# Chapter 5 - Lists

## Lists

## List Iteration

## Learning Objectives - List Iteration

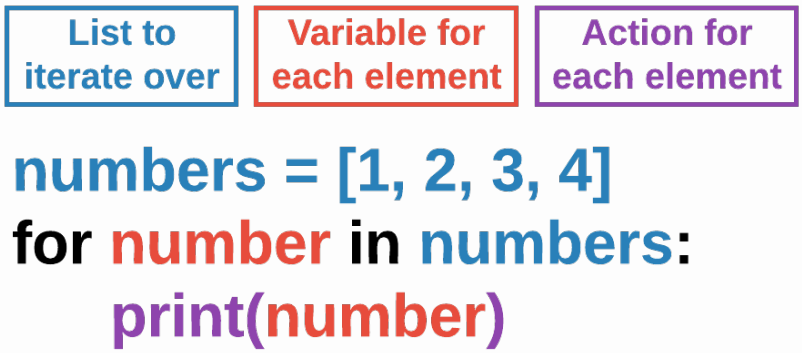
---

- **Define list iteration**
- **Identify which loops are used for list iteration**
- **Describe what happens when you iterate over a list?**

# List Iteration - For Loop

## Iterating Over Lists

The previous topics about lists have been about dealing with the list as a whole (sort, reverse, etc.) or dealing with individual elements of the list (pop, max, etc.). Iterating over a list allows you to deal with all of the elements in a list individually. Iterating over the list means to start with the element at index 0, and then progress until you reach the end of the list. A for loop is used to iterate over a list.




Iteration Variable

▼ **Number & Numbers**

In the example below, the iteration variable is `number` and the list is named `numbers`. This is a very common practice in Python. The list is always plural, while the iterating variable is the singular of the list name. Python will not throw an error if this convention is not followed. However, `for number in numbers` helps with the readability of your code. You should follow this convention as often as possible.

```
numbers = [1, 2, 3, 4]
for number in numbers:
    print(number)
```

challenge

## What happens if you:

- Change the value of `numbers` to `[101, 12, 36, 24, 55, 6]`?
- Change the value of `numbers` to `["cat", "dog", "mouse", "bird", "fish"]`?
- Change the `print` statement to `print(numbers)`?

## What Happens Behind the Scene

Use the code visualizer below and step through the code. Notice how the variable `number` is the value of the element. The index of the list is never referenced.

# List Iteration - While Loop

---

## Iteration with While Loops

For loops are almost exclusively used to iterate over lists. It is possible to use a while loop. Be aware of all of things you manually do when using a while loop over a for loop.

```
numbers = [1, 2, 3, 4]
length = len(numbers)
i = 0

while i < length:
    print(numbers[i])
    i += 1
```

Code Visualizer

challenge

## What happens if you:

- Change the loop to `while i <= length:`?
- Change the `print` statement to `print(i)`?
- Remove `i += 1`?

## Comparing While & For Loops

**While Loop**

```
numbers = [1, 2, 3, 4]
length = len(numbers)
i = 0

while i < length:
    print(numbers[i])
    i += 1
```

**For Loop**

```
numbers = [1, 2, 3, 4]

for number in numbers:
    print(number)
```

While and For Loops

The for loop is more efficient than a while loop when iterating over a list. You do not need to declare variables for the length of the list (red text), declare a variable for the index of the list (blue text), or increment the index variable (orange text). All of this is handled by the `in` statement. In for loops, you can use the iteration variable to reference the list element. With a while loop, however, you need to use the list and index to reference the element (purple text).