Sequences and Dictionaries worksheet
CS51 - Spring 2026
ANSWER KEY

Practice Problem 1 - Strings
You are given the string `message = 'I love cs51!'`. What will the following expressions evaluate to?

a. `message[1:4]` `' lo'`

b. `message[:2]` `'I '`

c. `message[2:]` `'love cs51!'`

d. `message[:]` `'I love cs51!`

e. `message[2:11:2]` `'lv s1'`

f. `message[:11:2]` `'Ilv s1'`

g. `message[3::2]` `'oec5!'`

h. `message[-1]` `'!'`

i. `message[-1]='0'` `Type error: 'str' object does not support item assignment`

j. `message.upper()` `'I LOVE CS51!]`

k. `message.isupper()` False

Practice Problem 2 - Lists

a. Define a list `favorite_numbers` with two elements, the numbers 24 and 47. `favorite_numbers = [24, 47]`

b. Create an empty list. There are two ways you can achieve this.

```
empty_list = []
empty_list = list()
```

c. Convert the string `'cs51'` into a list of its individual characters. `list('cs51')`

d. Consider the lists `list_1 = [1, 2]` and `list_2 = [3, 4]`. What will `list_1 + list_2` return? `[1,2,3,4]`

e. Consider the list `list_1 = [1, 2]`. What will the contents of `list_2` be if `list_2 = list_1 * 3`? `list_2=[1, 2, 1, 2, 1, 2]`

f. Consider the list `cheeses = ['Feta','Cheddar', 'Edam', 'Gouda']`. What will happen after the execution of each of the following lines?

```
cheeses.append('Haloumi')
cheeses.extend(['Emmental', 'Gruyere'])
cheeses.insert(1, 'Brie')
```

```
cheeses = [‘Feta’, ‘Cheddar’, ‘Edam’, ‘Gouda’, ‘Haloumi’]
[‘Feta’, ‘Cheddar’, ‘Edam’, ‘Gouda’, ‘Haloumi’, ‘Emmental’, ‘Gruyere’]
[‘Feta’, ‘Brie’, ‘Cheddar’, ‘Edam’, ‘Gouda’, ‘Haloumi’, ‘Emmental’, ‘Gruyere’]
```

g. Consider the list `cheeses = [‘Feta’,‘Cheddar’, ‘Edam’, ‘Gouda’]`. What will happen after the execution of each of the following lines?

```
removed_cheese = cheeses.pop(1)
cheeses.remove(‘Cheddar’)

removed_cheese = ‘Cheddar’
cheeses = [‘Feta’, ‘Edam’, ‘Gouda’]
```

h. How can you break the string `motto = ‘cs51 is my favorite class’` into a list of individual words? `motto.split()`

i. How can you concatenate a list of strings into a single string? E.g., how can you turn the list `favorite_list = [‘cs51’, ‘is’, ‘my’, ‘favorite’, ‘class’]` into `‘cs51 is my favorite class’`?

```
delimiter = ‘ ’
favorite_string = delimiter.join(favorite_list)
```

Practice Problem 3 - Tuples

a. Define a tuple `tup` with the characters `‘c’,‘s’,‘5’, ‘1’`.

```
tup = (‘c’,‘s’,‘5’, ‘1’)
#or tup = ‘c’,‘s’,‘5’, ‘1’
```

b. Create an empty tuple. There are two ways you can achieve this.

```
tup = ()
#or tup = tuple()
```

c. Convert the string `‘cs51’` into a tuple of its individual characters. `tuple(‘cs51’)`

d. Consider the code `tuple(‘lup’) + (‘i’, ‘n’)` . What will happen? `(‘l’, ‘u’, ‘p’, ‘i’, ‘n’)`

e. Consider the code `tuple(‘spam’) * 2`. What will happen? `(‘s’, ‘p’, ‘a’, ‘m’, ‘s’, ‘p’, ‘a’, ‘m’)`

f. Consider the code `tuple(‘cs51’)`. What will happen if we execute `tuple[3]=‘0’`? `TypeError: ‘tuple’ object does not support item assignment`

Practice Problem 4 - Ranges

a. What will the following code do?

```
for i in range(1,10,2):
    print(i)
```

Print:

```
1
3
5
7
9
```

Practice Problem 5 - Dictionaries

a. Create an empty dictionary `d`. There are two ways you can achieve this

```
d = dict()
# or d = {}
```

b. Create an dictionary that associates English words with their translation to a language of your choice. Insert two such pairs of words. Print in pairs the keys and associated values.

```
english_spanish_dict = {}
english_spanish_dict['hello']='hola'
english_spanish_dict['goodbye']='adios'
for english, spanish in english_spanish_dict.items():
    print(english, spanish)
```

Practice Problem 6 - Sorting strings

```
def compare_class(class_number):
    if class_number < 'cs51':
        print(class_number, 'comes before cs51.')
    elif number > 'cs51':
        print(class_number, 'comes after s51.')
    else:
        print('Hello, cs51')
```

What will happen if I call `compare_class('cs50')` and next `compare_class('CS62')`?

```
cs50 comes before cs51.
CS62 comes before cs51.
```

Practice Problem 7 - String functions

Define a function `str_odd_indices` that takes one parameter `s` (a string) and returns a string comprised of only the odd indexed characters of `s`. For example, `str_odd_indices('hello!')` would return `'el!'`.

```
#def str_odd_indices(s):
#    return s[1::2]
def str_odd_indices(s):
    result = ''
    for i in range(len(s)):
        if i % 2 == 1:
            result += s[i]
    return result
```

Practice Problem 8 - Aliasing

You are given the function

```
def augment_twice(a_list, val):
    a_list.append(val)
    a_list.append(val)
```

What happens to `numbers` after executing the following code?

```
numbers = [1,2,3]
augment_twice(numbers, 47)
```

```
[1, 2, 3, 47, 47]
```

What if we were to change the function to this and execute the same code?

```
def augment_twice(a_list, val):
    a_list = a_list + [val, val]
```

```
[1, 2, 3]
```

Practice Problem 9 - Recursion

Write a recursive Python function called `rec_linear_search` that takes as input a list, an element, and an index that has a default value of 0, and returns the index of the first encounter of the element in the list, if it exists, or -1 if it does not.

```
def rec_linear_search(lst, element, index=0):
    if index == len(lst):
       return -1
    elif lst[index] == element:
        return index
    else:
        return rec_linear_search(lst, element, index+1)
```

Practice Problem 10 - Loop invariants

Given a list `my_list` (of size `n`) of numbers, write an iterative function `sum_of_list_numbers` that calculates the sum of the numbers in `my_list`.

```
def sum_of_list_numbers(my_list):
    answer = 0
    for i in range(len(my_list)):
     answer += my_list[i]
    return answer
```

1. What are the pre-and post-conditions? A list of n numbers and a number that is equal to the sum of the n numbers in `my_list`
2. What is a good loop invariant? At the start of iteration `i` of the loop, the variable answer should contain the sum of the numbers from the sublist `my_list[0:i]`.
3. Use loop invariants to prove that your function works correctly.
    - *Initialization:* At the start of the first loop, the sum of the numbers in an empty slice is 0, and this is what `answer` has been set to.
    - *Maintenance:* Assume that the loop invariant holds at the start of iteration `i` . Then it must be that answer contains the sum of numbers in slice `my_list[0:i]`. In the body of the loop, we add `my_list[i]` to answer. Therefore, at the end of iteration and i and before `i+1` iteration begins, answer will contain the sum of numbers in `my_list[0:i+1]`, which is what we needed to prove.
    - *Post-condition:* When the loop terminates, `i` should be equal to `n` and the loop invariant gives that answer contains the sum of all numbers in slice `my_list[0:n]` which is equal to list. Thus, we will indeed get the sum of all numbers in `my_list`.
    - *Termination:* `i` increases by 1 in every iteration and ranges from 0 to maximum `n==len(my_list)`. The for loop will terminate in a finite number of steps.