for Loops

A `for` loop acts as an iterator in Python; it goes through items that are in a *sequence* or any other iterable item. Objects that we've learned about that we can iterate over include strings, lists, tuples, and even built-in iterables for dictionaries, such as keys or values.

Here's the general format for a `for` loop in Python:

```
for item in object:
    statements to do stuff
```

In [1]:
```
list1 = [1,2,3,4,5,6,7,8,9,10]
```

In [2]:
```
for num in list1:
    print(num)
```

```
1
2
3
4
5
6
7
8
9
10
```

Great! Hopefully this makes sense. Now let's add an `if` statement to check for **even** numbers. We'll first introduce a new concept here--the modulo.

Modulo

The modulo allows us to get the remainder in a division and uses the % symbol. For example:

In [3]:
```
13 % 5
```

Out[3]: 3

This makes sense since 13 divided by 5 is 2 remainder 3. Let's see a few more quick examples:

In [4]:
```
# 3 Remainder 1
10 % 3
```

Out[4]: 1

In [5]:
```
# 2 Remainder 4
18 % 7
```

Out[5]: 4

In [6]:
```
# 2 no remainder
4 % 2
```

Out[6]: 0

Notice that if a number is fully divisible with no remainder, the result of the modulo call is 0. We can use this to test for even numbers, since if a number modulo 2 is equal to 0, that means it is an even number!

Back to the `for` loops!

Let's print only the even numbers from that list!

In [7]:
```
for num in list1:
    if num % 2 == 0:
        print(num)
```

```
2
4
6
8
10
```

We could have also put an `else` statement in there:

In [8]:
```
for num in list1:
    if num % 2 == 0:
        print(num)
    else:
        print('Odd number')
```

```
Odd number
2
Odd number
4
Odd number
6
Odd number
8
Odd number
10
```

Another common idea during a `for` loop is keeping some sort of running tally during multiple loops. For example, let's create a `for` loop that sums up the list:

In [9]:
```
# Start sum at zero
list_sum = 0

for num in list1:
    list_sum = list_sum + num

print(list_sum)
```

```
55
```

Also we could have implemented a += to perform the addition towards the sum. For example:

In [10]:
```
# Start sum at zero
list_sum = 0

for num in list1:
    list_sum += num

print(list_sum)
```

```
55
```

We've used `for` loops with lists, how about with strings? Remember strings are a sequence so when we iterate through them we will be accessing each item in that string.

In [11]:
```
for letter in 'I am Abhishek.':
    print(letter)
```

```
I

a
m

A
b
h
i
s
h
e
k
.
```

Let's now look at how a `for` loop can be used with a tuple:

In [12]:
```
tup = (1,2,3,4,5)

for t in tup:
    print(t)
```

```
1
2
3
4
5
```

Let's start exploring iterating through Dictionaries to explore this further!

In [16]:
```
d = {'k1':1,'k2':2,'k3':3}
```

In [17]:
```
for item in d:
    print(item)
```

```
k1
k2
k3
```

Conclusion

We've learned how to use for loops to iterate through tuples, lists, strings, and dictionaries.