# KEY_Lesson10_Loops1

January 2, 2020

## 1 Loops

Earlier, we learned about lists, and how they are super useful ways to store a lot of information in one variable. Suppose you had a list of animals and you wanted to print your list so that each item was on its own line.

[3]:
```
# create a list of animals
animals = ['turtle', 'penguin', 'platypus', 'zebra', 'giraffe']
```

[4]:
```
# print animals
print(animals)
```

```
['turtle', 'penguin', 'platypus', 'zebra', 'giraffe']
```

We see that using the `print` function on our list prints all the values in brackets and separated by commas on the same line. This isn't quite what we want. But, Python has a useful tool we can use for this problem called the for loop.

A for loop has three key parts:

```
for [item] in [list of items]:
  [command]
```

A for loop will look at each individual item in the list it is given, then run the given command on each item.

Let's take a look at how we might use a for loop to solve our problem

[3]:
```
# create a for loop to print each item in our animals list
for animal in animals:
    print(animal)
```

```
turtle
penguin
platypus
zebra
giraffe
```

A Python function that is really useful when writing loops is the `range` function. `range` creates a list of numbers between the two numbers given to the function.

If we use the output of the `range` function in a for loop, then whatever code we write in our for loop will execute on each number between the two parameters of `range`.

[1]:
```
# use a for loop to print each number in range(0,5)
for value in range(0,5):
  print(value)
```

```
0
1
2
3
4
```

The `range` function can be useful when working with lists. To iterate over a whole list, we can use the `range` function with 0 as the start and the list length as the end. Let's test this out with our animals list!

[5]:
```
# use a for loop and the range function to print every animal in the list␣
↪animals
for index in range(0,len(animals)):
    print(animals[index])
```

```
turtle
penguin
platypus
zebra
giraffe
```

We don't just have to use range, though. Let's say we only wanted to print out some of the animals in our list. We could give our for loop a list of indices in our list we wanted to print

[6]:
```
# create a list called indices containing indices of the list animals to print
indices = [0,1,4]

# use a for loop to only print animals at the above indices
for index in indices:
  print(animals[index])
```

```
turtle
penguin
giraffe
```

Earlier, we learned about 2D lists. Can for loops be helpful in working with 2D lists? Let's find out!

[8]:
```
# create a list of names for our animals (remember, we have 5 animals!)
names = ['Shelly', 'Tuxedo', 'Perry', 'Stripes', 'Bob']

# create a 2D list combining animals and names
zoo = [animals, names]
```

```
# print zoo so we can look at its structure
print(zoo)
```

```
[['turtle', 'penguin', 'platypus', 'zebra', 'giraffe'], ['Shelly', 'Tuxedo',
'Perry', 'Stripes', 'Bob']]
```

Say we want to print our whole list so that each animal has its name printed next to it.

[10]:
```
# use a for loop to print each item in the 2D list zoo.
for index in range(0,len(zoo)):
    print(zoo[index])
```

```
['turtle', 'penguin', 'platypus', 'zebra', 'giraffe']
['Shelly', 'Tuxedo', 'Perry', 'Stripes', 'Bob']
```

Hmm, that isn't quite what we wanted. But there is another way to do this. We can do something called *nesting* for loops. This means we put two for loops inside one another. Let's see what this might look like.

```
for [item1] in [list1]:
  for [item2] in [list2]:
    [command using item1 and item2]
```

It is **very important** to remember that `item1` and `item2` have different names. Otherwise, Python might not know which list you want to iterate through, and your code will not run quite how you expected!

So how might we nest two for loops inside one another to print out our animal names?

[12]:
```
# use a nested for loop to print each animal and its name
for column in range(0,len(zoo[0])):
    animal_info = ""
    for row in range(0,len(zoo)):
        animal_info += zoo[row][column] + " "
    print(animal_info)
```

```
turtle Shelly
penguin Tuxedo
platypus Perry
zebra Stripes
giraffe Bob
```

We can also have code that is inside the first, or *outer*, for loop, but is not inside the second, or *inner*, for loop. We can use this idea to make our list above a bit easier to read.

Excellent work! You just learned about for loops. You learned: - How to write a for loop to iterate over items in a list - How to write a for loop using the `range` function - How to *nest* for loops inside one another