Python - Recursão

Uma função que chama a si mesma é chamada de função recursiva. Este método é usado quando um determinado problema é definido em termos dele mesmo. Embora isso envolva iteração, usar uma abordagem iterativa para resolver tal problema pode ser entediante. A abordagem recursiva fornece uma solução muito concisa para problemas aparentemente complexos.

O exemplo mais popular de recursão é o cálculo fatorial. Matematicamente fatorial é definido como -

```
n! = n × (n-1)!
```

Pode-se ver que usamos o próprio fatorial para definir fatorial. Portanto, este é um caso adequado para escrever uma função recursiva. Vamos expandir a definição acima para o cálculo do valor fatorial de 5.

```
5! = 5 × 4!
   5 × 4 × 3!
   5 × 4 × 3 × 2!
   5 × 4 × 3 × 2 × 1!
   5 × 4 × 3 × 2 × 1
= 120
```

Embora possamos realizar esse cálculo usando um loop, sua função recursiva envolve chamá-lo sucessivamente, diminuindo o número até chegar a 1.

Exemplo 1

O exemplo a seguir mostra como você pode usar uma função recursiva para calcular fatorial -

```
def factorial(n):

   if n == 1:
      print (n)
      return 1
   else:
      print (n,'*', end=' ')
      return n * factorial(n-1)

print ('factorial of 5=', factorial(5))
```

Ele produzirá a seguinte **saída** -

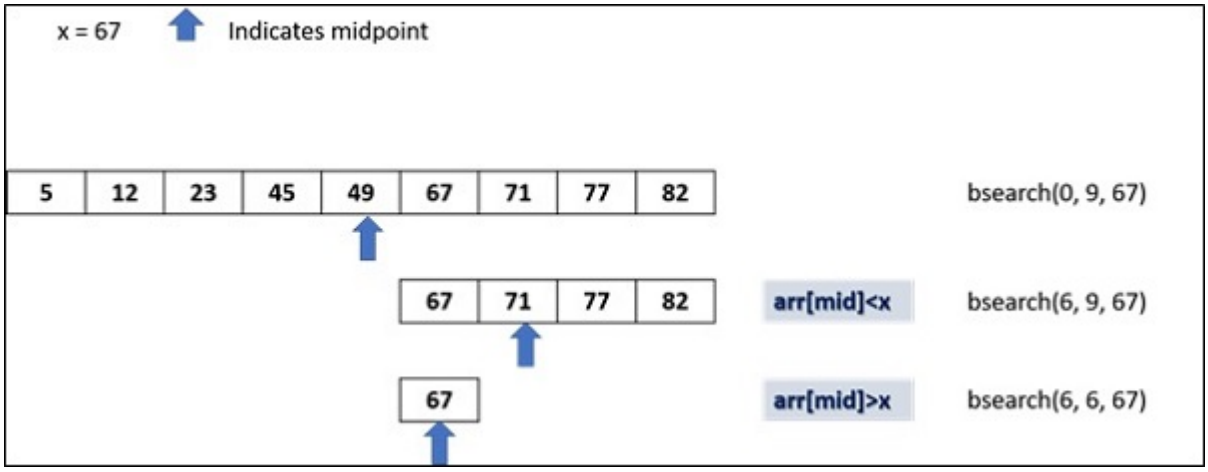
```
5 * 4 * 3 * 2 * 1
factorial of 5= 120
```

Vejamos outro exemplo para entender como funciona a recursão. O problema em questão é verificar se um determinado número está presente em uma lista.

Embora possamos realizar uma pesquisa sequencial por um determinado número na lista usando um loop for e comparando cada número, a pesquisa sequencial não é eficiente, especialmente se a lista for muito grande. O algoritmo de pesquisa binária que verifica se o índice 'alto' é maior que o índice 'baixo. Com base no valor presente na variável 'mid', a função é chamada novamente para procurar o elemento.

Temos uma lista de números, organizados em ordem crescente. Encontramos o ponto médio da lista e restringimos a verificação à esquerda ou à direita do ponto médio, dependendo se o número desejado é menor ou maior que o número no ponto médio.

O diagrama a seguir mostra como funciona a pesquisa binária -



Exemplo 2

O código a seguir implementa a técnica de pesquisa binária recursiva -

```
def bsearch(my_list, low, high, elem):
   if high >= low:
      mid = (high + low) // 2
      if my_list[mid] == elem:
         return mid
      elif my_list[mid] > elem:
         return bsearch(my_list, low, mid - 1, elem)
      else:
         return bsearch(my_list, mid + 1, high, elem)
   else:
      return -1

my_list = [5,12,23, 45, 49, 67, 71, 77, 82]
num = 67
```

```
print("The list is")
print(my_list)
print ("Check for number:", num)
my_result = bsearch(my_list,0,len(my_list)-1,num)

if my_result != -1:
   print("Element found at index ", str(my_result))
else:
   print("Element not found!")
```

Ele produzirá a seguinte **saída** -

```
The list is
[5, 12, 23, 45, 49, 67, 71, 77, 82]
Check for number: 67
Element found at index 5
```

Você pode verificar a saída de diferentes números na lista ou não na lista.