Lecture 05: Introduction to Lists

Swakkhar Shatabda

B.Sc. in Data Science
Department of Computer Science and Engineering
United International University

January 17, 2024

Lists

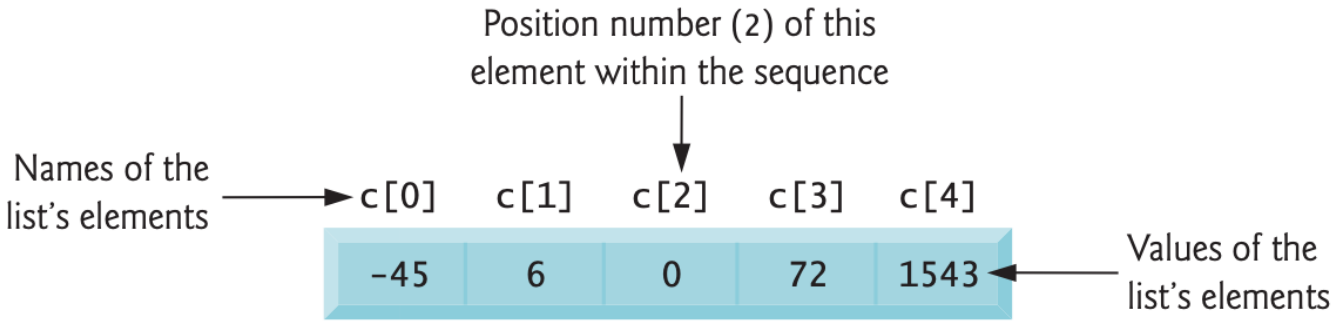
- A list is a collection of items.
- You can put anything you want into a list.
- In Python, square brackets ([]) indicate a list, and individual elements in the list are separated by commas.

```
bicycles = ['trek', 'cannondale', 'redline', 'specialized']
print(bicycles)
```

```
['trek', 'cannondale', 'redline', 'specialized']
```

Accessing Elements of a List

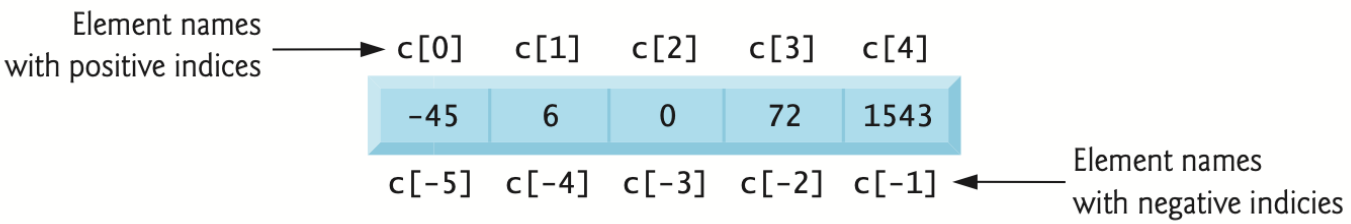
```
c = [-45, 6, 0, 72, 1543]
```




```
bicycles = ['trek', 'cannondale', 'redline', 'specialized']
print(bicycles[0])
```

Negative Indices

```
c = [-45, 6, 0, 72, 1543]
```



Modifying Lists

- The syntax for modifying an element is similar to the syntax for accessing an element in a list.
- To change an element, use the name of the list followed by the index of the element you want to change, and then provide the new value you want that item to have.

```
motorcycles = ['honda', 'yamaha', 'suzuki']
print(motorcycles)
motorcycles[0] = 'ducati'
print(motorcycles)
```

```
['honda', 'yamaha', 'suzuki']
['ducati', 'yamaha', 'suzuki']
```

Adding Elements - End

- The simplest way to add a new element to a list is to append the item to the list.
- When you append an item to a list, the new element is added to the end of the list.

```
motorcycles = ['honda', 'yamaha', 'suzuki']
print(motorcycles)
motorcycles.append('ducati')
print(motorcycles)
```

```
['honda', 'yamaha', 'suzuki']
['honda', 'yamaha', 'suzuki', 'ducati']
```

List may start empty!

- You can start with an empty list and then add items to the list using a series of append() calls.

```
motorcycles = []

motorcycles.append('honda')
motorcycles.append('yamaha')
motorcycles.append('suzuki')

print(motorcycles)
```

```
['honda', 'yamaha', 'suzuki']
```

Inserting Elements into a List!

- You can add a new element at any position in your list by using the insert() method.
- You do this by specifying the index of the new element and the value of the new item.

```
motorcycles = ['honda', 'yamaha', 'suzuki']

motorcycles.insert(0, 'ducati')
print(motorcycles)
```

```
['ducati', 'honda', 'yamaha', 'suzuki']
```

Removing Elements from a List!

- If you know the position of the item you want to remove from a list, you can use the del statement.

```
motorcycles = ['honda', 'yamaha', 'suzuki']
print(motorcycles)

del motorcycles[1]
print(motorcycles)
```

```
['honda', 'yamaha', 'suzuki']
['honda', 'suzuki']
```

Removing an Item Using pop()

- The pop() method removes the last item in a list, but it lets you work with that item after removing it.

```
motorcycles = ['honda', 'yamaha', 'suzuki']
print(motorcycles)

popped_motorcycle = motorcycles.pop()
print(motorcycles)
print(popped_motorcycle)
```

```
['honda', 'yamaha', 'suzuki']
['honda', 'yamaha']
suzuki
```

Popping Items from Any Position

- You can use pop() to remove an item from any position in a list by including the index of the item you want to remove in parentheses.

```
motorcycles = ['honda', 'yamaha', 'suzuki']
print(motorcycles)

popedItem = motorcycles.pop(1)
print(motorcycles)
print(popedItem)
```

```
['honda', 'yamaha', 'suzuki']
['honda', 'suzuki']
yamaha
```

Removing an Item by Value

- Sometimes you won't know the position of the value you want to remove from a list.
- If you only know the value of the item you want to remove, you can use the remove() method.
- The remove() method deletes only the first occurrence of the value you specify.

```
motorcycles = ['honda', 'yamaha', 'suzuki', 'ducati']
print(motorcycles)

motorcycles.remove('ducati')
print(motorcycles)
```

```
['honda', 'yamaha', 'suzuki', 'ducati']
['honda', 'yamaha', 'suzuki']
```

Indexing Errors

- Using an out-of-range list, tuple or string index causes an IndexError.

```
c = [-45, 6, 0, 72, 1543]
print(c[5])
```

```
Traceback (most recent call last):
  File "/Users/swakkhar/hello.py", line 2, in <module>
    print(c[5])
          ~^^^
IndexError: list index out of range
```

- For last time use negative index to reduce chances of error.

Iterating

- The for statement allows you to repeat an action or several actions.
- The for statement performs its action(s) for each item in a sequence of items.

```
c = [-45, 6, 0, 72, 1543]
for x in c:
    print(x,end=" ")
```

```
-45 6 0 72 1543
```