Aula 6: Lista, Tuplas, Sets e Dicionários

Por que precisamos de Estruturas de Dados?

Se você precisar armazenar **todos os nomes dos seus colegas de turma** (18 pessoas) em uma única variável chamada `turma`. Você criaria `nome1`, `nome2`,... até `nome18` ?

Na vida real e na programação, lidamos com **coleções** (listas de compras, registros de clientes, cardápios, pautas de aula). Precisamos de variáveis que aguentem múltiplas informações e permitam que as manipulemos facilmente.

Em Python, Listas, Tuplas, Sets e Dicionários são essas estruturas. A principal diferença entre elas está em:

- **Mutabilidade:** Posso mudar o conteúdo depois de criado?
- **Ordem:** A ordem dos elementos é mantida?
- **Duplicatas:** Posso ter o mesmo valor repetido?

Listas

É a estrutura mais flexível e comum.

Propriedade	Lista
Mutável?	SIM
Ordenada?	SIM
Aceita Duplicatas?	SIM
Sintaxe:	`[ ]` Colchetes

```
# CRIANDO UMA LISTA
frutas = ['Maçã', 'Banana', 'Morango', 'Maçã', 'Pera']

print(f"Lista Original: {frutas}")
```

```
# 1. ACESSO (Sempre por índice, começando em 0)
primeira_fruta = frutas[0]
ultima_fruta = frutas[-1]

print(f"Primeira fruta: {primeira_fruta}")

# 2. INCLUSÃO
frutas.append('Kiwi') # Adiciona no FINAL da lista
frutas.insert(1, 'Laranja') # Adiciona na POSIÇÃO 1

print(f"Lista após inclusão: {frutas}")

# 3. ALTERAÇÃO (Mutabilidade)
frutas[0] = 'Manga' # Troca 'Maçã' por 'Manga'

print(f"Lista após alteração: {frutas}")

# 4. REMOÇÃO
frutas.remove('Morango') # Remove a PRIMEIRA ocorrência do valor
frutas.pop(3)          # Remove o item da POSIÇÃO 3 (removendo 'Maçã')

print(f"Lista após remoção: {frutas}")
```

Tuplas

Funciona como uma lista, mas é "travada" (imutável). Usamos para dados que não devem ser alterados (como coordenadas geográficas, dados de um registro que não mudam).

Propriedade	**Tupla**
Mutável?	NÃO
Ordenada?	SIM
Aceita Duplicatas?	SIM
Sintaxe:	`( )` Parênteses

```
# CRIANDO UMA TUPLA
coordenadas = (40.71, -74.00, 10) # Latitude, Longitude, Altitude
```

```
estados = ('RJ', 'SP', 'MG')

print(f"Tupla de Coordenadas: {coordenadas}")

# 1. ACESSO (Funciona igual à lista)
latitude = coordenadas[0]
print(f"Latitude: {latitude}")

# 2. TENTATIVA DE ALTERAÇÃO
coordenadas[2] = 15 #TypeError

# 3. INCLUSÃO/REMOÇÃO
# Não é possível. Para "alterar" uma tupla, você deve criar uma nova tupla ou fazer uma cópia mutável dessa tupla.
tupla_nova = estados + ('BA',) # Cria uma tupla nova a partir da antiga.

print(f"Nova Tupla: {tupla_nova}")
```

Sets

O **Set (Conjunto)** elimina duplicatas e verifica presença de itens rapidamente. A ordem dos itens não é garantida, nesse caso.

Propriedade	**Set**
Mutável?	SIM
Ordenada?	NÃO
Aceita Duplicatas?	NÃO (principal)
Sintaxe:	`{ }` Chaves (sem pares)

```
# CRIANDO UM SET COM DUPLICATAS INTENCIONAIS
numeros = {10, 20, 30, 10, 40, 30} # Note os 10 e 30 repetidos

print(f"Set Original: {numeros}") # Saída mostrará apenas valores únicos e sem ordem garantida

# 1. ACESSO
print(numeros[0]) #TypeError
```

```
# 2. INCLUSÃO (Tenta adicionar um valor, mas ignora se já existe)
numeros.add(50)
numeros.add(10) # Será ignorado, pois 10 já existe

print(f"Set após adicionar 50: {numeros}")

# 3. VERIFICAÇÃO RÁPIDA (Decisão)
if 20 in numeros:
    print("O número 20 está no conjunto.")

# 4. REMOÇÃO
numeros.remove(40)
print(f"Set após remover 40: {numeros}")
```

Dicionários

É a forma mais útil de estruturar dados complexos, armazenando informações em pares **Chave: Valor** (como um índice de telefone ou um registro de cliente).

Propriedade	**Dicionário**
Mutável?	SIM
Ordenada?	A partir do Python 3.7 ou superior (sim, acessível pela chave)
Aceita Duplicatas?	Não, as **Chaves** devem ser **únicas**.
Sintaxe:	`{chave: valor}` Chaves com pares

```
# CRIANDO UM DICIONÁRIO (Registro de um livro)
livro = {
    'titulo': 'Dom Quixote',
    'autor': 'Cervantes',
    'ano': 1605,
    'preço': 27.90
}

print(f"Dicionário Original: {livro}")
```

```
# 1. ACESSO
titulo_livro = livro['titulo']
print(f"O livro é: {titulo_livro}")

# 2. INCLUSÃO (Basta definir uma nova chave)
livro['editora'] = 'L&PM'

# 3. ALTERAÇÃO
livro['preço'] = 50.00 # Altera o valor da chave 'preço'

print(f"Dicionário após alteração: {livro}")

# 4. REMOÇÃO
livro.pop('ano') # Remove o par Chave: Valor baseado na Chave

print(f"Dicionário após remover 'ano': {livro}")
```

Atividade Assistida

Vamos desenvolver um programa que leia 5 nomes de filmes e armazene nas quatro formas, mostrando as diferenças na prática.

Coleta dos Dados:

```
dados_entrada = []
print("Digite 5 nomes de filmes:")
for i in range(5):
    nome = input(f"Filme {i+1}: ")
    dados_entrada.append(nome) # Usando LISTA para coletar os dados

print("-" * 30)
```

Armazenamento e Exibição:

```
# 1. LISTA:
lista_filmes = dados_entrada

# 2. TUPLA:
tupla_filmes = tuple(dados_entrada)

# 3. SET:
set_filmes = set(dados_entrada)

# 4. DICIONÁRIO: Usando o índice como chave (ex: 1: 'Filme A')
dicionario_filmes = {}
for i in range(len(dados_entrada)): # Repetição pelo tamanho da lista
    # Usando o índice (i+1) como CHAVE e o nome como VALOR
```

```
    dicionario_filmes[i + 1] = dados_entrada[i]

print(f"LISTA (Flexível): {lista_filmes}")
print(f"TUPLA (Fixa): {tupla_filmes}")
print(f"SET (Apenas Únicos): {set_filmes}")
print(f"DICIONÁRIO (Chave: Valor): {dicionario_filmes}")
```

Desafios (ugprade):

Média Escolar para 5 Estudantes

Use um **for** loop para iterar 5 vezes. Dentro do loop, realize a leitura das notas e a decisão (if/elif/else) da média. Crie uma lista vazia (resultados = []). A cada repetição, adicione uma string (ex: "Aluno 1 - Aprovado") a esta lista usando .append().

Cadastro Seletivo de Candidatos

Use um **for** loop para iterar 5 vezes. Dentro do loop, use um **if/else** para checar se o candidato é menor de 18 anos (rejeição). Crie uma lista principal: candidatos_validos = []. Se o candidato for **válido**, crie um **Dicionário** (ex: candidato = {'nome': '...', 'email': '...'}). Adicione este Dicionário à lista: candidatos_validos.append(candidato).