Universidade Federal do Espírito Santo

Departamento de Computação e Eletrônica - CEUNES
Programação I / Programação Funcional
Prof. Oberlan Romão

Aula Prática 6 – Recursão

Atenção: Alguns dos arquivos .py estão disponíveis no AVA, baixem-os para facilitar a implementação dos exercícios.

Recursão é um método de resolução de problemas que envolve quebrar um problema em subproblemas menores e menores até chegar a um problema pequeno o suficiente para que ele possa ser resolvido trivialmente. Normalmente recursão envolve uma função que chama a si mesma. Embora possa não parecer muito, a recursão nos permite escrever soluções elegantes para problemas que, de outra forma, podem ser muito difíceis de programar.

Toda função recursiva deve obedecer a três leis importantes:

1. Deve chamar a si mesmo, recursivamente;
2. Deve ter um caso básico (chamado de **caso base**);
3. Deve mudar o seu estado e se aproximar do caso básico (chamado de **passo recursivo**).

Como visto em sala de aula, o fatorial de um número natural n, representado por $n!$, é o produto de todos os inteiros positivos menores ou iguais a n, conforme definido na equação matemática abaixo:

$$n! = \begin{cases} 1, & \text{se } n \leq 1 \\ n \times (n-1)!, & \text{caso contrário} \end{cases}$$

De forma funcional, isso pode ser expresso facilmente em Python como:

```
1 def fatorial(n):
2     if n <= 1:
3         return 1
4     else:
5         return n * fatorial(n - 1)
```

Existem algumas ideias-chave nesse programa para se estudar. Em primeiro lugar, na linha 2 estamos verificando se o valor de n é menor ou igual a 1. Esse teste é fundamental, chamado caso base, e é a nossa cláusula de escape da função. Por definição, se n for 0 ou 1, o valor do fatorial será, de forma trivial, 1. Em segundo lugar, na linha 5, a função chama a si mesma! Esta é a razão pela qual chamamos `fatorial` de função recursiva. Por simplicidade, considere que n sempre será um inteiro positivo maior ou igual a 0.

Entre no site Python Tutor, cole o código da função `fatorial` abaixo (`fat.py`). Após colar o código, clique no botão "`Visualize Execution`" e, em seguida, clique (várias vezes) no botão "`Next >`" para que você veja o passo a passo da execução do programa. Se precisar/quiser, você pode voltar um passo clicando em "`< Prev`".

```python
def fatorial(n):
    print(f"Iniciando a função fatorial({n})")
    if n <= 1:
        print(f"Finalizando a função fatorial({n}) com valor 1")
        return 1
    else:
        fat = fatorial(n-1)
        print(f"Finalizando a função fatorial({n}) com valor {n*fat}")
        return n*fat

print("5! =", fatorial(5))
```

A Figura 1 mostra a série de chamadas recursivas necessária para calcular o valor de $5!$. Você deve pensar nessa série de chamadas como uma sequencia de simplificações. Cada vez que fazemos uma chamada recursiva estamos resolvendo um problema menor, até chegar ao ponto em que o problema não pode ficar menor.

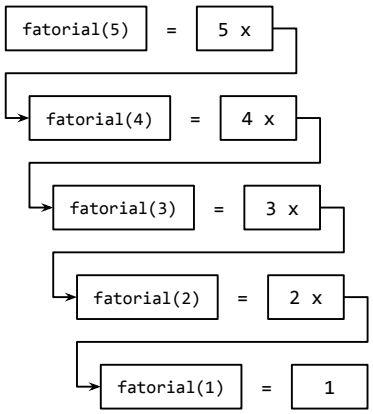

Figura 1: Série de chamadas recursivas da função `fatorial`

Quando chegarmos ao ponto em que o problema é tão simples quanto ele pode ficar, começamos a juntar as soluções de cada um dos pequenos problemas até que o problema inicial seja resolvido. A Figura 2 mostra as multiplicações que são executadas a medida que `fatorial` percorre o seu caminho para trás através de uma série de chamadas. Quando `fatorial` retorna do problema mais alto, obtemos a solução para todo o problema.

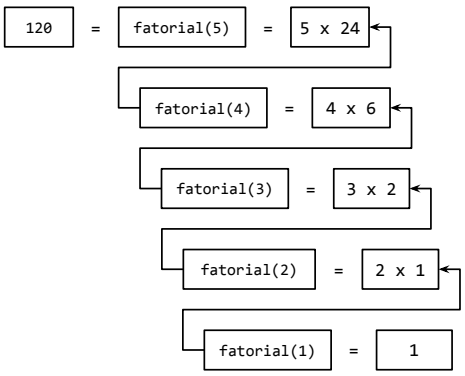

Figura 2: Série de retornos recursivos da função `fatorial`

Exercício 1: (ex1.py) Observe o trecho de código abaixo. Antes de executá-lo tente pensar se há ou não algum problema. Se sim, qual o problema e por que ele ocorre? Modifique a função para que ela aceite apenas valores válidos.

```
def fatorial(n):
    return 1 if n == 0 or n == 1 else n*fatorial(n-1)

def main():
    n = 2.5
    print(f"{n}! = {fatorial(n)}")

    n = -2
    print(f"{n}! = {fatorial(n)}")

main()
```

Exercício 2: Entre no site Python Tutor e cole o seguinte código (visto em sala de aula):

```
def imprime(i):
    if i > 0:
        print(i)
        imprime(i - 1)

imprime(5)
```

Clique em "`Visualize Execution`" e, em seguida, clique (várias vezes) em "`Next >`". Com isso será exibido, passo a passo, a execução do programa.

Exercício 3: Em outra aba do navegador, entre novamente no site Python Tutor e implemente uma função que imprime os número de 1 a m. Chame sua função passando como parâmetro o valor 5. Se tudo saiu como esperado, teste sua função com outros argumentos.

Exercício 4: Em outra aba, implemente uma função que imprima os números de n a m. Chame sua função passando 5 e 10 como parâmetros. Se tudo saiu como esperado, teste sua função com outros argumentos.

Exercício 5: (ex5.py) Analise a função `f1` abaixo. O que você acha que essa função irá retornar?

```
def f1(n, i = 0, soma = 0):
   if i <= n:
       return f1(n, i + 1, soma + i)
   else:
       return soma
```

Faça uma função `main()` que chama/utiliza essa função com diferentes parâmetros.

Implemente outra função (chamada `imprimeSoma`) que, usando a função `f1`, imprima o valor de $\sum_{i=1}^{k} i = 1 + 2 + \cdots + k$, com $1 \leq k \leq m$, de todos os números menores que m. A função `imprimeSoma` deve ser de tal forma que ao ser chamada como:

```
imprimeSoma(5)
```

gere a seguinte saída:

```
1 = 1
2 = 3
3 = 6
4 = 10
5 = 15
```

As saídas anteriores poderiam ser obtidas utilizando o código abaixo, mas o objetivo da função `imprimeSoma` é evitar essa replicação de códigos. Perceba que para $m = 5$ replicamos o código 5 vezes e se m fosse 100 ou 1000, faz sentindo copiar/colar esse código 100 ou 1000 vezes?

```python
print(f"1 = {f1(1)}")
print(f"2 = {f1(2)}")
print(f"3 = {f1(3)}")
print(f"4 = {f1(4)}")
print(f"5 = {f1(5)}")
```

Exercício 6: (ex6.py) A função abaixo tem por objetivo fazer a leitura de números positivos enquanto não for digitado um número negativo, que indica o fim da leitura. No entanto, por um descuido, ela não funciona como esperado. Qual o problema? Corrija-o.

```python
def leitura(soma = 0, qt = 0):
    n = float(input("Digite um valor (um numero <0 indica o fim da leitura): "))
    if n < 0:
        return soma, qt
    else:
        leitura(soma + n, qt + 1)

def main():
    soma, qt = leitura()
    print("Quantidade:", qt)
    print("Soma:", soma)
    if qt > 0:
        print("Média:", soma / qt)

main()
```

Exercício 7: (ex7.py) Analise as funções abaixo e tente descobrir o que cada uma faz.

```python
def desenha1(n, c = "*", i = 1):
    if i <= n:
        print(c*n)
        desenha1(n, c, i + 1)

def desenha2(n, c = "*", i = 1):
    if i <= n:
        print(c*i)
        desenha2(n, c, i + 1)


```

```
12 def desenha3(n, c = "*", i = 1):
13     if n % 2 == 0:
14         print("n deve ser impar")
15     if i <= n:
16         ne = (n-i)//2
17         print(" "*ne + c*i)
18         desenha3(n, c, i + 2)
```

Exercício 8: Execute o arquivo `ex8.py`. Lindo, né? Teste o programa com alguns destinos. O que acontece se você digitar um destino inválido (um valor negativo ou maior que 5)? Sua tarefa é corrigir a função `escolheDestino` de forma que ela solicite o destino ao usuário até que seja digitado um valor válido. Veja um exemplo simplificado (os valores sublinhados foram digitados pelo usuário):

```
--> Escolha o seu destino: 8
Destino inválido! Você deve digitar um valor entre 1 e 5.
--> Escolha o seu destino: -1
Destino inválido! Você deve digitar um valor entre 1 e 5.
--> Escolha o seu destino: 2
```

Perceba que nesse código (por simplicidade), os nomes das cidades e as respectivas distâncias foram duplicadas nas funções `inicio` e `imprimeCusto`, o que não é uma boa ideia (se tivermos que trocar a distância de alguma cidade, teremos que lembrar de trocar em todos os lugares onde a informação deve ser alterada). Isso será resolvido quando aprendermos listas. Por enquanto, podemos deixar como está.
Analise o código com bastante atenção. Se você entender muito bem o que cada função faz e como foi implementada, ele te ajudará/guiará na implementação dos EPs. Copiar e colar não irá te ajudar.

Obs.: As cores foram colocadas apenas para conhecimento e para sairmos do tradicional. Nos EPs você não precisa mudar a cor da fonte, mas sinta-se a vontade para tal.