SI@UCF Python Program: List Function Practice

Objectives

1. To learn how to use lists.
2. To write functions that take in lists as parameters.
3. How to return a list from a function.
4. To integrate the working functions into a full program.

Problem A: valsInRange Function (inrange.py)

Write a function that takes in a list of integers, **`vals`**, an integer **`low`**, and an integer **`high`** and returns the number of integers in **`vals`** that are in between **`low`** and **`high`**, inclusive. To make sure your function is working, test it using the test function given below:

```
def valsInRange(vals,low,high):
    # Fill in your code here.

def testValsInRange():
    print(valsInRange([3,12,6,5,2,8,9], 4, 9))
    print(valsInRange([100, 99, 98, 97], 0, 100))
    print(valsInRange([30,40,50,55,59,20], 60, 100))
    print(valsInRange([36,16,25,24,36,23,23,20,32], 23,35))

testValsInRange()
```

When you run the test, it should print 4, 4, 0, and 5, respectively, on separate lines.

Problem B: getValsInRange Function (listinrange.py)

Write a function that takes in a list of integers, **`vals`**, an integer **`low`**, and an integer **`high`** and returns a list with each value in **`vals`** that is in between **`low`** and **`high`**, inclusive.

```
def getValsInRange(vals,low,high):
    # Fill in your code here.

def testValsInRange():
    print(getValsInRange([3,12,6,5,2,8,9], 4, 9))
    print(getValsInRange([100, 99, 98, 97], 0, 100))
    print(getValsInRange([30,40,50,55,59,20], 60, 100))
    print(getValsInRange([36,16,25,24,36,23,23,20,32], 23,35))

testValsInRange()
```

When you run the test it should print the following:

```
[6, 5, 8, 9]
[100, 99, 98, 97]
[]
[25,24,23,23,32]
```

Problem C (Optional): Direction Changes Function (dirchange.py)

Write a function that takes in a list of integers, **`vals,`** and returns the number of direction changes in the list. The current direction of a list of numbers is defined by the first consecutive pair of numbers that are unequal. The pair (a, b) is defined as "up" if a < b and is defined as "down" if a > b. We can look at each pair of consecutive numbers in a sequence and define their change as either "up", "down" or "neither" (for 2 equal numbers). Consider converting a list of n values into a list of n-1 of these changes then ignoring each "neither" change. A direction change is either an "up"-"down", or "down"-"up"

```
def numChanges(vals):
    # Fill in your code here.

def testNumChanges():
    print(numChanges([3,12,6,5,5,5,2,2,2,8,8,9]))
    print(numChanges([100, 99, 98, 97]))
    print(numChanges([30,30,40,50,55,59,20]))
    print(numChanges([36,16,25,24,36,23,23,40,32]))
    print(numChanges([2]))
    print(numChanges([3,3,3,3,3,3,3,3,3,3,3]))
    print(numChanges([3,3,3,3,3,3,3,3,3,3,3,4]))
    print(numChanges([3,4,3]))
    print(numChanges([3,3,3,3,3,3,4,4,4,3,3,3,3,3,3,3]))


testNumChanges()
```

When you run the test it should print the following:

```
2
0
1
6
0
0
0
1
1
```