**Python Code Challenges: Loops**

**Python Code Challenges involving loops.**

This lesson will help you review Python loops by providing some challenge exercises involving loops.

Some of these challenges are difficult! Take some time to think about them before starting to code.

You might not get the solution correct on your first try — look at your output, try to find where you’re going wrong, and iterate on your solution.

Finally, if you get stuck, use our solution code! If you “Check Answer” twice with an incorrect solution, you should see an option to get our solution code. However, truly investigate that solution — experiment and play with the solution code until you have a good grasp of how it is working. Good luck!

**Function and Loop Syntax**

As a refresher, function syntax looks like this:

def some\_function(some\_input1, some\_input2):  
  … do something with the inputs …  
  return output

For example, a function that prints all odd numbers in a list would look like this:

def odd\_nums(lst):  
  for item in lst:  
    if item % 2 == 1:  
      print(item)

And this would produce output like:

>>> odd\_nums([4, 7, 9, 10, 13])  
7  
9  
13

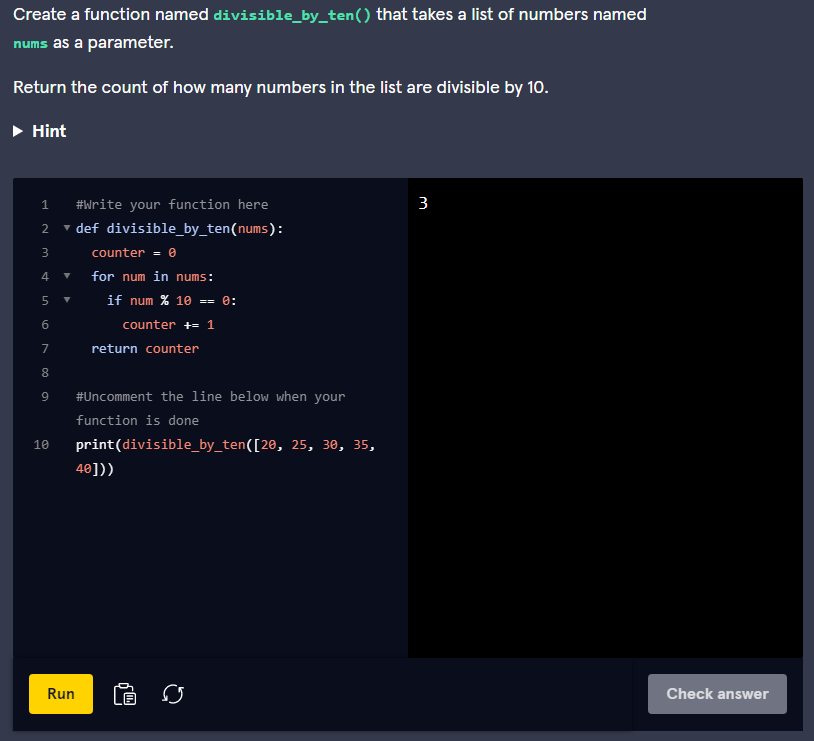
**Challenges**

We’ve included 5 challenges below. Try to answer all of them and polish up your problem-solving skills and your loop expertise.

### 1. Divisible By Ten

Let’s start our code challenges with a function that counts how many numbers are divisible by ten from a list of numbers. This function will accept a list of numbers as an input parameter and use a loop to check each of the numbers in the list. Every time a number is divisible by 10, a counter will be incremented and the final count will be returned. These are the steps we need to complete this:

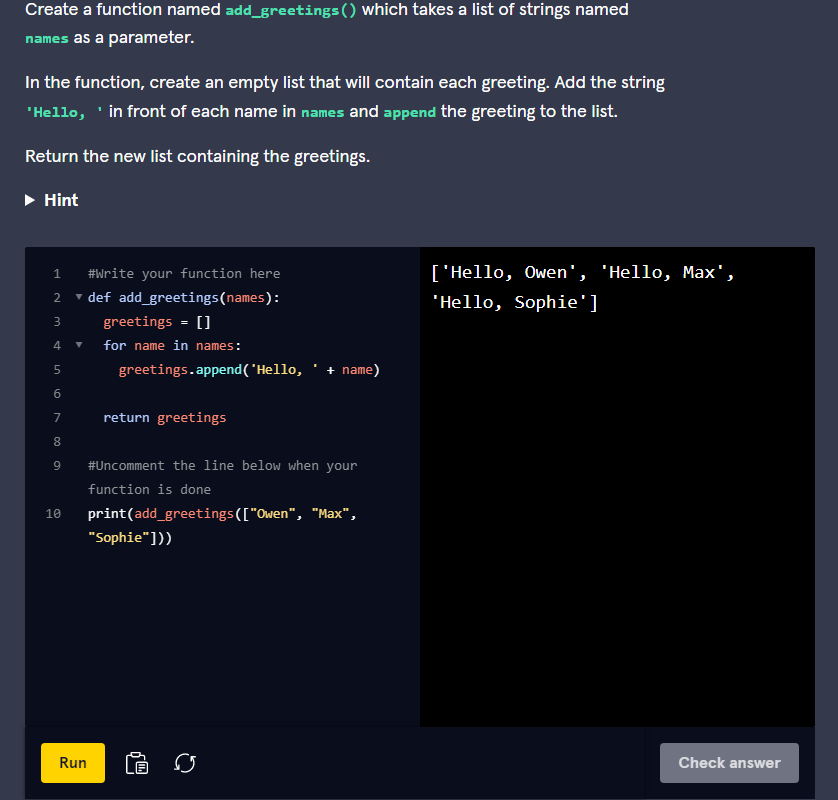
1. Define the function to accept one input parameter called **nums**
2. Initialize a counter to 0
3. Loop through every number in **nums**
4. Within the loop, if any of the numbers are divisible by 10, increment our counter
5. Return the final counter value



### 2. Greetings

You are invited to a social gathering, but you are tired of greeting everyone there. Luckily we can create a function to accomplish this task for us. In this challenge, we will take a list of names and prepend the string **'Hello, '** before each name. This will require a few steps:

1. Define the function to accept a list of strings as a single parameter called **names**
2. Create a new list of strings
3. Loop through each name in **names**
4. Within the loop, concatenate **'Hello, '** and the current name together and append this new string to the new list of strings
5. Afterwards, return the new list of strings



This is one way to solve it:

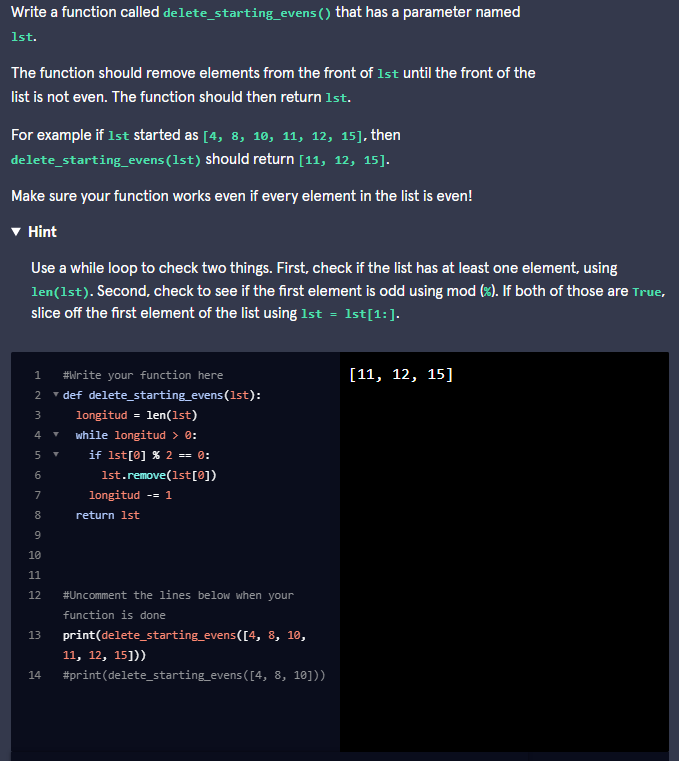
def add\_greetings(names):  
  new\_list = []  
  for name in names:  
    new\_list.append("Hello, " + name)  
  return new\_list

First, we set up our function to accept the list of strings and we initialized a new list of strings to hold our greetings. We iterate through each name and we append and concatenate the strings at the same time within our loop. Finally, we return the list containing all of our eloquent greetings.

### 3. Delete Starting Even Numbers

Let’s try a tricky challenge involving removing elements from a list. This function will repeatedly remove the first element of a list until it finds an odd number or runs out of elements. It will accept a list of numbers as an input parameter and return the modified list where any even numbers at the beginning of the list are removed. To do this, we will need the following steps:

1. Define our function to accept a single input parameter **lst** which is a list of numbers
2. Loop through every number in the list if there are still numbers in the list and if we haven’t hit an odd number yet
3. Within the loop, if the first number in the list is even, then remove the first number of the list
4. Once we hit an odd number or we run out of numbers, return the modified list



This is the way we solved it:

def delete\_starting\_evens(lst):  
  while (len(lst) > 0 and lst[0] % 2 == 0):  
    lst = lst[1:]  
  return lst

After defining our method, we use a **while** loop to keep iterating as long as some provided conditions are true. We provide two conditions for the **while** loop to continue. We will keep iterating as long as there is at least one number left in the list **len(lst) > 0** and if the first element in the list is even **lst[0] % 2 == 0**. If both of these conditions are true, then we replace the list with every element except for the first one using **lst[1:]**. Once the list is empty or we hit an odd number, the **while** loop terminates and we return the modified list.