1) Returns the value 3
The word is “karel” and each item in the list is being compared to it, starting with the first item. Since the first item in the list is “karel”, it equals the word “karel” and so x goes up by 1. We don’t take the “else” path since we took the most recent “if” path before it. However we do try the next “if” condition: Is x (which is 1) greater than y (which is 0)? Yes, so y takes on the value 1. The next item in the list is “karel” too, so x goes up by 1 again for the same reason, and because x (which is 2) is greater than y (which is 1), y takes on the value 2.

Since the next item in the list is not “karel”, the first “if” condition does not get satisfied, so we take the “else” path, which resets x to 0. Since x (which is 0) is not greater than y (which is 2), y does not take on the value of x, and so y stays at the value 2.

x will starting rising again as we see more “karel”s, but only when it gets to the third “karel” in a row will the value of x exceed the value of y, at which point y will take on the value of x, which will be 3 at that point.

In other words, y will only take on the value of x if the value of x has exceeded its previous maximum value, which is based on the number of consecutive times that the word (in this case “karel”) appears in the list. In other words, this code will always return the absolute maximum number of times that a particular word appears consecutively in a particular list.

2) Returns the expression 5*x.
\[ x + \text{result} = x + 0 = x, \text{ so result} = x \text{ after the first iteration.} \]
On the second iteration, we have \[ x + \text{result} = x + x, \]
which means \[ \text{result} = x + x, \text{ or } 2*x. \]
In the third iteration, \[ \text{result} + x = 2*x + x = 3*x, \text{ so result} = 3*x, \text{ and so on.} \]
After the fifth and final iteration is done, result = 5 * x by this pattern.

3) Returns the expression 2 * 3 * x
The inner loop results in 3*x by the same logic as in question # 2. When we repeat the inner loop a second time, the result starts as 3*x, so result + x = 3*x + x = 4x, then on the second iteration, result + x = 4*x + x = 5x, then on the third iteration, result + x = 5*x + x, which finally gives us 6*x.

4) Displays the value 19
what = 1
what + 1 = 1 + 1 = 2, so does = 2.
what + does + 3 = 1 + 2 + 3 = 6, so the = 6.
the + what = 6 + 1 = 7, so fox = 7.
say = 3, so what + does + the + fox + say = 1 + 2 + 6 + 7 + 3 = 19, so result = 19.

5) Choice 3
Let x = 3 and y = 4 (you can choose any values since this is a generalization).
Choice 1 is wrong, because this is what it does:
y = 3, temp = 3, x = 3, so all variables end up taking the value of x.
Choice 2 is wrong, because this is what it does:
x = 4, y = 4, so all variables end up taking the value of y.
Choice 3 is correct, because this is what it does:
temp = 3, x = 4, y = 3, so x and y have successfully swapped values.
Choice 4 is wrong, because this is what it does:
temp = 3, y = 3, y = 3, so all variables end up taking on the value of x.

6) Only program 1 (not program 2)
Let number_list = [2, 1, 4, 3] (you can choose any values since this is a generalization)
In program 1, \( \text{sum} + \text{number} = 0 + 2 \) (in the first iteration), so \( \text{sum} = 2 \). In the second iteration, \( \text{sum} + \text{number} = 2 + 1 = 3 \), so \( \text{sum} = 3 \). In the next iteration, \( \text{sum} + \text{number} = 3 + 4 \), so \( \text{sum} = 7 \). In the next iteration, \( \text{sum} + \text{number} = 7 + 3 = 10 \), so \( \text{sum} = 10 \). The sum is then divided by the length of the list, which is 4 (the number of values in the list), and so we get \( 10/4 \), which is exactly what we would do when calculating the average of this particular list.

In the second program, \( \text{sum} \) also reaches the actual sum of the list, which in our example is 10. However, counter reaches 5, because it started at 1 rather than at 0, so we end up with \( 10/5 \), which is not the correct way to calculate the average of this list. If the counter had started at 0, then it would have ended up at 4 and our result would be \( 10/4 \) which is the correct average of this list.