Copy linked list with random pointer

Question: A linked list is given such that each node contains an additional random pointer which could point to any

node in the list or null.

Return a deep copy of the list.

<u>Solutions:</u>

```
class RandomListNode:
    def __init__(self, x):
        self.val = x
        self.next = None
        self.random = None


class Solution:
    # @param ll, a RandomListNode
    # @return a RandomListNode
    def copyRandomList(self, ll):
        # copy and combine copied list with original list
        current = ll
        while current:
            copied = RandomListNode(current.val)
            copied.next = current.next
            current.next = copied
            current = copied.next
        # update random node in copied list
        current = ll
        while current:
            if current.random:
                current.next.random = current.random.next
            current = current.next.next

        # split copied list from combined one
        dummy = RandomListNode(0)
        copied_current, current = dummy, ll
        while current:
            copied_current.next = current.next
            current.next = current.next.next
            copied_current, current = copied_current.next, current.next
        return dummy.next


if __name__ == "__main__":
    ll, ll.next = RandomListNode(1), RandomListNode(2),
    ll.random = ll.next
    result = Solution().copyRandomList(ll)
    print ( result.val )
    print ( result.next.val )
    print ( result.random.val )
```