find_element() vs find_elements()

- ***find_element()*** -> returns a single web element *Object*
- ***find_elements()*** -> returns multiple elements as a *List Collection*

Both methods take Locator as an argument, i.e. a tuple of a locator type and value.

Sample locator for a List Collection containing multiple elements:
Locate all links in the footer section:
//div[@class='footer']//a
↑
Jump directly to the link section/tag

Only with a single element, I get the *.send_keys()* method or *.text* property.

*element**.text***
- Capture/extract text (value) of any element/object.
- Every link, button, etc., has some name, which can be retrieved with *.text*.

If the element is not present on the web page,the **find_element()** method returns NoSuchElementException. Doesn't matter if the method points to single or multiple locators. If the locator does not find the element, then it throws NoSuchElementException.
When I pass an incorrect locator, it isn't able to find element(s) on the page. Unable to locate an element address, when the locator is not matching with any elements on the page.

When the **find_elements()** method cannot locate an element, it returns '0', not an exception as in **find_element()**.

- **find_elements()** method always returns a List Collection
- **find_element()** method always returns a WebElement Object.

I can use the *.send_keys()* method or *.text* property on a single web element object, but not on a List. Once I extract one element from the List Collection, I can use *.send_keys()* or *.text -> elements[0].text*.

Even if **find_elements()** returns one element, it will be a member of a List Collection.

I can get the size/count of the List with ***len(elements)***.

Since *.text* can only extract value from a single element, I need to write a loop to iterate through every item in the List.
Print text of multiple elements using for-loop:

```
for e in elements:
    print(e.text)
```