Quiz 07

Due Oct 16, 2019 at 11:59pm **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	137 minutes	10 out of 10

Correct answers are hidden.

Score for this quiz: **10** out of 10
Submitted Oct 16, 2019 at 2:11pm
This attempt took 137 minutes.

Question 1

2 / 2 pts

Explain what's wrong in the second call to arg_cnt(). How do you fix the problem?

```
def arg_cnt(args, expect):
    if len(args) != expect:
        print(f"expected {expect} but found {len(args)}"
    else:
        print("ok")

>>> arg_cnt(('hello', 'world'), 2)
ok

>>> arg_cnt(('hello'), 1)
expected 1 but found 5  <-- WHY?  What's wrong?  How to fix?
```

Your Answer:

```
arg_cnt(('hello', 'world'), 2)>> In this argument python considers
the length of the argument is 2
```

```
arg_cnt(('hello'), 1) >>in this argument python considers
the length of the argument as 5 because there is only 'hello' passed in the
argument where as in the above argument 2 different values were passed in th
e
same argument

How to Fix::
arg_cnt((['hello']), 1)>> in this the python considers its length as one .
```

('hello', 'world') is a tuple with two values in arg_cnt(('hello', 'world'), 2)

('hello') evaluates to the string 'hello' in arg_cnt(('hello'), 1)

The fix is to add a comma after 'hello' to force Python to create a tuple ('hello',)

Question 2

2 / 2 pts

Describe a situation where lists are not appropriate but tuples are a good match.

Your Answer:

A situation where u should not change the value of the variable tuples are a good match because tuples are immutable and lists are mutable.

Tuples are more efficient in tasks like swapping values , returning multiple values and tuples can be used as keys in dict.

Tuples can be used as the key in a dictionary but lists can't be used a the key for a dictionary

Question 3

2 / 2 pts

You need to write a function that manages a customer waiting list. As the customer enters the store, she adds her name to the waiting list. When an employee becomes available, the employee identifies the customer who has been waiting the longest and then removes that customer's name from the list, and helps the customer. Which builtin Python data structure is most appropriate? Which methods would you use?

Your Answer:

```
from collections import deque
list1=deque(['cust1','cust2','cust3'])

list1.append('cust4')#customer enters the line

print(list1.popleft()) #removes the customer from the waitlist
list1.append('cust5') #customer enters the line
print(list1.popleft()) #removes the customer from the waitlist
```

deque built in function will be appropriate

because there u can use popleft() which will remove the left most element of the list i.e the customer who has been waiting the longest

Use a queue (FIFO) which can be implemented with a list, and either

list.insert(0, value), and list.pop() # insert at the beginning, pop from the end

or

list.append(value) and list.pop(0) # append at the end and pop from the beginning

a deque works but a simple list would also work just asa well because you add to one end and pop from the other

Question 4

2 / 2 pts

Explain how memoization works and how it can improve performance for some algorithms. Under what conditions does memoization help? Under what conditions, does it **not** offer much benefit?

Your Answer:

It can increase the speed of calculation by creating and maintaining the cache memory of the last result or previous result.

It can improve the performance of recursive factorial, recursive Fibonacci

In memoization space is used in order to speed up things

after memoization if the speed is increased in a good way then its a help.

If after memoization speed is not increased that much but the space is used so this is not a helpful case where it does not offer much help.

Memoization stores intermediate results to avoid recalculation. This is very helpful for problems where the same request may be made many times, e.g. fibonacci, factorial, etc. Memoization does not help if the solution computes a value a small number of times.

Question 5

2 / 2 pts

You are asked to write a program that reads an arbitrary file and identify the 10 most frequently used words. Which Python container would you choose and why?

Your Answer:

I choose Counter

we can Use Counter from Collections which has a property most common (k) where k is the number of most common values you want to print

for eg:

`Counter.most_common(` `4` `)`

`#this prints 4 most common value like:`

```
[('Geeks', 5), ('to', 4), ('and', 4), ('article', 3)]
credit geeks for geeks
```

defaultdict(int) or Counter()

Quiz Score: **10** out of 10