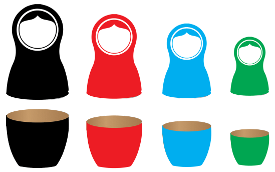
Overview of Recursion

Repetition in Programming



We have seen ways of a producing repetition in a program through **loops (https://maryville.instructure.com/courses/43640/pages/control-flow)** .

We can also utilize **recursion** in order to accomplish this. Recursion is a technique by which a function makes one or more calls to itself during execution. A physical example of this would be a Russian nesting doll. When opening the doll, there is another doll present inside of it until there is an end point.

How Recursive Commands Work

Let's also look at a mathematical example. What would we do if we wanted to calculate a list of numbers like [2, 4, 6, 8, 10]? Remember that addition is just a function between two numbers. We would just have to group the numbers together. It would probably look something like this without any while or for loops applied to it:

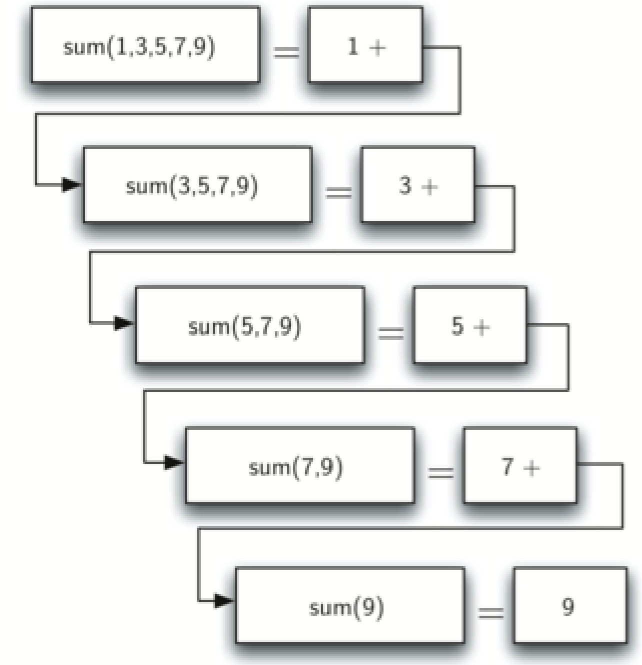
```
total = (1+(3+(5+(7+9))))

total = (1+(3+(5+16)))

total = (1+(3+21))

total = (1+24)

total = 25
```

Now let's code this out:

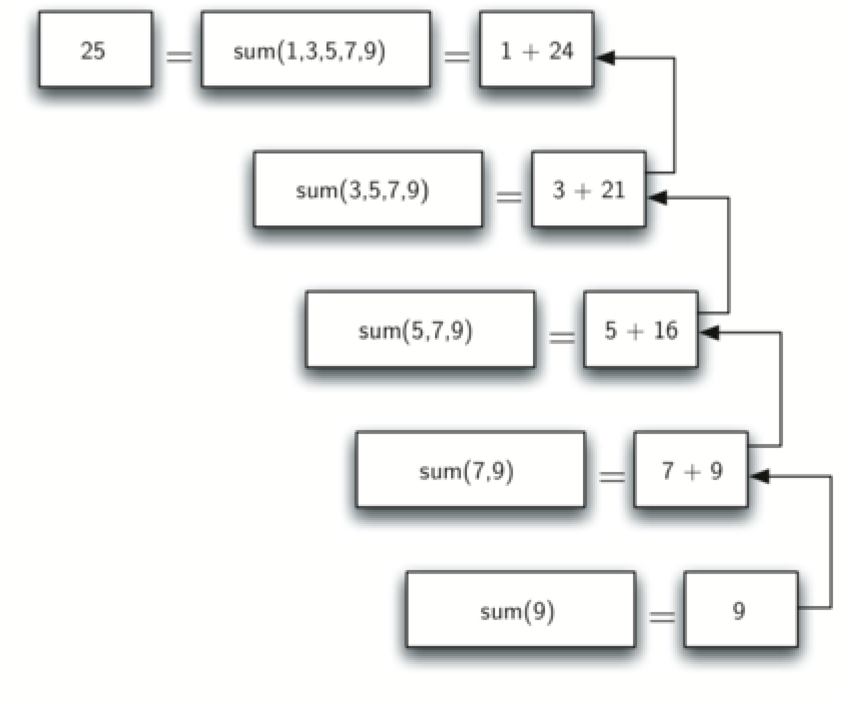
```
def listsum(numList):

if len(numList) == 1:

return numList[0]

else:

return numList[0] + listsum(numList[1:])
```

```
print(listsum([1, 3, 5, 7, 9])
```

What we can see is that the code in the second line is calling to check if the list is one element long. We follow that up where in line 5, it calls upon itself. This is the repetition of recursion. Below is a graphical display of how these recursive commands are working.



What is happening is that we are breaking apart the operation into the simplest parts that it can be. Once we reach that point, we then begin to do the calculations.



When we hit the topmost part of this tower, we then have our answer.

Source: Problem Solving and Algorithms in Python (http://interactivepython.org/runestone/static/pythonds/index.html#) from Bradley Miller on www.interactivepython.org (http://interactivepython.org/runestone/static/pythonds/index.html#) .