form

A. sum  
B. x  
C. total  
D. accum  
E. none of the above  
Check meCompare me

Bottom of Form

✔️Correct.

1. Yes, total is a good name for accumulating numbers.
2. Yes, accum is a good name. It's both short and easy to remember.

seqmut-1-16: Which of these are good names for an iterator (loop) variable? Select as many as apply.

Top of Form

A. item  
B. y  
C. elem  
D. char  
E. none of the above  
Check meCompare me

Bottom of Form

✔️Correct.

1. Yes, item can be a good name to use as an iterator variable.
2. Yes, elem can be a good name to use as an iterator variable, especially when iterating over lists.
3. Yes, char can be a good name to use when iterating over a string, because the iterator variable would be assigned a character each time.

seqmut-1-17: Which of these are good names for a sequence variable? Select as many as apply.

Top of Form

A. num\_lst  
B. p  
C. sentence  
D. names  
E. none of the above  
Check meCompare me

Bottom of Form

✔️Correct.

1. Yes, num\_lst is good for a sequence variable if the value is actually a list of numbers.
2. Yes, this is good to use if the for loop is iterating through a string.
3. Yes, names is good, assuming that the for loop is iterating through actual names and not something unrelated to names.
4. seqmut-1-18: Given the following scenario, what are good names for the accumulator variable, iterator variable, and sequence variable? You are writing code that uses a list of sentences and accumulates the total number of sentences that have the word ‘happy’ in them.
5. Top of Form
6. A. accumulator variable: x | iterator variable: s | sequence variable: lst  
   B. accumulator variable: total | iterator variable: s | sequence variable: lst  
   C. accumulator variable: x | iterator variable: sentences | sequence variable: sentence\_lst  
   D. accumulator variable: total | iterator variable: sentence |sequence variable: sentence\_lst  
   E. none of the above
7. Bottom of Form

ans: D

For each character in the string saved in ael, append that character to a list that should be saved in a variable app.

ael = "python!"

app=list()

for a in ael:

app.append(a)

print(app)

For each string in wrds, add ‘ed’ to the end of the word (to make the word past tense). Save these past tense words to a list called past\_wrds.

wrds = ["end", 'work', "play", "start", "walk", "look", "open", "rain", "learn", "clean"]

past\_wrds=[]

for word in wrds:

past\_wrds.append(word+"ed")

print(past\_wrds)