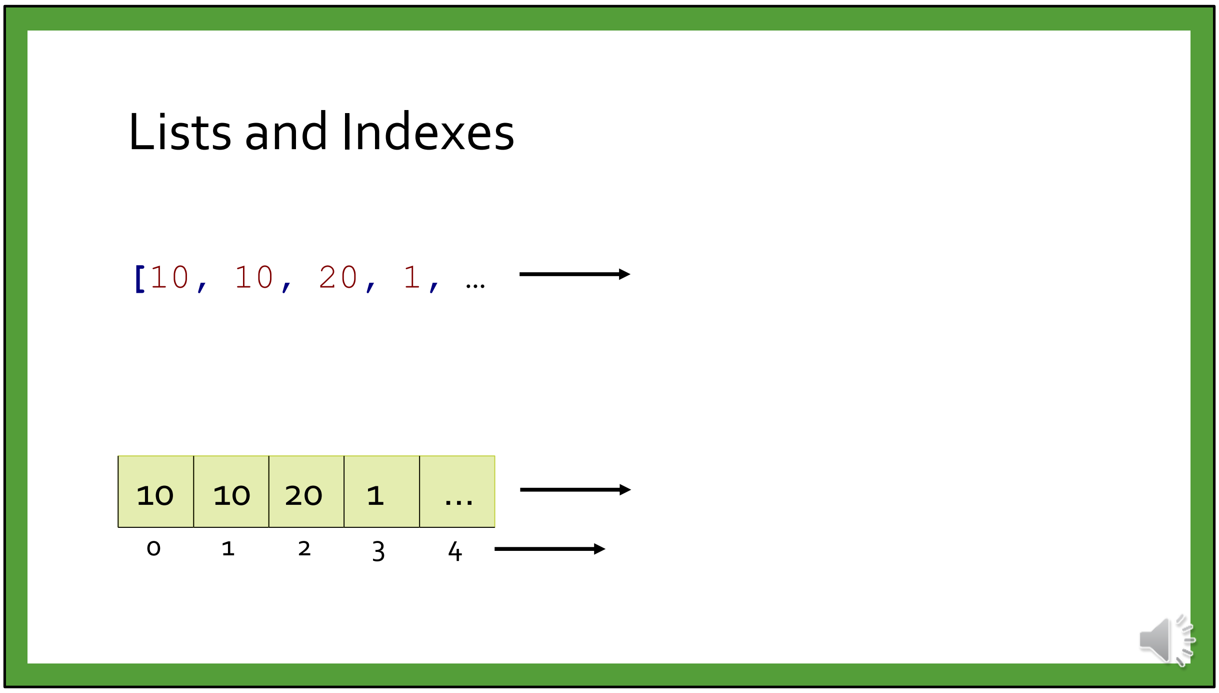
LISTS AND INDEXES

An Introduction to Computer Science

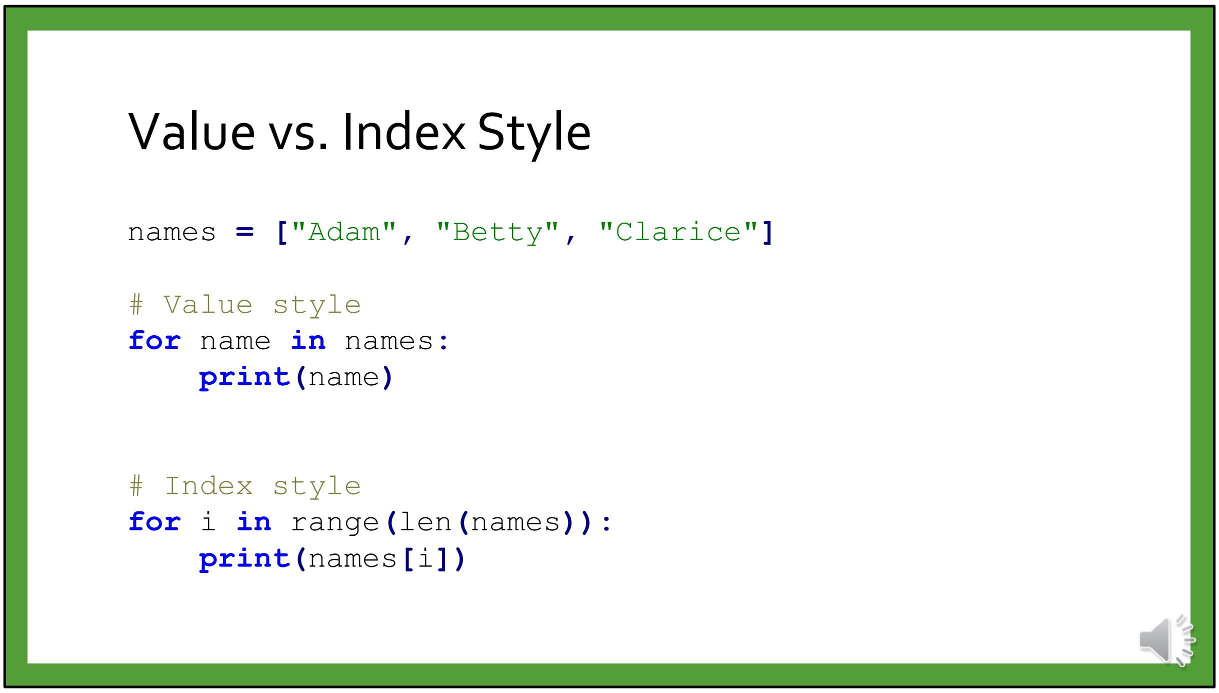
Let's learn about Lists and Indexes.



Lists and Indexes

```
[10, 10, 20, 1, …
```

So far, we have iterated through a list using the for-each syntax to get each value of the list.
In many languages, indexes are used to access the values of a list.
The index of the list are the integers, starting from 0 and counting up, that uniquely identify each element.



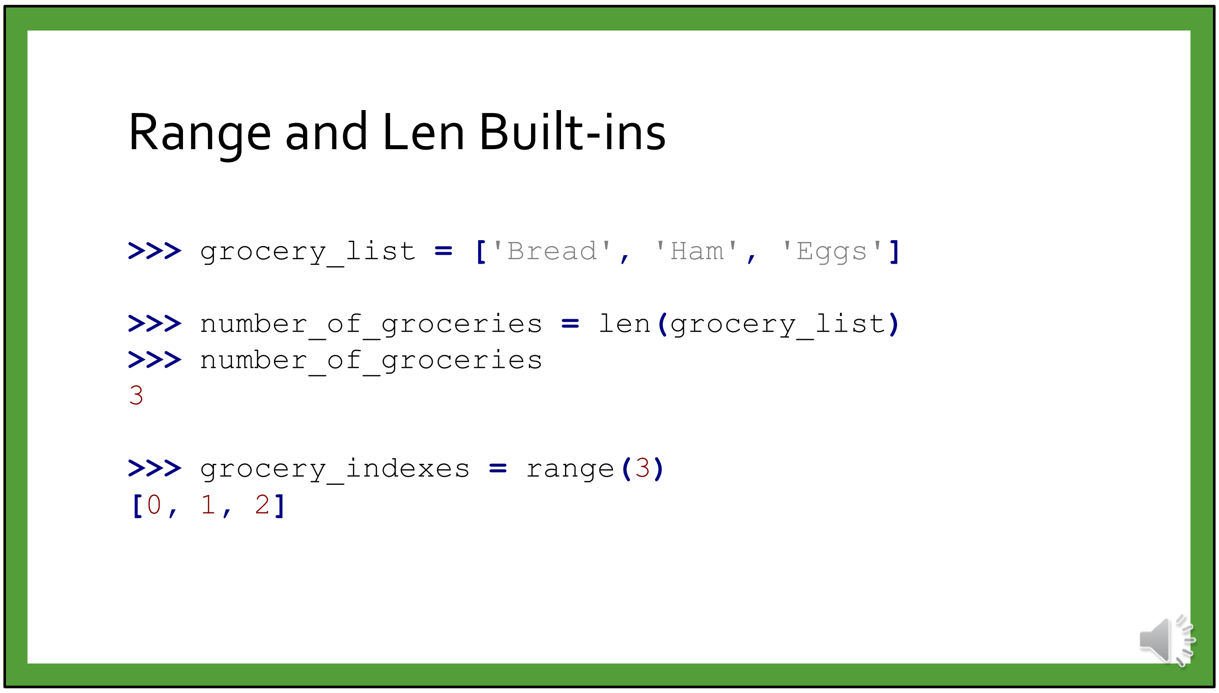
Value vs. Index Style

```
names = ["Adam", "Betty", "Clarice"]

# Value style
for name in names:
    print(name)


# Index style
for i in range(len(names)):
    print(names[i])
```

The top code shown here is the preferred way to process a list, where we iterate through each value.
However, you can also iterate by using a combination of the range and len functions, as shown here.
Both the Value style and the Index style shown will print the same result.



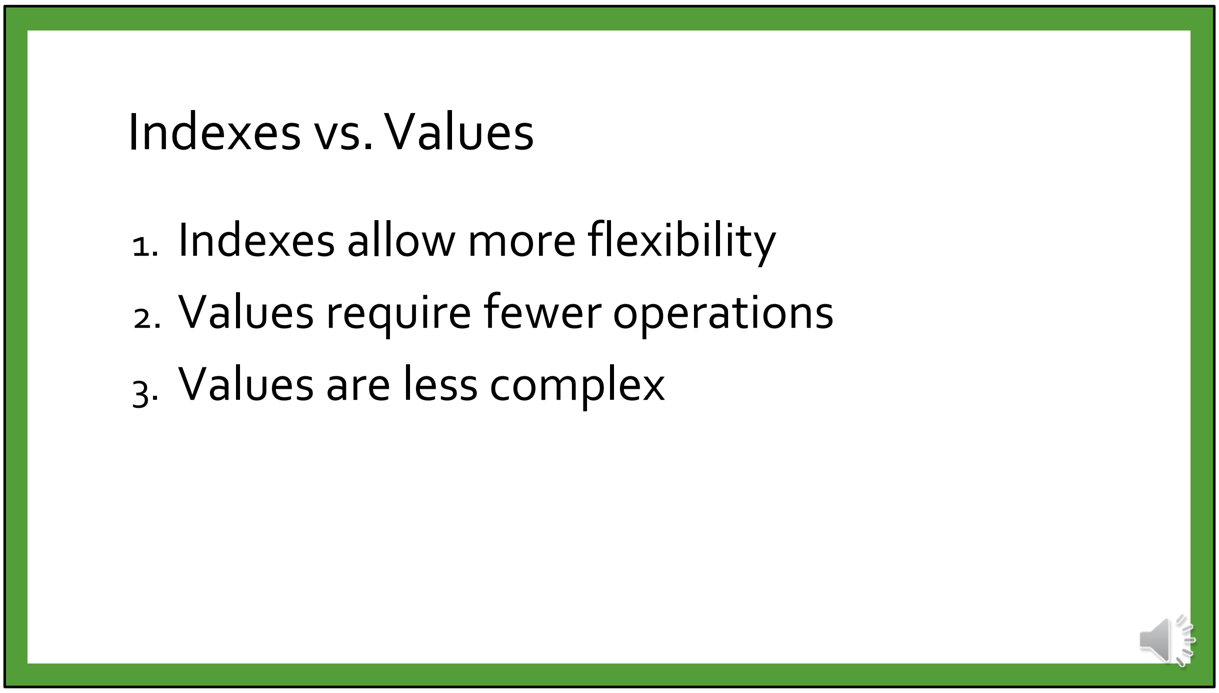
Range and Len Built-ins

```
>>> grocery_list = ['Bread', 'Ham', 'Eggs']

>>> number_of_groceries = len(grocery_list)
>>> number_of_groceries
3

>>> grocery_indexes = range(3)
[0, 1, 2]
```

The `range` and `len` functions are built-in to Python.
The `len` function consumes a list or other iterable and produces an integer representing its length.
The `range` function consumes an integer and produces a list of integers from 0 up till that number.
You can combine these two functions to create the list of indexes corresponding to a list.



Indexes vs. Values

1. Indexes allow more flexibility
2. Values require fewer operations
3. Values are less complex

So when do you use indexes and when do you use values?
Usually, you should stick to value style iteration unless you have an explicit reason to use indexes.
Indexes allow more flexibility and are necessary for certain kinds of specialized algorithms. However, they require more operations and are more complex to reason about.