Recursion

in Computer Programming

Recursion in Python

We know that in python you can create functions. It's also possible for a function can call other functions.

Did you know that it's even possible for the function to call **itself**?

When a function calls itself we refer to this as a **recursive function.**

The principle of **recursion** comes from mathematics. (Sometimes a mathematical function can be defined as a function of itself).

An example of a recursive function:
finding the factorial of a number: 6! = 6x5x4x3x2x1

```
def calc_factorial(x):

    if x == 1:
        return 1
    else:
        return (x * calc_factorial(x-1))

num = int(input("enter a number"))
print("The factorial of", num, "is",
calc_factorial(num))
```

Key points:

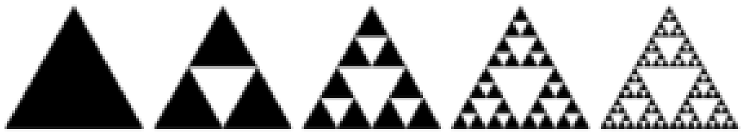
- In Programming **Recursion** is a way to **repeat actions** without `while`, or `for` loops.
- When you see a function that calls itself, this means you have a created a an action that will repeat itself (similar to a loop).

When to use it:

For now, you won't use recursion often, but it is an important computer science concept that can help you solve specific problems. **You must be able to recognize and understand recursion when it appears in code.** You will notice that most of the recursive problems we look at in this section could be solved using a simple `for` or `while` loop. This won't always be the case.

Exercise 1 ***Enter in the example at the top of the page the code above and try it out in the IDE of your choice.***

This is an example of a recursive pattern. ***A pattern that is made up of itself***



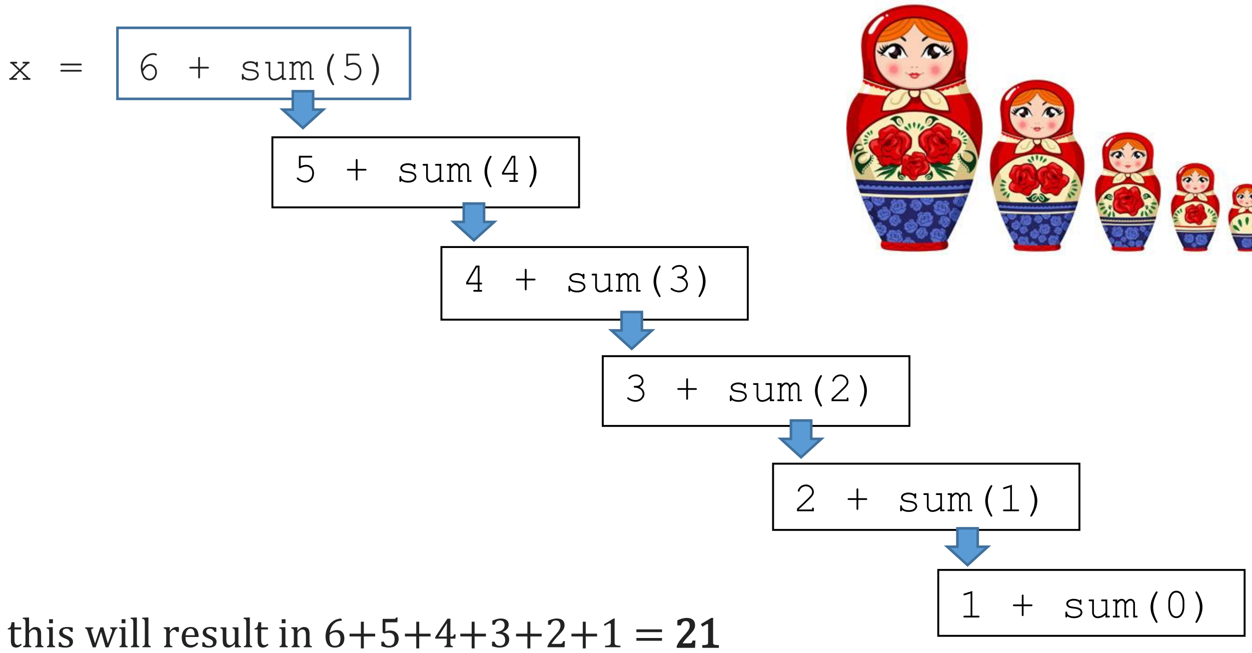
Recursive patterns: the product is made by repeating itself!

```
def sum(x):    #Function called "sum" is defined

    if (x!=0):
        return x + sum(x-1) #sum() function calls itself!
    else:
        return x;
print(sum(6))
```

This line repeats again and again because "sum" calls itself…so every time it is called,…it will be called again and again and again! if (x!=0)

if we input x = 6, the result would be:



this will result in 6+5+4+3+2+1 = **21**

Look carefully!

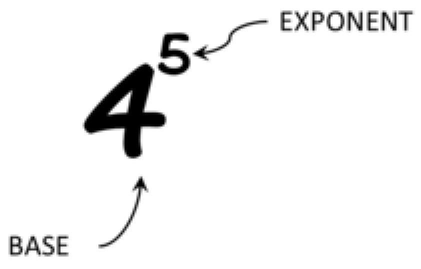
1. **the function** `sum()` **is called with a value of x=6**
2. `sum(6)` **will add the number 6 to the result of** `sum(5)`…
3. **but** `sum(5)` **calls itself as** `sum(4)`
4. `sum(4)` **calls itself as** `sum(3)` **….and so on..**
5. **the function keeps adding versions of itself until x = 0.**

Exercise 2

Enter the code above and try it out in the IDE of your choice.

Exercise 3 (try and then check)

Write a program that will find the value of one number to the power of another number (you must use recursion).



Sample input:

```
Base? 4
Exponent? 2
```

Sample output:

```
4 to the power of 2 is 16
```

```
Algorithm:

   1. A function is created that takes two inputs:
      Base and Exponent
   2. The function will return whatever the Base number is

But…

   3. If power is greater than 1 the function will return:
     (Base number) x (the function itself)

      and then the program will reduced power value by 1


Try on your own before looking at the solution below.
```

Sample solution:

```
def pow(base, power):
  if power == 1:
    return base
  else:
    return base*pow(base, power-1)
print(pow(3,4))
```

Exercise 4 Recursive patterns!

The code below will create a drawing of a tree which uses recursion to draw a series of branches. **Each branch is made up of smaller branches identical to the branch itself.** Look at the code to see if you can predict what will happen and then cut and paste the code into an IDE to see what happens

```
#Recursive Tree Challenge

from turtle import *
from random import randint

#Recursive function to draw a tree, branch by branch

def drawTree(level,size,angle,ratio):
  if level >= 0:
    forward(size)
    left(angle)
    drawTree(level-1,size/ratio,angle,ratio)
    right(2*angle)
    #Draw right branch of the tree
    drawTree(level-1,size/ratio,angle,ratio)
    left(angle)
    forward(-size)
  else:
    #Stop the recursion
    return

#Main Program Starts Here

speed(0)
penup()
goto(0,-180)
left(90)
pendown()

#Draw a tree using a recursive function

size = randint(80,120)
angle = randint(20,40)
ratio = randint(14,18)/10
level = randint(4,6)

drawTree(level,size,angle,ratio)
```

Exercise 4

Use recursion to find the max value in a list.

Possible algorithm:

If there is a single element, return it.

Else return the maximum between the following two values:

a) Last Element

b) Value returned by recursive call of the *next* element

do this until you have reached the end of the list.

Sample solution below:
(try on your own first)

Sample Solution:

```
def maxElement(list1):
  if len(list1) == 1:
    return list1[0]
  else:
    max = maxElement(list1[1:])
    if list1[0] > max:
      return list1[0]
    else
      return max
```

Exercise 5 (try and then check)

Write a program that will find the number of digits in an integer (using recursion).

Sample input:

```
Enter your number:   486584
```

Sample output:

```
Number of digits: 6
```

Before you try:

The standard algorithm for this problem is to:

1. divide the entered number by 10.
2. Then check to see if the result is less than zero.
3. If you keep track of the number of times you divide before getting zero,…then you have found the number of digits.

Example: **675**

675/10 = 67.5   67.1/10=6.75   6.71/10=0.671

Had to divide **3** times before result was less than zero.

Sample Solution

```
def DigitCount(n):

    if n < 10:
        return 1
    else:
        return 1 + DigitCount(n / 10)
```