Week 4 Discussion Worksheet

1. Given the following code:

```
def indexer(item, index, new_val):
    """
    Function to assign a new value to an iterable
    item at a specific index.

    args:
        item (iterable): iterable object to be manipulated
        index (int): int representing index of iterable
        new_val (any): item to be assigned to index.

    returns:
        None
    """
    item[index] = new_val
    return
```

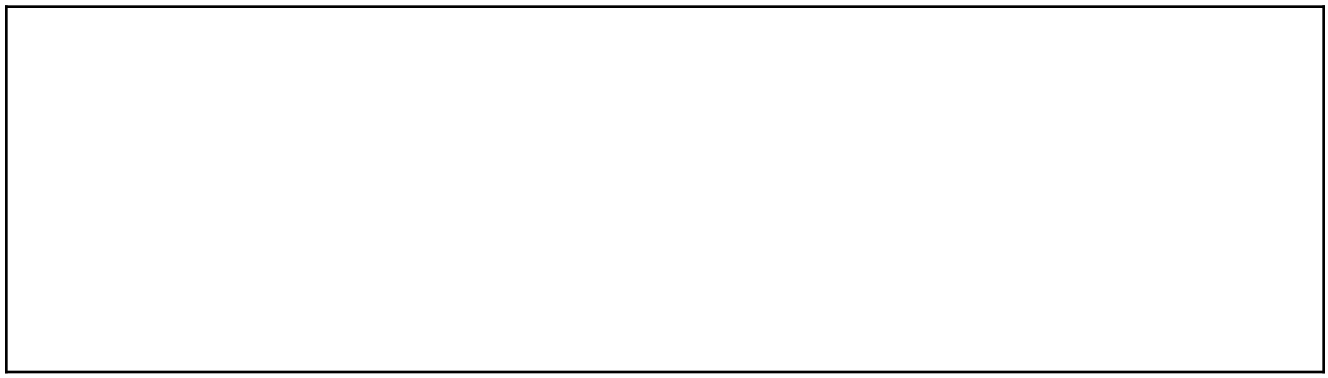
What is the result of running: `indexer('star wars', 2, '@')` ? Explain why.

2.

```
def extract_text(sentences):
    """
    Write a function that processes sentences by creating a dictionary that
    tracks the index of the sentences in which the word shows up.

    Args:
        sentences (list): list of strings to be considered
    Returns:
        a dictionary that tracks each word and the index
        of the sentences it appeared in.
    Throws:
        AssertionError: if sentences is not a list
        AssertionError: if elements of sentences is not a string

        >>> sentences = ["a quick brown fox jumps",
    "a brown dog jumps at the fox", "dog"]
    >>> extract_text(sentences)
    {'a': [0, 1], 'quick': [0], 'brown': [0, 1], 'fox': [0, 1], \
    'jumps': [0, 1], 'dog': [1, 2], 'at': [1], 'the': [1]}
    """
```

3.

```
def string_comparer(lst, comparer):
    """
    Write a function that counts the number of equivalent strings in a list.
    Requirement: 1 line list comp

    Args:
        lst (list): list of strings to be considered
        comparer (str): string to be considered
    Returns:
        The number of equivalent elements
    Throws:
        AssertionError: if lst is not a list
        AssertionError: if comparer is not a string
        AssertionError: if there are non-strings in lst

    >>> test(['good', 'luck', 'on', 'mt', 'luck'], 'luck')
    2
    >>> test([], 'lol')
    0
    """
```



4.

```
def process_file(file_path, sub):
    """
    Write a function that processes a file by replacing each instance of a specified string with another
    and returns the number of substitutes that occur as well as the corrected file content.

    args:
        file_path (str): filepath for relevant file
        sub (tup): tuple representing (target, replacement)
            target(str): target to be replaced
            replacement(str): replacement value

    returns:
        the number of substitutes made during processing

    raises:
        AssertionError: if file_path is not a string
        AssertionError: if sub is not a tuple
        AssertionError: if either element of sub is not a str

    >>> process_file('files/review.txt', ('hard', 'easy'))
    ("DSC20 is so easy. It's probably the easyest class I've taken! I have so many easy classes this quarter. ",
    3)
    """
```