Recursion

Who answers when you call yourself ?

But first, what is recursion ?

It is a method of solving problems that uses a function that calls itself from inside.

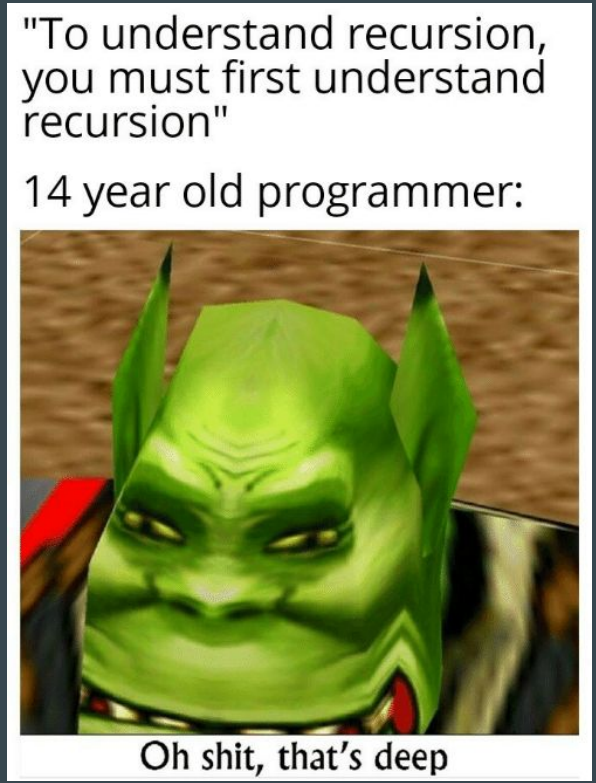
It looks something like that:

```
def myRecursiveFunction():

       return myRecursiveFunction()

print(myRecursiveFunction())
```

Warning : this code might show what recursion looks like, but it has several problems, can you spot it ?



So, what is wrong with it ?

```
def myRecursiveFunction():

        return myRecursiveFunction()

print(myRecursiveFunction())
```

1. The function has no parameters =>
    i. The function calls an identical version of itself
    ii. It is impossible to change the behaviour of the function
2. There is no base cases => what do we return when we hit the end of the loop

```
def myRecursiveFunction(n):
        if n==0:
                return 1
        return myRecursiveFunction(n-1)*n

print(myRecursiveFunction(9))
```

When to use it ?

-Main problem is divisible in similar smaller problems

-We could use it almost everywhere :

Iteration ⇔ Recursion

So why use recursion at all ?

-Combine MULTIPLE sub-problems to get the final solution

Some examples

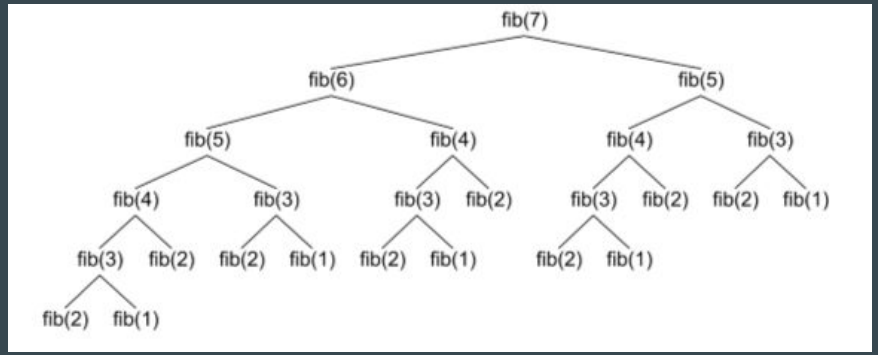
Fibonacci sequence : $F_0 = 0$ | $F_1 = 1$ | $F_n = F_{n-1} + F_{n-2}$

What we see: To know F_n we must first know the two termes before.

In summary, F_n is divisible in the same problems but smaller => Let's use recursion

```
def fibonacci(n):
      if n==0: return 0
      if n==1: return 1
      return fibonacci(n-1)+fibonacci(n-2)

print(fibonacci(7))
```



Works but be careful of the complexity !

Complexity : $O(2^n)$

Some examples

We have a family tree, and want to count the number of descendants of one person.

=> it is the sum of each child of this person and the number of descendants of each child

We can write numDesc(person) = numChild + sum(numDesc(child) for each child)

```
def numDesc(person):
	childs = person.childs
	numD = len(person.childs)
	for(child in childs):
		numD += numDesc(child)
	return numD
```

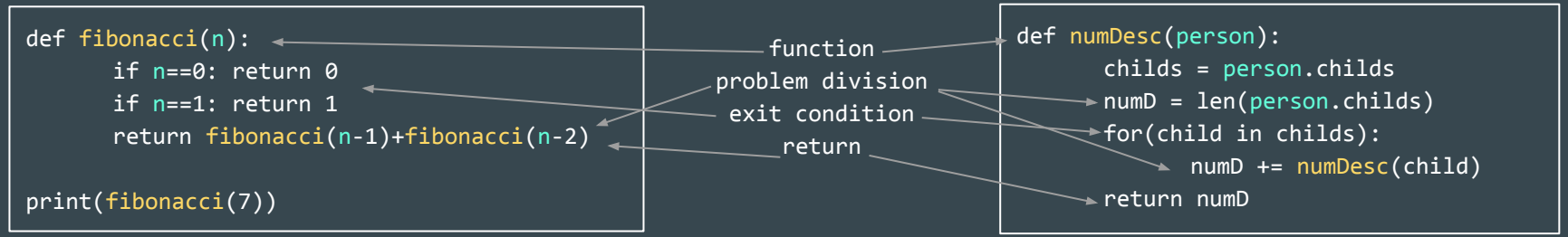
How to make it ?

-Create a function (python syntax : `def myFunction(param):`)

-Divide the main problem in smaller problem (find the equation between problems)

-Identify how to stop the loop (base cases/no more parsing possible)

-Remember to return a result



Question ?