

Name: _____ Section: 1 2 3 4

1 = Mutchler, 1st-2nd periods. 2 = Mutchler, 3rd-4th periods. 3 = Anderson, 7th-8th periods. 4 = Anderson, 9th-10th periods.

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 and revise as needed then. **Please print two-sided.**

Throughout, where you are asked to “circle your choice”, you can underline or circle it (whichever you prefer).

Handout: **Overloading the Plus Operator**

1. Fill in the blanks:

11 + 22 evaluates to _____

'11' + '22' evaluates to _____

'11' + str(3 + 3) + '22' evaluates to _____

'11' + 33 evaluates to _____ (this one is a trick question)

2. Suppose you have variables *x*, *y* and *z*, each of whose values are integers. Write a single **print** statement that would print those 3 values separated by commas, with NO spaces. For example, the **print** statement might yield:

4,72,3 or **101,55,17**

Note that something like **print(x, y, z)** will NOT solve this problem – it puts spaces between each item that it prints.

Handout: **Accumulating Sequences**

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

```
def make_list(n):
    """ Returns the list [1, 2, 3, 4, 5, ... n],
        where n is the given argument.
        Precondition: The argument is a positive integer. """
```

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

```
def make_string(n):  
    """ Returns the string '12345678910111213...' where the last number in the string is the given integer.
```

Video: *Patterns for Iterating Through Sequences* [15:21 minutes]

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

```
def find_negative(seq):  
    """ Returns the index of the first negative number in the given sequence of numbers (that contains at least one negative number).  
    Precondition: The argument is a sequence of numbers. """
```

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

```
def max_stutter(string):  
    """ Returns the largest of times a letter is repeated twice-in-a-row in the given string.  
    For example:  
        'xhhbrs' returns 2  
        'xxxx' returns 3  
        'xaxaxa' returns 0  
    Precondition: The argument is a string. """
```

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

```
def max_element(seq, n):  
    """ Returns the largest number in the first n numbers of the  
        given sequence of numbers.  
        Preconditions: The first argument is a sequence of  
            numbers, the second argument is a positive integer,  
            and the length of the given sequence is at least  
            the given n. """
```

Textbook Reading: Section 6.1 – Basic Properties of Lists (pages 278 – 283); you were supposed to read this before Session 10. Section 6.2 – List Operations (pages 163 – 165)

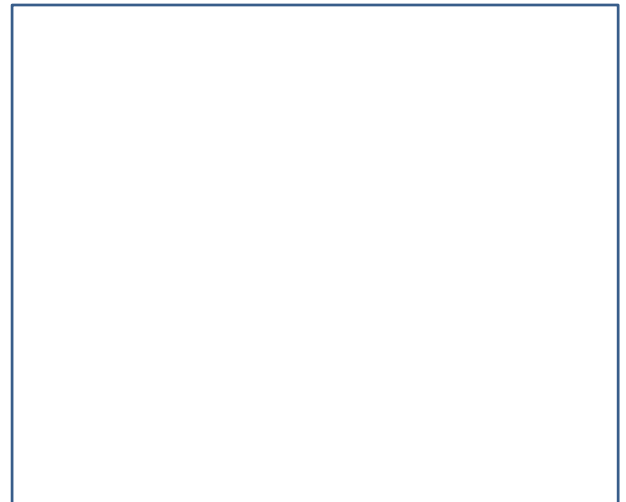
Try to figure out the answers. If you are not sure, try them in the Pydev Console. If you cannot figure out why Python does what it does, ask in class.

8. What does the following Python code print? Answer here: _____

```
seq1 = ["John", "Paul", "George", "Pete"]  
seq2 = seq1  
seq2[3] = "Ringo"  
print(seq1[3], seq2[3])
```

9. What does the following Python code print?
Answer here:

```
numbers = [5, 9, 2]  
numbers.append(3)  
print("appended", numbers)  
numbers.insert(2, 1)  
print("inserted", numbers)  
numbers.pop(3)  
print("popped", numbers)  
numbers.remove(9)  
print("removed", numbers)
```



10. Show a very simple modification that we could make to the code in the Beatles problem above so that its output would be
Pete Ringo