LOOP PATTERNS 2

An Introduction to Computer Science

Let's learn about more about Loop Patterns

Many Patterns

- Count
- Sum
- Accumulate
- Map
- Filter
- **Find**
- **Take When**
- **Min/Max**

We've already seen the Accumulate, Map, and Filter patterns.
Now we're going to cover a few more patterns that have more complex conditionals.

Find Pattern

```
found = None
for item in a_list:
    if ___:
        found = item
```

We have a list of items and want to find a specific value based on a condition.
This requires us to walk through the entire list and check whether that value satisfies our condition.
When we are done, the desired value will be in the found variable, or found will have the value None.

Take Pattern

```
taking = True
new_list = []
for item in a_list:
    if ___:
        taking = False
    elif taking:
        new_list.append(item)
```

We have a list of items, and we want a copy of the items up until a certain point.
We again have to walk through the list, but this time we will stop taking elements after we satisfy our condition.
Notice that the body of this loop is more complicated, using an `elif` statement to prevent us from taking elements after we have met the condition.

Min/Max Pattern

```
maximum = a_list[0]
for item in a_list:
    if item > maximum:
        maximum = item

print(maximum)
```

Must be a non-empty list!

The last pattern is used to find the highest or lowest element in a list.
At first, it may appear similar to the Filter pattern, because it uses an IF statement.
However, you can see that instead of comparing each element to a constant value, the conditional compares each value to the accumulated value.
Using this pattern, the accumulated value (in this case a maximum) will end up as the highest value in the list.