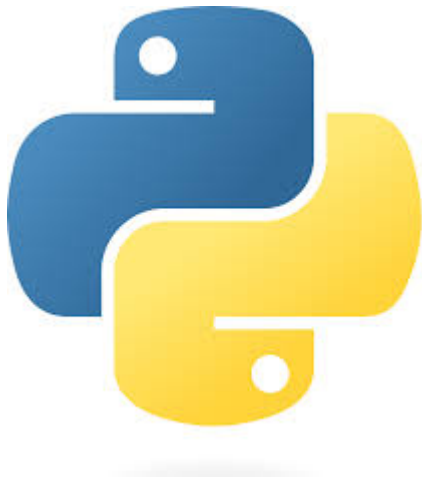


Nome: Ruan Gabriel O. Santos

Turma: N2

Disciplina: ADS_Tecnicas de desenvolvimento de algoritmos



✓ Aplicações com Lista, Tuplas e Dicionários

```
1 # MINI PROJETO 1: Cadastro Simplificado de Alunos (Lista de Dicionários)
2 # Objetivo: Gerenciar informações de alunos, simulando operações CRUD (C
3 # [cite_start]Estrutura: Lista (alunos_db) de Dicionários (cada aluno).
4
5 # Base de dados inicial (Lista de Dicionários)
6 alunos_db = [
7     {"matricula": "2023001", "nome": "Ana Silva", "curso": "Sistemas de
8     {"matricula": "2023002", "nome": "João Gomes", "curso": "Engenharia
9     {"matricula": "2023003", "nome": "Maria Souza", "curso": "Ciência de
10 ]
11
12 print("1. Sistema Simplificado de Cadastro de Alunos")
13 print("-" * 40)
14
15 # 1. Adicionar Novo Aluno (CREATE)
16 print("- Adicionando novo aluno...")
17 novo_aluno = {"matricula": "2023004", "nome": "Pedro Costa", "curso": "R
18 alunos_db.append(novo_aluno)
19 print(f"Aluno {novo_aluno['nome']} adicionado com sucesso.")
20
21 # 2. Buscar Aluno por Matrícula (READ)
22 matricula_busca = "2023002"
23 # [cite_start]Uso de compreensão de lista para buscar o dicionário do al
```

```

24 aluno_encontrado = next((aluno for aluno in alunos_db if aluno["matricul
25
26 if aluno_encontrado:
27     print(f"\nAluno encontrado (Matrícula: {matricula_busca}):")
28     # [cite_start]Acessa os valores do dicionário usando suas chaves. [c
29     print(f"Nome: {aluno_encontrado['nome']}, Curso: {aluno_encontrado['
30 else:
31     print(f"\nAluno com matrícula {matricula_busca} não encontrado.")
32
33 # 3. Atualizar Informação do Curso (UPDATE)
34 matricula_update = "2023003"
35 for aluno in alunos_db:
36     if aluno["matricula"] == matricula_update:
37         # [cite_start]Dicionários são mutáveis, permite alterar o valor
38         aluno["curso"] = "Segurança da Informação"
39         print(f"\nCurso do aluno {aluno['nome']} atualizado para: {aluno
40         break
41
42 print("-" * 40)
43 # Visualiza a lista completa após as operações
44 print(alunos_db)

```

```

1 # MINI PROJETO 2: Gerenciamento de Coordenadas (Lista de Tuplas)
2 # [cite_start]Objetivo: Armazenar sequências de valores Fixos (latitude,
3 # [cite_start]Estrutura: Lista (locais) de Tuplas (cada coordenada), ond
4
5 # Base de dados inicial (Lista de Tuplas: (latitude, longitude, nome_do_
6 locais = [
7     (-23.55052, -46.63330, "São Paulo"),
8     (-15.7801, -47.9292, "Brasília"),
9     (40.7128, -74.0060, "Nova York")
10 ]
11
12 print("\n2. Gerenciamento de Coordenadas Geográficas")
13 print("-" * 40)
14
15 # 1. Adicionar Nova Coordenada
16 print("-> Adicionando nova coordenada (Tupla)...")
17 # [cite_start]Cria uma nova tupla para garantir que a coordenada seja re
18 novo_local = (34.0522, -118.2437, "Los Angeles")
19 # [cite_start]A lista é mutável, permitindo adicionar a nova tupla. [cit
20 locais.append(novo_local)
21 print(f"Local {novo_local[2]} adicionado.")
22
23 # 2. Acessar e Desempacotar (Unpack) uma Tupla
24 print("\n-> Desempacotando a última coordenada...")
25 # [cite_start]Desempacota os elementos da última tupla da lista em variá
26 lat, lon, nome = locais[-1]
27 print(f"Latitude: {lat}, Longitude: {lon}, Local: {nome}")
28
29 # 3. Iterar e Formatar Coordenadas
30 print("\n-> Coordenadas registradas:")
31 for coordenada in locais:
32     # Acessa os elementos da tupla usando índices.

```

```

33     print(f"Local: {coordenada[2]} | Lat/Lon: ({coordenada[0]}, {coorden
34
35 # [cite_start]Importante: Tuplas são imutáveis; tentar alterar um elemen
36 print("-" * 40)

```

```

1 # MINI PROJETO 3: Catálogo de Produtos com Busca Rápida (Dicionário)
2 # [cite_start]Objetivo: Mapear um ID de produto (chave) para os detalhes
3 # [cite_start]Estrutura: Dicionário (catalogo), onde a chave é o ID do p
4
5 # Catálogo de produtos (Dicionário aninhado)
6 catalogo = {
7     "P001": {"nome": "Teclado Mecânico", "estoque": 25, "preco": 350.00}
8     "P002": {"nome": "Monitor LED 27", "estoque": 18, "preco": 1200.00},
9     "P003": {"nome": "Mouse Gamer", "estoque": 40, "preco": 150.00}
10 }
11
12 print("\n3. Catálogo de Produtos com Busca Rápida")
13 print("-" * 40)
14
15 # 1. Adicionar Novo Produto (CREATE)
16 novo_produto_id = "P004"
17 # [cite_start]Adiciona um novo par chave:valor. [cite: 494]
18 catalogo[novo_produto_id] = {"nome": "Webcam HD", "estoque": 50, "preco":
19 print(f"Produto {catalogo[novo_produto_id]['nome']} adicionado ao catálo
20
21 # 2. Buscar por Chave (Busca O(1) Rápida) (READ)
22 id_busca = "P002"
23 # [cite_start]Acessar diretamente pelo ID (chave) é a forma mais rápida
24 if id_busca in catalogo:
25     produto_detalhes = catalogo[id_busca]
26     print(f"\nDetalhes do Produto {id_busca}:")
27     print(f"Nome: {produto_detalhes['nome']}, Preço: R$ {produto_detalhe
28 else:
29     print(f"\nProduto com ID {id_busca} não encontrado.")
30
31 # 3. Atualizar Estoque (UPDATE)
32 id_update = "P001"
33 # [cite_start]Acessa o dicionário interno pela chave e atualiza o valor.
34 catalogo[id_update]["estoque"] = 5
35 print(f"\nEstoque do {catalogo[id_update]['nome']} atualizado. Novo esto
36
37 print("-" * 40)

```

```

1 # MINI PROJETO 4: Agrupamento de Vendas por Região (Dicionários Aninhado)
2 # [cite_start]Objetivo: Agrupar dados de vendas (Lista de Tuplas) por um
3 # [cite_start]Estrutura: Dicionário (dados_vendas) onde a chave é a regi
4
5 # Dados de vendas (Dicionário Aninhado)
6 dados_vendas = {
7     "SUDESTE": {
8         "total_anual": 150000.00,
9         # [cite_start]Lista de Tuplas: Cada tupla (imutável) é um regist
10     "vendas_detalhe": [

```

```

11         ("V001", 5000.00),
12         ("V002", 12000.00),
13         ("V003", 8000.00)
14     ]
15 },
16 "SUL": {
17     "total_anual": 95000.00,
18     "vendas_detalhe": [
19         ("V004", 5500.00),
20         ("V005", 10000.00)
21     ]
22 }
23 }
24
25 print("\n4. Agrupamento de Vendas por Região")
26 print("-" * 40)
27
28 # 1. Adicionar Nova Venda à Região SUDESTE
29 regioao = "SUDESTE"
30 nova_venda = ("V006", 9500.00)
31
32 # [cite_start]Acessa o dicionário 'SUDESTE', depois a chave 'vendas_deta
33 dados_vendas[regiao]["vendas_detalhe"].append(nova_venda)
34 print(f"Nova venda {nova_venda[0]} adicionada na região {regiao}.")
35
36 # 2. Calcular o Valor Total de Vendas Detalhadas no SUDESTE
37 vendas_sudeste = dados_vendas["SUDESTE"]["vendas_detalhe"]
38 # [cite_start]Usa compreensão de lista e a função sum() para somar o seg
39 soma_detalhes = sum(venda[1] for venda in vendas_sudeste)
40
41 print(f"\nSoma dos detalhes de vendas SUDESTE: R$ {soma_detalhes:.2f}")
42 print(f"Total anual registrado: R$ {dados_vendas[regiao]['total_anual']}:
43
44 # 3. Listar as Regiões (Chaves)
45 print("\nRegiões em análise:")
46 # [cite_start]O método .keys() retorna as chaves do dicionário principal
47 for reg in dados_vendas.keys():
48     print(f"- {reg}")
49
50 print("-" * 40)

```

```

1 # MINI PROJETO 5: Mapeamento de Níveis de Acesso (Dicionário com Tuplas)
2 # [cite_start]Objetivo: Definir conjuntos de permissões fixas por nível
3 # [cite_start]Estrutura: Dicionário (permissoes) mapeia o nível de acess
4
5 # Mapeamento de Níveis e Permissões
6 permissoes = {
7     # [cite_start]Tupla de permissões para Admin [cite: 622]
8     "ADMIN": ("CREATE_USER", "READ_ALL", "UPDATE_ANY", "DELETE_ALL"),
9     # [cite_start]Tupla de permissões para Manager [cite: 623]
10    "MANAGER": ("READ_ALL", "UPDATE_OWN"),
11    # [cite_start]Tupla de um elemento para Viewer (vírgula obrigatória)
12    "VIEWER": ("READ_ALL",)
13 }

```

```
14
15 print("\n5. Mapeamento de Níveis de Acesso e Permissões")
16 print("-" * 40)
17
18 # 1. Acessar Permissões de um Nível Específico (READ)
19 nivel_acesso = "ADMIN"
20 # [cite_start]Acessa a tupla de permissões usando a chave do dicionário.
21 perms_admin = permissoes.get(nivel_acesso)
22
23 if perms_admin:
24     print(f"Permissões do nível {nivel_acesso}:")
25     # [cite_start]Itera sobre a tupla de permissões. [cite: 634]
26     for perm in perms_admin:
27         print(f"- {perm}")
28 else:
29     print(f"Nível de acesso '{nivel_acesso}' não definido.")
30
31 # 2. Verificar uma Permissão Específica
32 perm_verificar = "DELETE_ALL"
33 # [cite_start]Usa o operador 'in' para checar se a permissão existe dent
34 if perm_verificar in permissoes["ADMIN"]:
35     print(f"\nADMIN possui a permissão: {perm_verificar}")
36 else:
37     print(f"\nADMIN NÃO possui a permissão: {perm_verificar}")
38
39 # 3. Tentativa de Adicionar uma Permissão (Imutabilidade da Tupla)
40 # print("Tentando adicionar permissão ao VIEWER...")
41 # [cite_start]permissoes["VIEWER"].append("DELETE_OWN") # Erro: 'tuple'
42 # [cite_start]A tupla garante a integridade do conjunto de permissões. [
43 print("\nA Integridade do conjunto de permissões é garantida pela Tupla
44 print("-" * 40)
```