



course_1_assessment_8

Due: 2018-11-25 01:22:00

Description: Assessment for Sequence Mutation lesson.

Score: 5.0 of 5 = 100.0%

Questions

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

Score: 1.0 / 1

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

Comment: autograded

☒ 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-1-6: Could aliasing cause potential confusion in this problem?

Score: 1.0 / 1

```
sent = "Holidays can be a fun time when you have good company!"  
phrase = sent  
phrase = phrase + " Holidays can also be fun on your own!"
```

Comment: autograded

☐ A. yes

☒ B. no

Check me

Compare me

✓ Since a string is immutable, aliasing won't be as confusing. Beware of using something like `item = item + new_item` with mutable objects though because it creates a new object. However, when we use `+=` then that doesn't happen.

Multiple Choice (assess_question3_3_1_4)

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

Score: 1.0 / 1

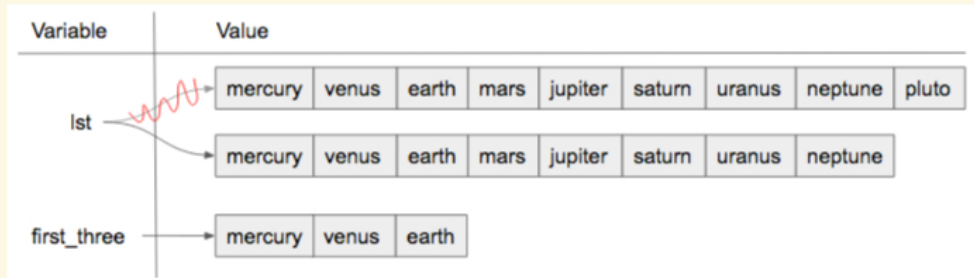
```
lst = ['mercury', 'venus', 'earth', 'mars', 'jupiter', 'saturn', 'uranus', 'neptune', 'pluto']
lst.remove('pluto')
first_three = lst[:3]
```

Comment: autograded

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)

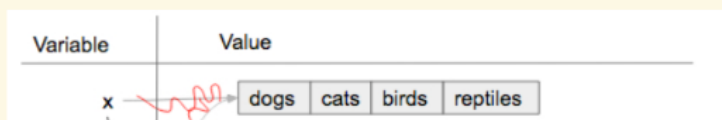
seqmut-1-7: Which of these is a correct reference diagram following the execution of the following code?

Score: 1.0 / 1

```
x = ["dogs", "cats", "birds", "reptiles"]
y = x
x += ['fish', 'horses']
y = y + ['sheep']
```

Comment: autograded

1.





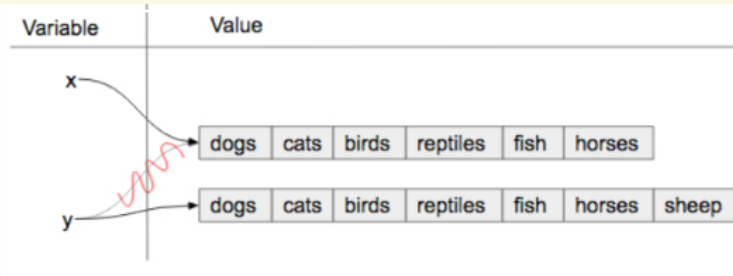
2.



3.



4.



- ☐ A. I.
- ☐ B. II.
- ☐ C. III.
- ☒ D. IV.

Check me

Compare me

✔ Yes, the behavior of `obj = obj + object_two` is different than `obj += object_two` when `obj` is a list. The first version makes a new object entirely and reassigns to `obj`. The second version changes the original object so that the contents of `object_two` are added to the end of the first.

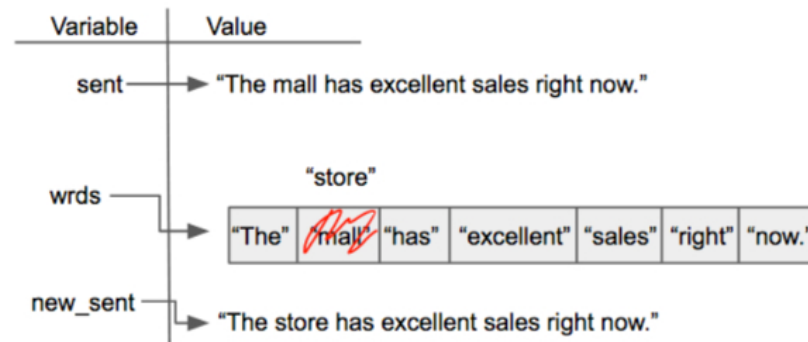
seqmut-1-8: Which of these is a correct reference diagram following the execution of the following code?

Score: 1.0 / 1

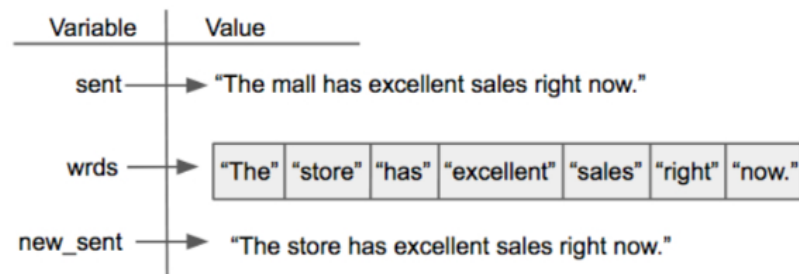
Comment: autograded

```
sent = "The mall has excellent sales right now."  
wrds = sent.split()  
wrds[1] = 'store'  
new_sent = " ".join(wrds)
```

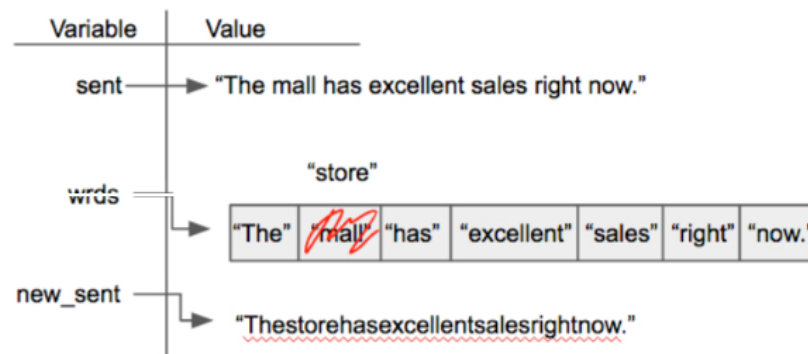
1.



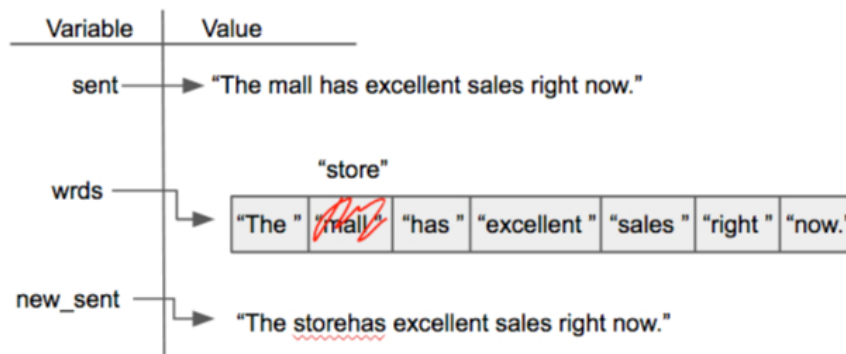
2.



3.



4.



- ☒ A. I.
- ☐ B. II.
- ☐ C. III.
- ☐ D. IV.

Check me

Compare me

✔ Yes, when we make our own diagrams we want to keep the old information because sometimes other variables depend on them. It can get cluttered though if there is a lot of information.

Multiple Choice (assess_question4_1_3_1)

Score Me