Sobre a Disciplina Programação Imperativa

Prof. Alberto Costa Neto
DComp/UFS

Sobre a Disciplina PI

- Disciplina: Programação Imperativa (COMP0334)
- Equivalentes:
  - Introdução à Ciência da Computação
  - Programação Imperativa (COMP0197)
- Carga horária: 60 horas
- Créditos: 4

Ementa

Noções fundamentais sobre algoritmos e sobre a execução de programas. Análise e síntese de problemas. Identificadores, tipos, constantes, variáveis, tipos. Operadores e expressões. Comandos condicionais e de repetição. Variáveis compostas homogêneas e heterogêneas. Procedimentos, funções e passagem de parâmetros. Noções sobre o uso de arquivos em programação. Algoritmos básicos de ordenação. Recursividade. Uma linguagem imperativa. Convenções de código. Boas práticas de programação.

Objetivos

Geral

- Apresentar os conceitos básicos e principais técnicas de desenvolvimento de programas de computador, tornando-o apto a compreendê-los e aplicá-los.

Específicos

- Tornar o aluno capaz de implementar programas básicos usando uma linguagem de programação imperativa.
- Habilitar o aluno a criar programas para executar computação científica na sua área de conhecimento.
- Colocar em prática os conhecimentos aprendidos no curso, desenvolvendo aplicações de pequeno porte em Python.

Conteúdo Programático

1º Unidade

- Motivação para Programar
- Hardware, software e princípios
- Visão Geral da Linguagem Python
- Preparação do Ambiente de Desenvolvimento
- Instruções primitivas: atribuição, entrada e saída
- Expressões
- Tipos
- Comandos Condicionais (if)
- Tratamento de exceções (try / except)
- Funções
- Laço While
- Strings
- Laços For

2º Unidade

- Listas
- Compreensão de Listas
  Recursividade
- Conjuntos (Aula 16)
- Algoritmos de Ordenação
  (Inserção, Seleção e Bolha)
- Busca Binária
- Matrizes
- Dicionários
- Tuplas
- Arquivos

Afinal, por que o nome PI?

- Vem da denominação do Paradigma que vamos estudar: Paradigma Imperativo
  - Você escreve explicitamente as ordens e o computador obedece
  - Mais próximo do funcionamento real do computador
  - Existem outros paradigmas, como por exemplo:
    - Funcional
    - Orientado a Objetos

Método de Ensino

Metodologia - Presencial

- Conteúdo teórico estará disponível pelo Youtube
- Sistema que permite programar e tem autoavaliação
- Exercícios podem ser feitos em casa ou laboratório
- Tempo de aula será focado em exercícios e tirar dúvidas

Recursos Didáticos

As aulas serão ministradas presencialmente utilizando as seguintes ferramentas:

- Youtube, para exposição das videoaulas.
- Ferramentas de Videoconferência: Google Meet.
- Editores de programas: VS Code, PyCharm, Notepad++ ou Sublime Text.
- Interpretador da linguagem Python, que permite a verificação de erros de sintaxe e execução de programas em Python.
- Apps que permitem elaborar, executar e testar programas em smartphones e tablets.
- Ambientes Virtuais de Aprendizagem (AVA) SIGAA
- Questionários com Problemas de Programação no site The Huxley
- Provas práticas na plataforma Koude

Correção de Questões

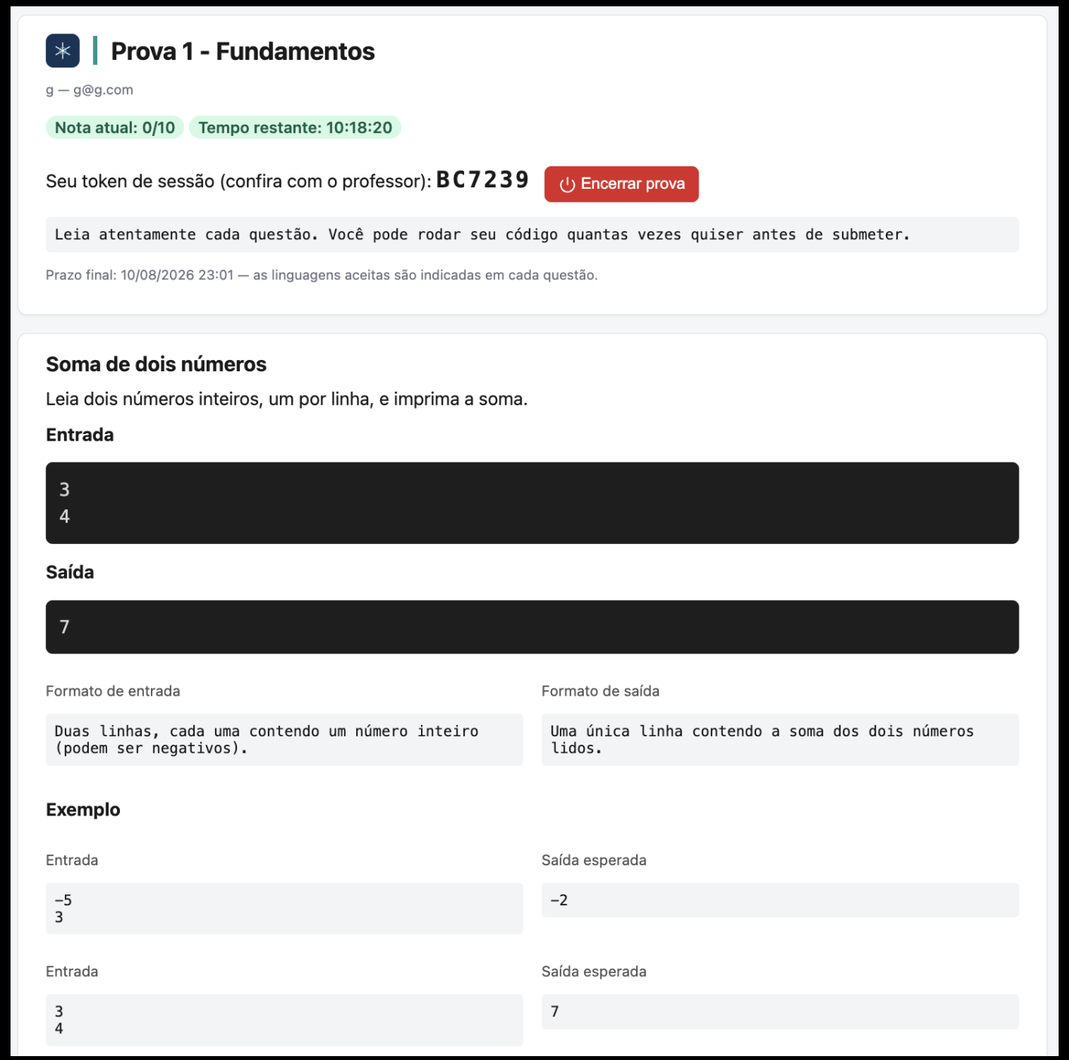
- Imagine se seu professor terá como corrigir 100 questões de cada um dos 50 alunos…
  Façamos as contas:
  - São 5.000 questões!
  - Supondo que o professor gaste 6 min por questão, seriam necessários 30.000 minutos, ou seja, 500 horas!
- Seria interessante ter uma ferramenta que ajudasse o professor, concordam?

Fonte: http://2.bp.blogspot.com/_Q4jxiezF5Hk/TNbebADQ2FI/AAAAAAAAABM/gnjeS8-S2I0/s1600/estres-laboral-y-enfermedad-periodontal.jpg

The Huxley

- Uma ferramenta Web que oferece um banco de problemas de programação (juiz *on-line*).
- Os alunos podem enviar soluções (programas em várias linguagens de programação).
- O The Huxley executa a solução com entradas presentes em casos de teste e compara com o resultado esperado.
- Com esta ferramenta o aluno tem um *feedback* imediato

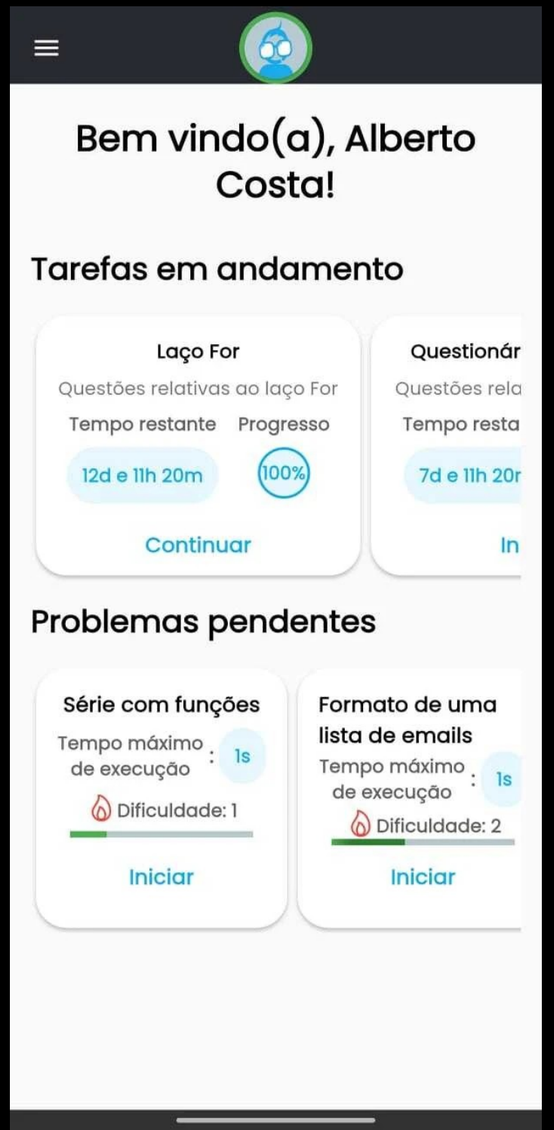
Koude



- Uma ferramenta Web para realização de provas online de programação.
- Os alunos podem enviar suas respostas (programas em várias linguagens de programação).
- O The Koude executa a solução com entradas presentes em casos de teste e compara com o resultado esperado.
- O professor gerencia o acesso à prova e dispõe de mecanismos de segurança para controle de acesso.

The Huxley Mobile

- Uma interface para o The Huxley para dispositivos móveis
- Desenvolvido e mantido por alunos da UFS
- Disponível na Play Store



Avaliação

Critério de Avaliação

Através de testes presenciais, obedecendo à fórmula:

$$\text{Nota Final} = (NT1 + NT2) / 2$$

Onde:

- NT1 = Média das provas da 1º unidade
- NT2 = Média das provas da 2º unidade

Observação: Em cada unidade, haverá pelo menos 3 provas obrigatórias e com datas definidas no plano de ensino. Pode haver provas surpresa, mas estas serão consideradas como extra e servem apenas para descartar notas mais baixas em provas obrigatórias ou repor falta em alguma prova obrigatória.

Prova de Reposição

- Observação: Haverá uma prova de reposição no final do semestre apenas para os alunos com falta justificada em algum teste, desde que a justificativa esteja prevista nas normas acadêmicas.

- Entretanto, caso o aluno tenha feito provas surpresa e atingido 3 provas por unidade, não terá direito à prova de reposição por já ter tido oportunidade de fazer a avaliação perdida em outro dia.

Calendário de Provas

As provas serão realizadas presencialmente:

- No horário da aula
- Segundo calendário e orientações divulgados no Plano de Ensino e no SIGAA
- As provas surpresa não têm data para acontecer

Controle de Frequência

Controle de Frequência (Turmas Presenciais)

- O aluno é obrigado a estar presencialmente nas aulas.
- Assim, a frequência dos alunos será computada através da Lista de presença.
- Caso o aluno assine a lista de presença e se ausente da sala de aula sem autorização do professor, terá sua presença computada parcialmente.

Bibliografia

Referências Bibliográficas (Básica)

- Python Para Todos: Explorando Dados com Python 3. Charles R. Severance. Publicação independentes; 1º edição, 2020; ISBN: 979-8635191408

- Think Python: How to Think Like a Computer Scientist. Allen Downey. 2024. 3rd Edition. Disponível em https://allendowney.github.io/ThinkPython/

Referências Bibliográficas (Complementares)

- Fundamentos da Programação de Computadores. Ana Fernanda Gomes Ascencio / Edilene Aparecida Veneruchi De Campos. 3˚ edição; 2012, Pearson; ISBN 978-8564574168
- Algoritmos e Lógica de Programação. Marco A. Furlan de Souza, Marcelo M. Gomes, Marcio V. Soares, Ricardo Concilio. Editora Cengage Learning, 2ª edição, 2011.
- Algoritmos: Lógica para Desenvolvimento de Programação de Computadores. José Augusto N. G. Manzano, Jayr Figueiredo de Oliveira. Editora Érica, 17ª edição, 2005

Contatos dos Professores

- Alberto Costa Neto – T08 e T09
  - alberto@dcomp.ufs.br
  - albertocn@academico.ufs.br

Como proceder em caso de dificuldade?

- Sempre que identificar alguma dificuldade, dúvida sobre conceitos das videoaulas ou problemas, entre em contato com o(s) professor(es) responsáveis pela sua turma.
- Caso não consiga acessar os AVAs ou sites, também entre em contato com o(s) professor(es).

Não deixe de tirar suas dúvidas!

E sejam bem-vindos ao curso de PI!!!