Week 2

Intro to Python

cadtx.pw/intropyfa18-2

Topics Covered Today

- Review
- Lists

Review from last week

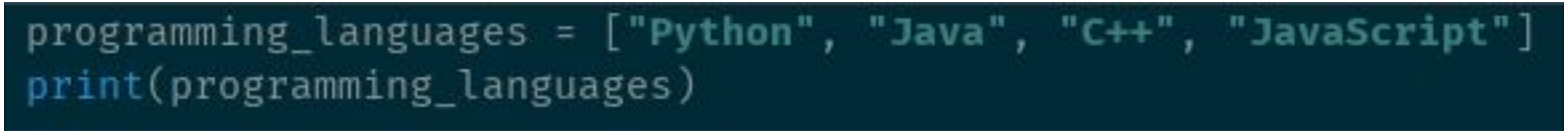
- We can print out values to the console using print()
- You can assign values to variables using ‘=’
- Variables have data types associated with them
  - Strings are a series of alphanumeric characters, denoted by “” or ‘’ in Python
  - Integers are whole numbers
  - Floats are any numbers with decimal points
- You can add (+), subtract(-), multiply(*), and divide(/) integers and floats
- ** represents exponents
- You can use + to “add” strings - this is called concatenation

What are lists?

- Ordered collections of items
- Can be anything
    - All the letters of the alphabet
    - Your favorite foods
    - Digits 0-9
- Any data type, don’t have to be related

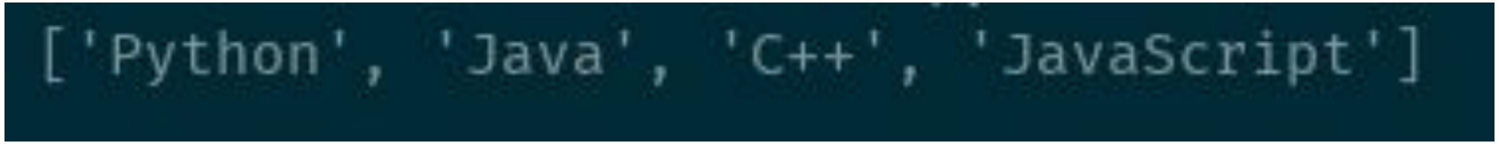
Creating Lists

- Square brackets , [] indicate a list in Python



```
programming_languages = ["Python", "Java", "C++", "JavaScript"]
print(programming_languages)
```

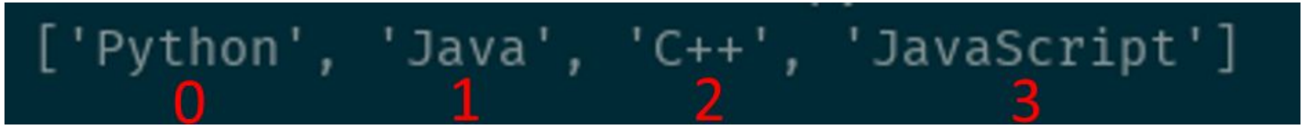
- Output is Python’s interpretation of a list, including brackets:



```
['Python', 'Java', 'C++', 'JavaScript']
```

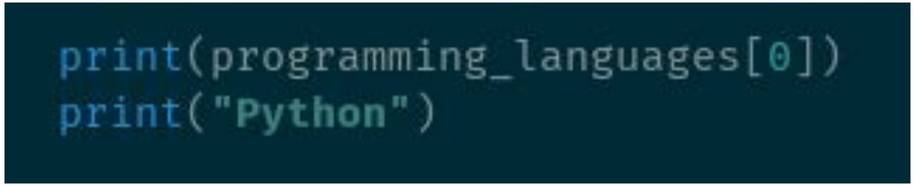
Accessing Elements in a List

- As lists are ordered collections, you can access any element by giving Python its position, or *index*
- Indices in Python start at 0, not 1



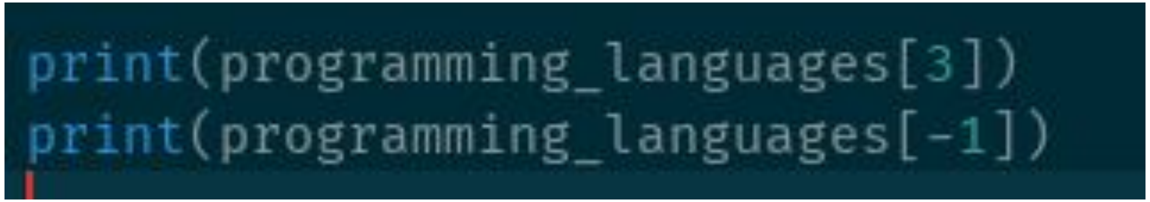
```
['Python', 'Java', 'C++', 'JavaScript']
     0        1      2         3
```

- Access elements of a list using the following syntax:
  - List[index]



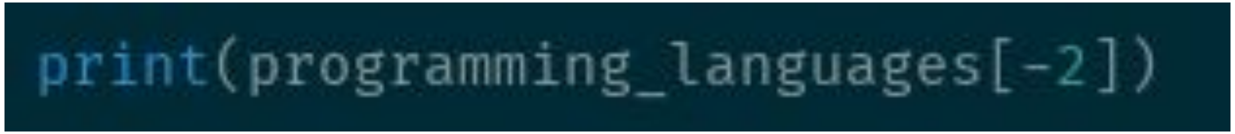
```
print(programming_languages[0])
print("Python")
```

- Accessing the last element of a list



```
print(programming_languages[3])
print(programming_languages[-1])
```

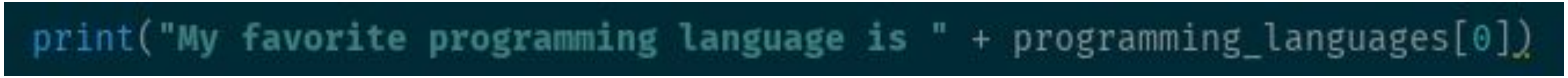
- Both of these accomplish the same thing, but the second is useful if you don’t know the length of the list
- Similarly, you can access the second to last, third, last, etc. using this syntax



```
print(programming_languages[-2])
```

Using items from a list:

- You can use elements of a list much like you would variables

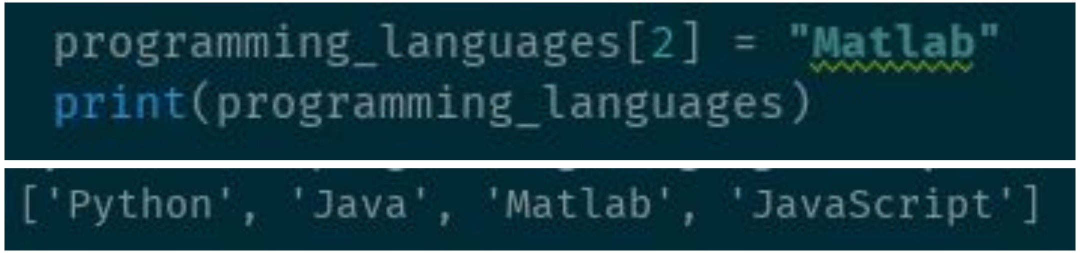


```
print("My favorite programming language is " + programming_languages[0])
```

- Try it yourself: store the names of a few of your friends in a list called *names*, then print each person's name with a personalized message.

Modifying Elements in a List

- Similar syntax to accessing items from a list

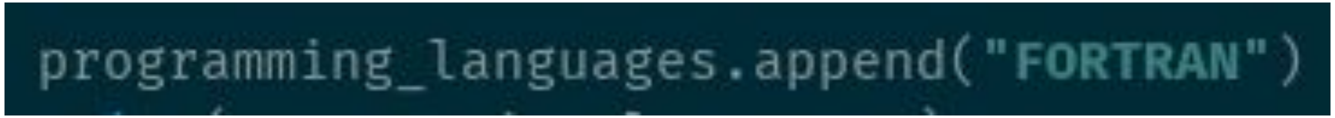


```
programming_languages[2] = "Matlab"
print(programming_languages)
```

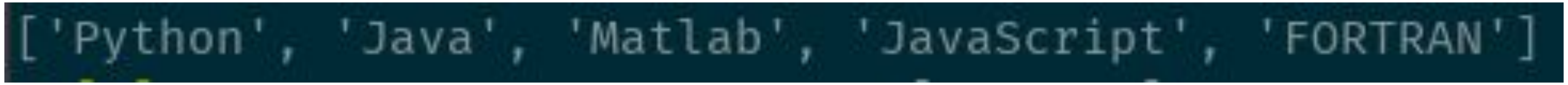
```
['Python', 'Java', 'Matlab', 'JavaScript']
```

Adding Elements to a List

- The easiest way is to add items to the end of the list

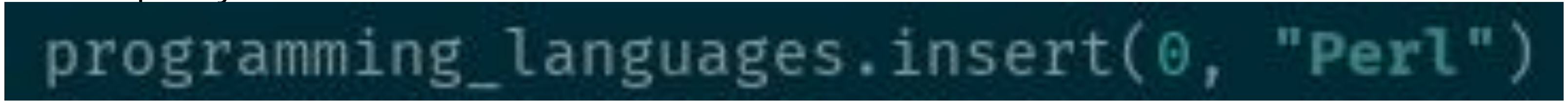


```
programming_languages.append("FORTRAN")
```



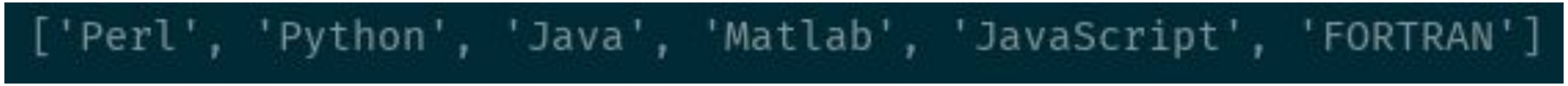
```
['Python', 'Java', 'Matlab', 'JavaScript', 'FORTRAN']
```

- Inserting elements into a list
    - Specify the index of the new element and the value of the new item



```
programming_languages.insert(0, "Perl")
```

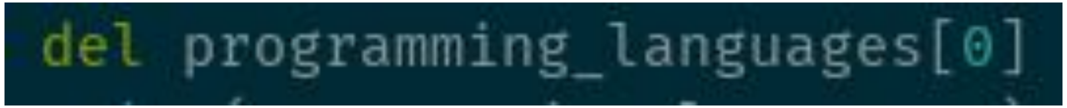
    - Opens up a space at index 0 and shifts every other value one position to the right



```
['Perl', 'Python', 'Java', 'Matlab', 'JavaScript', 'FORTRAN']
```

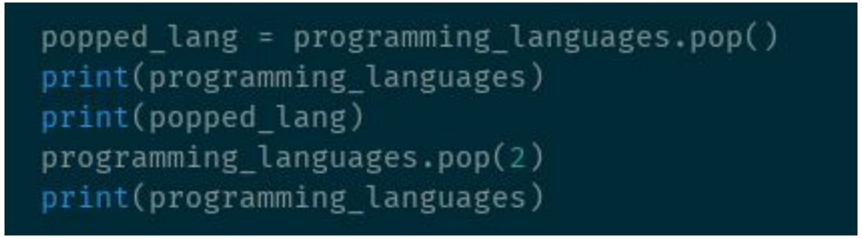
Removing Elements from a List

- If you know the position of the item you want to remove, you can use *del*

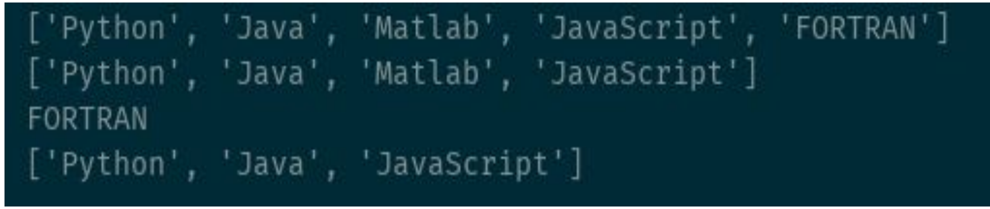


```
del programming_languages[0]
```

- If you want to use the value of an item after you remove it, you can use pop()
  - You can also use pop to remove an item at any position by specifying the index.

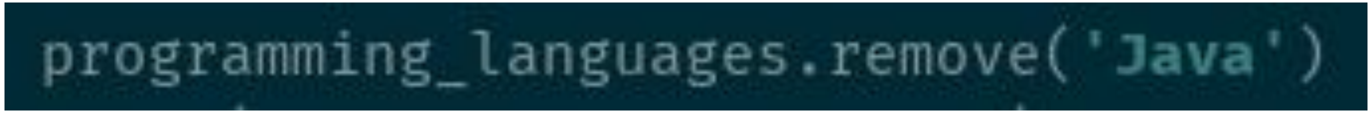


```
popped_lang = programming_languages.pop()
print(programming_languages)
print(popped_lang)
programming_languages.pop(2)
print(programming_languages)
```

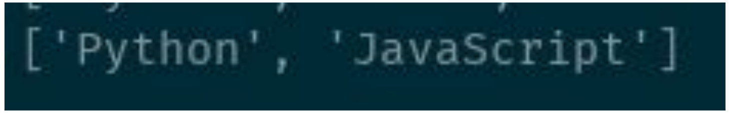


```
['Python', 'Java', 'Matlab', 'JavaScript', 'FORTRAN']
['Python', 'Java', 'Matlab', 'JavaScript']
FORTRAN
['Python', 'Java', 'JavaScript']
```

- You can also remove items by value, using *remove()*



```
programming_languages.remove('Java')
```

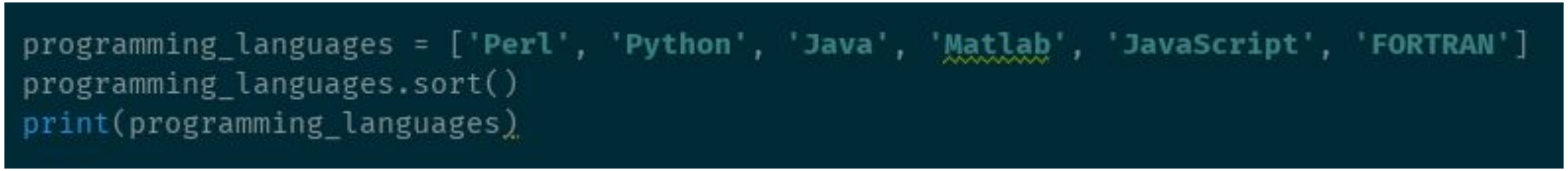


```
['Python', 'JavaScript']
```

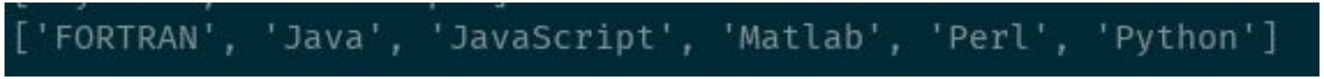
- Note that this only removes the first occurence of the value you specify. If you want to remove all occurences, you’ll have to use a loop - we’ll go over those later.

Organizing lists

- You can sort a list permanently using the sort() method.

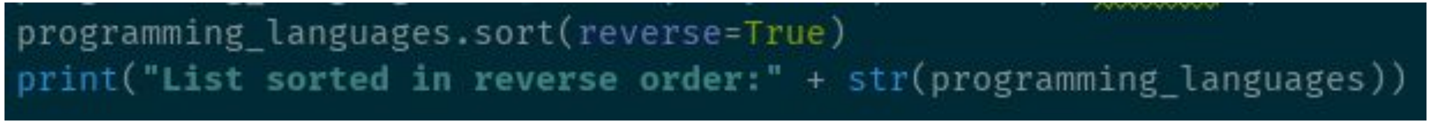


```
programming_languages = ['Perl', 'Python', 'Java', 'Matlab', 'JavaScript', 'FORTRAN']
programming_languages.sort()
print(programming_languages)
```

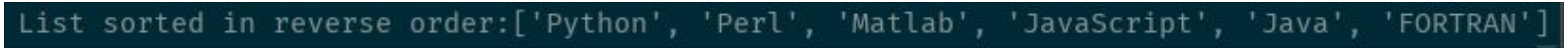


```
['FORTRAN', 'Java', 'JavaScript', 'Matlab', 'Perl', 'Python']
```

- You can sort a list in reverse order by giving the sort() method arguments

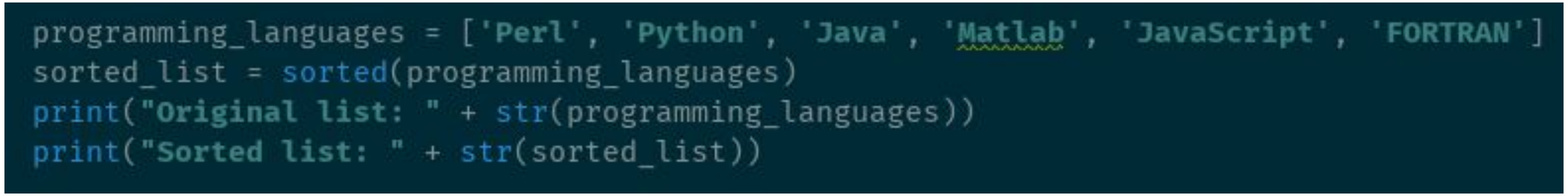


```
programming_languages.sort(reverse=True)
print("List sorted in reverse order:" + str(programming_languages))
```

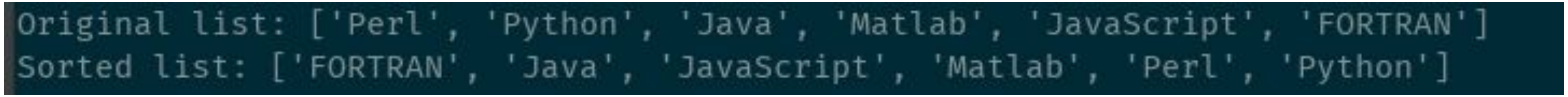


```
List sorted in reverse order:['Python', 'Perl', 'Matlab', 'JavaScript', 'Java', 'FORTRAN']
```

- If you want to keep a copy of the old list, use sorted()

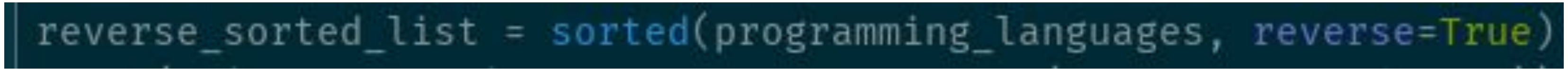


```python
programming_languages = ['Perl', 'Python', 'Java', 'Matlab', 'JavaScript', 'FORTRAN']
sorted_list = sorted(programming_languages)
print("Original list: " + str(programming_languages))
print("Sorted list: " + str(sorted_list))
```



```
Original list: ['Perl', 'Python', 'Java', 'Matlab', 'JavaScript', 'FORTRAN']
Sorted list: ['FORTRAN', 'Java', 'JavaScript', 'Matlab', 'Perl', 'Python']
```

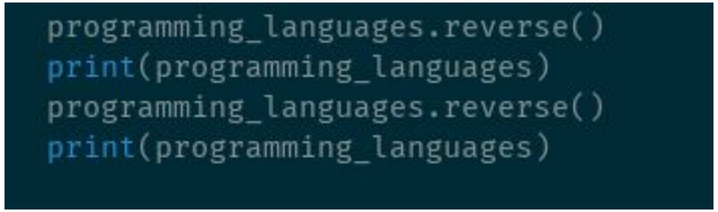
- sorted() will also take the arguments reverse=True



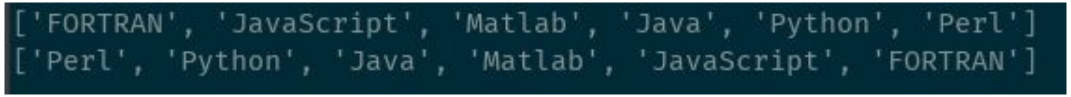
```python
reverse_sorted_list = sorted(programming_languages, reverse=True)
```

Other list functions

- You can reverse a list using reverse(). This modifies the list - to get to the original, just call the function again.

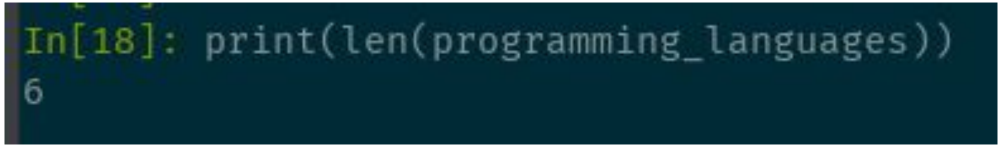


```
programming_languages.reverse()
print(programming_languages)
programming_languages.reverse()
print(programming_languages)
```

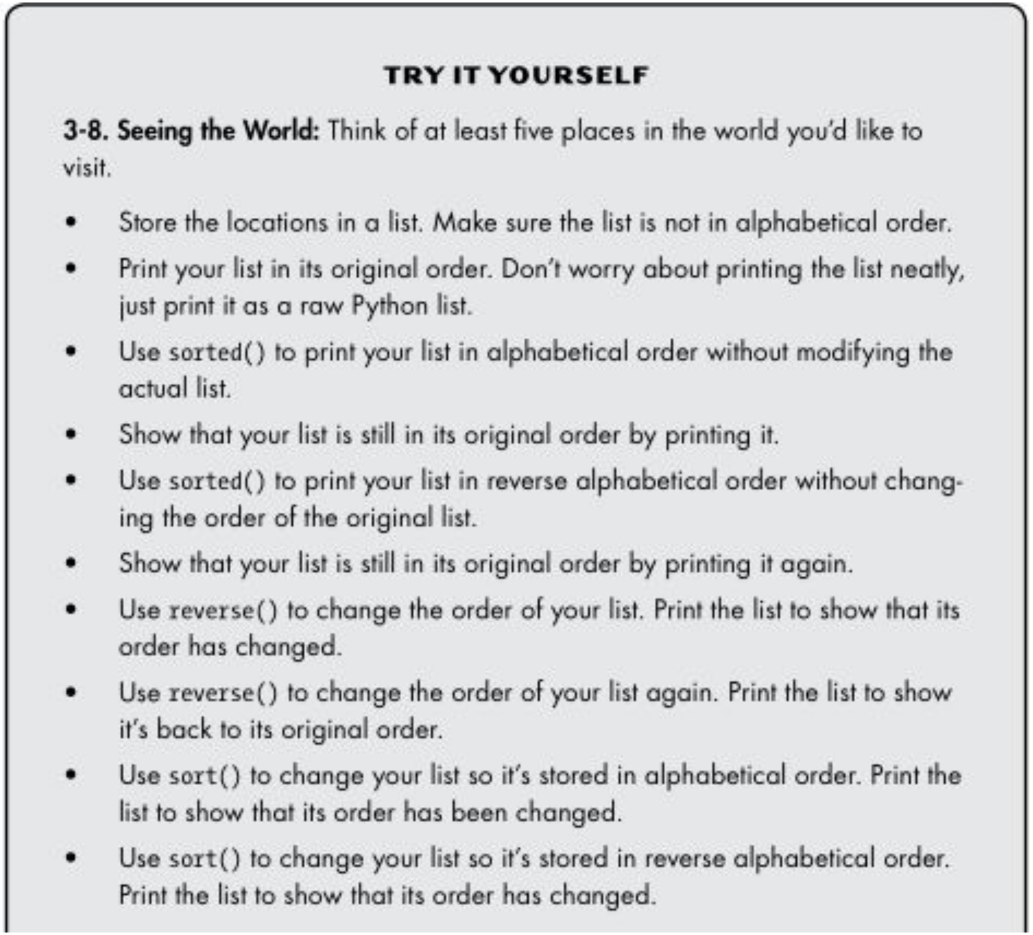


```
['FORTRAN', 'JavaScript', 'Matlab', 'Java', 'Python', 'Perl']
['Perl', 'Python', 'Java', 'Matlab', 'JavaScript', 'FORTRAN']
```

- Use len() to find the the length of a list. Python will count the length starting at one.



```
In[18]: print(len(programming_languages))
6
```



TRY IT YOURSELF

3-8. Seeing the World: Think of at least five places in the world you'd like to visit.

- Store the locations in a list. Make sure the list is not in alphabetical order.
- Print your list in its original order. Don't worry about printing the list neatly, just print it as a raw Python list.
- Use `sorted()` to print your list in alphabetical order without modifying the actual list.
- Show that your list is still in its original order by printing it.
- Use `sorted()` to print your list in reverse alphabetical order without changing the order of the original list.
- Show that your list is still in its original order by printing it again.
- Use `reverse()` to change the order of your list. Print the list to show that its order has changed.
- Use `reverse()` to change the order of your list again. Print the list to show it's back to its original order.
- Use `sort()` to change your list so it's stored in alphabetical order. Print the list to show that its order has been changed.
- Use `sort()` to change your list so it's stored in reverse alphabetical order. Print the list to show that its order has changed.