Educação Profissional Paulista

Técnico em **Ciência de Dados**

Secretaria da Educação

SÃO PAULO
GOVERNO DO ESTADO

Lógica de programação e algoritmos

Algoritmos de contagem e acumulação

Aplicações avançadas e otimizações

Código da aula: [DADOS]ANO1C3B4S27A3

Mapa da Unidade 1 Componente 4

Lógica de programação e algoritmos

semana **25**

Prática de busca e ordenação

semana **27**

Você está aqui!
Algoritmos de contagem e acumulação

Pilhas e filas

semana **28**

Lógica de programação e algoritmos

Mapa da Unidade 1 Componente 4

27

Você está aqui!

Algoritmos de contagem e acumulação

Aula 3

Código da aula: [DADOS]ANO1C3B4S27A3

Objetivos da Aula

- Introduzir os fundamentos dos Algoritmos de Contagem e Acumulação.

Recursos Didáticos

- Recurso audiovisual para exibição de vídeos e imagens;
- Acesso ao laboratório de informática e/ou internet.

Duração da Aula

50 minutos.

Competências Técnicas

- Identificar e resolver problemas relacionados a dados e análises;
- Compreender e dominar técnicas de manipulação de dados.

Competências Socioemocionais

- Adaptar-se a novas tecnologias, técnicas e tendências sem perder o foco, as metas e os objetivos da organização;
- Colaborar efetivamente com outros profissionais, como cientistas de dados e engenheiros de dados; trabalhar em equipes multifuncionais colaborando com colegas, gestores e clientes.

Construindo o **conceito**

Aplicações avançadas e otimizações

Combinação de contagem e acumulação

Algoritmos frequentemente combinam contagem e acumulação, como contar e somar valores simultaneamente.

Casos de uso em ciência de dados

Algoritmos de contagem e acumulação são amplamente usados em análise de dados, como calcular estatísticas descritivas, somar valores de vendas, ou contar ocorrências de eventos.

Otimização de algoritmos

Otimizar algoritmos envolve melhorar sua eficiência em termos de tempo de execução e uso de memória. Técnicas incluem minimizar o número de operações e evitar cálculos repetidos.

Técnicas de *Profiling* e *Debugging*

- *Profiling* é usado para medir o desempenho de um programa, identificando gargalos;
- *Debugging* é o processo de encontrar e corrigir erros no código.

Construindo o **conceito**

Exemplo 1: média de valores com contagem e acumulação

A função **media_valores** calcula a média dos valores em uma lista, combinando contagem e acumulação.

```
def media_valores(lista):
    total = 0
    contagem = 0
    for numero in lista:
        total += numero
        contagem += 1
    return total / contagem if contagem != 0 else 0

numeros = [10, 20, 30, 40, 50]
print(media_valores(numeros))
```

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

Construindo o **conceito**

Exemplo 2: otimização de contagem com dicionário

Esta função **contar_elementos** utiliza um dicionário para otimizar a contagem de elementos em uma lista.

```
def contar_elementos(lista):
    contagem = {}
    for elemento in lista:
        if elemento in contagem:
            contagem[elemento] += 1
        else:
            contagem[elemento] = 1
    return contagem

elementos = ['a', 'b', 'a', 'c', 'b', 'a']
print(contar_elementos(elementos))
```

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

Construindo o **conceito**

Exemplo 3: análise de dados com *profiling*

Neste exemplo, usamos **cProfile** para analisar o desempenho da função **analisar_dados**, identificando possíveis otimizações.

```
import cProfile

def analisar_dados(numeros):
    total = 0
    for numero in numeros:
        total += numero
    return total

numeros = [i for i in range(100000)]
cProfile.run('analisar_dados(numeros)')
```

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

Colocando em **prática**

Exercício: análise avançada de desempenho de vendas

Crie um programa em Python que analise detalhadamente o desempenho de vendas de uma empresa, combinando algoritmos de contagem e acumulação.

O programa deve contar a frequência de vendas de cada produto, calcular a receita total acumulada, e também determinar a média de vendas diárias para cada produto. Use técnicas de otimização para garantir a eficiência do código.

20 minutos

Duplas

Código de programação

Colocando em **prática**

Exercício: análise avançada de desempenho de vendas

20 min

Duplas

Código de programação

1 Crie dicionários para armazenar a contagem de vendas, a receita acumulada e a soma das vendas diárias de cada produto.

2 Para cada venda registrada, atualize os dicionários de contagem de vendas, receita acumulada e soma das vendas diárias.

3 Calcule a média de vendas diárias para cada produto.

4 No final, exiba a contagem de vendas, a receita total e a média de vendas diárias para cada produto.

Colocando em **prática**

Exercício: análise avançada de desempenho de vendas

Entrada:

Uma lista de tuplas, em que cada tupla contém o nome do produto, a quantidade vendida, o preço unitário e o dia da venda. Use a tabela abaixo no teu código:

```
vendas = [
    ("Produto A", 10, 5.0, 1),
    ("Produto B", 5, 12.0, 1),
    ("Produto A", 3, 5.0, 2),
    ("Produto C", 8, 7.5, 2),
    ("Produto B", 7, 12.0, 3)
]
```

Saída:

Exibir a contagem de vendas, a receita acumulada e a média de vendas diárias para cada produto.

20 minutos

Duplas

Código de programação

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

Ser
sempre +

Situação

Você trabalha em uma equipe de analistas de dados em uma empresa de comércio eletrônico.

Recentemente, a empresa lançou um novo produto, e os executivos estão ansiosos para ver os resultados das vendas.

Sua equipe está sobrecarregada com várias tarefas e prazos apertados.

Um dos membros da equipe, que é novo e ainda está aprendendo, cometeu um erro ao analisar os dados de vendas, resultando em um relatório incorreto enviado à gerência. Você percebeu o erro antes dos executivos, mas agora enfrenta a decisão de como lidar com a situação.

Situação fictícia elaborada especialmente para o curso.

Ser
sempre +

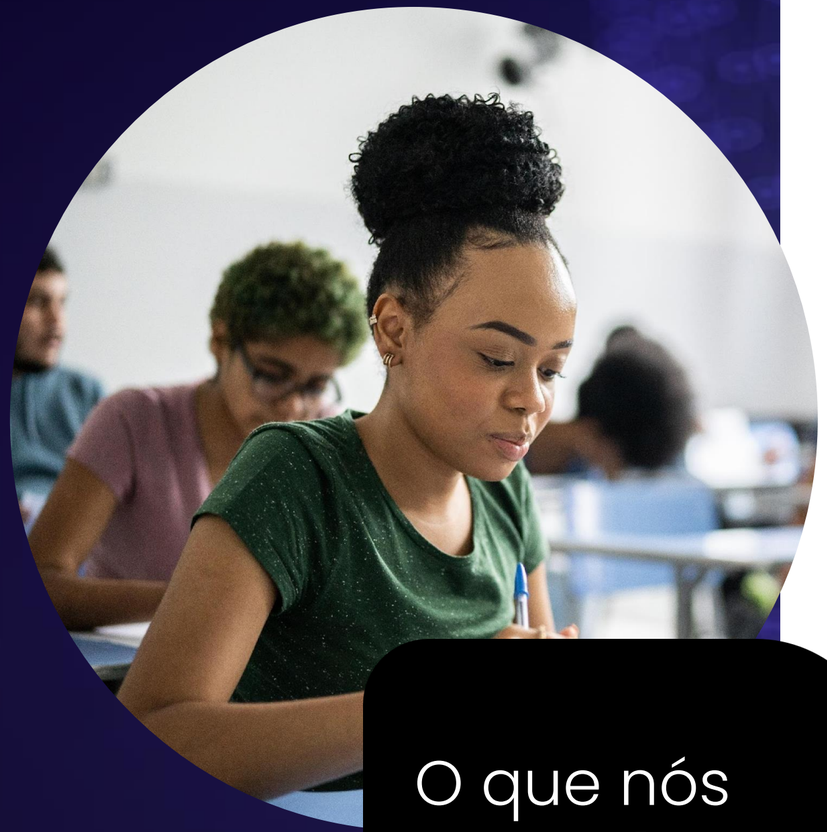
Ação

1. Como você abordaria o colega que cometeu o erro, considerando que ele é novo e ainda está aprendendo?

2. Qual seria a melhor maneira de comunicar o erro à gerência, garantindo que a confiança na equipe seja mantida?

3. Como você pode ajudar sua equipe a lidar com a sobrecarga de trabalho e prevenir futuros erros em situações de alta pressão?

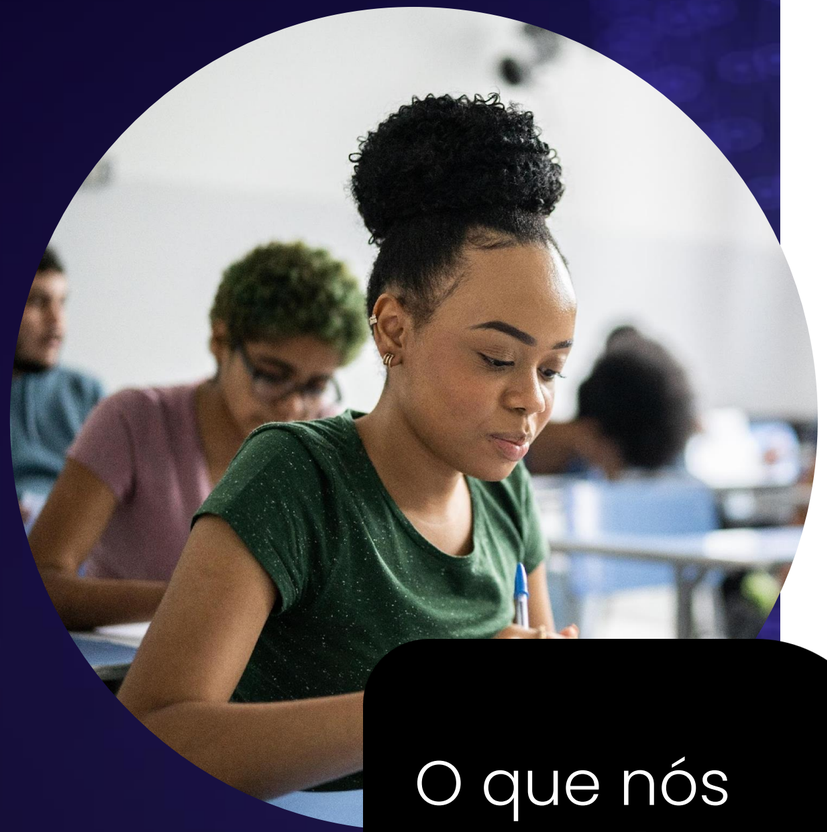
Diálogo



© Getty Images

O que nós **aprendemos hoje?**

Nuvem de palavras



© Getty Images

O que nós **aprendemos hoje?**

Então ficamos assim...

1. Exploramos a combinação de contagem e acumulação, mostrando como esses algoritmos podem ser utilizados simultaneamente para obter resultados mais completos e precisos em análises de dados;

2. Discutimos técnicas de otimização de algoritmos, focando na melhoria da eficiência em termos de tempo de execução e uso de memória. Minimizar operações e evitar cálculos repetidos foram pontos-chave;

3. Foram apresentados casos de uso em ciência de dados, como calcular estatísticas descritivas e somar valores de vendas. As técnicas de *profiling* e *debugging* foram explicadas para medir desempenho e corrigir erros no código.

Saiba mais

Quer organizar suas listas em Python de forma rápida e eficiente?

FELIPE, A. Ordenando listas no Python. Alura, 7 fev. 2017. Disponível em: https://www.alura.com.br/artigos/ordenando-listas-no-python/. Acesso em: 16 jul. 2024.

Referências da aula

Identidade visual: imagens © Getty Images

FORBELLONE, A. L. V.; EBERSPÄCHER, H. F. Lógica de programação: a construção de algoritmos e estruturas de dados com aplicações em Python. Porto Alegre: Bookman, 2022.

Educação Profissional Paulista

Técnico em **Ciência de Dados**

Secretaria da Educação

SÃO PAULO
GOVERNO DO ESTADO