socrative

Week 3 Quiz

7 Questions

1. Why is recursion impo tant for traversing tree like st uctures such as file systems?

2/56 **A** It reduces code complexity

0/56 **B** It suppo ts the infinite expandability of tree like st uctures

2/56 **C** It *ca n* be more efficient than loops

2/56 **D** A and B only

50/56 **E** All of the above

2. In the function pictured here, what will be the output?

```
def recursive_sum(nested_list) -> int:
    total = 0

    for obj in nested_list:
        if type(obj) == list:
            total += recursive_sum(obj)

    return total

print(recursive_sum([1,2,[3,4],[5],6]))
```

26/55 **A** 0

22/55 **B** 21

6/55 **C** 11

1/55 **D** 1

i The value for total is never changed, so the final output will be 0

3. Which of the following code snippets will fix the previous function?

```
def recursive_sum(nested_list) -> int:
    total = 0

    for obj in nested_list:
        if type(obj) == list:
            total += recursive_sum(obj)

    return total

print(recursive_sum([1,2,[3,4],[5],6]))
```

51/58 **A**

```
for obj in nested_list:
    if type(obj) == list:
        total += recursive_sum(obj)
    else:
        total += obj
```

5/58 **B**

```
for obj in nested_list:
    if type(obj) == list:
        total += recursive_sum(obj)

    total += obj
```

2/58 **C**

```
for obj in nested_list:
    if type(obj) == list:
        total += recursive_sum(obj)
    else:
        total += 1
```

i obj is the value we are attempting to sum, so when obj is not of type list, it must be an integer and therefore should be added to the total.

6. Are you using the IDLE debugger or any Python debugger to analyze your code when it doesn't work as expected?

27/59 A Yes

33/59 B No

7. Do you know how to use a debugger to analyze your code at runtime?

20/57 A Yes

40/57 B No