universidade de aveiro
escola superior de tecnologia e gestão de águeda

Introdução à Programação

Recursividade

Recursividade

universidade de aveiro
escola superior de tecnologia e gestão de águeda

Termo usado de maneira mais geral para descrever o processo de repetição de um objeto de um jeito similar ao que já fora mostrado.

Por exemplo, observando a seguinte função:

```
def countdown(n):
    if n <= 0:
        print('Boom!')
    else:
        print(n)
        countdown(n-1)
```

Se n for 0 ou negativo, imprime a palavra "Boom!" Caso contrário, imprime n e chama novamente a função countdown (ela própria) passando n-1 como um argumento.

Recursividade

universidade de aveiro
escola superior de tecnologia e gestão de águeda

```
>>> countdown(3)
```

A execução da função countdown começa com n = 3 e, como n é maior que 0, ele imprime o valor 3 e, em seguida, chama-se a si mesmo...

> A execução da função countdown começa com n = 2 e, como n é maior que 0, ele imprime o valor 2 e, em seguida, chama-se a si mesmo...
>
> > A execução da função countdown começa com n=1, e como n é maior que 0, ele imprime o valor 1 e, em seguida, chama-se a si mesmo...
> >
> > > A execução da função countdown começa com n = 0 e, como n não é maior que 0, ele imprime a palavra “Boom!” e depois retorna.
> >
> > A função countdown que obteve n=1 retorna.
>
> A função countdown que obteve n=2 retorna.

A função countdown que obteve n=3 retorna.

O output fica assim:

3

2

1

Boom!

Recursão Infinita

Se uma recursão nunca atingir o caso base, ela continuará a fazer chamadas recursivas para sempre e o programa nunca terminará.

Um exemplo de um programa mínimo com uma recursão infinita:

```
def recurse():
    recurse()
```

Na maioria dos ambientes de programação, um programa com recursão infinita não é realmente executado para sempre.

O Python reporta uma mensagem de erro quando a profundidade máxima de recursão é atingida.

RuntimeError: Maximum recursion depth exceeded

Exemplo de factorial

O **factorial** de um número **n**, que se representa por **n!** (lê-se **n factorial**) é:

- O produto de todos os números inteiros positivos até n.

  $n! = n \times (n-1) \times (n-2) \times \ldots \times (1)$

- Também pode ser descrito por:

  $n! = n \times (n-1)!$

  ou seja, é o produto de número x o factorial do número uma unidade menor que n

Nota. Por definição, o factorial de 0 é 1

$$0! = 1$$

O factorial de um número depende do fatorial do número anterior e não pode ser calculado sem conhecer o anterior

Exemplo de factorial(4)


```
def factorial(n):
    if n == 0:              # caso base
        return 1
    else:
        return n * factorial(n - 1)   # chamada recursiva
```

- Descrição de execução

factorial(1) espera por resultado de factorial(0)

factorial(2) espera por resultado de factorial(1)

factorial(3) espera por resultado de factorial(2)

factorial(4) espera por resultado de factorial(3)

Esquema da execução (verificar com o debugger)

```
factorial(4)
        4 * factorial(3)
                3 * factorial(2)
                        2 * factorial(1)
                                1 * factorial(0)
                                        return 1          #caso base
```