Lists and Loops

Danny Zou

Back to Java

```
int[] joe = {1, 2, 41}
```

This is what you do in Java.

First, you define joe as an array of type int, then you set its values.

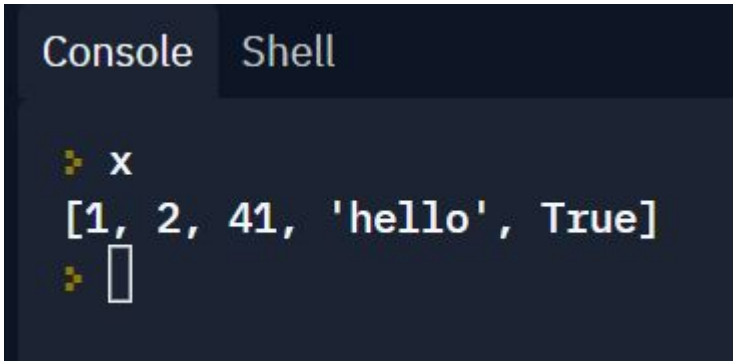
But what about Python?

Arrays Don’t Exist in Python!

Just like Pythonic variables don’t have a type, Pythonic lists don’t either! They’re like a box that can hold anything!

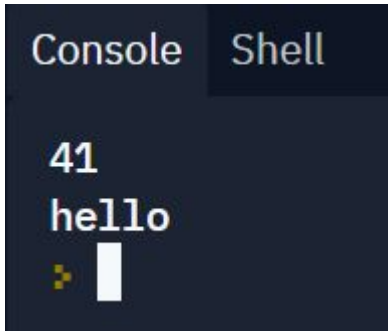
For example:

```
1    x = [1, 2, 41, "hello", True]
```



Everything Else is the Same

```
1   x = [1, 2, 41, "hello", True]
2
3   print(x[2])
4   print(x[3])
```



As you can see, the index of values in a list still starts at 0!

List Methods

Method	Description
append()	Adds an element at the end of the list
clear()	Removes all the elements from the list
copy()	Returns a copy of the list
count()	Returns the number of elements with the specified value
extend()	Add the elements of a list (or any iterable), to the end of the current list
index()	Returns the index of the first element with the specified value
insert()	Adds an element at the specified position
pop()	Removes the element at the specified position
remove()	Removes the item with the specified value
reverse()	Reverses the order of the list
sort()	Sorts the list

From W3Schools

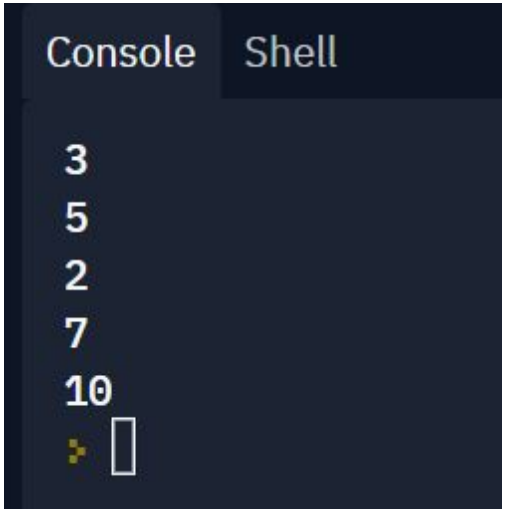
Examples (You can skip this)

```
x = [1, 2, 41, "hello", True]
#I've created a list named x
y = ["this is a second list", 31, False]
#I create another list named y
x.append(5)
print(x)
#I add 5 to the end of x.
y.clear
print(y)
#I remove all elements from y
y = x.copy()
print(y)
#I copy all the elements from x into y
x.extend(y)
print(x)
#I copy all the elements in y and attach them to x
print(x.count(2))
#I print out the number of times the element 2 occurs
print(x.index(41))
#prints out the index of the first element with the value 41
x.insert(3, "tada!")
print(x)
#I insert a value at the 3rd index
x.pop(3)
print(x)
#I remove the value
x.remove(41)
print(x)
#I remove the first element with the value 41
x.reverse()
print(x)
#I flip the list around!
```

```
[1, 2, 41, 'hello', True, 5]
['this is a second list', 31, False]
[1, 2, 41, 'hello', True, 5]
[1, 2, 41, 'hello', True, 5, 1, 2, 41, 'hello', True, 5]
2
2
[1, 2, 41, 'tada!', 'hello', True, 5, 1, 2, 41, 'hello', True, 5]
[1, 2, 41, 'hello', True, 5, 1, 2, 41, 'hello', True, 5]
[1, 2, 'hello', True, 5, 1, 2, 41, 'hello', True, 5]
[5, True, 'hello', 41, 2, 1, 5, True, 'hello', 2, 1]
```

Why is this important?

```
1   joe = [2, 4, 1, 6, 9]
2
3   for i in joe:
4     print(i + 1)
```



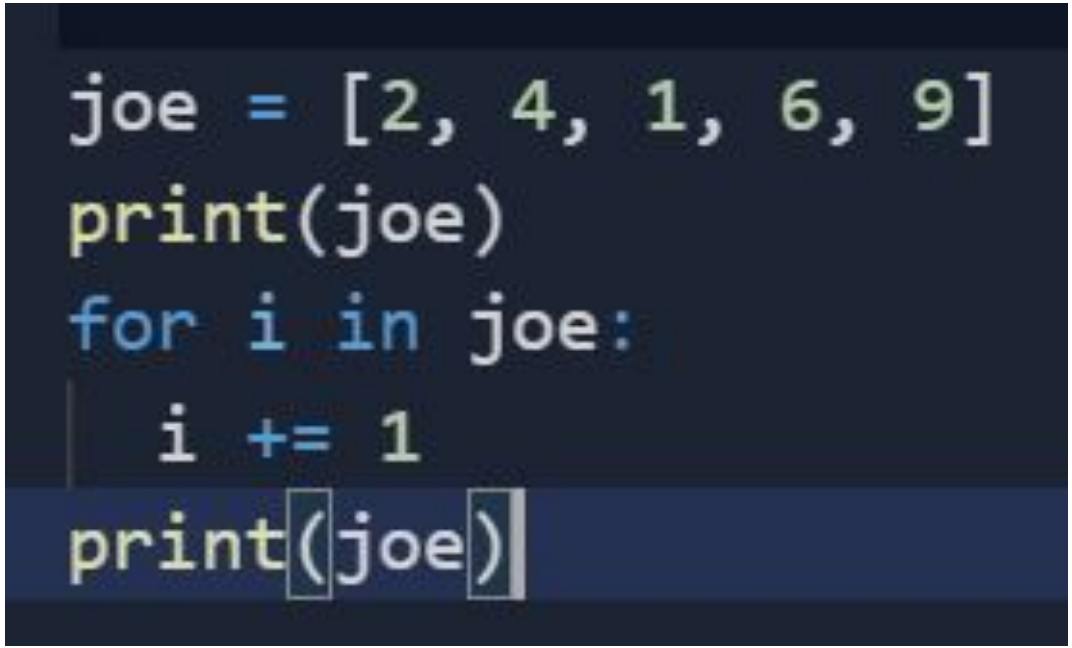
Lists allow us to iterate through multiple values and change them all quickly - using loops!

Loops

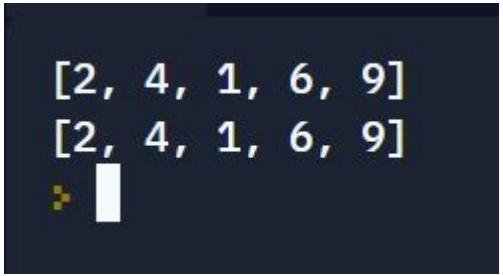
```
joe = [2, 4, 1, 6, 9]

for i in joe:
  print(i + 1)

while len(joe) > 3:
  joe.remove(joe[-1])
print(joe)
```

Python’s for loop is like Java or C#’s foreach loop. There is no Java-style for loop. Why is this important?

What's wrong with this code?



Nothing happens!



The variable i has changed, but that didn't change what was inside of joe!
How do we change things then?

What do we do then?

```
joe = [2, 4, 1, 6, 9]
print(joe)
for i in joe:
    joe[joe.index(i)] += 1
print(joe)
```

```
[2, 4, 1, 6, 9]
[3, 5, 2, 7, 10]
>
```

Lab: Shuffle the deck

Ask for a string of letters and numbers and shuffle them using loops and lists!
Hint: use random