# Quiz 06

**Due** Oct 18 at 10pm **Points** 10 **Questions** 5 **Time Limit** None

## Instructions

Answer the following questions in your own words. Do NOT simply cut and paste the information from the slides. You will receive a score of 0 if you copy the prose from the slides.

## Attempt History

| | Attempt | Time | Score |
|---|---|---|---|
| **LATEST** | Attempt 1 | 32 minutes | 10 out of 10 |

ⓘ Correct answers are hidden.

Score for this quiz: **10** out of 10
Submitted Oct 18 at 8:51pm
This attempt took 32 minutes.

### Question 1 — 2 / 2 pts

Describe the four types of Python containers. How are they different from each other?

Your Answer:

Python Containers:

1. **Lists:** Lists have arbitrary values with mutable elements and the elements are ordered with elements of any type.
2. **Tuples:** Tuples are also similar to lists but the elements of lists are not mutable with elements of any type.
3. **Dictionaries:** Dictionaries have a unique key for every value, it contains a pair of key, value. They are similar to maps and keys are unique. Dictionaries are mutable.
4. **Set:** Sets are similar to lists but can only store unique values, are unordered, and are mutable.

Lists are ordered and mutable with elements of any type.

Tuples are ordered, but not mutable with elements of any type.

Dicts map hashable keys to values. Dicts are mutable.

Sets contain distinct hashable values and are mutable.

## Question 2

2 / 2 pts

Explain the difference between list.append() and list.extend(). Include an example in your answer.

Your Answer:

list.append() -> This appends the value of the new element at the end of the list.

list.extend() -> This concatenates two lists.

Example:

list.append([5,6]) #will add the parameter as a list at the end of the list.

list.extend([5,6]) #will add 5 and 6 as individual elements at the end of the list

list.append(value) appends the value a new element at the end of the list.

list.extend(sequence) concatenates list and the sequence.

## Question 3

2 / 2 pts

Write a code fragment that shows how to emulate the behavior of list.remove(value) using only list.pop() and list.index()

Your Answer:

```
def lists_remove( i , j):
    if i in j:
        pos = j.index(i)
        return (j.pop(pos)
lists_remove(1, [1,2,3,4])
```

**list.pop(list.index(value)**

```
def my_remove(t, l):
    if t in l:
       l.pop(l.index(t))
```

## Question 4

**2 / 2 pts**

What is the difference between sorted(list) and list.sort()? Show examples of both.

Your Answer:

**sorted(list)** method sorts the given sequence (keeping in mind that list is of similar type) in ascending or descending order and returns a sorted list. This method does not affect the original sequence.

**list.sort()** is a method of list class and can only be used with lists. list.sort() is in-place sorting, which means it does not take additional space to sort the list. it returns nothing and makes changes to the original sequence.

Example:

list1: list = [2, 1 , 3]

sorted(list1) will return [1,2,3] where list1 is not changed whereas

list1.sort() will change the list1 in place and list1 is now going to be [1,2,4].

sorted(list) **returns a copy** of the list whereas list.sort() **modifies the actual list in place**.

sorted(list) returns a sorted **copy** of the list

list.sort() sorts the list in place

Example from slides:

```
lst1 = [3, 1, 2]
sorted(lst1) returns a new list [1, 2, 3].  lst1 is not changed

lst1.sorted() sorts lst1 in place, changing the list so
lst1 == [1, 2, 3]
```

## Question 5

**2 / 2 pts**

Consider the code fragment:

```
x = [1, 2, 3]
y = [x, x]
x[0] = 4
```

What is the value of y?  Why?

Your Answer:

x = [1, 2, 3]

y = [x,x]

so, y = [[1,2,3],[1,2,3]]

If we change x[0]= 4, then y will be **[[4,2,3],[4,2,3]]**, because y has two instance of x, and directly has the reference of it, so if anything is changed in x its directly reflected back in y.

```
[[4, 2, 3], [4, 2, 3]]
```

Y consists of two instances of x so changing x also changes y.

Quiz Score: **10** out of 10