**Fábio Lucas Romeiro de Castro**

**Fixação 1.1:**

def Comb(n,k):

if k == 1:

return n

if k == n:

return 1

if 1<k and k<n:

return Comb(n-1, k-1) + Comb(n-1, k)

**Fixação 1.2:**

def Max(V):

maiorValor = 0

for n in V:

if n > maiorValor:

maiorValor = n

return maiorValor

**Fixação 1.3:**

9 e 23

**Fixação 2.1:**

É uma função que dentro do seu bloco ela chama a si mesma.

**Fixação 2.2:**

A vantagem é a de que o código fica mais enxuto, menos verboso. A desvantagem é que a execução do código fica mais lenta por precisar realizar muita alocação e desalocação de memória.

**Fixação 2.3:**

A recursividade é boa quando utilizada em contextos na qual é necessário a tentativa e erro, manipulação de árvores ou analisadores léxicos recursivos de compiladores.

**Complementar 1:**

def somaNumerosReais(lista):

if len(lista) == 1: return lista[0]

else: return lista[0] + somaNumerosReais(lista[1:])

**Complementar 2:**

def ocorrencia(n,k):

if n==0: return 0

return ocorrencia(n/10,k) + (n%10==k)

**Complementar 3:**

def convert(num):

if num == 0: return 0

return convert(num/2)

**Complementar 4:**

def mdc(x,y):

   if x == y: return x

   if x < y: return mdc(y, x)

   return mdc(x ­ y, y)

**Complementar 5:**

def pot(b,n):

if p == 0: return b

else: return pot(b\*2,p-1)