LOOP PATTERNS

An Introduction to Computer Science

Let's learn about Loop Patterns

Many Patterns

- Count
- Sum
- Accumulate
- Map
- Filter

There are so many ways to use For loops.
In this lesson, we will review a few common patterns.
These patterns shown here are just starting points.
Think of them as templates that can be adapted and combined to solve more complex problems.

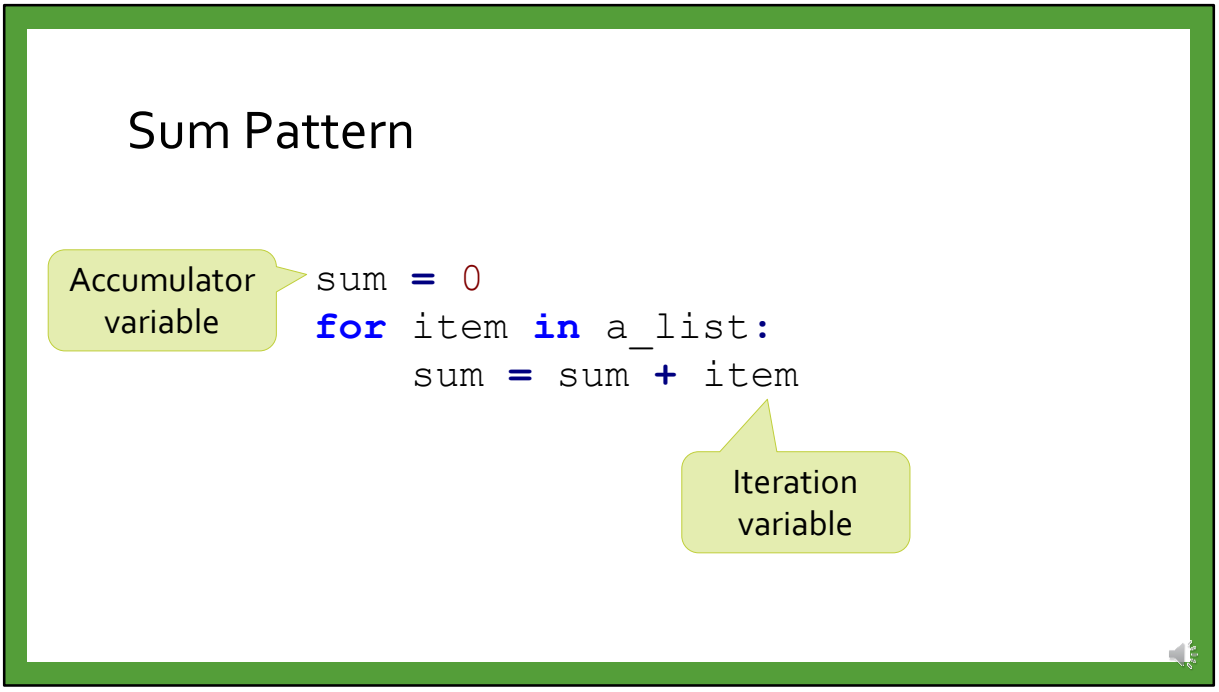
Count Pattern

Accumulator variable

```
count = 0
for item in a_list:
    count = count + 1
```

We have a list of items, and want to know how many there are.
A simple algorithm is, starting with an initial value of 0, to add 1 for each element we see to a "count" variable.
When the loop is finished, the "count" variable will have the length of the list.

Sum Pattern



We have a list of numbers, and want to add them all up.
The plus operator can only take two items at a time, however.
Therefore, we add each element one at a time to a "sum" variable, which is also initialized to 0.
As you can see the, the sum pattern is similar to the Count pattern, except instead of adding 1, we are adding the iteration variable.

Accumulator Pattern

General Template

```
result = ___
for item in a_list:
    result = result ___ item
```

Strings (join together)

```
result = ""
for item in a_list:
    result = result + item
```

Booleans (any true?)

```
result = False
for item in a_list:
    result = result or item
```

Booleans (all true?)

```
result = True
for item in a_list:
    result = result and item
```

The Sum and Count patterns are both more specific examples of the accumulator pattern. In general, this pattern allows us to start with an initial value and use any function or operation that takes in two values.

This pattern can be applied to numbers, but it also works for strings, booleans, and even lists.

If you accumulate Strings using addition, you join them together into one big string.

If you accumulate Booleans using or, you ask if any element in the list is True.

If you accumulate Booleans using and, you ask if all elements in the list are True.

The initial value and the combining operation need to be chosen whenever you use the template.

This process of accumulation is also sometimes known as "reducing" or "folding" a list.

Map Pattern

```
copied_list = []
for item in old_list:
    copied_list.append(___)
```

Replace the blank with an expression involving `item` such as `item * 2`

What happens when we accumulate a list?
If we start with an empty list as our initial value, and append each value one at a time, we end up with a copy of the original list.
As we're appending values, we can also modify them.
For example, you could double each value from the old list, or convert each temperature from Fahrenheit to Celsius.

Filter Pattern

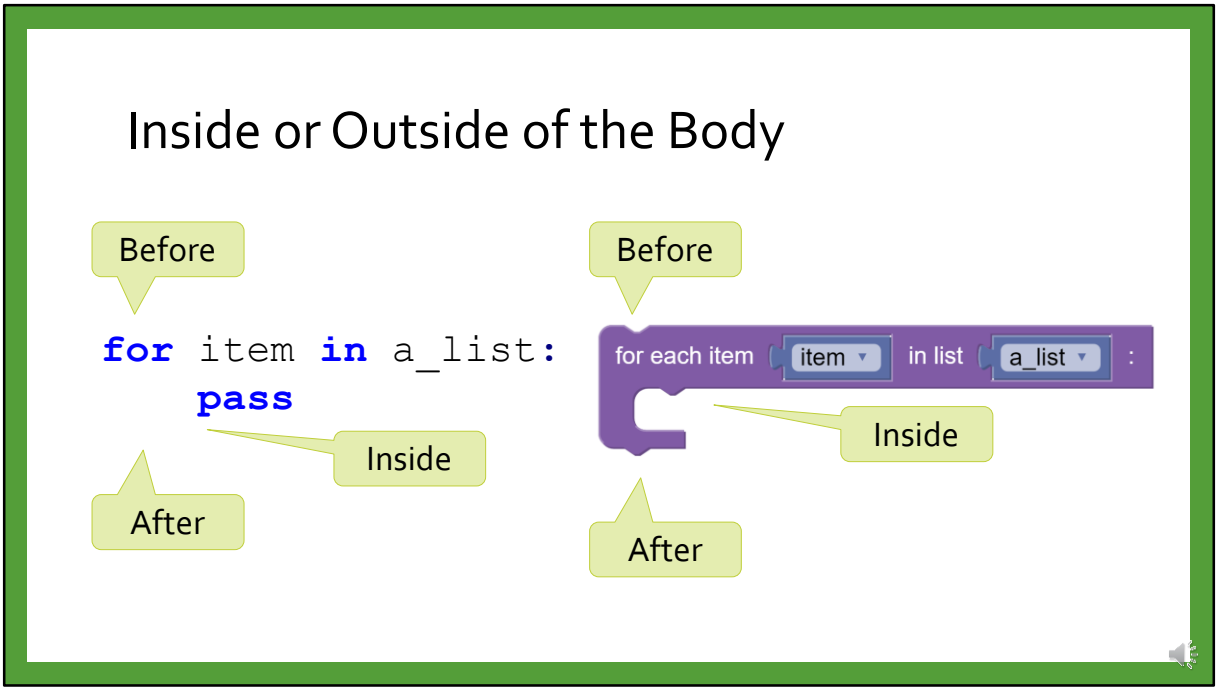
Filter + Count

```
result = 0
for item in a_list:
    if item < 100:
        result = result + 1
```

Filter + Map

```
result = []
for item in a_list:
    if item < 100:
        result.append(item)
```

We have a list of numbers, and want to ignore some of them according to a rule.
By embedding an IF statement inside the loop, we can optionally include or not include elements in our accumulation.
The Filter pattern is very compatible with the other patterns.



Novices always struggle with what goes inside or outside of a loop body.
Remember, every statement inside the body is executed for each element.
Only put things inside if they should happen for each element.
The patterns can help you keep track of where things go, but ultimately you have to think critically to know.