CSC148 Worksheet 16 Solution

Hyungmo Gu

April 26, 2020

Question 1

a. The doctests for the base case is

```
    """
    >>> nested_list_contains(1,1)
    True
    >>> nested_list_contains(1,2)
    False
    """

```

Using this fact, we can write

```
    from typing import Union, List

    def nested_list_contains(obj: Union[int, List], item: int) $\to$
   bool:
        """Return whether the given item appears in <obj>.
        Note that if <obj> is an integer, this function checks whether
        <item> is equal to <obj>.

        >>> nested_list_contains(1,1)
        True
        >>> nested_list_contains(1,2)
        False
        """

        if isinstance(self, int):
            return obj == item

```

Listing 1: worksheet_16_q1a_solution

b. Consider the following doctest

```
    """
    >>> nested_list_contains([4,2,2,[6,5,7,[8]]],8)
    True
    """

```

Using the base case from question 1.a, and the basic recursive design recipe, we can conclude the algorithm will behave as follows

1) $4 \rightarrow 4 == item? \rightarrow$ False

2) $2 \rightarrow 2 == item? \rightarrow$ False

3) $2 \rightarrow$ False

4)[6,5,7,[8]] $\rightarrow$ Recursion

    5) $6 \rightarrow 6 == item? \rightarrow$ False

    6) $5 \rightarrow 5 == item? \rightarrow$ False

    7) $7 \rightarrow 7 == item? \rightarrow$ False

    8) [8] $\rightarrow$ Recursion

        9) $8 \rightarrow 8 == item? \rightarrow$ True (function terminates)

11) Function Terminates until the end of recursion

Now, no new parameters other than *obj* and *item* are required, since

1. for the traversing and checking of elements, they are done using the two parameters.
2. for bringing the value 'True' to user, it is done by repeatedly ending the recursive function call early with the value
3. for brining the value 'False' to user, it is done by returning False at the end.

Question 2