

# Lists

February 13, 2023



## 1 Introduction to Python

### 1.1 Lists & Dictionaries & Tuples

- Slides by [Ahmed Moustafa](#)
- Content modified from [Pierian Data](#)

## 2 Lists

- Lists can be thought of the most general version of a sequence in Python.
- Unlike strings, they are **mutable**, i.e. elements inside a list can be changed.
- Lists are constructed with brackets [ ] and commas , separating every element in the list.

## 3 Creating a List

```
[2]: weights = [65.0, 70.5, 72.3, 68.0, 77.2] # list of numbers
weights
```

```
[2]: [65.0, 70.5, 72.3, 68.0, 77.2]
```

```
[3]: cities = ["London", "Paris", "New York", "Tokyo", "Berlin"] # list of strings
cities
```

```
[3]: ['London', 'Paris', 'New York', 'Tokyo', 'Berlin']
```

```
[4]: types = [1, 2.5, "hello", "world", 42, "python"] # list of different data types
types
```

```
[4]: [1, 2.5, 'hello', 'world', 42, 'python']
```

## 4 List of Lists

We can also create a list of lists. For example, combining the two list we just created, `cities` and `weights` into a new list `my_list`:

```
[5]: my_list = [cities, weights]
      my_list
```

```
[5]: [['London', 'Paris', 'New York', 'Tokyo', 'Berlin'],
      [65.0, 70.5, 72.3, 68.0, 77.2]]
```

```
[6]: len(my_list)
```

```
[6]: 2
```

## 5 Indexing and Slicing 1/3

Indexing and slicing work just like in strings:

```
[7]: cities[0]
```

```
[7]: 'London'
```

```
[8]: cities[1:]
```

```
[8]: ['Paris', 'New York', 'Tokyo', 'Berlin']
```

```
[9]: cities[::-1]
```

```
[9]: ['Berlin', 'Tokyo', 'New York', 'Paris', 'London']
```

```
[10]: cities + ["Cairo", "Alexandria"]
```

```
[10]: ['London', 'Paris', 'New York', 'Tokyo', 'Berlin', 'Cairo', 'Alexandria']
```

## 6 Indexing and Slicing 2/3

```
[11]: cities
```

```
[11]: ['London', 'Paris', 'New York', 'Tokyo', 'Berlin']
```

```
[12]: cities += ["Cairo", "Alexandria"]
      cities
```

```
[12]: ['London', 'Paris', 'New York', 'Tokyo', 'Berlin', 'Cairo', 'Alexandria']
```

```
[13]: cities * 2
```

```
[13]: ['London',  
      'Paris',  
      'New York',  
      'Tokyo',  
      'Berlin',  
      'Cairo',  
      'Alexandria',  
      'London',  
      'Paris',  
      'New York',  
      'Tokyo',  
      'Berlin',  
      'Cairo',  
      'Alexandria']
```

## 7 Indexing and Slicing 3/3

```
[14]: my_list
```

```
[14]: [['London', 'Paris', 'New York', 'Tokyo', 'Berlin', 'Cairo', 'Alexandria'],  
      [65.0, 70.5, 72.3, 68.0, 77.2]]
```

```
[15]: len(my_list)
```

```
[15]: 2
```

```
[16]: my_list[0]
```

```
[16]: ['London', 'Paris', 'New York', 'Tokyo', 'Berlin', 'Cairo', 'Alexandria']
```

```
[17]: my_list[1][2]
```

```
[17]: 72.3
```

## 8 List Methods: append

The `append()` method adds an item to the end of the list

```
[18]: print(cities)  
      len(cities)
```

```
['London', 'Paris', 'New York', 'Tokyo', 'Berlin', 'Cairo', 'Alexandria']
```

```
[18]: 7
```

```
[19]: cities.append("Aswan")  
      print(cities)
```

```
len(cities)
```

```
['London', 'Paris', 'New York', 'Tokyo', 'Berlin', 'Cairo', 'Alexandria',  
'Aswan']
```

```
[19]: 8
```

## 9 List Methods: pop

The `pop()` method removes the item at the given index from the list and returns the removed item

```
[20]: cities.pop() # pop (remove) the last element
```

```
[20]: 'Aswan'
```

```
[21]: print(cities)  
len(cities)
```

```
['London', 'Paris', 'New York', 'Tokyo', 'Berlin', 'Cairo', 'Alexandria']
```

```
[21]: 7
```

```
[22]: cities.pop(1) # pop (remove) at the given index
```

```
[22]: 'Paris'
```

```
[23]: print(cities)  
len(cities)
```

```
['London', 'New York', 'Tokyo', 'Berlin', 'Cairo', 'Alexandria']
```

```
[23]: 6
```

## 10 List Methods: reverse

The `reverse()` method reverses the elements of the list

```
[27]: print ("Before: ", cities)
```

```
Before:  ['Alexandria', 'Cairo', 'Berlin', 'Tokyo', 'New York', 'London']
```

```
[28]: cities.reverse()
```

```
[29]: print ("After: ", cities)
```

```
After:  ['London', 'New York', 'Tokyo', 'Berlin', 'Cairo', 'Alexandria']
```

```
[ ]:
```