Section 6

Lists

Mohammed Asir Shahid

2021-08-02

Contents

1 The List Data type

A list is a data type that contains multiple ordered items. Lists begin and end with square brackets and are separated by commas. They can be assigned to variables just like any other value.

```
["cat", "bat", "rat", "elephant"]

spam=["cat", "bat", "rat", "elephant"]

print(spam)
```

```
['cat', 'bat', 'rat', 'elephant']
```

We can use indices to access items within the list. These indices also begin and end with square brackets.

```
["cat", "bat", "rat", "elephant"]
```

```
spam=["cat", "bat", "rat", "elephant"]

print(spam)

print(spam[0])
print(spam[1])
print(spam[2])
print(spam[3])
```

```
['cat', 'bat', 'rat', 'elephant']
cat
bat
rat
elephant
```

The items inside of a list can be of any data type, including other lists. In those cases, we can use 2 indices in order to find the items in the list inside of the list. We can also use negative values for the index in order to start counting from the end.

```
spammed=["cat", "bat", "rat", "elephant"]

spam=[["cat", "bat"], [10,20,30,40,50]]

print(spam)

print(spam[0][0])
print(spam[1][0])
print(spam[1][4])
print(spammed[-1])
print(spammed[-2])
```

```
[['cat', 'bat'], [10, 20, 30, 40, 50]]
cat
10
50
elephant
```

```
rat
```

We also have slices which can give us several values from inside of a list. This works similar to the range function. A slice of 1 3 starts at index 1 and goes up to, but does not include the value at 3.

```
spammed=["cat", "bat", "rat", "elephant"]

spam=[["cat", "bat"], [10,20,30,40,50]]

print(spam)

print(spam[0][0])
print(spam[1][0])
print(spam[1][4])
print(spammed[0:2])
```

```
[['cat', 'bat'], [10, 20, 30, 40, 50]]
cat
10
50
['cat', 'bat']
```

We can also change the values of a list.

```
spammed=["cat", "bat", "rat", "elephant"]

spam=[["cat", "bat"], [10,20,30,40,50]]

print(spammed)

spammed[0]="Hello"

print(spammed)

spammed[1:3]=["Good", "Bye"]

print(spammed)
```

```
['cat', 'bat', 'rat', 'elephant']
['Hello', 'bat', 'rat', 'elephant']
['Hello', 'Good', 'Bye', 'elephant']
```

We have some shortcuts when it comes to lists. Leaving out the first index is the same as 0, or the beginning of the list. Leaving out the second index is the same as using the length of the list which will slice to the end of the list.

```
spammed=["cat", "bat", "rat", "elephant"]

spam=[["cat", "bat"], [10,20,30,40,50]]

print(spammed[:2])
print(spammed[2:])
```

```
['cat', 'bat']
['rat', 'elephant']
```

Del statements can delete values from the list.

```
spammed=["cat", "bat", "rat", "elephant"]

spam=[["cat", "bat"], [10,20,30,40,50]]

print(spammed)
del spammed[2]

print(spammed)
```

```
['cat', 'bat', 'rat', 'elephant']
['cat', 'bat', 'elephant']
```

We can get the number of items in a list using the len function. We can also do list concatenation similar to string concatenation. We can also do list replication.

```
spammed=["cat", "bat", "rat", "elephant"]

spam=[["cat", "bat"], [10,20,30,40,50]]

print(len(spammed))

print(spammed*3)

print(spammed+spam)
```

```
4
['cat', 'bat', 'rat', 'elephant', 'cat', 'bat', 'rat', 'elephant', 'cat', 'bat', 'rat'
['cat', 'bat', 'rat', 'elephant', ['cat', 'bat'], [10, 20, 30, 40, 50]]
```

There is also a list function that converts our values into a list, similar to the int or str functions.

```
spammed=["cat", "bat", "rat", "elephant"]

spam=[["cat", "bat"], [10,20,30,40,50]]

print(list("Hello"))
```

```
['H', 'e', 'l', 'l', 'o']
```

We can use the in or not in operators with lists.

```
spammed=["cat", "bat", "rat", "elephant"]

spam=[["cat", "bat"], [10,20,30,40,50]]

print("cat" in spammed)
print("dog" in spammed)
print("dog" not in spammed)
```

```
True
False
True
```

2 For Loops with Lists, Multiple Assignment, and Augmented Operators

For Loops can be used to execute a block of code a certain number of times.

```
for i in range(4):
    print(i)
```

```
0
1
2
3
```

Python interprets the range object similarly to a list of integers. For example, the above code can also be written as follows:

```
for i in [0,1,2,3]:
    print(i)
```

```
0
1
2
3
```

We can also use the list function in order to convert the range object into a list.

```
print(list(range(4)))
```

```
[0, 1, 2, 3]
```

So if you want to make a long list of integers that follows a pattern, using the range function can be better than writing it out.

```
print(list(range(0,100,2)))
```

```
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44
```

One way to use range is to use a for loop over the range of the length of some list.

```
supplies=["pens","staplers","flame-throwers","binders"]

for item in range(len(supplies)):
    print("Index {} in supplies is {}".format(item,supplies[item]))
```

```
Index 0 in supplies is pens
Index 1 in supplies is staplers
Index 2 in supplies is flame-throwers
Index 3 in supplies is binders
```

The nice thing about the above for loop is that it'll work regardless of the length of the supplies list.

```
cat=["fat","orange","loud"]

size=cat[0]

color=cat[1]

disposition=cat[2]

# Instead of 3 lines of code for something like this, we can assign mutiple variables a

size, color, disposition = cat
print(size)
print(color)
print(disposition)

# We could also have mutiple variables on the right side.

size, color, disposition = "skinny","black","quiet"
print(size)
print(color)
print(disposition)
```

```
fat
orange
loud
skinny
black
quiet
```

We can use the multiple assignment for swapping variables.

```
a="AAA"
b="BBB"

a,b=b,a

print(a,b)
```

```
BBB AAA
```

Python also lets us use Augmented Assignment Operators.

```
spam=42
spam=spam+1
spam+=1
```

Instead of doing “spam=spam+1”, we can do “spam+=1”. This also applies to other operators such as plus, minus, multiplication, division, and modulus.