Estrutura de Dados

Aula 01

Prof. Luiz Antonio Schalata Pacheco, Dr. Eng.

Instituto Federal de Santa Catarina
Câmpus Garopaba
Curso Superior de Tecnologia em Sistemas para Internet

schalata@ifsc.edu.br

09/02/2023

Revisão de Python e Orientação a Objetos

Introdução: Boas-vindas e conteúdo do curso

- Objetivos: Entender a teoria e implementar/testar as estruturas de dados.
- Metodologia: Apresentação da teoria, aplicação de questionários e exercício de implementação.
  - Não usaremos bibliotecas prontas.
  - Codificação feita passo a passo.
- Pré-requisito: Lógica de programação e o básico da linguagem Python, incluindo POO.

Programação básica em Python

- Revisão básica da Linguagem Python com aplicação de exercícios, antes de iniciar propriamente o conteúdo de Estruturas de Dados.

Notação Big-O para análise de algoritmos

- Serve para fazer análise de algoritmos.
- Vamos supor que existem dois algoritmos que fazem a mesma função, mas foram implementados por programadores distintos. É necessário comparar para avaliar qual deles é o melhor.
- A área de estrutura de dados está relacionada a otimização de recursos de memória, por exemplo, criando diversos tipos de estruturas de dados para esse tipo de otimização.

Vetores não ordenados e Vetores ordenados

- Em vetores ordenados e não ordenados, veremos várias operações como inserção, inclusão e pesquisa.
- Vamos trabalhar com vários tipos de pesquisa de dados em vetores.
- Analisaremos as vantagens e desvantagens de cada uma dessas estruturas.

Pilhas, filas e deques

- Na sequência trabalharemos com três tipos de estruturas de dados diferentes que são as pilhas, filas e os deques.

Listas encadeadas

- Depois avançaremos para as listas encadeadas ao invés de usar estruturas de vetores, que é algo mais estático.
- Vamos criar a ligação de um objeto no outro que é ideia de ponteiro, onde se trabalha com referência de memória.

Algoritmos de ordenação

- Depois veremos algoritmos de ordenação, estudando os principais algoritmos: Bubble sort (bolha), Selection Sort (seleção), Insertion Sort (seleção), que são mais básicos.
- Depois os mais avançados (pois utilizam conceito de recursão) que são Shell Sort, Merge Sort, Quick Sort fazendo ao final comparativos entre os mesmos.

Árvores

- Estudaremos árvores, fazendo todo o processo de desenvolvimento.
- Vamos fazer operações de inserção, exclusão, pesquisa para que se entenda o funcionamento e a construção desse tipo de estrutura e dados.

Grafos

- Por fim, trabalharemos com os Grafos, que são estruturas um pouco mais complexas, utilizadas para resolver problemas do mundo real.

Considerações finais

- Implementação passo a passo.
- Não usaremos bibliotecas específicas de Estruturas de Dados.
- Por exemplo, para construir uma fila, nós podemos chamar uma classe do Python e simplesmente passar alguns parâmetros e a fila já estaria construída.
- Como o objetivo é aprender é necessário programar as estruturas do zero. Depois recomenda-se utilizar algum tipo de biblioteca pronta.
- O objetivo principal é entender a teoria sobre cada uma das estruturas, implementar/testar as estruturas de dados.
- Pré-requisito: Lógica de programação e o básico da linguagem Python.

Avaliação

- São previstas 3 avaliações: 2 provas e 1 trabalho prático.
- Plano de Ensino: Disponível no SIGAA

Revisão de Python e Orientação a Objetos

Objetivos

- Revisar conceitos básicos da linguagem Python
- Relembrar o paradigma da Orientação a Objetos
- Proporcionar prática através de exercícios propostos

Conceitos Básicos de Python

- Variáveis, operadores, estruturas de controle e funções
- Tipos de dados: inteiros, ponto flutuante, strings, listas, tuplas e dicionários
- Python é uma linguagem dinâmica, onde os tipos de dados são determinados em tempo de execução, e é também uma linguagem interpretada, o que significa que o código pode ser executado diretamente, sem a necessidade de compilação prévia

Relembrando...

- Variáveis são identificadores utilizados para armazenar valores. Eles podem ser de diferentes tipos, como int, float, strings, boolean, list, tuple ou dict
- Os operadores em Python incluem operadores aritméticos (+, -, *, /, //, %), operadores de comparação (==, !=, >, <, >=, <=) e operadores lógicos (and, or, not).
- As estruturas de controle permitem que controle do fluxo de execução do seu código. Três estruturas de controle principais: if, for e while.
- As funções são blocos de código que realizam uma tarefa específica e podem ser reutilizadas ao longo do código. Elas podem aceitar argumentos e retornar valores.

Listas, Tuplas e Dicionários

- Listas: são coleções ordenadas e mutáveis de itens. Você pode adicionar, remover, modificar e acessar itens de uma lista.
- Tuplas: são semelhantes às listas, mas são imutáveis. Você não pode adicionar, remover ou modificar itens em uma tupla depois que ela é criada.
- Dicionários: são coleções não ordenadas de pares chave-valor. Você pode adicionar, remover, modificar e acessar itens de um dicionário.

Listas, Tuplas e Dicionários

```
1 frutas = ["abacate", "banana", "laranja"]
2 cores = ("vermelho", "verde", "azul")
3 pessoa = {"nome": "Luiz", "idade": 48}
```

- Em resumo, as listas são úteis para armazenar coleções de dados ordenados e mutáveis, as tuplas são úteis para armazenar coleções de dados imutáveis e ordenadas, e os dicionários são úteis para armazenar coleções de dados não ordenadas com pares chave-valor.

Praticando...

1. Calcule a soma de dois números e armazene o resultado em uma variável.
2. Verifique se um número é par ou ímpar.
3. Utilize uma estrutura de repetição para imprimir todos os números de 1 a 10.
4. Leia duas strings e verifique se são iguais.
5. Calcule a fatorial de um número.
6. Leia uma lista de números e imprima o maior número da lista.
7. Verifique se um número é primo ou não.
8. Escreva um programa que utilize uma estrutura de controle de repetição para somar todos os números ímpares entre 1 e 100.

Relembrando listas e dicionários

9. Crie uma lista com números inteiros aleatórios e exiba o maior e o menor número da lista.
10. Crie uma lista de strings e conte a quantidade de ocorrências de cada palavra na lista.
11. Escreva um programa que receba uma lista de números e retorne uma nova lista com os elementos duplicados removidos.
12. Crie duas listas e verifique se elas têm elementos comuns. Se sim, exiba esses elementos.
13. Crie uma lista com números inteiros e ordene a lista em ordem crescente ou decrescente. O usuário deve escolher a ordem.

Orientação a Objetos

- O paradigma de Orientação a Objetos é uma forma de modelar a lógica de programação para representar conceitos do mundo real como objetos. Em Python, isso é feito criando classes, que são modelos para objetos do mundo real.
- Uma classe tem atributos, que são variáveis que descrevem as características de um objeto, e métodos, que são funções que definem a comportamento de um objeto.

Orientação a Objetos

14. Crie uma classe chamada aluno com os seguintes atributos: Nome, Nota 1 e Nota. Crie um construtor para a classe.
15. Crie as seguintes funções (métodos):
    - Calcula média, retornando a média aritmética entre as notas;
    - Mostra dados, que somente imprime o valor de todos os atributos;
    - Resultado, que verifica se o aluno está aprovado ou reprovado (se a média for maior ou igual a 6.0, o aluno está aprovado).
16. Crie dois objetos (aluno1 e aluno2) e teste as funções.