```python
accumulator = accumulator + 1
```

Acumuladores em Python

Aprenda sobre uma das estruturas fundamentais da programação: os acumuladores. Essenciais para cálculos dinâmicos e processamento de dados.

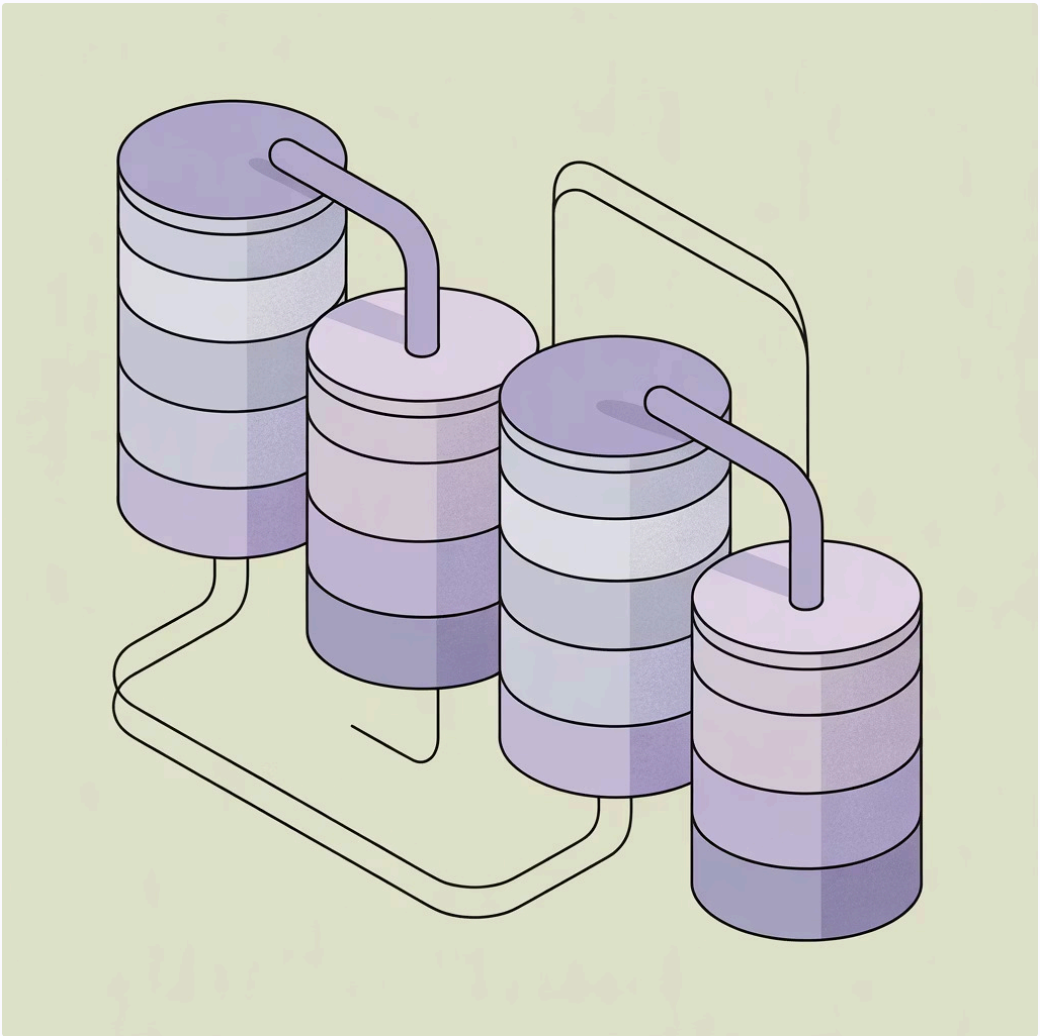
Começar Agora | Ver Exemplos

O que são Acumuladores?

Definição

Acumuladores são variáveis especiais que armazenam valores que mudam durante a execução do programa. Diferentemente dos contadores, que incrementam valores fixos, os acumuladores somam valores variáveis fornecidos pelo usuário ou calculados pelo programa.

A principal característica dos acumuladores é que o valor adicionado não é constante - ele varia conforme a situação ou entrada do usuário.



Diferença: Contador vs Acumulador

Contador

- Incremento constante
- Sempre soma +1
- Controla repetições
- Exemplo: n = n + 1

Acumulador

- Incremento variável
- Soma valores diferentes
- Armazena resultados
- Exemplo: soma = soma + x

Exemplo Prático: Soma de 10 Números

```
n = 1        # contador
soma = 0     # acumulador
while n <= 10:
    x = int(input("Digite o %d número:" % n))
    soma = soma + x  # acumulação
    n = n + 1        # incremento
print("Soma: %d" % soma)
```

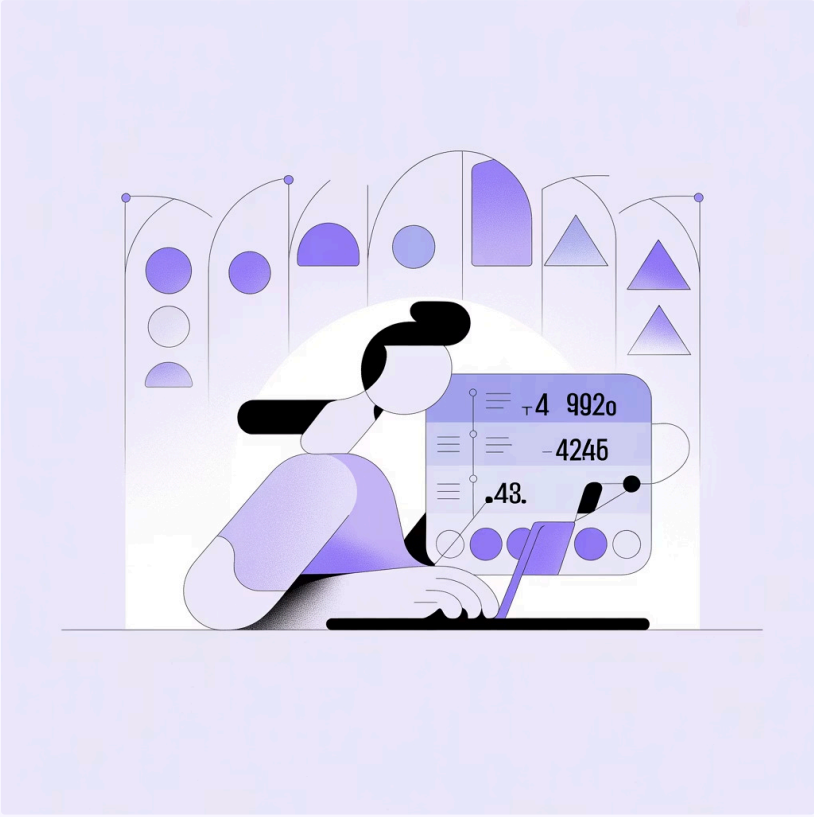
Neste exemplo, **soma** é o acumulador que recebe valores diferentes a cada iteração, enquanto **n** funciona como contador controlando o loop.

Conceito de Média Aritmética

A média aritmética é calculada dividindo a soma de vários números pela quantidade total de números. É uma das aplicações mais comuns dos acumuladores em programação.

$$média = (n1 + n2 + n3 + n4 + n5)/5$$

Em vez de criar cinco variáveis separadas, utilizamos um acumulador para somar os valores conforme são digitados pelo usuário.



Implementando Cálculo de Média

```
x = 1       # contador
soma = 0    # acumulador inicializado
while x <= 5:
    n = int(input("%d Digite o número:" % x))
    soma = soma + n    # acumulação do valor
    x = x + 1
print("Média: %5.2f" % (soma/5))
```

A variável soma é inicializada com zero e vai acumulando os valores digitados. O incremento não é constante porque depende da entrada do usuário, caracterizando um verdadeiro acumulador.

Exercícios Práticos com Acumuladores

Exercício 5.11 - Poupança

Crie um programa que calcule os rendimentos de uma poupança durante 24 meses, considerando depósito inicial e taxa de juros mensal.

Exercício 5.12 - Depósitos Mensais

Modifique o programa anterior para incluir depósitos mensais regulares no cálculo dos juros.

Exercício 5.13 - Quitação de Dívida

Calcule quantos meses são necessários para quitar uma dívida com juros mensais e pagamentos fixos.

Interrompendo Repetições com Break

A estrutura `while` verifica sua condição apenas no início de cada repetição. Porém, às vezes precisamos interromper o loop no meio da execução usando a instrução break.

Esta técnica é especialmente útil quando a condição de parada depende de uma entrada do usuário ou de um cálculo realizado dentro do loop.

Exemplo: Loop com Break

```
s = 0
while True:  # loop infinito
    v = int(input("Digite um número a somar ou 0 para sair:"))
    if v == 0:
        break    # interrompe o loop
    s = s + v
print(s)
```

Usando `while True` criamos um loop infinito, mas controlamos sua execução com `break` quando o usuário digita zero.

Exercícios com Break

01

Leitura até Zero

Programa que lê números até o usuário digitar 0, exibindo quantidade, soma e média dos números digitados.

02

Máquina Registradora

Sistema que processa códigos de produtos e quantidades, calculando o total da compra usando uma tabela de preços.

03

Contagem de Cédulas

Programa que calcula quantas cédulas de cada valor são necessárias para pagar um determinado montante.

Tabela de Códigos de Produtos

Código	Preço (R$)
1	0,50
2	1,00
3	4,00
5	7,00
9	8,00

Esta tabela será utilizada no exercício da máquina registradora para converter códigos em preços e calcular o total das compras.

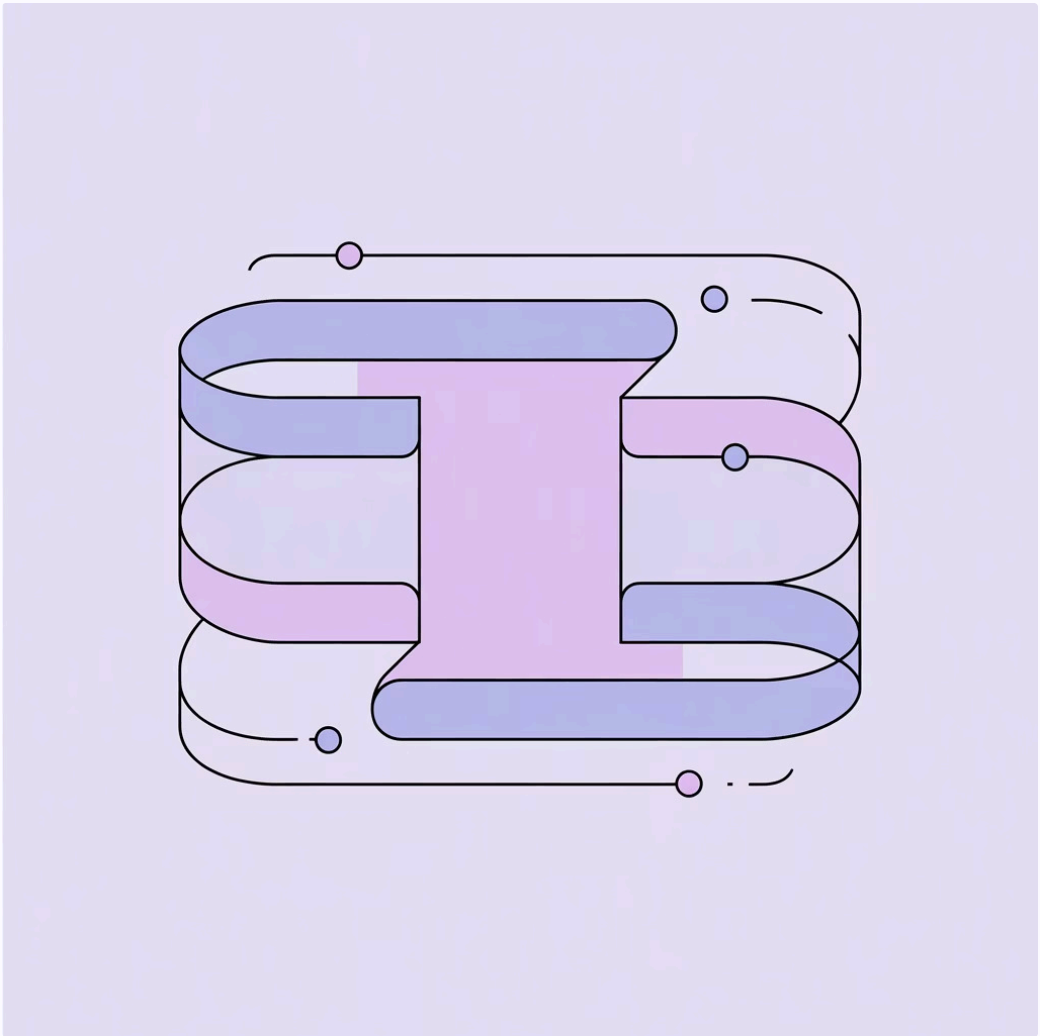
Sistema de Contagem de Cédulas

```
valor = int(input("Digite o valor a pagar:"))
cédulas = 0
atual = 50
apagar = valor
while True:
 if atual <= apagar:
 apagar -= atual
 cédulas += 1
 else:
 print("%d cédula(s) de R$%d" % (cédulas, atual))
 if apagar == 0:
 break
 # Troca para próxima cédula
 if atual == 50: atual = 20
 elif atual == 20: atual = 10
 elif atual == 10: atual = 5
 elif atual == 5: atual = 1
 cédulas = 0
```

Repetições Aninhadas (Nested Loops)

Repetições aninhadas ocorrem quando temos um loop `while` dentro de outro. Esta técnica é fundamental para resolver problemas que envolvem duas ou mais dimensões de iteração.

O exemplo clássico é a impressão de tabuadas, onde precisamos de um loop para cada tabuada (1 a 10) e outro para cada multiplicação (1 a 10).



Exemplo: Tabuadas de 1 a 10

```
tabuada = 1
while tabuada <= 10:      # loop externo
    número = 1            # reinicialização importante
    while número <= 10:   # loop interno
        print("%d x %d = %d" % (tabuada, número, tabuada * número))
        número += 1
    tabuada += 1
```

A reinicialização da variável número dentro do loop externo é crucial para que cada tabuada comece do 1.

Alternativa Sem Repetições Aninhadas

```
tabuada = 1
número = 1
while tabuada <= 10:
    print("%d x %d = %d" % (tabuada, número, tabuada * número))
    número += 1
    if número == 11:
        número = 1
        tabuada += 1
```

Esta versão usa apenas um loop, mas requer lógica adicional para controlar quando mudar de tabuada. As repetições aninhadas tornam o código mais legível e organizado.

Exercícios Avançados

Menu de Operações

Crie um programa com menu de opções para diferentes operações matemáticas, exibindo a tabuada correspondente até que o usuário escolha sair.

Números Primos

Implemente verificação de números primos calculando o resto das divisões por 2 e todos os ímpares até o número lido.

Raiz Quadrada

Use o método de Newton para calcular raiz quadrada com aproximação, parando quando a diferença for menor que 0,0001.

Desafios Matemáticos

1

Divisão por Subtração

Calcule o resto da divisão inteira usando apenas operações de soma e subtração, sem usar o operador módulo.

2

Números Palíndromos

Verifique se um número permanece igual quando seus dígitos são invertidos (exemplos: 454, 10501).

Aplicações Práticas dos Acumuladores

1

Sistemas Financeiros

Cálculo de juros, prestações e saldos bancários

2

Processamento de Dados

Análise estatística e agregação de informações

3

Jogos e Simulações

Pontuação, recursos e progressão de personagens

Dicas para Usar Acumuladores

Inicialização Correta

Sempre inicialize acumuladores com zero (ou valor neutro) antes do loop para evitar resultados incorretos.

Posicionamento da Atualização

Coloque a operação de acumulação dentro do loop, mas cuidado com a ordem das operações.

Validação de Entrada

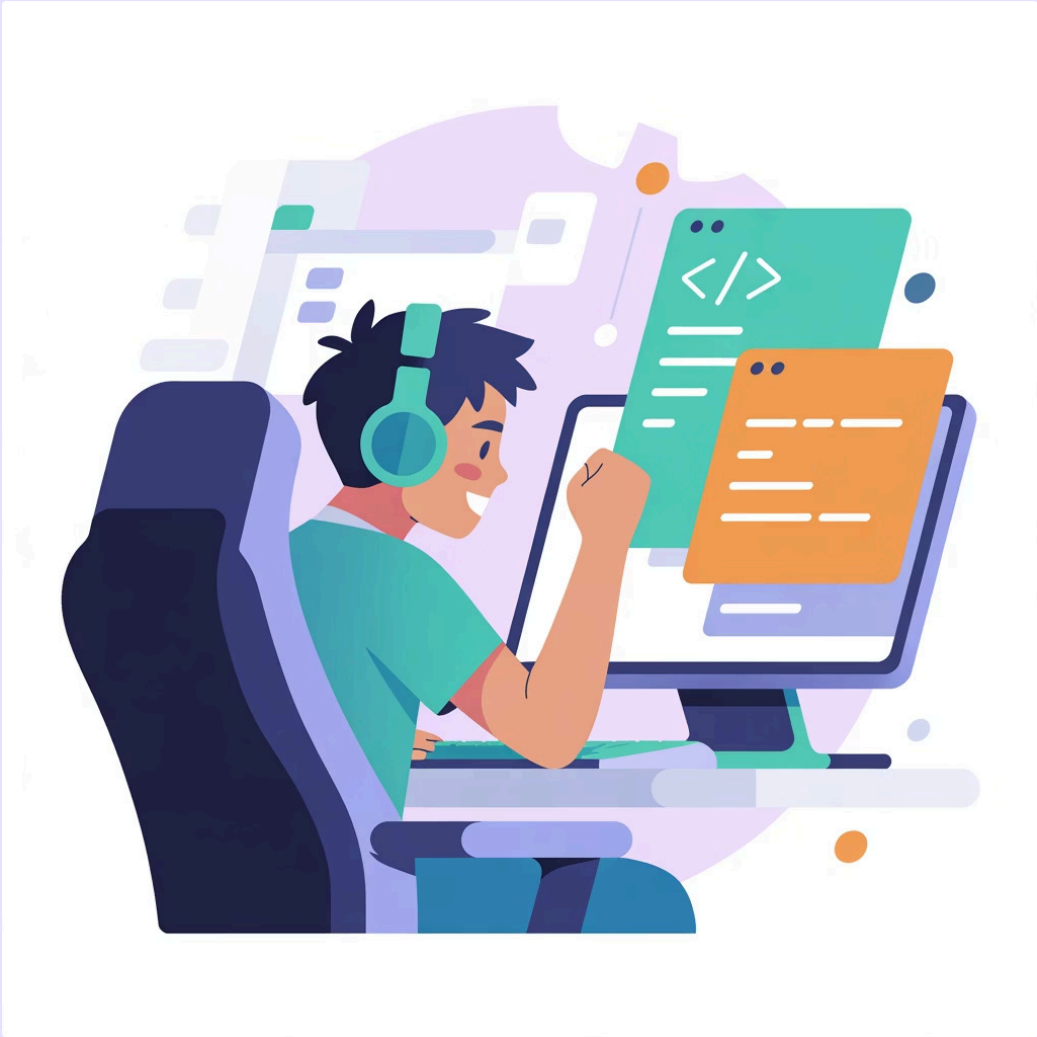
Sempre valide os dados do usuário antes de acumular para evitar erros ou comportamentos inesperados.

Próximos Passos

Continue Praticando

Os acumuladores são fundamentais em programação. Practice com os exercícios propostos para dominar completamente este conceito essencial.

- Implemente todos os exercícios do capítulo
- Experimente variações dos exemplos
- Crie seus próprios desafios



Resolver Exercícios

Próximo Capítulo