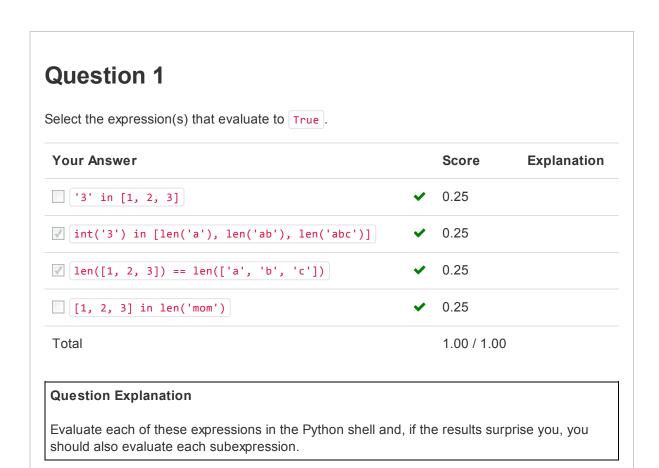
Feedback — Week 5 Exercise

You submitted this homework on **Mon 23 Sep 2013 7:59 PM PDT (UTC -0700)**. You got a score of **13.75** out of **14.00**. You can attempt again, if you'd like.

These exercises are intended to get you to open IDLE and work on small pieces of code. If you have been copying/pasting/editing/running the code and trying out expressions in the Python shell, then congrats, you're doing the right thing!

If, on the other hand, you have been avoiding using IDLE, and you're struggling with the exercises, then you aren't learning as much as you should. Open up IDLE and try out all the code as you do the exercise!



Question 2

Consider this code:

```
def mystery(s):
    i = 0
    result = ''

while not s[i].isdigit():
    result = result + s[i]
    i = i + 1

return result
```

Select the function call(s) that result in an error.

Your Answer		Score	Explanation
<pre>mystery('abc')</pre>	~	0.25	
mystery('abc123')	~	0.25	
mystery('123abc')	~	0.25	
mystery('123')	~	0.25	
Total		1.00 / 1.00	

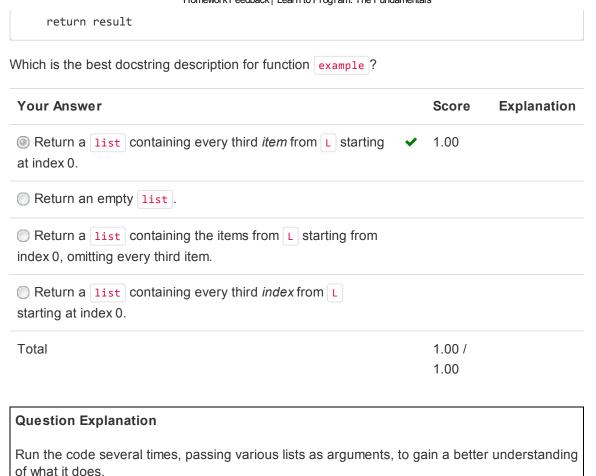
Question Explanation

Run the code. To understand why the error occurs (or does not), trace the code in the visualizer.

Question 3

Consider this code:

```
def example(L):
    """ (list) -> list
    """
    i = 0
    result = []
    while i < len(L):
        result.append(L[i])
        i = i + 3</pre>
```



```
def compress_list(L):
    """ (list of str) -> list of str

    Return a new list with adjacent pairs of string elements from L
    concatenated together, starting with indices 0 and 1, 2 and 3,
    and so on.

Precondition: len(L) >= 2 and len(L) % 2 == 0

>>> compress_list(['a', 'b', 'c', 'd'])
    ['ab', 'cd']
    """
    compressed_list = []
    i = 0

while i < len(L):</pre>
```

compressed_list.append(L[i] + L[i + 1])
MISSING CODE HERE
return compressed_list

Select the missing line of code.

Your Answer		Score	Explanation
(i = i + 2)	~	1.00	
(i = i + 1)			
(i = i + i			
(i = i * 2)			
Total		1.00 / 1.00	

Question Explanation

Run the code to gain a better understanding of what it does and what effect each choice has.

Question 5

What is the sum of the **odd** numbers from 1523 through 10503, inclusive? Hint: write a while loop to accumulate the sum and print it. Then copy and paste that sum. For maximum learning, do it with a for loop as well, using range.

You entered:

27004383

Your Answer		Score	Explanation
27004383	~	1.00	
Total		1.00 / 1.00	

Question Explanation

Start by solving a simpler problem, such as printing the odd integers between 1 and 9. Make

sure that you print 1, 3, 5, 7, and 9. Then introduce a variable, perhaps called sum, that starts at 0, and in each iteration adds the current integer to sum. The sum of the odd numbers from 1 through 9, inclusive, is 25. Then reattempt the question.

Question 6

Consider this code:

```
def while_version(L):
    """ (list of number) -> number
    """
    i = 0
    total = 0

while i < len(L) and L[i] % 2 != 0:
    total = total + L[i]
    i = i + 1

return total</pre>
```

The while loop stops as soon as an even number is found, and the sum of all the previous numbers is returned. The four functions below use a for loop to try to accomplish the same task, although they keep iterating through all of the numbers in L regardless of whether the numbers are even or odd. Only one of them returns the same value as function while_version. Which one is it?

```
def for_version(L):
   found even = False
   total = 0
   for num in L:
     if num % 2 != 0:
        total = total + num
     elif not found even:
        found even = True
   return total
def for_version(L):
   found even = False
   total = 0
   for num in L:
     if num % 2 != 0:
        total = total + num
        found even = True
   return total
def for_version(L):
   found_even = False
   total = 0
   for num in L:
     if num % 2 != 0:
        total = total + num
     found_even = True
   return total
Total
                                                             1.00 / 1.00
Question Explanation
Pay close attention to the variables involved in the loop condition: i and L. Read through
the code or trace it in the visualizer to determine what that while loop does and when it exits.
You should also run each of the for_version functions.
```

Consider this code:

```
>>> letters = ['b', 'd', 'a']
>>> # MISSING CODE HERE
>>> print(letters)
['a', 'b', 'd']
```

Which of the following code fragments(s) could be the missing code in the program above?

Your Answer		Score	Explanation
<pre>✓ letters.sort()</pre>	~	0.25	
sort(letters)	•	0.25	
<pre>letters = letters.sort()</pre>	•	0.25	
<pre>letters = sort(letters)</pre>	~	0.25	
Total		1.00 / 1.00	

Question Explanation

Run the code and try each code fragment.

Question 8

Consider this code:

```
veggies = ['carrot', 'broccoli', 'potato', 'asparagus']
veggies.insert(veggies.index('broccoli'), 'celery')
print(veggies)
```

What is printed by the code above?

Your Answer		Score	Explanation
<pre>['carrot', 'celery', 'broccoli', 'potato', 'asparagus']</pre>	~	1.00	
<pre>['carrot', 'celery', 'potato', 'asparagus']</pre>			

```
['celery', 'carrot', 'broccoli', 'potato',
'asparagus']

['carrot', 'broccoli', 'celery', 'potato',
'asparagus']

Total

1.00 /
1.00

Question Explanation

Run the code and read the help for list.index and list.insert.
```

Your younger sibling has just discovered music from the 1970's. They have put together a playlist of the same 5 songs repeated again and again. Here are the songs:

- ABC by The Jackson 5
- · Venus by Shocking Blue
- · Lola by the Kinks
- · Let It Be by the Beatles
- · Cecilia by Simon and Garfunkel

Here is an example of their playlist:

```
['Lola', 'Venus', 'Lola', 'Lola', 'Let It Be', 'Lola', 'ABC', 'Cecilia', 'Lola', 'Lola']
```

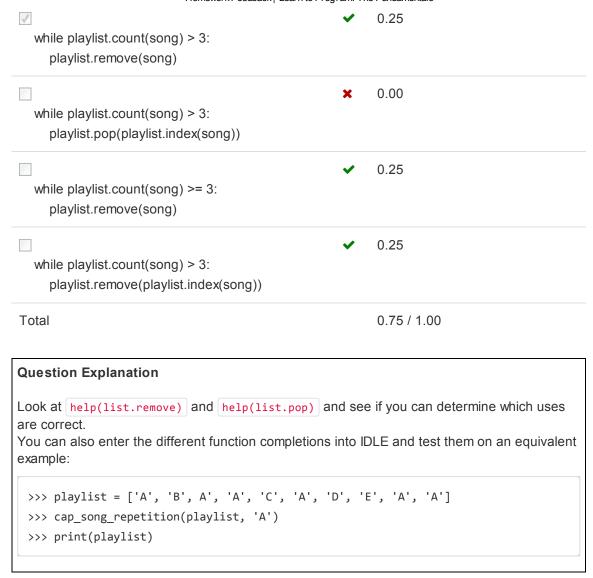
You want to make sure that Lola gets played at most 3 times, so you want to complete this function that edits the playlist:

```
def cap_song_repetition(playlist, song):
    '''(list of str, str) -> NoneType

Make sure there are no more than 3 occurrences of song in playlist.
    '''
```

Select the loop(s) that accomplish this.

Your Answer Score Explanation



Consider this code:

```
>>> a = [1, 2, 3]
>>> b = a
>>> # MISSING CODE HERE
>>> print(a, b)
[1, 'A', 3] [1, 'A', 3]
```

Which of the following code fragments(s) could be the missing code in the program above?

Your Answer	Score	Explanation

		Japan I Dan I Digital I Tio I di Idanio I Ida
a[1] = 'A'	~	0.25
b[-2] = 'A'	~	0.25
a = [1, 'A', 3] b = [1, 'A', 3]	~	0.25
b[1] = 'AB' a[1] = a[1][0]	~	0.25
Total		1.00 / 1.00

Question Explanation

Run the code, substituting each code fragment in for the missing code. Note that if a[1] is "CAT", then a[1][0] is "C".

Question 11

Consider this code:

Which of the following code fragments(s) could be the missing code in the program above?

Your Answer		Score	Explanation
a = [1, 'A', 3] b = [1, 'A', 3]	~	0.25	
a[1] = 'A'	~	0.25	
b[1] = 'AB'	~	0.25	

a[1] = a[1][0]		
b[-2] = 'A'	~	0.25
Total		1.00 / 1.00

Question Explanation

Run the code, substituting each code fragment in for the missing code. Note that if a[1] is "CAT", then a[1][0] is "C".

Question 12

Consider this code:

```
def increment_items(L, increment):
    i = 0
    while i < len(L):
        L[i] = L[i] + increment
        i = i + 1

values = [1, 2, 3]
print(increment_items(values, 2))
print(values)</pre>
```

What is printed by the program above?

Your Answer		Score	Explanation
•	~	1.00	
None			
[3, 4, 5]			
0			
None			
[1, 2, 3]			
0			
[3, 4, 5]			
None			
0			
[3, 4, 5]			

```
[1, 2, 3]

Total 1.00 / 1.00

Question Explanation
Run the code.
```



Call help(range) to learn about the function range and its two optional arguments.

Question 14

Select the function calls to range that, when used to fill in the blank, cause the code to produce the results below.

```
for num in _____:
    print(num)
```

The loop should print this:

3 11

19

Your Answer	Score	Explanation
<pre> ✓ range(3, 20, 8)</pre>	0.25	
<pre> ✓ range(3, 23, 8)</pre>	0.25	
range(3, 8, 20)	0.25	
range(3, 19, 8)	0.25	
Total	1.00 / 1.00	

Question Explanation

Call help(range) to learn about the function range and it's two optional arguments.