Call Stack – demonstração da execução de código

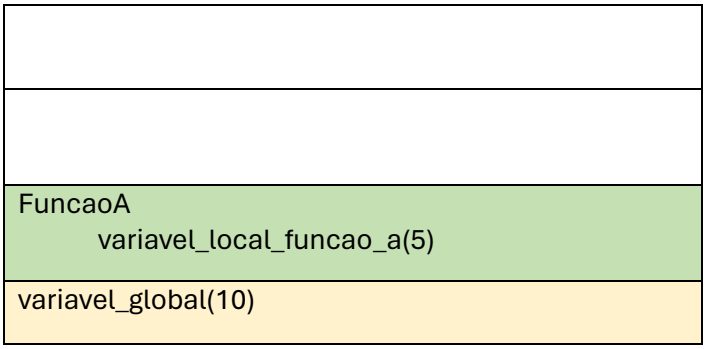
```python
variavel_global=10

def FuncaoA():
    variavel_local_funcao_a=5
    print("Esta é a função A")

def FuncaoB():
    variavel_local_funcao_b=15
    print("Inicio da função B")
    FuncaoA()
    print("Fim da função B")

print("No programa principal")
FuncaoA()
print("No programa principal")
FuncaoB()
print("No programa principal")

```

Depois de executar cada linha

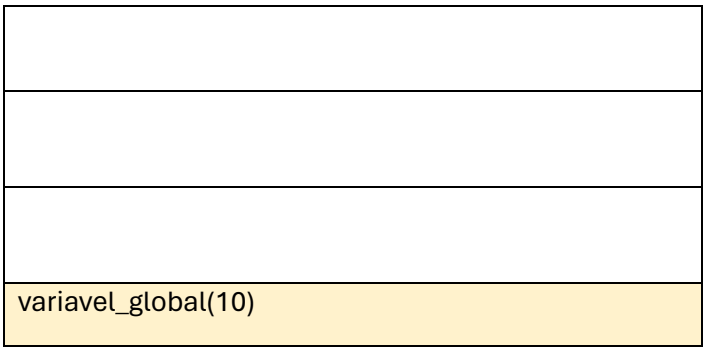
>> Linha 1 – é definida a variável global

variavel_global(10)

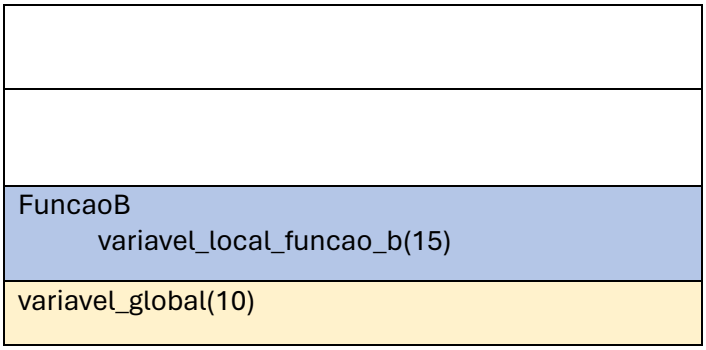
>> Linha 14 – é chamada a função FuncaoA e por isso é criada uma frame nova no topo da stack



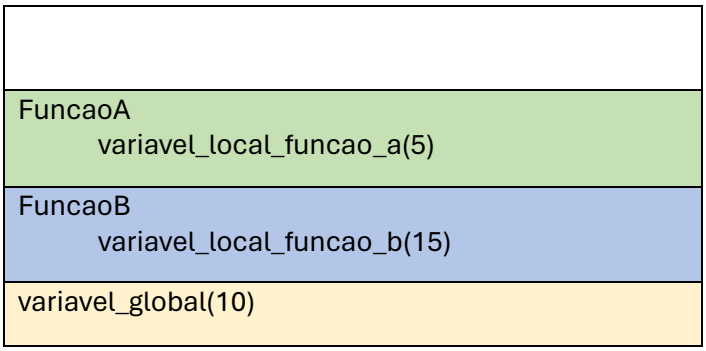
>> Linha 15 – depois de sair da função FuncaoA a frame desta é removida da stack



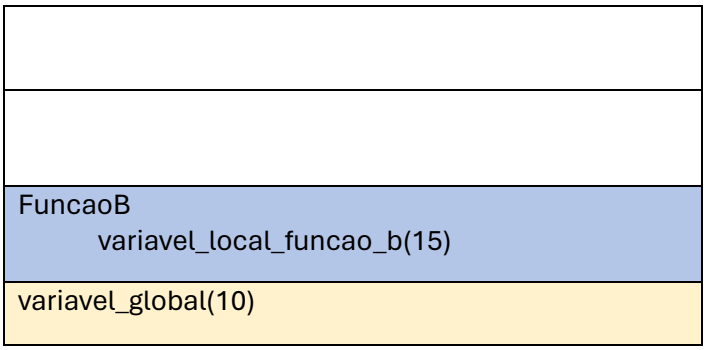
>> Linha 16 – depois de chamar a função FuncaoB é criada uma frame nova no topo da stack



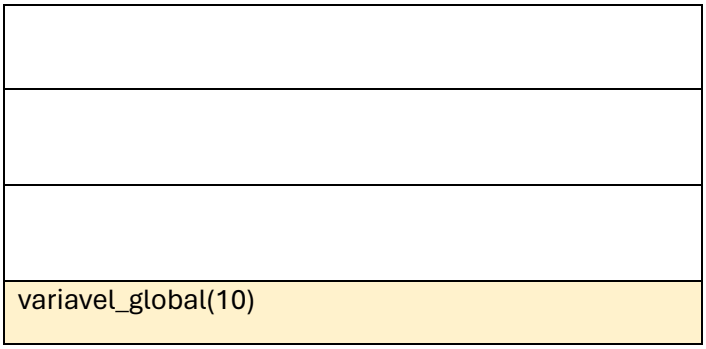
>> Linha 10 – ainda dentro da função FuncaoB é chamada outra função, assim é necessário criar uma frame no topo da stack



>> Linha 11 – terminou a função FuncaoA e a frame desta é retirada, mas ainda não terminou a função FuncaoB



>> Linha 17 – Terminaram todas as funções e a execução volta ao programa principal



Quando o programa termina o sistema operativo “reclama” a memória utilizada libertando-a para outros programas.