Name: _____

Use this quiz to help make sure you understand the videos/reading. ***Answer all questions.*** Make additional notes as desired. ***Not sure of an answer?*** Ask your instructor to explain in class.

Video: ***Sequences*** [8:43 minutes]

1. Sequences are powerful because:

2. What gets printed by the program shown below when *main* runs?

```
def main():
    list1 = [74, 34, 13, 30, 4004]
    string1 = 'Expert texpert!'

    examples_of_indexing(list1)
    examples_of_indexing(string1)

def examples_of_indexing(sequence):
    length = len(sequence)
    print('Length is', length)

    print('At indices 0, 1, 4:')
    print(sequence[0])
    print(sequence[1])
    print(sequence[4])

    print('List them all:')
    for k in range(len(sequence)):
        print(sequence[k])
    print()

    print('Cool stuff:')
    print(sequence[-1])

    print('Last item:')
    length = len(sequence)
    print(sequence[length - 1])
```

Just take a wild guess and ask in class!

Output
(fill in the blanks)

```
Length is _______

At indices 0, 1, 4:

_____________

_____________

_____________

List them all:

_____________

_____________

_____________

_____________

_____________


Cool stuff:

_____________

Last item:

_____________
```

(output continues)

```
Length is _______

At indices 0, 1, 4:

_____________

_____________

_____________

List them all:

_____________

_____________

_____________
```

. . . [don't bother doing all of the letters, just skip to the last one]

```
_____________


Cool stuff:

_____________

Last item:

_____________
```

3. Implement (here, on paper) the following function, per its specification.

```
def count_negative(seq):
    """
    Returns the number of items in the given sequence of numbers
    that are negative.

    Precondition: The argument is a sequence of numbers.
    """
```

4. Implement (here, on paper) the following function, per its specification.

```
def count_short_ones(seq_of_lists):
    """
    The argument is a sequence of lists.  Returns the number of
    items in that sequence whose length is less than 3.

    For example, if the argument is:
        [ [3, 5], [3, 9, 0, 4], [5], [5], [], [9, 8, 7], [5, 6] ]
    then this function returns 5, since 5 of the 7 lists in the
    above sequence have length less than 3
    (I have displayed those 5 lists in red.)

    Precondition: The argument is a sequence of lists.
    """
```

Video: ***The Last Item in a Sequence*** [3:20 minutes]

5. Given a sequence called ***seq*** that contains `10` items, write expressions for:

   - The beginning item (element) in ***seq***: _____
   - The last item (element) in ***seq***: _____

6. Suppose that you have a ***list*** called ***my_list*** that contains `10` items. What error message appears if you attempt to access ***my_list*****[10]**?

7. Suppose that you have a ***string*** called ***my_string*** that contains `10` characters. What error message appears if you attempt to access ***my_string*****[10]**?

8. In Eclipse, when a run-time *exception* (error) occurs, how do you go straight to the line at which the exception occurred?

9. Is that line always the line that is wrong? **Yes** **No** (circle your choice)

10. What should you do if the exception occurred in the *zellegraphics* or *new_create* modules?

11. In the two code fragments below, assume that the variable **s** has been assigned as its value some sequence. The code on the left (below) causes a run-time exception. The code on the right (below) does NOT cause a run-time exception. But both use the same expression **len(s)**. Explain why the code on the left breaks (and is wrong) but the code on the right does not break (and is correct).

```
... s[len(s)] ...
```

```
for k in range(len(s)):
    ... s[k] ...
```

Video: ***Iterating Through a Sequence*** [11:35 minutes]

In each of the following problems, assume that there is a variable **x** that is a sequence of numbers.

12. Write a loop that prints all the elements of **x**, backwards. For example, if x were the list [9, 4, 12, 88, 17], then your loop should print 17, 88, 12, 4 and 9.

13. Write a loop that prints all the elements of **x** between indices 4 and 7, backwards. For example, if x were the list [9, 4, 12, 88, 17, 33, 20, 21, 22], then your loop should print 21, 20, 33 and 17. Assume that the sequence has at least 8 elements.

14. Write a loop that prints every 3rd element of **x**, starting at index 1. For example, if x were the list [9, 4, 12, 88, 17, 33, 20, 21, 22, 5, 7, 77], then your loop should print 4, 17, 21 and 7. Assume that the sequence has at least 2 elements.

15. Write a loop that prints all the elements of **x** that are greater than 10. For example, if x were the list [9, 4, 12, 88, 17, 33, 20, 21, 22, 5, 7, 77], then your loop should print 12, 88, 17, 33, 20, 21, 22 and 77.