Guia Completo: Entendendo a Instrução return em Python

1. Introdução ao Conceito de Retorno em Funções

No universo da programação em Python, as funções (definidas pela palavra-chave def) são como **rotinas** . Imagine a sua rotina matinal: acordar, tomar banho e ir à padaria. No código, automatizar tarefas repetitivas para que não seja necessário escrever os mesmos comandos várias vezes.No entanto, existe uma diferença fundamental entre uma função que apenas executa uma tarefa e uma que entrega algo. Pense na rotina "ir à padaria": se você apenas vai e volta, você executou a tarefa, mas não trouxe nada. Agora, se você vai à padaria e **traz o pão** , esse pão é o seu return. A instrução return permite que uma função envie um valor (um dado) de volta para quem a chamou, permitindo que esse resultado seja guardado e utilizado posteriormente no **Programa Principal** .

2. A Diferença Fundamental: print() vs. return

Uma das maiores confusões para quem está começando é distinguir o print() do return.

- **O print()** é visual. Ele funciona como um "relâmpago": ele brilha na tela, mostra a informação ao usuário e desaparece. O programa não consegue "segurar" esse valor para usá-lo em um cálculo depois.
- **O return** é funcional. Ele entrega um objeto físico para o programa. É como se a função colocasse o resultado nas mãos do programador para que ele decida se quer imprimir, salvar em um banco de dados ou usar em outra conta.| Comando | Função Principal | Persistência do Dado || ------ | ------ | ------ || print() | Exibir um resultado visual na tela (para o humano). | **Temporária** : O dado "morre" após ser exibido. || return | Disponibilizar um valor para o sistema (para o código). | **Permanente** : O valor pode ser armazenado em variáveis. |

3. Parâmetros Formais vs. Parâmetros Reais

Antes de avançarmos para a captura de valores, é vital entender como os dados entram na função. No material do Prof. Guanabara, distinguimos dois tipos de parâmetros:

1. **Parâmetros Formais:** São as variáveis definidas na assinatura da função (no def). Exemplo: Em def soma(a, b):, a e b são os parâmetros formais.
2. **Parâmetros Reais:** São os valores que você passa no momento da chamada. Exemplo: Em soma(5, 3), os números 5 e 3 são os parâmetros reais.O Python realiza uma cópia automática: o valor do parâmetro real é copiado para dentro do parâmetro formal para que a rotina possa processá-lo.

4. Captura de Valores e Personalização

Quando uma função retorna um valor, a chamada da função é "substituída" pelo resultado no Programa Principal. Para não perder esse dado, devemos capturá-lo em variáveis.

```
def somar(a, b, c=0): # 'a, b, c' são parâmetros formais
    s = a + b + c
    return s

# --- PROGRAMA PRINCIPAL ---
# Capturando os resultados (os parâmetros reais são 3, 2, 5, etc.)
```

```
r1 = somar(3, 2, 5)
r2 = somar(2, 2)
r3 = somar(6, 0)

print(f"Os resultados foram {r1}, {r2} e {r3}.")

# Exemplo de personalização: Usando o retorno em uma condicional
if somar(10, 5) > 10:
    print("A soma é maior que dez!")
```

A captura permite que você personalize a saída. Se a função tivesse apenas um print, você estaria preso à mensagem que o criador da função escreveu. Com o return, você decide como exibir o dado.

5. Exemplos Práticos: Cálculos Matemáticos

Soma Personalizada

Aqui, a função calcula o valor, mas o **Programa Principal** decide como formatar a mensagem.

```
def soma(a, b):
    return a + b

# --- PROGRAMA PRINCIPAL ---
res = soma(4, 5)
print(f"O valor total da operação foi: {res}")
```

Cálculo de Fatorial

No exemplo do fatorial, utilizamos uma variável acumuladora f. Note que iniciamos f = 1 porque 1 é o elemento neutro da multiplicação.

```
def fatorial(n=1):
    f = 1
    for c in range(n, 0, -1):
        f *= c
    return f  # O retorno acontece apenas APÓS o término do laço
for

# --- PROGRAMA PRINCIPAL ---
num = 5
print(f"O fatorial de {num} é igual a {fatorial(num)}")
```

O return f devolve o produto acumulado final para o programa, permitindo que usemos esse número em qualquer outro cálculo.

6. Exemplos Práticos: Verificações Lógicas (Predicados)

Funções que retornam True ou False são extremamente poderosas e facilitam a leitura do código, sendo chamadas de funções booleanas ou predicados.

```
def par(n):
    if n % 2 == 0:
        return True
    else:
        return False

# --- PROGRAMA PRINCIPAL ---
num = 8
if par(num): # A função retorna True, então o 'if' é executado
    print(f"O número {num} é par.")
else:
    print(f"O número {num} é ímpar.")
```

7. Exemplos Práticos: Retornos Literais (Status de Votação)

Uma função também pode processar uma lógica e retornar uma frase pronta (string literal). Isso é útil para garantir que a lógica de negócio (idade de voto) esteja separada da interface (o print).

```
def voto(ano_nascimento):
    from datetime import date
    atual = date.today().year
    idade = atual - ano_nascimento

    if idade < 16:
        return f"Com {idade} anos: NÃO VOTA."
    elif 16 <= idade < 18 or idade > 65:
        return f"Com {idade} anos: VOTO OPCIONAL."
    else:
        return f"Com {idade} anos: VOTO OBRIGATÓRIO."

# --- PROGRAMA PRINCIPAL ---
nasc = int(input("Em que ano você nasceu? "))
print(voto(nasc))
```

8. Conclusão e Melhores Práticas

O uso do return é o que separa um script básico de um sistema modular e profissional. Ele oferece:

- **Modularização:** Você separa a lógica (o que o código faz) da exibição (como o usuário vê).
- **Reutilização:** Uma função que retorna um número pode ser usada em mil lugares diferentes do seu programa.
- **Flexibilidade:** O Programa Principal mantém o controle total sobre os dados.

3 Dicas para decidir entre return e print:

1. **Precisa do valor depois?** Se o resultado da função for ser usado em uma soma, um if ou guardado em uma lista, o return é obrigatório.
2. **Interface Limpa:** Use return para que sua função não "suje" o console com mensagens fixas, permitindo que quem a chame escolha a mensagem ideal.
3. **Criação de Bibliotecas e Módulos:** Se você pretende criar arquivos de funções para usar em outros projetos (como veremos na Aula 22), você **deve** usar return. Sem ele, suas funções serão inúteis para outros scripts que precisam processar dados, não apenas lê-los.