Chapter 6: Lists

SECTION 6.1

BASIC PROPERTIES OF LISTS

Lecture Goals

- Lists are the fundamental mechanism in Python for collecting multiple values that are stored in adjacent memory locations.
- In this lecture you will learn how:
  - To create lists
  - To access list elements
  - To use the `for` loop for traversing lists
  - To use list references

Creating a List

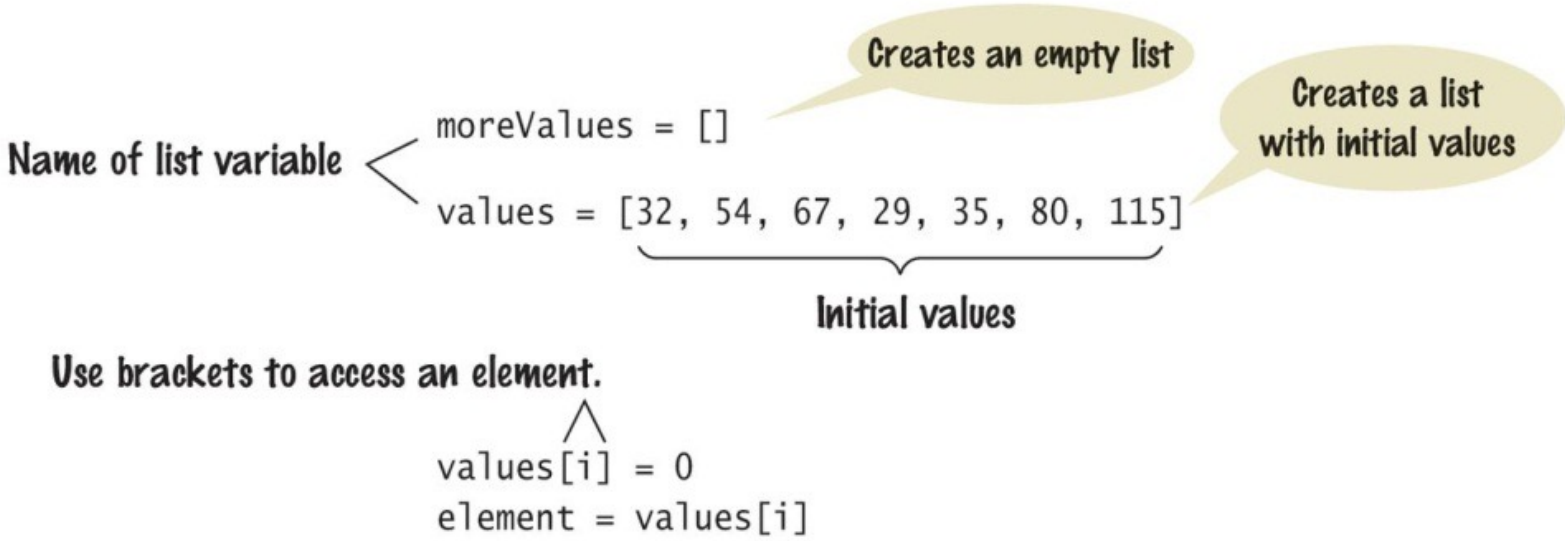
Code Academy

- Specify a list variable with the subscript operator []

Syntax

To create a list: $[value_1, value_2, \ldots]$

To access an element: *listReference*[*index*]

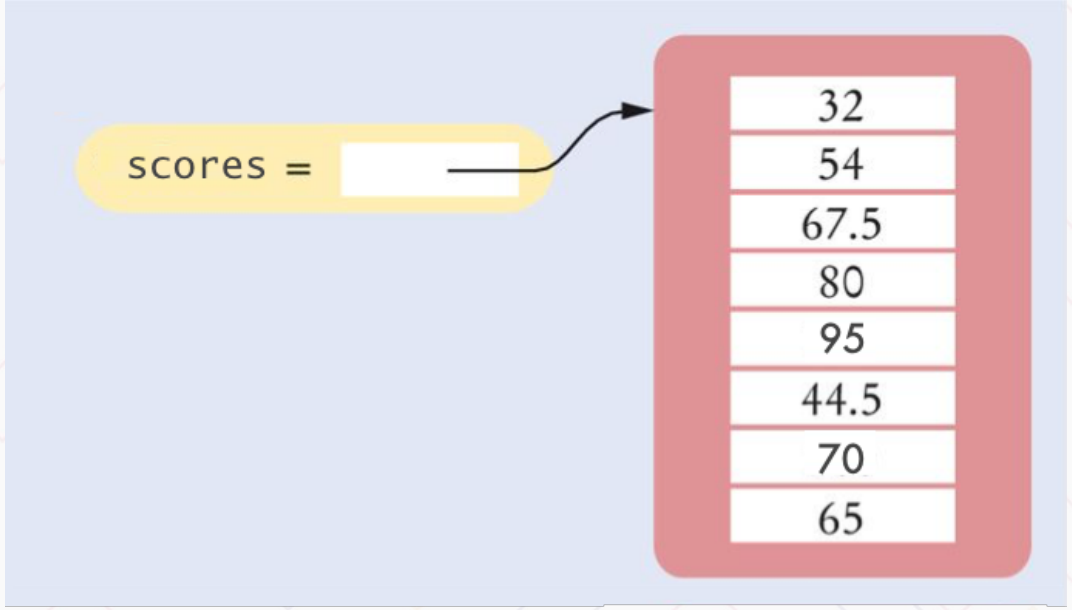


Example

Code Academy

```
scores= [32, 54, 67.5, 80, 95, 44.5, 70, 65]
```

- This statement creates a list named `scores` with initial values 32, 54, 67.5, 80, 95, 44.5, 70, and 65.
- The square brackets indicate that we are creating a list.
- The items are stored in the order they are provided.



Accessing List Elements

- A list is a sequence of *elements*, each of which has an integer position or *index*
- To access a list element, you specify which index you want to use. That is done with the subscript operator in the same way that you access individual characters in a string
- The first element in the list always has subscript 0.

Accessing list elements

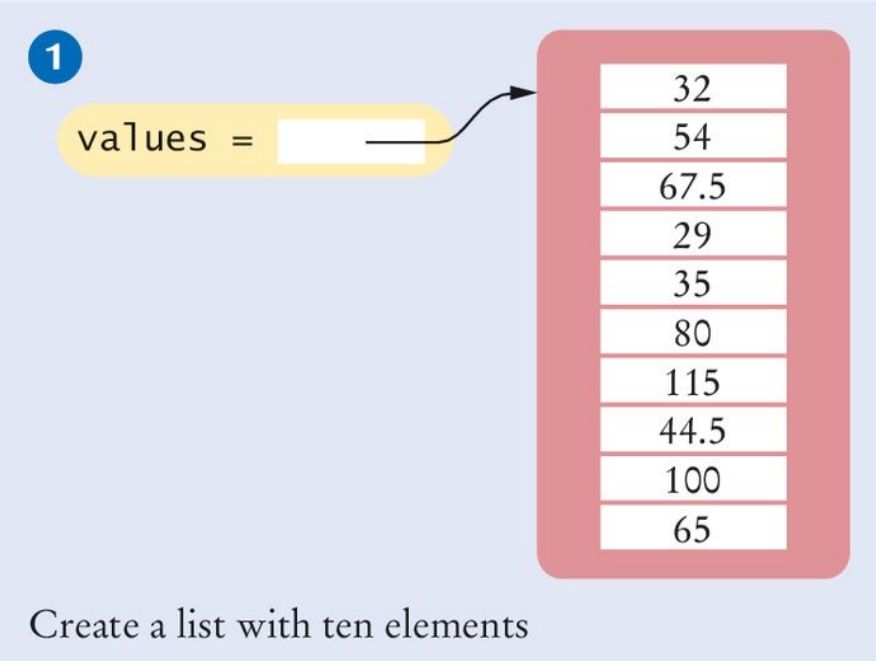
```
print(values[5])
```

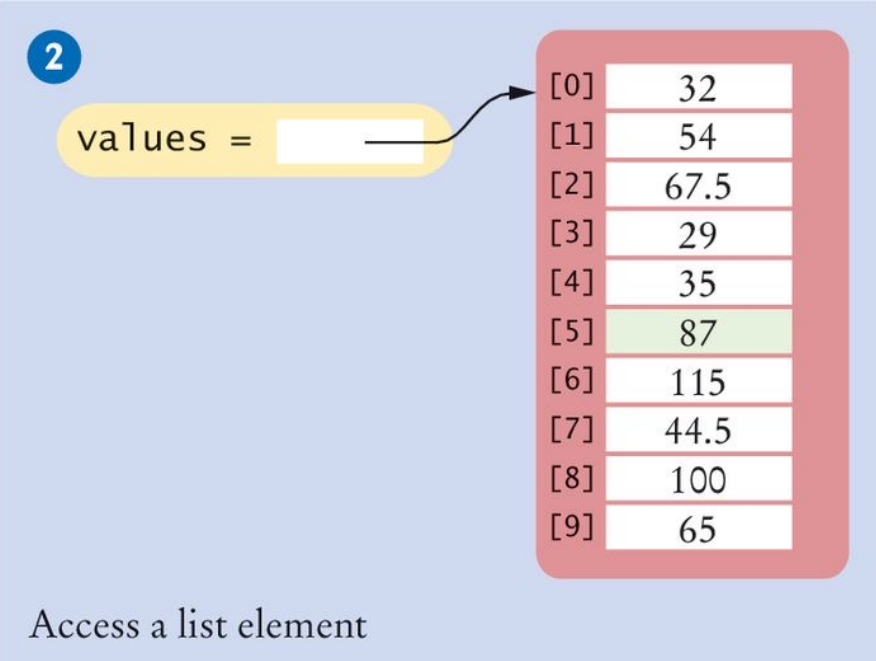
Replacing list elements

```
values[5] = 87
```

Code Academy

Creating Lists/Accessing Elements






```
# 1: Creating a list
values = [32, 54, 67.5, 29, 35, 80, 115, 44.5, 100, 65]

# 2: Accessing a list element
values[5] = 87
```

Exercises

- Define a list of integers containing the first five prime numbers.

```
primes = [2, 3, 5, 7, 11]
```

- Define a list containing two strings “Yes” and “No”

```
words = ["Yes", "No"]
```

Lists Vs. Strings

- Both lists and strings are **sequences**, and the [] operator is used to access an element in any sequence
- There are two differences between lists and strings:
  - Lists can hold values of any type, whereas strings are sequences of characters
  - Moreover:
    - strings are immutable— you cannot change the characters in the sequence
    - Lists are *mutable* – you can replace one list element with another, like this:

```
values[5] = 87
```

Now the element at index 5 is filled with 87.

Out of Range Errors

Code Academy

- Perhaps the most common error in using lists is accessing a nonexistent element

```
values = [2.3, 4.5, 7.2, 1.0, 12.2, 9.0, 15.2, 0.5]
values[8] = 5.4
# Error--values has 8 elements,
# and the index can range from 0 to 7
```

- If your program accesses a list through an out-of-range index, the program will generate an exception at run time

Determining List Length

- You can use the len() function to obtain the length of the list; that is, the number of elements:

```
numElements = len(values)
```

- Example:

```
scores= [32, 54, 67.5, 80, 95, 44.5, 70, 65]
print(len(scores))  # displays 8
```

Using The Square Brackets

- Note that there are two distinct uses of the square brackets.
  - When the square brackets immediately follow a variable name, they are treated as the subscript operator:

```
values[4]
```

  - When the square brackets follow an “=“ they create a list:

```
values = [4]
```

Code Academy

Traversing Lists

- There are two ways of visiting all elements of a list.
  - Loop over the index values.
    - Given the values list that contains 10 elements, we want to set a variable, say i, to 0, 1, 2, and so on, up to 9

```
# First version (list index used)
for i in range(10) :
    print(i, values[i])
```

```
# Better version (list index and length used)
for i in range(len(values)) :
    print(i, values[i])
```

  - Loop over the elements themselves.

```
# Third version: index values not needed (traverse list elements)
for element in values :
    print(element)
```

Example

- Write a for loop that iterate over the elements of `myList` for computing the sum of all elements in a list.

  `myList = [3, 5, 1, 7, 2, 8]`

  - Using list index

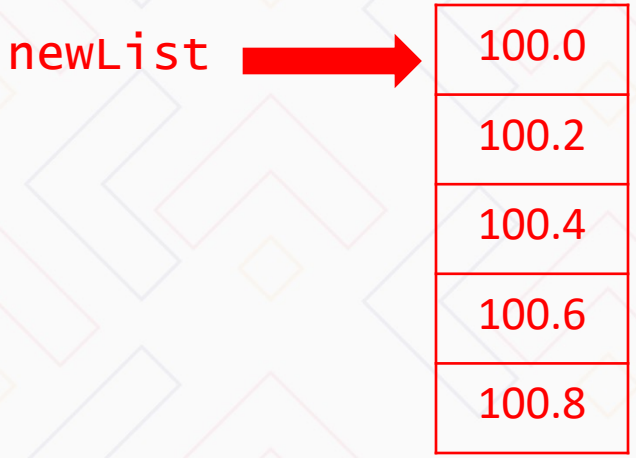
```
total = 0
for i in range(len(myList)) :
    total += myList[i]
print(total)
```

  - Without using list index

```
total = 0
for element in myList :
    total += element
print(total)
```

What values are assigned?

```
newList = [0, 0, 0, 0, 0]
for i in range(5):
        newList[i] = 100.0 + i * 0.2
```



- Given m = 3, what is the value of newList[m+1] and newList[m]+1 ?

newList[3+1] = newList[4] = 100.8

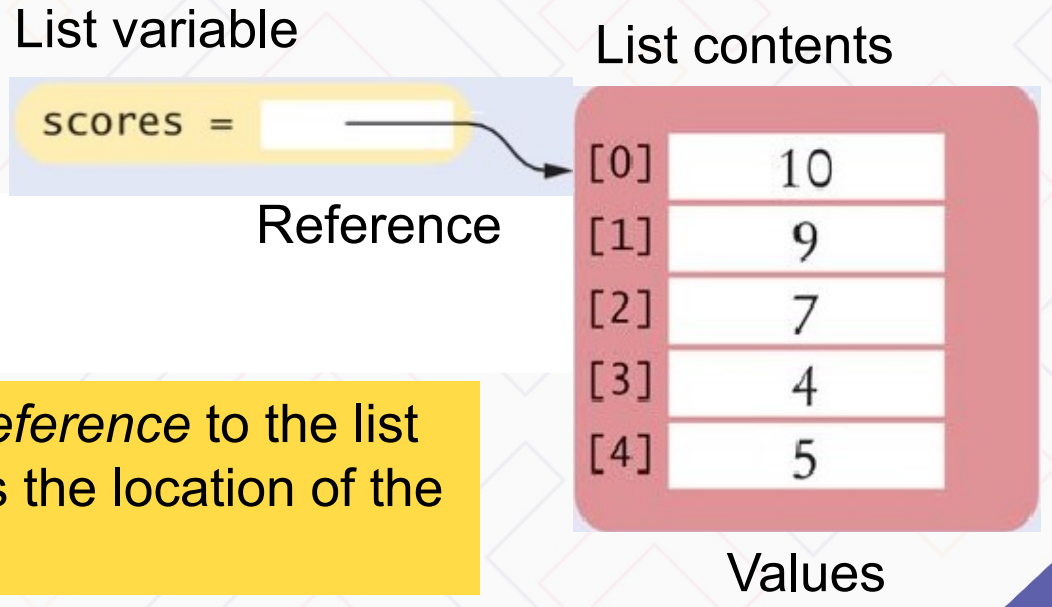
newList[3]+1 = 100.6 + 1 = 101.6

List References

Code Academy

- Make sure you see the difference between the:
  - List variable: The named 'alias' or pointer to the list
  - List contents: Memory where the values are stored

```
scores= [10, 9, 7, 4, 5]
```



A list variable contains a *reference* to the list contents. The *reference* is the location of the list contents (in memory).

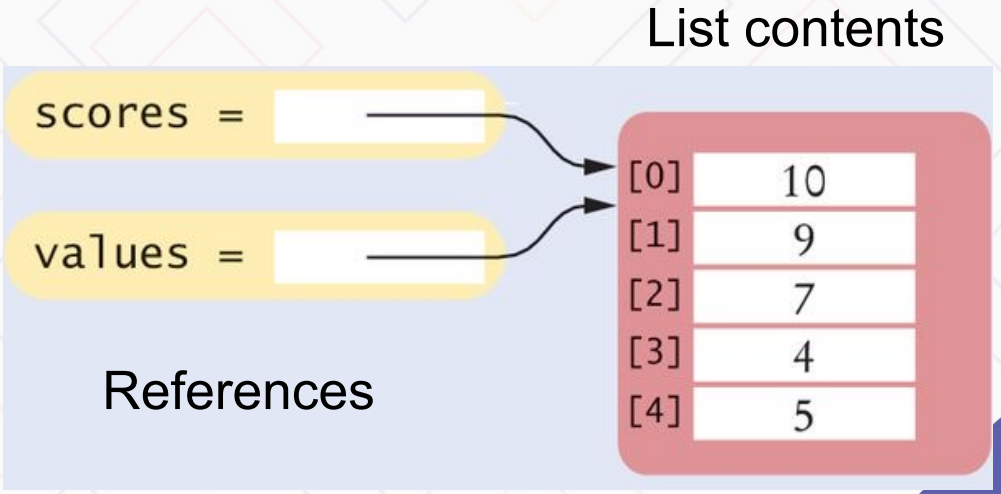
List Aliases

Code Academy

- When you copy a list variable into another, both variables refer to the same list
  - The second variable is an *alias* for the first because both variables reference the same list

```
scores = [10, 9, 7, 4, 5]
values = scores      # Copying list reference
```

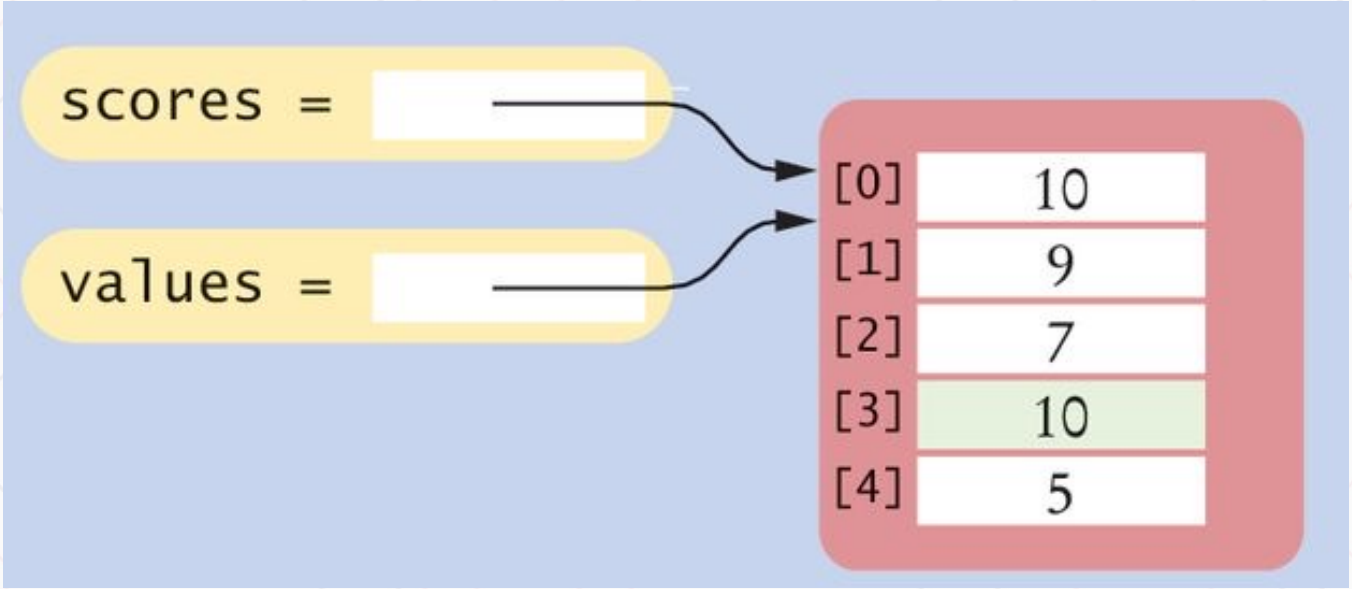
A list variable specifies the location of a list. Copying the reference yields a second reference to the same list.



Modifying Aliased Lists

- You can modify the list through either of the variables:

```
scores[3] = 10
print(values[3])   # Prints 10
```



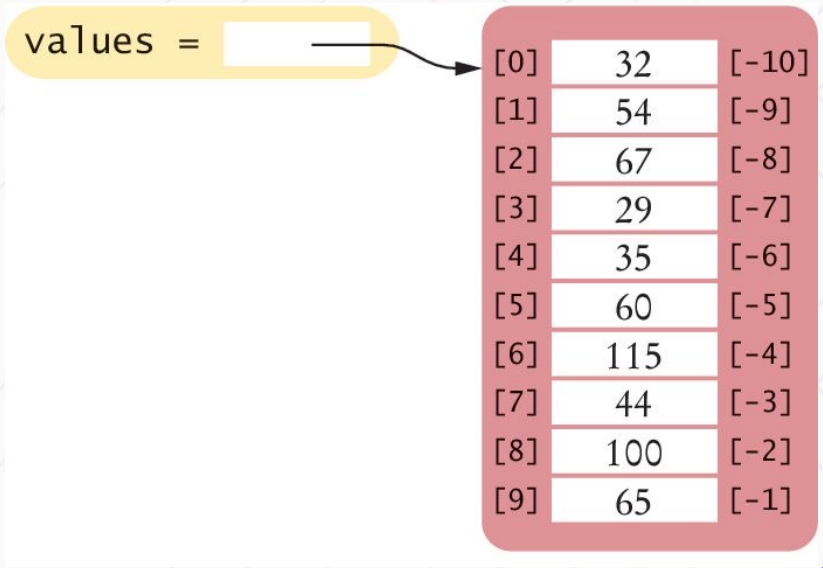
Reverse Subscripts

Code Academy

- Python, unlike other languages, uses negative subscripts to provide access to the list elements in reverse order.
  - For example, a subscript of –1 provides access to the last element in the list
  - Similarly, values[-2] is the second-to-last element.

Just because you can do this, does not mean you should…

```
last = values[-1]
print("The last element in the list is", last)
```



Code Academy

Practice More…

1. ***Solve exercises(R6.4, R6.7) on Page 370***
2. Write for loops that iterate over the elements of a list without the use of the range function for the following tasks.
    a. Printing all elements of a list in a single row, separated by spaces.
    b. Computing the product of all elements in a list.
    c. Counting how many elements in a list are negative.

Summary: Lists

- A list is a container that stores a sequence of values
- Each individual element in a list is accessed by an integer index i, using the notation `list[i]`
- A list index must be at least zero and less than the number of elements in the list
- An out-of-range error, which occurs if you supply an invalid list index, can cause your program to terminate
- You can iterate over the index values or the elements of a list
- A list reference specifies the location of a list. Copying the reference yields a second reference to the same list