

# Discussion 3: Domain and Range, Higher Order Functions

## Domain and Range

1. Determine the domain and range of the following Snap! blocks:

a.



Domain: Lists (first blank), any type (second blank)

Range: Booleans

Explanation: We can tell the domain by the input shapes. Since this block is hexagon-shaped, it is a predicate, meaning its range is Booleans.

b.



Domain of foo: Numbers

Range of foo: Booleans

Data type of var: Boolean

Explanation: Similarly to the last example, we can tell the domain of foo is numbers because of the shape of its input slot. Because foo is hexagon-shaped, it is a predicate, so it outputs Booleans. var is then set to the output of foo, so the data type of var is also Booleans.

c.






Domain of foo: Lists (first blank), numbers (second blank)

Range of foo: Lists

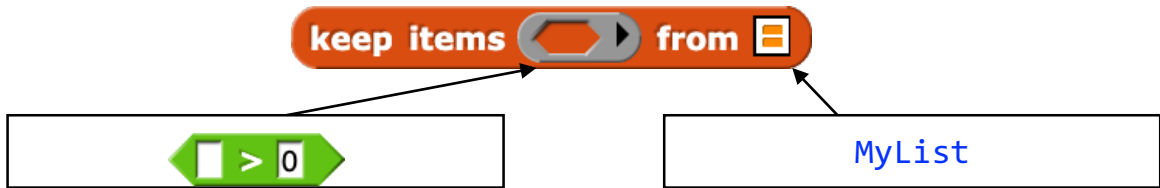
Explanation: We first look at the outer foo block. This block has two inputs: one is the output of foo, and the other is the output of the + block. Thus, we know that the domain of the first blank is equal to the range of foo, and that the domain of the second blank is equal to the range of the + block (numbers). Since foo is passed into the item block, which takes in lists, the range of foo (and thus the domain of the first blank) is lists.

2. Fill in the table below with the domain and range of the following higher-order functions:

Higher Order Function	Domain	Range	Notes
 <p>The image shows a Scratch 'map over' block. It consists of a red rounded rectangle with the word 'map' on the left, a grey arrow pointing right in the middle, and the word 'over' on the right. There are two small square icons: one on the left side of the arrow and one on the right side of the 'over' text.</p>	<p>First blank: reporter Second blank: list</p>	<p>List, with the function applied to each item in the input list</p>	<p>The output list has the same length as the input list, and generally, the items in the output list have been modified.</p>
 <p>The image shows a Scratch 'keep items from' block. It consists of a red rounded rectangle with the text 'keep items' on the left, a grey arrow pointing right in the middle, and the text 'from' on the right. There are two small square icons: one on the left side of the arrow and one on the right side of the 'from' text.</p>	<p>First blank: predicate Second blank: list</p>	<p>List, with only the items from the input list that satisfy the predicate</p>	<p>The output list has length less than or equal to that of the input list. The items that satisfy the predicate are not modified.</p>
 <p>The image shows a Scratch 'combine using' block. It consists of a red rounded rectangle with the text 'combine' on the left, a small square icon on the right, the text 'using' in the middle, and a grey arrow pointing right on the far right.</p>	<p>First blank: list Second blank: reporter</p>	<p>A value (the type of the value is equivalent to the output type of the reporter)</p>	

## Higher Order Functions

1. Fill in the blanks so the keep block returns a list of the positive numbers from MyList. You may assume MyList only contains numbers.



2. Write an expression that returns the sum of the squares of the numbers in YourList. You may assume that YourList only contains numbers.

Answer:



3. Describe in words what the following block outputs. You may assume OurList is a list of words.



It reports a word made up of the last letter of each word in OurList.

4. Write an expression that takes in a list, var, and returns whether there is a word with over 5 letters in the list. You may not use 'length of' or 'contains thing' in your solution.

Answer:



5. What is the output of the following block?



Answer: list 5 9 6

## Challenge

1. For the questions below, determine if the two expressions are equivalent for all possible reporters **F** and lists **DATA**.

a. ,  no

Explanation: In the first block, F is taking in DATA directly, whereas in the second block, F takes in each item of DATA.

An example of where these wouldn't be equal is if F was the plus block, and DATA was any list of numbers.

b. ,  yes

Explanation: In the left block, each item of DATA is first being passed into F, and then the output of that would be passed into F again, meaning you are calling F twice on each item of DATA. In the right block, we are doing just that: we first apply F once to each item of DATA in the inner call to map, and then do it again in the outer call to map.

c. ,  no

Explanation: In the left block, we are first applying F directly to DATA (not to the items of DATA). In the right block, as explained above, we are applying F twice to each item of DATA.

d. ,  no

Explanation: In the left block, we are first applying F to each item of DATA, then applying F to the list of modified items. In the right block, we are applying F directly to DATA (not the items of DATA) twice.

2. What is the output of the following block?



The block is a 'combine' block with the following structure:

- item 1 of
  - map
    - keep items: 3 < [ ] and [ ] < 9
    - from: [ ] (list icon)
  - keep items: is [ ] a list?
  - from: [ ] (list icon) and [ ] (list icon) containing [5, 6, 8, 9] and [1, 2, 5, 3, 4, 7]
- using: +

3. Which higher order function(s) could we use to solve the following problems? If there are multiple ways to solve the problem, indicate the most concise way to solve it (i.e. if there is a solution that uses 2 higher order functions and another that uses just 1, answer with the solution that uses just 1). You may assume you have access to any helper functions, as long as they don't use any loops in them.

a. Given a list of numbers, find the smallest number above 10.

Keep fed into combine. We would first use keep to find all of the numbers above 10, and then combine to find the smallest of those numbers.

b. Given a list of words, we want to find the word that comes first alphabetically.

Combine. We could use combine with a reporter that takes in two words and reports which one comes first alphabetically.

c. Given a list of lists, return the first item from each list.

Map. We could map  over the list of lists.