**List accessing**

This exercise goes over just pulling information from a list, which we've covered in a previous section!

Remember that elements in a list start from index 0 and they are accessed in the form

x[n]

where x is the name of the list and n is the index in that list that you're trying to access.

# List element modification

You've already learned how to modify elements of a list in a previous section. This exercise is just a recap of that!

An item in a list in Python can be set to a value using the form

x[n] = v

where x is the name of the list, nis the index in the array and v is the value you want to set.

# Appending to a list

Here, we'll quickly recap how to .append()elements to the end of a list.

The function to append to the end of a list is

x.append(item)

where x is the name of the list and item is the object you want to append.

# Removing elements from lists

This exercise will expand on ways to remove items from a list. You actually have a few options. For a list called n:

1. n.pop(index) will remove the item at index from the list and return it to you:

n = [1, 3, 5] n.pop(1) # Returns 3 (the item at index 1) print n # prints [1, 5]

1. n.remove(item) will remove the actual item if it finds it:

n.remove(1) # Removes 1 from the list, # NOT the item at index 1 print n # prints [3, 5]

1. del(n[1]) is like .pop in that it will remove the item at the given index, but it won't return it:

del(n[1]) # Doesn't return anything print n # prints [1, 5]

# Changing the functionality of a function

In this exercise, you will just be making a minor change to a function to change what that function does.

# More than one argument

This exercise will recap how to use more than one argument in a function.

Define a function called add\_functionthat has 2 parameters x and y and adds them together.

|  |
| --- |
| def add\_function(x,y):  return x+y |

# Strings in functions

This is a basic recap on using strings in functions.

Write a function called string\_function that takes in a string argument (s) and then returns that argument concatenated with the word 'world'. Don't add a space before world!

|  |
| --- |
| n = "Hello"  # Your function here!  def string\_function(s):  return s+" world" |

# Passing a list to a function

You pass a list to a function the same way you pass any other argument to a function.

|  |
| --- |
| def list\_function(x):  return x  n = [3, 5, 7]  print list\_function(n) |

**Using an element from a list in a function**

Passing a list to a function will store it in the argument (just like with a string or a number!)

def first\_item(items): print items[0] numbers = [2, 7, 9] first\_item(numbers)

1. In the example above, we define a function called first\_item. It has one argument called items.
2. Inside the function, we print out the item stored at index zero of items.
3. After the function, we create a new list called numbers.
4. Finally, we call the first\_item function with numbers as its argument, which prints out 2.

**Modifying an element of a list in a function**

Modifying an element in a list in a function is the same as if you were just modifying an element of a list outside a function.

def double\_first(n): n[0] = n[0] \* 2 numbers = [1, 2, 3, 4] double\_first(numbers) print numbers

1. We create a list called numbers.
2. We use the double\_first function to modify that list.
3. Finally, we print out [2, 2, 3, 4]

When we pass a list to a function and modify that list, like in the double\_firstfunction above, we end up modifying the original list.

# List manipulation in functions

You can also append or delete items of a list inside a function just as if you were manipulating the list outside a function.

my\_list = [1, 2, 3] my\_list.append(4) print my\_list # prints [1, 2, 3, 4]

The example above is just a reminder of how to append items to a list.

# Printing out a list item by item in a function

This exercise will go over how to utilize every element in a list in a function. You can use the existing code to complete the exercise and see how running this operation inside a function isn't much different from running this operation outside a function.

Don't worry about the range function quite yet—we'll explain it later in this section.

# Modifying each element in a list in a function

This exercise shows how to modify each element in a list. It is useful to do so in a function as you can easily put in a list of any length and get the same functionality. As you can see, len(n) is the length of the list.

Create a function called double\_listthat takes a single argument x(which will be a list) and multiplies each element by 2 and returns that list. Use the existing code as a scaffold.

|  |
| --- |
| n = [3, 5, 7]  def double\_list(x):  for i in range(0, len(x)):  x[i] = x[i] \* 2  return x  # Don't forget to return your new list!  print double\_list(n) |

**Passing a range into a function**

Okay! Range time. The Python range()function is just a shortcut for generating a list, so you can use ranges in all the same places you can use lists.

range(6) # => [0, 1, 2, 3, 4, 5] range(1, 6) # => [1, 2, 3, 4, 5] range(1, 6, 3) # => [1, 4]

The range function has three different versions:

1. range(stop)
2. range(start, stop)
3. range(start, stop, step)

In all cases, the range() function returns a list of numbers from start up to (but not including) stop. Each item increases by step.

If omitted, start defaults to 0 and stepdefaults to 1.

# Iterating over a list in a function

Now that we've learned about range, we have two ways of iterating through a list.

**Method 1** - for item in list:

for item in list: print item

**Method 2** - iterate through indexes:

for i in range(len(list)): print list[i]

Method 1 is useful to loop through the list, but it's not possible to modify the list this way.

Method 2 uses indexes to loop through the list, making it possible to also modify the list if needed. Since we aren't modifying the list, feel free to use either one on this lesson!

# Using two lists as two arguments in a function

Using multiple lists in a function is no different from just using multiple arguments in a function!

a = [1, 2, 3] b = [4, 5, 6] print a + b # prints [1, 2, 3, 4, 5, 6]

The example above is just a reminder of how to concatenate two lists.

**Using a list of lists in a function**

Finally, this exercise shows how to make use of a single list that contains multiple lists and how to use them in a function.

list\_of\_lists = [[1, 2, 3], [4, 5, 6]] for lst in list\_of\_lists: for item in lst: print item

1. In the example above, we first create a list containing two items, each of which is a list of numbers.
2. Then, we iterate through our outer list.
3. For each of the two inner lists (as lst), we iterate through the numbers (as item) and print them out.

We end up printing out:

1 2 3 4 5 6

|  |
| --- |
| n = [[1, 2, 3], [4, 5, 6, 7, 8, 9]]  # Add your function here  def flatten(lists):  results = []  for numbers in lists:  for i in numbers:  results.append(i)  return results  print flatten(n) |