course_1_assessment_11

Due: 2018-11-25 01:25:00

Description: Assessment for Way of Programmer Week four.   Score: 2.0 of 11 = 18.2%
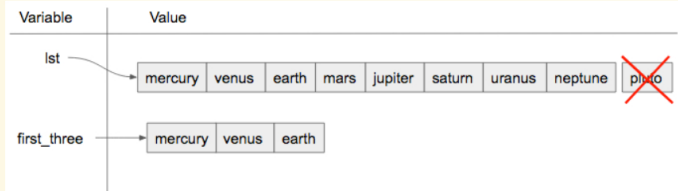
Questions

seqmut-1-1: Which of these is a correct reference diagram following the execution of the following code?   Score: 1.0 / 1
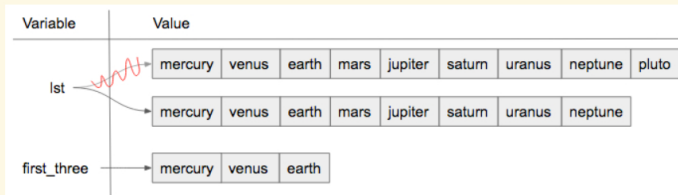
Comment: autograded

```
lst = ['mercury', 'venus', 'earth', 'mars', 'jupiter', 'saturn', 'uranus', 'neptune', 'pluto']
lst.remove('pluto')
first_three = lst[:3]
```

1.

2.

- ⦿ A. I.
- ◯ B. II.
- ◯ C. Neither is the correct reference diagram.

[Check me] [Compare me]

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

Multiple Choice (assess_question4_1_1_1)

seqmut-1-4: What will be the value of a after the following code has executed?   Not yet graded

```
a = ["holiday", "celebrate!"]
quiet = a
quiet.append("company")
```

The value of a will be

[['holiday', 'celebrate!', 'compa]

[Check me] [Compare me]

Good work!

Fill in the Blank (assess_question3_3_1_1)

seqmut-1-5: Could aliasing cause potential confusion in this problem?   Score: 1.0 / 1

Comment: autograded

```
b = ['q', 'u', 'i']
z = b
b[1] = 'i'
z.remove('i')
print(z)
```

- ⦿ A. yes
- ◯ B. no

[Check me] [Compare me]

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

Multiple Choice (assess_question3_3_1_2)

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?   Not yet graded

1.

```
nums = [4, 5, 2, 93, 3, 5]
s = 0
```

```
    for n in nums:
        s = s + 1
```

2.

```
nums = [4, 5, 2, 93, 3, 5]
s = 0
for n in nums:
    s = n + n
```

3.

```
nums = [4, 5, 2, 93, 3, 5]
s = 0
for n in nums:
    s = s + n
```

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

[Check me] [Compare me]

✔ Yes, this will solve the problem.

Multiple Choice (assess_question5_2_1_1)

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?   **Not yet graded**

1.

```
lst = ['plan', 'answer', 5, 9.29, 'order, items', [4]]
s = 0
for n in lst:
    s = s + n
```

2.

```
lst = ['plan', 'answer', 5, 9.29, 'order, items', [4]]
for item in lst:
    s = 0
    if type(item) == type("string"):
        s = s + 1
```

3.

```
lst = ['plan', 'answer', 5, 9.29, 'order, items', [4]]
s = ""
for n in lst:
    s = s + n
```

4.

```
lst = ['plan', 'answer', 5, 9.29, 'order, items', [4]]
s = 0
for item in lst:
    if type(item) == type("string"):
        s = s + 1
```

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

[Check me] [Compare me]

✔ Yes, this will solve the problem.

Multiple Choice (assess_question5_2_1_2)

seqmut-1-15: Which of these are good names for an accumulator variable? Select as many as apply.   **Not yet graded**

☐ A. sum
☐ B. x
☑ C. total
☑ D. accum
☐ E. none of the above

[Check me] [Compare me]

✔ Correct.
   C. Yes, total is a good name for accumulating numbers.
   D. Yes, accum is a good name. It's both short and easy to remember.

Multiple Choice (assess_question5_2_1_3)

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

☑A. item
☐B. y
☑C. elem
☑D. char
☐E. none of the above

Check me **Compare me**

✔ Correct.
    A. Yes, item can be a good name to use as an iterator variable.
    C. Yes, elem can be a good name to use as an iterator variable, especially when iterating over lists.
    D. 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.

Multiple Choice (assess_question5_2_1_4)

---

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

☑A. num_lst
☐B. p
☑C. sentence
☑D. names
☐E. none of the above

Check me **Compare me**

✔ Correct.
    A. Yes, num_lst is good for a sequence variable if the value is actually a list of numbers.
    C. Yes, this is good to use if the for loop is iterating through a string.
    D. Yes, names is good, assuming that the for loop is iterating through actual names and not something unrelated to names.

Multiple Choice (assess_question5_2_1_5)

---

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.

○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

Check me **Compare me**

✔ Yes, this combination of variable names is the clearest.

Multiple Choice (assess_question5_2_1_6)

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

Save & Run   5/22/2020, 10:08:58 PM - 2 of 2   **Show in CodeLens**

```
1 ael = "python!"
2 app = []
3 for w in ael:
4     app.append(w)
5 print(app)
6
```

['p', 'y', 't', 'h', 'o', 'n', '!']

ActiveCode (access_ac_5_2_1_1)

Result	Actual Value	Expected Value	Notes	
Pass	['p',... '!']	['p',... '!']	Testing that app has the correct elements.	Expand Differences
Pass	'append'	'ael =...app)\n'	Testing that your code uses append.	Expand Differences

You passed: 100.0% of the tests

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`.

Save & Run   5/22/2020, 10:09:14 PM - 2 of 2   Show in CodeLens

```
1  wrds = ["end", 'work', "play", "start", "walk", "look", "open", "rain", "learn", "clean"]
2
3  past_wrds = [word + "ed" for word in wrds]
4  print(past_wrds)
```

['ended', 'worked', 'played', 'started', 'walked', 'looked', 'opened', 'rained', 'learned', 'cleaned

ActiveCode (access_ac_5_2_1_2)

Result	Actual Value	Expected Value	Notes	
Pass	['end...ned']	['end...ned']	Testing that past_wrds has the correct value.	Expand Differences
Pass	'for '	'wrds ...wrds)'	Testing that your code uses a for loop.	Expand Differences

You passed: 100.0% of the tests

Score Me