FAP / UFRN – Softex 2024 – Listas e dicionários – Junho de 2024

Prof. José Alfredo F. Costa

**1. Listas:**

* **Criando listas:**
* L = [] # Lista vazia

Z = [15, 8, 9] # Lista com 3 elementos

* **Acessando elementos:**

Z[0] # Acessa o primeiro elemento (índice 0)

* **Modificando elementos:**

Z[0] = 7 # Altera o primeiro elemento para 7

* **Operações com listas:**
  + **Tamanho:** len(L) retorna o número de elementos.
  + **Adição:**
    - append(valor): Adiciona um elemento ao final.
    - extend(outra\_lista): Adiciona os elementos de outra lista ao final.
    - L + [1]: Concatena uma lista com outro elemento ou lista.
  + **Remoção:** del L[1]: Remove o elemento de índice 1.
  + **Fatiamento:** L[1:3] retorna uma nova lista com elementos do índice 1 até (excluindo) o índice 3.
  + **Cópia:** V = L[:] cria uma nova lista com os mesmos elementos de L.
* **Utilizando listas como filas:**
  + **Inclusão:** fila.append(valor) (acrescenta ao final).
  + **Remoção:** fila.pop(0) (remove o primeiro elemento).
* **Utilizando listas como pilhas:**
  + **Inclusão:** pilha.append(valor) (acrescenta ao topo).
  + **Remoção:** pilha.pop(-1) (remove o último elemento).
* **Pesquisa sequencial:**

for x, e in enumerate(L):

if e == valor:

return x # Retorna o índice se encontrado

return None # Retorna None se não encontrado

* **Ordenação com o método "Bubble Sort":**

fim = len(L)

while fim > 1:

trocou = False

for x in range(fim - 1):

if L[x] > L[x+1]:

trocou = True

temp = L[x]

L[x] = L[x+1]

L[x+1] = temp

if not trocou:

break

fim -= 1

* **Usando a estrutura for para percorrer listas:**

for e in L: # Percorre cada elemento da lista

print(e)

* **Usando a função range para gerar sequências:**

for v in range(10): # Gera números de 0 a 9 (10 exclusivo)

print(v)

* **Usando a função enumerate para obter índice e valor simultaneamente:**
* for x, e in enumerate(L):

print("[%d] %d" % (x,e))

**2. Dicionários:**

* **Criando dicionários:**

tabela = {"Alface": 0.45, "Batata": 1.20}

* **Acessando valores:**

tabela["Alface"] # Retorna o valor associado à chave "Alface"

* **Modificando valores:**

tabela["Tomate"] = 2.50 # Altera ou adiciona o valor da chave "Tomate"

* **Verificando a existência de uma chave:**

"Manga" in tabela # Retorna True se "Manga" existir no dicionário

* **Obtendo chaves e valores:**
* tabela.keys() # Retorna um gerador com as chaves do dicionário

tabela.values() # Retorna um gerador com os valores do dicionário

* **Removendo uma chave:**

del tabela["Tomate"] # Remove a chave "Tomate" e seu valor

**3. Tuplas:**

* **Criando tuplas:**
* tupla = ("a", "b", "c")

tupla = 100, 200, 300

* **Acessando elementos:** Funciona como listas, utilizando índices.
* **Tuplas são imutáveis:** Não é possível modificar seus elementos.
* **Empacotamento e Desempacotamento:**
  + Empacotamento: tupla = 100, 200, 300 cria uma tupla.
  + Desempacotamento: a, b = 10, 20 atribui os elementos da tupla a variáveis.

O capítulo 6 introduz conceitos importantes para trabalhar com estruturas de dados em Python, sendo fundamental para o desenvolvimento de programas mais complexos.

***Exemplo de dicionário usando registros de alunos, com nomes, curso, matrícula, e outras informações. Podemos ter uma lista onde cada elemento é um dicionário (um registro) - um aluno***

Exemplo de Dicionário para Registros de Alunos

aluno1 = {

"nome": "João da Silva",

"curso": "Ciência da Computação",

"matricula": 2023001,

"notas": [8.5, 7.0, 9.0],

"presencas": 18,

"telefone": "1234-5678",

"email": "joao.silva@email.com"

}

aluno2 = {

"nome": "Maria Oliveira",

"curso": "Engenharia Civil",

"matricula": 2023002,

"notas": [9.5, 8.0, 10.0],

"presencas": 20,

"telefone": "9876-5432",

"email": "maria.oliveira@email.com"

}

aluno3 = {

"nome": "Pedro Santos",

"curso": "Administração",

"matricula": 2023003,

"notas": [6.0, 7.5, 8.5],

"presencas": 15,

"telefone": "5555-1111",

"email": "pedro.santos@email.com"

}

Neste exemplo:

* Cada dicionário aluno1, aluno2 e aluno3 representa um registro de um aluno.
* As chaves (nome, curso, matricula, etc.) representam as informações do aluno.
* Os valores associados a cada chave são os dados específicos de cada aluno.
* A chave notas armazena uma lista de notas, mostrando que um dicionário pode conter outros tipos de dados (neste caso, uma lista).

Lista de Dicionários (Alunos)

alunos = [aluno1, aluno2, aluno3]

Alunos é uma lista onde cada elemento é um dicionário, representando um aluno.

Isso permite organizar e manipular os dados de todos os alunos de forma eficiente.

Usando a Lista de Alunos

for aluno in alunos:

print("Nome:", aluno["nome"])

print("Curso:", aluno["curso"])

print("Matrícula:", aluno["matricula"])

print("---------------------")

Este código itera sobre a lista alunos, acessando os dados de cada dicionário (aluno) e imprimindo as informações relevantes.

Vantagens de usar Dicionários e Listas para Registros

* **Organização:** Os dados são estruturados de forma lógica e fácil de entender.
* **Flexibilidade:** Permite adicionar ou remover informações dos registros facilmente.
* **Eficiência:** As listas e dicionários do Python são otimizadas para acesso e manipulação de dados.

Você pode usar essa estrutura para criar sistemas que gerenciam informações de alunos, como um sistema de notas, um sistema de matrícula, etc.

***Poderíamos ter uma espécie de formulário para entrar os dados, e cada vez que eu inserisse, ele acrescentasse na lista dos alunos:***

alunos = [] # Lista para armazenar os alunos

while True:

print("\nMenu de Cadastro de Alunos:")

print("[1] Cadastrar novo aluno")

print("[2] Listar alunos cadastrados")

print("[0] Sair")

opcao = input("Escolha uma opção: ")

if opcao == "1":

# Cadastrar novo aluno

novo\_aluno = {} # Cria um dicionário vazio para o novo aluno

novo\_aluno["nome"] = input("Nome do aluno: ")

novo\_aluno["curso"] = input("Curso: ")

novo\_aluno["matricula"] = int(input("Matrícula: ")) # Convertendo para inteiro

# Adicionar outras informações se necessário...

alunos.append(novo\_aluno) # Adiciona o novo aluno à lista

print("Aluno cadastrado com sucesso!")

elif opcao == "2":

# Listar alunos cadastrados

if len(alunos) == 0:

print("Ainda não há alunos cadastrados.")

else:

for aluno in alunos:

print("Nome:", aluno["nome"])

print("Curso:", aluno["curso"])

print("Matrícula:", aluno["matricula"])

print("---------------------")

elif opcao == "0":

print("Saindo do programa...")

break

else:

print("Opção inválida. Por favor, escolha uma opção válida.")

**Explicação passo-a-passo:**

1. **Inicialização da lista:** alunos = [] cria uma lista vazia para armazenar os dados dos alunos.
2. **Loop principal:** Um loop while True executa o menu de opções indefinidamente até o usuário escolher sair.
3. **Menu de opções:** Imprime as opções para o usuário: cadastrar um novo aluno, listar os alunos cadastrados ou sair do programa.
4. **Entrada do usuário:** opcao = input("Escolha uma opção: ") recebe a opção do usuário como texto.
5. **Verificação da opção:**
   * **Opção 1 (Cadastrar novo aluno):**
     + Cria um dicionário vazio novo\_aluno = {} para armazenar os dados do novo aluno.
     + Solicita as informações do aluno usando input().
     + Converte a matrícula para inteiro usando int(input()).
     + Adiciona o dicionário novo\_aluno à lista alunos usando alunos.append(novo\_aluno).
     + Imprime uma mensagem de confirmação.
   * **Opção 2 (Listar alunos):**
     + Verifica se a lista está vazia usando len(alunos) == 0.
     + Se a lista estiver vazia, imprime uma mensagem informando que não há alunos cadastrados.
     + Se a lista não estiver vazia, itera sobre a lista alunos, acessando os dados de cada dicionário (aluno) e imprimindo as informações relevantes.
   * **Opção 0 (Sair):** Imprime uma mensagem de saída e encerra o loop while.
   * **Opção inválida:** Imprime uma mensagem de erro.

**Como usar:**

1. Execute o código.
2. O menu de opções será exibido.
3. Escolha a opção "1" para cadastrar um novo aluno.
4. Digite as informações do aluno solicitadas.
5. O aluno será cadastrado com sucesso.
6. Você pode continuar a cadastrar mais alunos ou escolher a opção "2" para listar os alunos cadastrados.
7. Escolha a opção "0" para sair do programa.

**Pontos importantes:**

* **Entrada de dados:** O código usa input() para receber informações do usuário.
* **Conversão de tipos:** int(input()) é usado para converter a matrícula, que é um número, para o tipo int.
* **Adicionar outras informações:** Você pode adicionar mais campos ao dicionário novo\_aluno para incluir outras informações como data de nascimento, endereço, etc.
* **Organização:** A estrutura usando dicionários e listas é eficiente para armazenar e gerenciar dados de alunos.

Este é um exemplo básico, mas você pode expandi-lo para criar um sistema de gerenciamento de alunos mais completo com funcionalidades como:

* **Atualizar dados de alunos:** Editar as informações de um aluno existente.
* **Remover alunos:** Excluir um aluno da lista.
* **Pesquisar alunos:** Encontrar alunos por nome, matrícula, curso, etc.
* **Calcular média:** Calcular a média das notas de um aluno.
* **Salvar e carregar dados:** Salvar a lista de alunos em um arquivo e carregar os dados de um arquivo.

Exercício de Programação em Python: Sistema de Gerenciamento de Alunos

**Objetivo:** Criar um sistema básico de gerenciamento de alunos em Python, utilizando listas e dicionários para armazenar e manipular os dados dos alunos.

**Requisitos:**

1. **Menu Interativo:** Implemente um menu com as seguintes opções:
   * **Cadastrar Novo Aluno:**
     + Permitir que o usuário digite o nome, curso, matrícula e outras informações relevantes do aluno.
     + Armazenar os dados do aluno em um dicionário.
     + Adicionar o dicionário do aluno a uma lista que contém todos os alunos cadastrados.
   * **Listar Alunos Cadastrados:**
     + Exibir os dados de todos os alunos cadastrados em um formato organizado, incluindo nome, curso e matrícula.
   * **Sair:** Encerrar o programa.
2. **Validação de Dados:**
   * **Matrícula:** Verifique se a matrícula informada é um número inteiro e se não está duplicada na lista de alunos.
   * **Nome e Curso:** Verifique se o nome e o curso não estão vazios.
   * Imprima mensagens de erro claras para o usuário em caso de dados inválidos.

**Dicas:**

* Utilize uma lista para armazenar os dicionários que representam os alunos.
* Use a função input() para receber dados do usuário.
* Utilize a função int() para converter a matrícula para o tipo inteiro.
* Use a estrutura for para iterar sobre a lista de alunos e exibir seus dados.

**Exemplo de Execução:**

Menu de Cadastro de Alunos:

[1] Cadastrar novo aluno

[2] Listar alunos cadastrados

[0] Sair

Escolha uma opção: 1

Nome do aluno: João da Silva

Curso: Ciência da Computação

Matrícula: 2023001

Aluno cadastrado com sucesso!

Menu de Cadastro de Alunos:

[1] Cadastrar novo aluno

[2] Listar alunos cadastrados

[0] Sair

Escolha uma opção: 2

Nome: João da Silva

Curso: Ciência da Computação

Matrícula: 2023001

---------------------

Menu de Cadastro de Alunos:

[1] Cadastrar novo aluno

[2] Listar alunos cadastrados

[0] Sair

Escolha uma opção: 0

Saindo do programa...

**Desafios (Opcionais):**

* Implemente funcionalidades adicionais, como:
  + **Atualizar Dados:** Editar as informações de um aluno existente.
  + **Remover Aluno:** Excluir um aluno da lista.
  + **Pesquisar Aluno:** Encontrar alunos por nome, matrícula ou curso.
  + **Calcular Média:** Calcular a média das notas de um aluno.
  + **Salvar e Carregar Dados:** Salvar a lista de alunos em um arquivo e carregar os dados de um arquivo.

**Entrega:**

* Submeta o código fonte do seu programa.

**Objetivo do Exercício:**

Este exercício visa consolidar o aprendizado sobre listas e dicionários em Python, além de introduzir conceitos básicos de interação com o usuário e validação de dados.