Aula 3: Revisão e Projeto Prático

Bem-vindos à nossa jornada prática de programação!

Objetivos da Aula:

Revisão Essencial

Revisitar os principais fundamentos de programação que você já conhece.

Mãos à Obra com Dados

Praticar o uso de **listas, tuplas e dicionários** para organizar informações.

Ciclo de Dados

Focar em **armazenamento, manipulação e exibição** eficiente de dados.

Projeto Aplicado

Desenvolver um **mini-projeto de cadastro de alunos** para fixar o aprendizado.

Revisão de Conceitos Fundamentais: A Base da Programação

- **Variáveis e Tipos**: **int, float, str, bool**
- **Entrada e Saída**: **input() e print()**
- **Operadores**: **aritméticos, relacionais e lógicos**
- **Estruturas de Controle**: **if, elif, else**
- **Laços de Repetição**: **while, for**

Atividade Prática

Peça ao usuário para digitar seu nome, idade e altura. Em seguida, exiba essas informações com uma string formatada. Lembre-se de converter os tipos de dados conforme necessário!

Listas: Estruturas Dinâmicas para Seus Dados

O que são? Estruturas mutáveis que armazenam vários valores, ideais para coleções de itens que podem mudar ao longo do tempo.

- **Acesso e Manipulação**: Permite indexação e fatiamento para acessar ou modificar elementos.
- **Métodos Essenciais**:
  - **append()**: Adiciona elementos ao final.
  - **remove()**: Remove a primeira ocorrência de um elemento.
  - **sort()**: Ordena os elementos da lista.
  - **count()**: Conta ocorrências de um elemento.
- **Iteração**: Facilmente percorrida com laços **for e while**.

Atividade Prática

Crie uma lista vazia e adicione três itens de supermercado nela, utilizando métodos de lista e, se quiser, laços de repetição. Exiba a lista final!

Tuplas: Dados Constantes e Seguros

O que são? Estruturas imutáveis, ou seja, seus elementos não podem ser alterados após a criação. Perfeitas para dados que não devem ser modificados.

- **Sintaxe**: Simples e direta, como:

```
tupla = (1, "texto", True)
```

- **Uso Ideal**: Para dados fixos, como coordenadas geográficas, configurações, ou coleções de valores relacionados que não precisam de alteração.
- **Acesso**: Suporta indexação e fatiamento, assim como listas, para ler elementos.
- **Iteração**: Pode ser percorrida eficientemente com o laço `for`.

Atividade Prática

Crie uma tupla contendo três itens de sua escolha (ex: cores, frutas, números). Depois, mostre apenas o último item da tupla. Como você faria para acessar esse elemento?

Dicionários: Mapeando Chaves a Valores

O que são? Coleções de itens no formato **chave : valor**, permitindo associar e acessar dados de forma lógica, como em um catálogo ou registro.

- **Exemplo**: `aluno = {"nome": "Ana", "idade": 20, "curso": "Engenharia"}`
- **Métodos Chave**:
  - **.keys()**: Retorna todas as chaves do dicionário.
  - **.values()**: Retorna todos os valores do dicionário.
  - **.items()**: Retorna pares de chave e valor.
- **Ideal para**: Representar objetos complexos, cadastros de usuários, configurações e qualquer dado que precise de um identificador único (chave) para ser acessado.

Atividade Prática

Crie um dicionário vazio. Adicione uma chave "nome" com seu valor e uma chave "altura" com seu respectivo valor. Ao final, exiba o dicionário completo para o usuário.

Integração de Estruturas: Poder Combinado

As verdadeiras aplicações da programação muitas vezes envolvem a combinação dessas estruturas de dados. Aprenda a aninhar e manipular coleções para criar sistemas mais robustos.

Lista de Dicionários

Crie uma lista onde cada elemento é um dicionário, perfeito para gerenciar coleções de objetos, como uma lista de alunos ou produtos.

Dicionário com Tuplas

Use tuplas como valores dentro de um dicionário, garantindo que certos dados associados às chaves permaneçam imutáveis.

Iteração Aninhada

Percorra coleções aninhadas usando múltiplos laços `for` para acessar e processar dados complexos.

Exemplo Prático: **alunos = [{"nome": "Ana", "notas": (7, 8, 9)}, {"nome": "Pedro", "notas": (6, 5, 7)}]**

Mini-Projeto: Cadastro de Alunos

O que será desenvolvido?

Um sistema de cadastro simples, mas funcional, que armazena informações cruciais sobre seus alunos:

- **Nome do aluno**
- **Notas** (utilizando a estrutura de listas).

Funcionalidades Principais

- **Adicionar alunos**: Inserir novos registros de forma organizada.
- **Remover alunos**: Manter o cadastro atualizado, removendo entradas existentes.
- **Calcular média**: Obter a média das notas para cada aluno.
- **Verificar situação**: Determinar se o aluno está aprovado ou reprovado com base na média (>7 aprovado, <7 reprovado).
- **Exibir lista**: Visualizar todos os alunos cadastrados e suas informações.

Este projeto unirá todos os conceitos vistos, desde variáveis até o uso integrado de listas e dicionários. Prepare-se para codificar!