Lab 6 Solutions **lab06.zip (lab06.zip)**

Solution Files

Topics

Consult this section if you need a refresher on the material for this lab. It's okay to skip directly to the questions and refer back here should you get stuck.

Required Questions

Mutability

Q1: WWPD: List-Mutation

Use Ok to test your knowledge with the following "What Would Python Display?" questions:

```
python3 ok -q list-mutation -u
```

Important: For all WWPD questions, type `Function` if you believe the answer is `<function...>`, `Error` if it errors, and `Nothing` if nothing is displayed.

```
>>> lst = [5, 6, 7, 8]
>>> lst.append(6)
------

>>> lst
------

>>> lst.insert(0, 9)
>>> lst
------

>>> x = lst.pop(2)
>>> lst
------

>>> lst.remove(x)
>>> lst
------

>>> a, b = lst, lst[:]
>>> a is lst
------

>>> b == lst
------

>>> b is lst
------

>>> lst = [1, 2, 3]
>>> lst.extend([4,5])
>>> lst
------

>>> lst.extend([lst.append(9), lst.append(10)])
>>> lst
------
```

Q2: Insert Items

Write a function which takes in a list `lst`, an argument `entry`, and another argument `elem`. This function will check through each item in `lst` to see if it is equal to `entry`. Upon finding an item equal to `entry`, the function should modify the list by placing `elem` into `lst` right

after the item. At the end of the function, the modified list should be returned.

See the doctests for examples on how this function is utilized.

Important: Use list mutation to modify the original list. No new lists should be created or returned.

> **Note:** If the values passed into `entry` and `elem` are equivalent, make sure you're not creating an infinitely long list while iterating through it. If you find that your code is taking more than a few seconds to run, the function may be in a loop of inserting new values.

```
def insert_items(lst, entry, elem):
    """Inserts elem into lst after each occurence of entry and then returns lst.

    >>> test_lst = [1, 5, 8, 5, 2, 3]
    >>> new_lst = insert_items(test_lst, 5, 7)
    >>> new_lst
    [1, 5, 7, 8, 5, 7, 2, 3]
    >>> double_lst = [1, 2, 1, 2, 3, 3]
    >>> double_lst = insert_items(double_lst, 3, 4)
    >>> double_lst
    [1, 2, 1, 2, 3, 4, 3, 4]
    >>> large_lst = [1, 4, 8]
    >>> large_lst2 = insert_items(large_lst, 4, 4)
    >>> large_lst2
    [1, 4, 4, 8]
    >>> large_lst3 = insert_items(large_lst2, 4, 6)
    >>> large_lst3
    [1, 4, 6, 4, 6, 8]
    >>> large_lst3 is large_lst
    True
    >>> # Ban creating new lists
    >>> from construct_check import check
    >>> check(HW_SOURCE_FILE, 'insert_items',
    ...       ['List', 'ListComp', 'Slice'])
    True
    """
    index = 0
    while index < len(lst):
        if lst[index] == entry:
            lst.insert(index + 1, elem)
            if entry == elem:
                index += 1
        index += 1
    return lst
```

Use Ok to test your code:

```
python3 ok -q insert_items
```

Video Walkthrough:

YouTube link (https://youtu.be/duHrRpS4TYo)

Iterators

Q3: WWPD: Iterators

Use Ok to test your knowledge with the following "What Would Python Display?" questions:

```
python3 ok -q iterators-wwpd -u
```

Important: Enter `StopIteration` if a `StopIteration` exception occurs, `Error` if you believe a different error occurs, and `Iterator` if the output is an iterator object.

```
>>> s = [1, 2, 3, 4]
>>> t = iter(s)
>>> next(s)
------

>>> next(t)
------

>>> next(t)
------

>>> iter(s)
------

>>> next(iter(s))
------

>>> next(iter(t))
------

>>> next(iter(s))
------

>>> next(iter(t))
------

>>> next(t)
------
```

```
>>> r = range(6)
>>> r_iter = iter(r)
>>> next(r_iter)
------

>>> [x + 1 for x in r]
------

>>> [x + 1 for x in r_iter]
------

>>> next(r_iter)
------

>>> list(range(-2, 4))   # Converts an iterable into a list
------
```

```
>>> map_iter = map(lambda x : x + 10, range(5))
>>> next(map_iter)
------

>>> next(map_iter)
------

>>> list(map_iter)
------

>>> for e in filter(lambda x : x % 2 == 0, range(1000, 1008)):
...     print(e)
------

>>> [x + y for x, y in zip([1, 2, 3], [4, 5, 6])]
------

>>> for e in zip([10, 9, 8], range(3)):
...   print(tuple(map(lambda x: x + 2, e)))
------
```

Q4: Count Occurrences

Implement `count_occurrences`, which takes in an iterator `t` and returns the number of times the value `x` appears in the first `n` elements of `t`. A value appears in a sequence of elements if it is equal to an entry in the sequence.

> **Note:** You can assume that `t` will have at least `n` elements.

```
def count_occurrences(t, n, x):
    """Return the number of times that x appears in the first n elements of iterator t.

    >>> s = iter([10, 9, 10, 9, 9, 10, 8, 8, 8, 7])
    >>> count_occurrences(s, 10, 9)
    3
    >>> s2 = iter([10, 9, 10, 9, 9, 10, 8, 8, 8, 7])
    >>> count_occurrences(s2, 3, 10)
    2
    >>> s = iter([3, 2, 2, 2, 1, 2, 1, 4, 4, 5, 5, 5])
    >>> count_occurrences(s, 1, 3)
    1
    >>> count_occurrences(s, 4, 2)
    3
    >>> next(s)
    2
    >>> s2 = iter([4, 1, 6, 6, 7, 7, 8, 8, 2, 2, 2, 5])
    >>> count_occurrences(s2, 6, 6)
    2
    """
    count = 0
    for _ in range(n):
        value = next(t)
        if value == x:
            count += 1
    return count
```

Use Ok to test your code:

```
python3 ok -q count_occurrences
```

Q5: Repeated

Implement `repeated`, which takes in an iterator `t` and returns the first value in `t` that appears `k` times in a row.

> **Note:** You can assume that the iterator `t` will have a value that appears at least `k` times in a row. If you are receiving a `StopIteration`, your `repeated` function is likely not identifying the correct value.

Your implementation should iterate through the items in a way such that if the same iterator is passed into `repeated` twice, it should continue in the second call at the point it left off in the first. An example of this behavior is in the doctests.

```
def repeated(t, k):
    """Return the first value in iterator T that appears K times in a row.
    Iterate through the items such that if the same iterator is passed into
    the function twice, it continues in the second call at the point it left
    off in the first.

    >>> s = iter([10, 9, 10, 9, 9, 10, 8, 8, 8, 7])
    >>> repeated(s, 2)
    9
    >>> s2 = iter([10, 9, 10, 9, 9, 10, 8, 8, 8, 7])
    >>> repeated(s2, 3)
    8
    >>> s = iter([3, 2, 2, 2, 1, 2, 1, 4, 4, 5, 5, 5])
    >>> repeated(s, 3)
    2
    >>> repeated(s, 3)
    5
    >>> s2 = iter([4, 1, 6, 6, 7, 7, 8, 8, 2, 2, 2, 5])
    >>> repeated(s2, 3)
    2
    """
    assert k > 1
    count = 1
    last_item = None
    while True:
        item = next(t)
        if item == last_item:
            count += 1
        else:
            last_item = item
            count = 1
        if count == k:
            return item
```

Use Ok to test your code:

```
python3 ok -q repeated
```