Sejam todos muito bem-vindos ao Período 2023.2

uniesp
Centro Universitário

Estrutura de Dados em Python

Prof. Nisston Moraes Tavares de Melo

uniesp
Centro Universitário

Conhecendo o professor

- Doutorando pelo PPGCC - Programa de Pós Graduação em Ciência da Computação da Universidade Federal de Campina Grande (UFCG) em 2020.1. Mestrado pelo PPGI - Programa de Pós Graduação em Informática da Universidade Federal da Paraíba (UFPB) em 2013, na área da Ciência da Computação, sub área Sistemas de Computação. Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistema pela Universidade Estácio da Paraíba em 2018. Bacharel em Ciências Econômicas pela UFPB em 1999. Especialista em Sistema de Informação e Redes de Computador pela UFPB em 2003, especialista em Design Instrucional pela Universidade de Itajuba RJ em 2008 e especialista em MBA de Gestão de Projetos pela Faculdade Estácio da Paraíba PB em 2015. Docente do ensino técnico (desde 1994) e superior (desde 2005). Experiência: Coordenador dos Cursos Técnico (Manutenção e Suporte em Informática) e Superior (Redes de Computadores) da Faculdade de Tecnologia IBRATEC de João Pessoa (08/2009 ? 08/2011). Professor substituto da Universidade Federal da Paraíba (08/2007 08/2009). Consultor credenciado pelo SEBRAE/PB (2008). Professor e tutor bolsista do IFPB - Instituto Federal de educação, Ciência e Tecnologia da Paraíba, do curso de Bacharel em Administração Pública, oferecido pela UAB (2013 - 2015). Gerente da Área de Educação do SENAIPB, na unidade Odilon Ribeiro Coutinho (09/2012 ? 12/2014) e Assessor de Inovação e Tecnologia do SENAIPB (02/2015 ? 12/2015). Professor Assistente III da Faculdade Estácio da Paraíba ? PB (desde 08/2012). Professor Mestre II da Faculdade DeVry de João Pessoa ? PB (desde 02/2016). Experiência em Linguagens de Programação e Banco de Dados. Atuando principalmente com os seguintes temas: Educação a Distância, Web Design, e Comercio, Processos e Tecnologia da Informação.
- http://lattes.cnpq.br/0388812363986090


DeVry


Estácio


SENAI


FACULDADE
IDEZ


TV TAMBAÚ


UNIPÊ
Centro Universitário
de João Pessoa


Senac


SAPIENTIA ÆDIFICAT


pitágoras


PORTAL DE EDUCAÇÃO CORPORATIVA
ESAT
CONHECIMENTOS
SEM LIMITES E
SEM FRONTEIRAS


GOVERNO
DA PARAÍBA
SECRETARIA
DA EDUCAÇÃO


RD


AUTO ESCOLA
CFCRP
RAINHA DA PAZ


REFRIGERAÇÃO
Polo Norte
3245-4333


uniesp
Centro Universitário


INSTITUTO
FEDERAL
Paraíba


Absoluta
ENGENHARIA

A disciplina

- Apresentação do Plano de Ensino da disciplina

Critério de avaliação

- Atividades de sala de aula;
- Atividades desafios de sala de aula;
- Listas de exercícios; e
- Avaliações.

Referências bibliográficas

- Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Algoritmos: teoria e prática.
- Goodrich, M. T., Tamassia, R., & Goldwasser, M. H. (2013). Estruturas de Dados e Algoritmos em Java.
- Zelle, J. M. (2010). Python Programming: An Introduction to Computer Science.

Algumas considerações

- Ordem dos assuntos no Plano de Ensino
- Conteúdo do Plano de Ensino
- Apresentação de casos problemas
- Participação dos alunos
- Celular em sala de aula
- Uso do computador
- Referências bibliográficas
- Como enviar material
- Realizar as atividades do professor

Metodologia

- Início com uma breve revisão da aula passada;
- Apresentação do conteúdo da aula;
- Aplicação de lista e atividade para fixação do conteúdo.

Questões que vamos responder.

- Como organizar e manipular minhas variáveis?
- Como criar algoritmos mais eficientes?
- Qual estrutura é mais eficiente?
- Como melhor estruturar minhas variáveis?
- Como otimizar meus recursos de memória e processamento?
- ...

Conteúdo programático

- Unidade 1
  - Introdução às estruturas de dados e algoritmos.
  - Listas em Python: manipulação e complexidade.
  - Pilhas e filas: conceitos e implementação.
  - Listas ligadas: tipos e operações.
- Unidade 2
  - Árvores: conceitos e propriedades.
  - Árvores binárias: percurso e busca.
  - Árvores de busca binária: inserção, remoção e balanceamento.

Conteúdo programático

- Unidade 3
  - Projeto prático: implementação de uma aplicação que utiliza Árvore binária.
  - Tabelas de espalhamento (*hash tables*): princípios e implementação.
  - Filas de prioridade: tipos e aplicações. Grafos: representações e tipos.
  - Grafos: busca em profundidade e busca em largura.
  - Grafos: algoritmos de caminho mínimo.
- Unidade 4
  - Projeto prático: implementação de uma aplicação que utiliza estruturas de dados.
  - Projeto prático: testes e finalização da implementação.


Revisão?

Revisão

- Revisão nos comandos e estruturas da linguagem Python:
  - Trabalhando com variáveis;
  - Estrutura condicional;
  - Estrutura de controle;
  - Vetores;
  - Operadores matemáticos;
- Revisar as funcionalidades da IDE utilizada no último semestre;
- Trabalhando com o GitHub.

Relembrando comandos básicos do Python!

- https://forms.gle/8NhRFZTPDxaGHH658

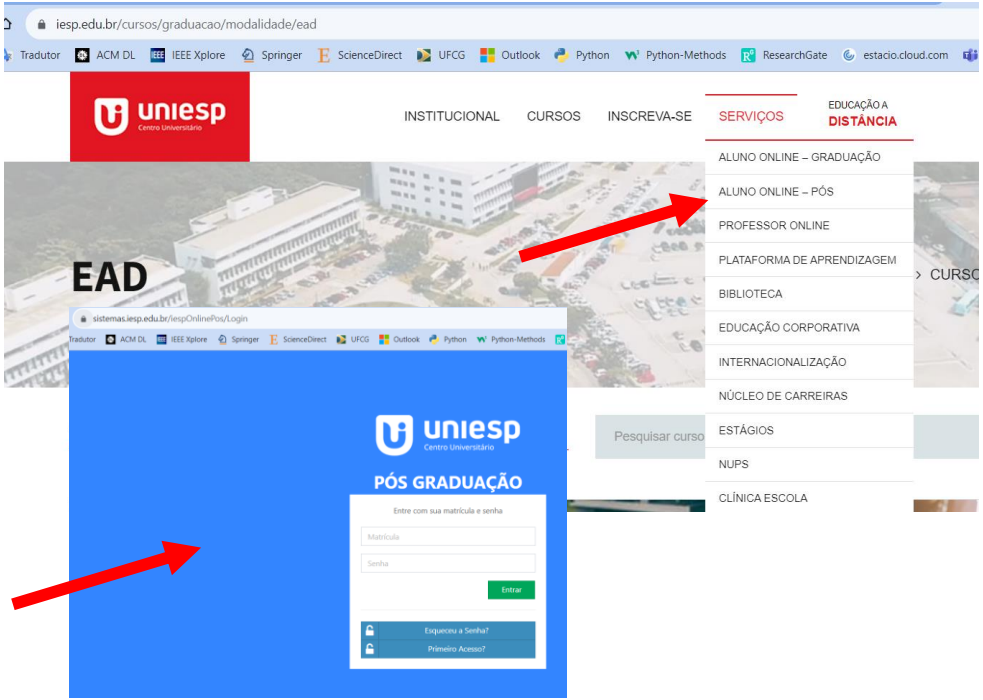
Relembrando conceitos de POO?

- https://forms.gle/fHuA73dW9biUU7rK9

Exercício de revisão.

Ambientes de interação

- Representante de turma
- Ambiente do aluno online
- E-mail do professor:
  - prof2279@iesp.edu.br



Qual a diferença entre eficiência e eficácia?

- Eficiência:
  - A eficiência se refere à capacidade de realizar uma tarefa ou atingir um objetivo com o mínimo de recursos, tempo, esforço ou desperdício possível.
  - É a relação entre os recursos utilizados e os resultados obtidos. Uma ação ou processo é considerado eficiente quando consegue alcançar seu objetivo com o menor custo possível.
- Eficácia:
  - A eficácia se refere à capacidade de atingir um objetivo ou realizar uma tarefa com sucesso e alcançar o resultado desejado.
  - É a medida de quão bem uma atividade ou processo alcança os resultados planejados, independentemente dos recursos utilizados.

Quais os recursos que utilizamos?

- Esses dois recursos são fundamentais para o funcionamento adequado de sistemas computacionais, afetando diretamente o desempenho e a capacidade de resposta das máquinas. Ambos desempenham papéis críticos na determinação da experiência geral do usuário e na eficiência operacional de dispositivos e sistemas.

Quais os recursos que utilizamos?

- Memória
  - Refere-se ao componente essencial que armazena temporariamente dados e instruções necessárias para a execução de programas. Existem vários tipos de memória em um sistema computacional, incluindo a memória principal (RAM) e a memória de armazenamento (disco rígido, SSD). A memória permite o acesso rápido e eficiente aos dados, garantindo que os programas possam ser executados de forma fluida. A capacidade e a velocidade da memória têm um impacto direto no desempenho geral de um sistema, influenciando a velocidade de inicialização, a abertura de aplicativos e a execução de tarefas.

Quais os recursos que utilizamos?

- Processamento
  - O processamento em computação refere-se à capacidade de realizar cálculos, executar instruções e manipular dados de acordo com os programas e algoritmos implementados. O processamento é realizado pela Unidade Central de Processamento (CPU), que é o "cérebro" do computador. A velocidade de processamento é medida em ciclos de clock e gigahertz (GHz) e determina a rapidez com que o computador pode realizar operações complexas, como cálculos matemáticos, lógica e manipulação de dados. Um processador poderoso é fundamental para a execução eficiente de aplicativos e tarefas computacionais intensivas.

Somos eficiente e eficaz na entrega dos sistemas de informação?

Importância das Estrutura de Dados

- O que são estruturas de dados.
- A importância do uso adequado de estruturas de dados.
- Como as estruturas de dados afetam a eficiência dos programas.

Conceito de Estrutura de Dados

"Estruturas de Dados são construções de uma linguagem de programação que agregam um ou mais elementos de dados para formar um tipo de dado que armazena uma quantidade maior de informações".(OLIVEIRA, R., TAVEIRA, G., BOTTINI, J., 2003, p.11)

Conceito de Estrutura de Dados

“O campo da Estruturas de Dados é concebido para construir ferramentas para serem incorporadas e usadas pelos programas de aplicação e para encontrar Estruturas de Dados que possam realizar certas operações rapidamente sem impor muita carga à memória do computador”.(DROZDEK, A, 2002, P.31)

Estrutura de Dados

- Envolve algoritmos que manipulam dados organizados em memória de maneira mais sofisticada do que as simples variáveis básicas que foram estudadas até o momento.
- A organização de dados em memória permite a construção de algoritmos sofisticados e eficientes As duas estruturas de dados elementares são:
  - vetores (ou array unidimencional);
  - matrizes (ou array multidimencional);

O que são Estruturas de Dados.

- As estruturas de dados são formas organizadas de armazenamento e gerenciamento de dados em um programa de computador. Elas fornecem uma maneira de representar, armazenar e manipular informações. Uma estrutura de dados pode ser vista como um recipiente que mantém um conjunto de valores, permitindo operações específicas, como inserção, remoção e pesquisa desses valores de maneira eficiente.

A importância do uso adequado das ED

- O uso adequado de estruturas de dados é fundamental para o desenvolvimento de software eficiente e escalável. Ao escolher a estrutura de dados correta para cada situação, você pode otimizar o desempenho do programa, economizar recursos computacionais e melhorar a legibilidade do código.
- Uma escolha inadequada de estrutura de dados pode resultar em programas lentos, consumo excessivo de memória e dificuldade de manutenção.

Como as estruturas de dados afetam a eficiência dos programas.

- As estruturas de dados têm um impacto direto na eficiência dos programas. Cada operação em um programa consome recursos computacionais, como tempo e memória. Diferentes estruturas de dados têm características distintas que as tornam mais adequadas para certas operações do que outras. Por exemplo:
  - Inserção e Remoção
  - Busca e acesso
  - Uso de memória

Continuação

- **Inserção e Remoção:** A escolha da estrutura de dados certa pode resultar em operações de inserção e remoção mais rápidas. Listas ligadas são eficientes para inserções e remoções no meio da estrutura, enquanto arrays são melhores para operações no final.
- **Busca e Acesso:** A eficiência de busca e acesso também varia com a estrutura de dados. As árvores de busca binária são ideais para buscas eficientes, enquanto arrays são mais rápidos para acessos aleatórios.
- **Uso de Memória:** Diferentes estruturas de dados têm requisitos de memória diferentes. Alguns podem consumir menos memória do que outros para armazenar a mesma quantidade de dados.

Tipos de Dados

- Tipos primitivos vs. tipos compostos.
- Dados homogêneos vs. heterogêneos.
- Exemplos de tipos de dados em linguagens de programação.

Os tipos de dados em programação

- **Tipos Primitivos:** Também conhecidos como tipos básicos, são os blocos de construção fundamentais de uma linguagem de programação. Eles representam valores simples, como números inteiros, números de ponto flutuante, caracteres e booleanos. São usados para armazenar informações individuais.
- **Tipos Compostos:** São criados a partir de tipos primitivos e são usados para agrupar vários valores em uma única unidade. Os tipos compostos incluem arrays, estruturas (structs) e classes. Eles permitem representar dados mais complexos e estruturados.

Dados homogêneos vs. heterogêneos.

- **Dados Homogêneos:** Em um conjunto de dados homogêneos, todos os elementos têm o mesmo tipo de dado. Por exemplo, um array de inteiros ou uma lista de strings são exemplos de conjuntos homogêneos. Eles são mais fáceis de gerenciar e processar, pois todos os elementos têm a mesma estrutura.
- **Dados Heterogêneos:** Em um conjunto de dados heterogêneos, os elementos podem ter diferentes tipos de dados. Isso permite representar informações variadas, mas pode ser mais complexo de lidar, pois cada elemento pode ter estruturas diferentes.

Exemplos de tipos de dados em ling. de prog.

- C e C++:
  - Tipos Primitivos: int, float, char, bool.
  - Tipos Compostos: array, struct, enum, union.
- Python:
  - Tipos Primitivos: int, float, str, bool.
  - Tipos Compostos: list, tuple, dict, set.
- Java:
  - Tipos Primitivos: int, float, char, boolean.
  - Tipos Compostos: array, class, interface, enum.
- JavaScript:
  - Tipos Primitivos: number, string, boolean.
  - Tipos Compostos: array, object.

Entendendo o uso das variáveis

- https://pythontutor.com/

Vetores e listas

- Definição de vetores e listas.
- Indexação e acesso aos elementos.
- Operações básicas de vetores/listas (inserção, remoção, atualização).

Definição de vetores e listas

- **Vetores:** Um vetor é uma estrutura de dados unidimensional que armazena uma sequência de elementos do mesmo tipo. Cada elemento é acessado através de um índice inteiro.
- **Listas:** Uma lista é uma estrutura de dados dinâmica que pode armazenar uma sequência de elementos do mesmo ou de diferentes tipos. As listas podem crescer ou diminuir de tamanho durante a execução do programa.


Dúvidas?


uniesp
Centro Universitário