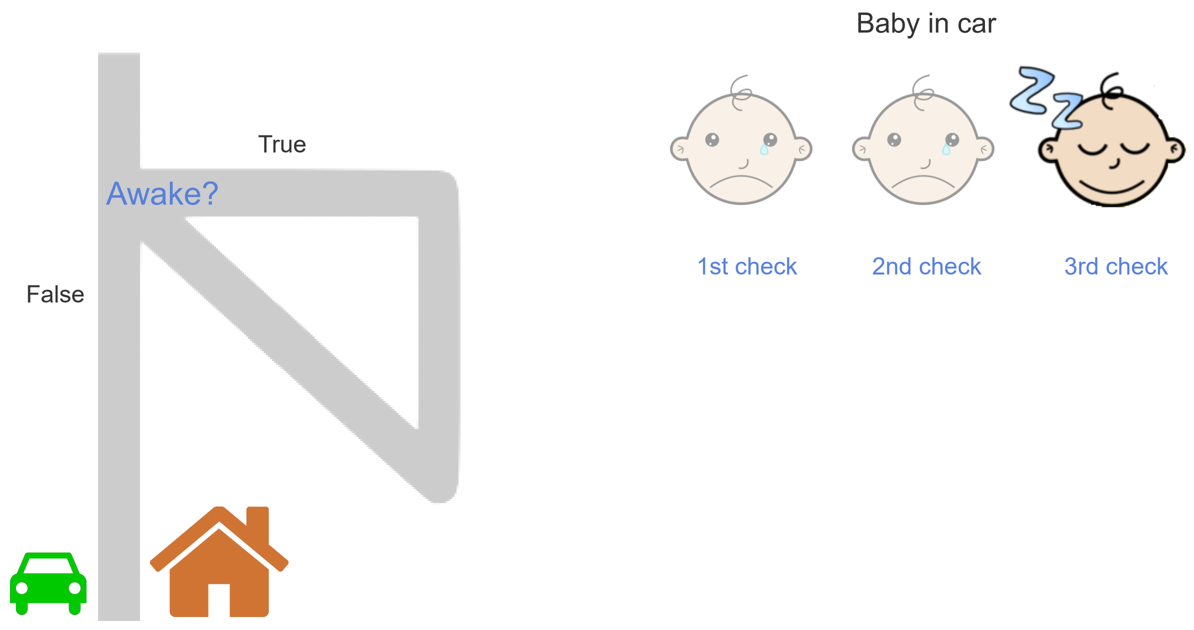
5.1 Loops (general)

Loop concept

People who have children may be familiar with looping around the block until a baby falls asleep.

PARTICIPATION ACTIVITY 5.1.1: Loop concept: Driving a baby around the block.

Start 2x speed



Captions

1. Parents may be familiar with this scenario: Driving home, the baby is awake. Parents circle the block, hoping the baby will fall asleep.
2. After the first loop, the baby is still awake, so the parents loop again.
3. After the second loop, the baby is asleep, so the parents head home for a peaceful evening.

PARTICIPATION ACTIVITY 5.1.2: Loop concept.

Consider the example above.

1) When the parents first checked, was the baby awake?

- Yes
- No

2) After the first loop, was the baby awake?

- Yes
- No

3) After the second loop, was the baby awake? ✓

- Yes
- No

4) How many loops around the block did the parents make? ✓

- 2
- 3

5) Where was the decision point for whether to loop: At the top of the street or the bottom? ✓

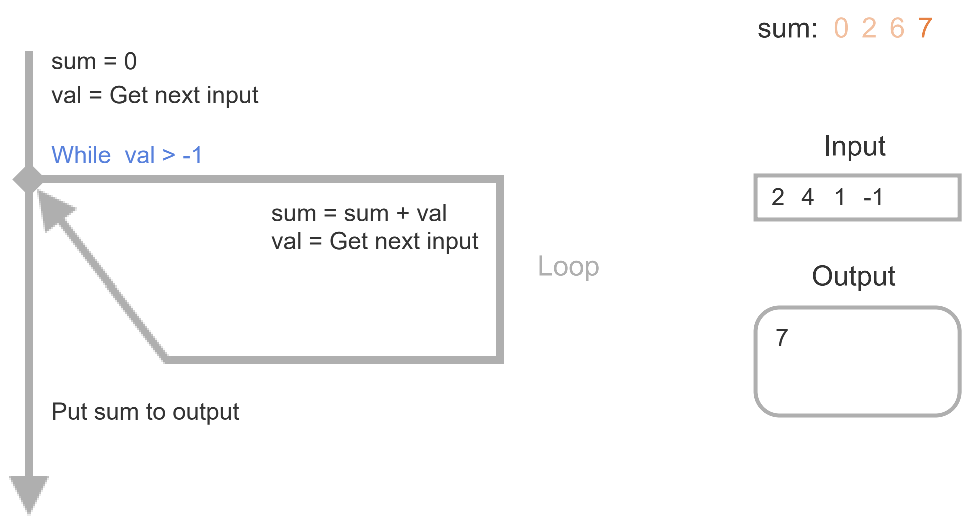
- Top
- Bottom

Loop basics

A ***loop*** is a program construct that repeatedly executes the loop's statements (known as the ***loop body***) while the loop's expression is true; when the expression is false, execution proceeds past the loop. Each time through a loop's statements is called an ***iteration***.

PARTICIPATION ACTIVITY 5.1.3: A simple loop: Summing the input values. ✓

Start ☐ 2x speed



Captions

1. A loop is like a branch, but the loop jumps back to the expression when done. Thus, the loop's statements may execute multiple times before execution proceeds past the loop.
2. This program receives an input value. If the value > -1, the program adds the value to a sum, receives another input, and repeats. val is 2, so the loop's statements execute, making sum 2.
3. The loop's statements ended by receiving the next input, which is 4. The loop's expression 4 > -1 is true, so the loop's statements execute again, making sum 2 + 4 or 6.

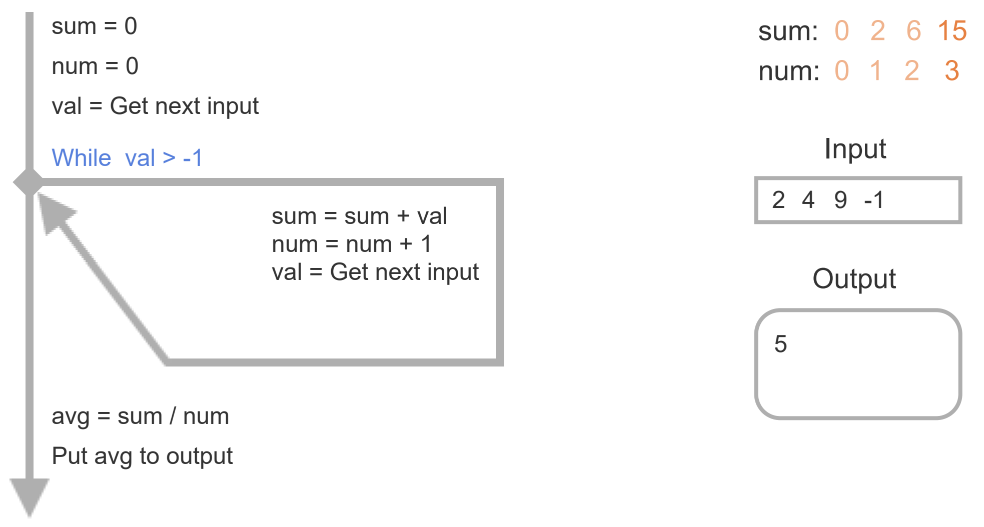
4. The loop's statements receive the next input of 1. The loop's expression 1 > -1 is true, so the loop's statements execute a third time, making sum 6 + 1 or 7.
5. The next input is -1. This time, -1 > -1 is false, so the loop is not entered. Instead, execution proceeds past the loop, where a statement puts sum, which is 7, to the output.

Loop example: Computing an average

A loop can be used to compute the average of a list of numbers.

PARTICIPATION ACTIVITY | 5.1.4: Loop example: Computing an average.

Start 2x speed



Captions

1. The program computes an average of a list of numbers (a negative ends the list). The first input is 2, so the loop is entered. Sum becomes 2, and num is incremented to 1.
2. The next input is 4. The loop is entered, so sum becomes 2 + 4 or 6, and num is incremented to 2.
3. The next input is 9, so the loop is entered. Sum becomes 6 + 9 or 15, and num is incremented to 3.
4. The next input is -1, so the loop is not entered. 15 / 3 or 5 is output.

PARTICIPATION ACTIVITY | 5.1.5: Loop example: Average.

Consider the computing an average example above.

1) In the example above, the first value received from input was 2. That caused the loop body to be _____.

- executed
- not executed

2) At the end of the loop body, the _____.

- next input is received
- loop is exited
- average is computed

3) With what value was sum initialized?

- -1
- 0

4) Each time through the loop, the sum variable is increased by _____.

- 0
- 1
- the current input value

5) What was variable num's value after the loop was done iterating?

- 1
- 2
- 3

6) Before the loop, the first input value is received. If that input was negative (unlike the data in the example above), the loop's body would _____.

- be executed
- not be executed

Example: Counting specific values in a list

Programs execute one statement at a time. Thus, using a loop to examine a list of values one value at a time and updating variables along the way, as in the above examples, is a common programming task.

Below is a task to help a person get accustomed to examining a list of values one value at a time. The task asks a person to count the number of negative values, incrementing a variable to keep count.

PARTICIPATION ACTIVITY 5.1.6: Counting negative values in a list of values.

Click "Increment" if a negative value is seen.

Start

List								Counter

Next value Increment

PARTICIPATION ACTIVITY | 5.1.7: Counting negative values. ✓

Complete the program such that variable count ends having the number of negative values in an input list of values (the list ends with 0). So if the input is -1 -5 9 3 0, then count should end with 2.

```
count = 0
val = Get next input

While val is not 0
   if __(A)__
      __(B)__

   val = Get next input
```

1) What should expression (A) be? ✓

- ○ val > 0
- ○ val < 0
- ○ val is 0

2) What should statement (B) be? ✓

- ○ val = val + 1
- ○ count = count + 1
- ○ count = val

3) If the input value is 0, does the loop body execute? ✓

- ○ Yes
- ○ No

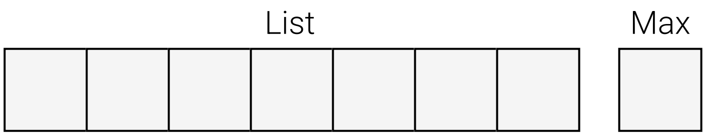
Example: Finding the max value

Examining items one at a time and updating a variable can achieve some interesting computations. The task below is to find the maximum value in a list of positive values. A variable stores the max value seen so far. Each input value is compared with that max, and if greater, that value replaces that max. The max value is initialized with -1 so that such comparison works even for the first input value.

PARTICIPATION ACTIVITY | 5.1.8: Find the maximum value in the list of values. ✓

Click "Store value" if a new maximum value is seen.

Start



Next value Store value

PARTICIPATION ACTIVITY | 5.1.9: Determining the max value. ✓

Complete the program so the variable max ends up having the maximum value in an input list of positive values (the list ends with 0). So if the input is 22 5 99 3 0, then max should end as 99.

```
max = -1
val = Get next input

while val is not 0
   If __(A)__
      __(B)__

   val = Get next input
```

1) What should expression (A) be? ✓

- ○ max > 0
- ○ max > val
- ○ val > max

2) What should statement (B) be? ✓

- ○ max = val
- ○ val = max
- ○ max = max + 1

3) Does the final value of max depend on the order of inputs? In particular, would max be different for inputs 22 5 99 3 0 versus inputs 99 3 5 22 0? ✓

- ○ Yes
- ○ No

4) For inputs 5 10 7 20 8 0, with what values should max be assigned? ✓

- ○ -1, 20
- ○ -1, 5, 10, 20
- ○ -1, 5, 10, 7, 20

Activity summary for assignment: C5

224 / 224 points

Due: 02/28/2026, 11:59 PM PST

Completion details