BCM0505 - Processamento de Informação
Plano de ensino (turma DB6)

Prof. Diogo S. Martins
Centro de Matemática, Computação e Cognição
Universidade Federal do ABC

Q2 2021
v.24/05

1 Informações básicas

- TPI: 3-2-5
- Horários oficiais:

	ter (teoria)	08-10h	semanal
	qua (teoria)	10-12h	quinzenal I
	qui (prática)	10-12h	semanal

- Ferramentas de comunicação:
  - Discord
    Para cadastrar-se na sala, utilizar link disponível no Moodle.
  - Comunicação com o professor
    - Prioritariamente via Discord, por mensagem pública ou privada
    - Por email (caso Discord esteja indisponível): `santana.martins@ufabc.edu.br`
- Plantão de dúvidas:

  ter 10-12h Discord

  Os plantões são nossos únicos eventos síncronos. O objetivo é esclarecer dúvidas e/ou reforçar temas vistos nas vídeo-aulas e outros materiais. Caso haja interesse em participar do plantão, é necessário agendar um slot no Google Calendar. Vide Moodle, para o link de agendamento.

  As comunicações com o professor, exceto no horário de plantão, serão assíncronas, em geral com resposta dentro de 24h em dias úteis.
- Sala no Moodle: `pi-2021.2-db6`

  `https://moodle.ufabc.edu.br/course/view.php?id=1875`

  Todos já foram convidados para a sala, em caso de dificuldade de acesso, entrar em contato com o professor.

2 Descrição da disciplina

A habilidade de programar computadores é essencial para todo estudante de ciência e tecnologia pois, ao lado da teoria e da experimentação, a computação tornou-se imprescindível para os avanços científicos e tecnológicos. O objetivo da disciplina de Processamento da Informação é preparar os alunos com conceitos e técnicas para processar informação digital eficientemente por meio de programas de computador. A disciplina é introdutória à programação de computadores, com foco no paradigma de programação estruturada/procedural, tomando como base a linguagem de programação Python. A parte teórica tende a focar nos aspectos *conceituais* da programação de computadores, por exemplo: abstração e modelagem de problemas, construção de algoritmos, execução de testes de algoritmos, etc. A

parte prática tende a enfatizar os aspectos *lógicos* e *físicos* da programação de computadores, por exemplo: uso de compiladores, testes online com conjuntos de dados sintéticos ou reais, ferramentas e técnicas de depuração, etc.

3 Requisitos recomendados

- BC0005 - Bases Computacionais da Ciência

4 Objetivos

- Conhecer os fundamentos de manipulação e tratamento da informação;
- Familiarizar-se com os conceitos de lógica de programação de computadores;
- Conhecer o paradigmas estruturado e procedural de programação;
- Conhecer estruturas de dados fundamentais;
- Saber transferir os conceitos estudados para práticas de programação.

Ao final do curso espera-se que o aluno, aprovado com conceito satisfatório, esteja familiarizado com os conceitos fundamentais de programação estruturada e procedural, sendo capaz de projetar algoritmos para problemas básicos de processamento de informação.

5 Ementa

1. Noções de organização de computadores;
2. Lógica de programação, algoritmos e programação (teoria e prática);
3. Sequenciamento de operações;
4. Estruturas de decisão;
5. Estruturas de repetição;
6. Modularização e abstração de dados;
7. Processamento de vetores e matrizes.

6 Bibliografia

Obras principais:

1. Downey, A. B. Think Python: How to think like a computer scientist. 2nd edition. O'Reilly, 2019.

   Disponível gratuitamente em: `https://greenteapress.com/wp/think-python-2e/`.

   Caso tenha dificuldade com a língua inglesa, há tradução em português: `https://github.com/PenseAllen/PensePython2e`

2. Severance, C. Python for Everybody: Exploring Data in Python 3 (1st. edition). Self published. 2016. URL: `https://www.py4e.com/book.php`

   Versão digital disponível gratuitamente. No website você encontra material complementar ao livro, inclusive videoaulas do Coursera, que podem ser usadas como material de reforço, caso necessário.

3. Sedgewick, R. & Wayne, K.. Introduction to Programming in Python: An Interdisciplinary Approach. (1st Ed.). Addison-Wesley Professional. 2015. 784 pages.

   Versão resumida (em HTML): `https://introcs.cs.princeton.edu/python/home/`

Vide a última versão do projeto pedagógico do BC&T[1] para as obras complementares dessa disciplina.

7 Metodologia de ensino-aprendizagem

Formato das aulas. Assíncronas, em formato de vídeo. Cada aula será disponibilizada no respectivo horário oficial da turma.

Interações. Em duas modalidades: *i)* assíncrona, via Discord canal de texto, com prazo máximo de 24 horas para resposta (em dias úteis); e ii) síncrona, via Discord canal de voz, no horário de plantão de dúvidas (vide seção 1).

Técnicas de ensino-aprendizagem:

- **Indissociabilidade entre teoria e prática.** O conteúdo englobará, além de discussões teóricas, também tutoriais com práticas de programação, os quais, idealmente, devem ser seguidos de forma síncrona, ou seja, que os alunos tentem implementar os exemplos ao mesmo tempo em que assistem ao vídeo.

  No ensino presencial, por questões logísticas, costuma-se organizar as ofertas de PI em turmas distintas de teoria e de prática, com salas diferentes e, muitas vezes, professores diferentes. No ensino remoto, essa divisão não faz sentido, portanto não haverá distinção entre as aulas teóricas e as práticas.
- **Aprendizagem ativa**[2]. As atividades das videoaulas enfatizam a análise, a síntese e a avaliação dos conteúdos, ou seja, haverá menor ênfase no consumo passivo de informações. Por esta razão, é importante seguir os tutoriais de programação na frente do computador, programando os exemplos sincronamente aos vídeos, de modo análogo ao que faria se estivesse num laboratório de programação.
- **Aprendizagem baseada em problemas**[3]. O conteúdo das aulas é estruturado em problemas-base concretos, os quais servem de contexto para abordar os temas conceituais/abstratos da ementa. O reforço do conhecimento teórico-conceitual é abordado via estudo prévio ou posterior com os *materiais complementares* (referências de estudo, que podem ser textos ou vídeos) indicados no roteiro da respectiva aula.

8 Avaliação

A avaliação somativa consiste nos componentes dados pela Equação 1, onde:

$$N_F = 0.5 \cdot N_{atv} + 0.25 \cdot N_{P1} + 0.25 \cdot N_{P2} \tag{1}$$

- N_{atv} é a média de 75% das atividades com as melhores notas. Como total para calcular a porcentagem, consideramos todas as atividades enunciadas durante o quadrimestre;
- N_{P1} e N_{P2} são as notas da Prova 1 e Prova 2, respectivamente;

O conceito final será obtido de acordo com a Equação 2.

$$C_F = \begin{cases} \texttt{A}, \text{ se } N_F \in [8.5, 10.0] \\ \texttt{B}, \text{ se } N_F \in [7.0, 8.5) \\ \texttt{C}, \text{ se } N_F \in [5.5, 7.0) \\ \texttt{D}, \text{ se } N_F \in [5.0, 5.5) \\ \texttt{F}, \text{ se } N_F \in [0.0, 5.0) \\ \texttt{O}, \text{ se ausência exceder } 25\% \end{cases} \tag{2}$$

[1] Projeto Pedagógico do BC&T UFABC: `http://prograd.ufabc.edu.br/bct/pps`
[2] `https://en.wikipedia.org/wiki/Active_learning`
[3] `https://en.wikipedia.org/wiki/Problem-based_learning`

Sobre as Atividades

Teremos um total de 5 atividades somativas, ou seja, valendo nota. Caberá ao aluno demonstrar disciplina e organização de tempo para executá-las, visto que são planejadas para preencher o componente I (5 horas/sem.) previsto oficialmente para a disciplina.

Auto-avaliação formativa. As atividades contarão, sempre que possível, com recursos de verificação automática de corretude e estilo de programação, com o objetivo de auxiliar o estudante a detectar, de modo automatizado, problemas recorrentes de programação. Na ausência de recursos de verificação automática, o estudante contará com uma rubrica para analisar a aderência do seu resultado ao que é esperado. Cabe ao aluno garantir que o programa passe por *todas* as verificações automáticas. Além disso, nas atividades em que haja rubrica, deve garantir que todos os critérios sejam atendidos. Submissões com falhas terão descontos substanciais na nota.

Avaliação somativa. Além dos testes automáticos, o professor analisará as submissões em atenção aos aspectos qualitativos e éticos detalhados neste plano de ensino. Ou seja, passar em todos os testes automáticos não implica necessariamente em nota máxima, visto que análise posterior pode implicar em desconto parcial ou total na nota da atividade.

Prazos. Cada atividade tem prazo de 15 dias para entrega. É possível entregar atividades atrasadas, sendo o limite máximo para entrega a data indicada no cronograma. Atividades atrasadas terão desconto de nota proporcional ao atraso.

Sobre as Provas

As provas serão remotas. Cada prova ficará aberta pelo prazo de 72h (entre terça-feira e quinta-feira), de modo que o aluno possa escolher o melhor dia desse prazo para executá-la. O aluno terá acesso ao enunciado somente quando iniciar explicitamente a prova. Após iniciada, haverá um prazo, pré-anunciado, de 2 a 3 horas, para entregá-la. O prazo será controlado automaticamente pelo sistema de submissão.

Não serão aceitas, em quaisquer hipóteses, provas atrasadas ou entregues por meios de submissão alternativos.

Sobre as submissões dos programas

Tanto as atividades quanto as provas serão submetidas via Moodle e via Github Classroom. O conteúdo da primeira semana de aula envolve treinamento sobre o uso combinado destas duas plataformas de submissão.

Sobre as Listas de Exercícios

Diferentemente das Atividades, as Listas de Exercícios não valem nota. O objetivo das listas é apenas servir como recurso complementar de estudo e de reforço. Ficarão disponíveis em plataforma de submissão com correção automática, para que os alunos possam auto-avaliar suas submissões. Ressalta-se que as listas servirão apenas como "treino" para o aluno e não serão avaliadas pelo professor, embora nada impeça que haja discussão de dúvidas sobre estes exercícios.

Critérios de avaliação

Todas as avaliações somativas, isto é, as atividades e as provas, qualificam-se como atividades de programação.

A nota máxima de cada atividade será obtida apenas se a mesma for entregue no prazo e executada correta e completamente.

Os programas solicitados em atividades de avaliação serão submetidos aos seguintes tipos de verificações:

- **Verificações automáticas**. Testes de corretude, de estilo de programação, de erros comuns e de detecção de plagiarismo.
- **Verificações manuais**. O professor inspecionará os programas para verificar os seguintes critérios gerais:
  - **Eficiência**: os programas desenvolvidos deverão ter bom desempenho, o que pode englobar o tratamento adequado dos seguintes fatores:
    * Ler e escrever dados nas quantidades mínimas necessárias para resolver o problema;

    - Não desperdiçar memória primária (RAM);
    - Acessar memória secundária (disco) somente quando necessário e sem redundância;
    - entre outros.
  - **Acurácia**: o programa deverá atender adequadamente a todos os requisitos enunciados para a atividade;
  - **Corretude**: a solução deverá estar funcionalmente correta;
  - **Estrutura e organização do código**: atentar principalmente aos seguintes aspectos:
    - **Auto-documentação**: nomes intuitivos para variáveis e funções;
    - **Modularização**: funções com alta concisão e baixo acoplamento, isto é, que sejam em sua maioria curtas, e que realizem preferencialmente uma única tarefa;
    - **Comentários**: documentação completa porém ao mesmo tempo concisa (sem poluição visual, apenas nos lugares adequados e necessários).
  - **Autenticidade**: o código é original e não foi copiado de outras fontes (i.e. web, livros, terceiros, etc.), sob pena das sanções previstas no código de honra.

8.1 Mecanismos de avaliação substitutivos

Teremos dois mecanismos de avaliação substitutiva:

- Para as atividades, como consideraremos 75% das maiores notas, para substituir basta entregar com atraso até atingir no mínimo 75% de entregas;
- No caso da prova, que é assíncrona com janela de uma semana, fica implícita a possibilidade de substituição, dada a possibilidade do aluno escolher o melhor momento para submeter.

8.2 Mecanismo de recuperação

A recuperação será aplicada apenas aos alunos que tiverem conceito final D ou F. Ocorrerá no início do quadrimestre subsequente, em formato similar ao estabelecido para as provas regulares.

A nota obtida na prova de recuperação (N_R) será usada para obter a nota final com recuperação (N_{FR}), que consiste na média estabelecida pela Equação 3.

$$N_{FR} = \frac{N_F + N_R}{2} \tag{3}$$

O conceito final obtido na recuperação (C_{FR}) é o conceito que entrará no histórico, obtido de acordo com os limiares para a nota final de recuperação (N_{FR}) dados pela Equação 4.

$$C_{FR} = \begin{cases} \texttt{C}, \text{ se } N_{FR} \geq 5.5 \\ \texttt{D}, \text{ se } N_{FR} \in (5.0, 5.5) \\ \texttt{F}, \text{ se } N_{FR} \leq 5.0 \end{cases} \tag{4}$$

No caso dos alunos que não participarem da recuperação, $C_{FR} = C_F$.

9 Código de honra

A aprovação na disciplina é baseada exclusivamente no esforço e trabalho pessoal do discente, ao qual cabe garantir que não ajudará ou receberá ajuda não-permitida em qualquer atividade avaliativa (e.g. provas, trabalhos, listas, etc.).

Exemplos de violação do código de honra incluem:

- Copiar atividades avaliativas (e.g. listas, trabalhos, provas, etc.) ou permitir que outros discentes copiem suas atividades avaliativas;
- Colaboração não-permitida entre indivíduos ou grupos (e.g. oferecer vantagens em troca de soluções prontas, doar trechos para o trabalho de outro grupo, etc.);

- Permitir que outros assumam sua identidade em atividades avaliativas (e.g. entregar trabalho que não fez ou permitir que outros façam provas por você);
- Plagiarismo (i.e. aplicável a textos, programas de computador, etc.), o que envolve copiar porções significativas de textos ou programas de terceiros, sem atribuição de autoria ou, mesmo que haja atribuição de autoria, demonstre-se que não houve trabalho original significativo;
- Receber ou conceder ajuda em atividades avaliativas quando o contexto mostra que não é sensato receber tal ajuda.

Como consequências de violação do código de honra tem-se:

- Reprovação automática na disciplina, com conceito F, sem direito ao mecanismo de recuperação;
- Denúncia na Comissão de Transgressões Disciplinares Discentes da Graduação, a qual decidirá sobre a punição adequada à violação, o que pode levar a advertência, suspensão ou desligamento, de acordo com os arts. 78-82 do Regimento Geral da UFABC.

10 Cronograma de aulas

O cronograma a seguir pode variar de acordo com o aproveitamento aferido nas turmas durante o quadrimestre.

Aula #	Data	Tema
01	25/05	Introdução à programação de computadores
02	26/05	Introdução ao ambiente de desenvolvimento
03	27/05	Tipos de dados, variáveis e expressões
04	01/06	Exercícios
–	03/06	Feriado
05	08/06	Funções
06	09/06	Funções
07	10/06	Exercícios
08	15/06	Seleção
09	17/06	Seleção
10	22/06	Repetição
11	23/06	Repetição
12	24/06	Repetição
13	29/06	Exercícios
14	01/07	Exercícios
15	06/07	Revisão + Prova 1
16	07/07	Revisão + Prova 1
17	08/07	Prova 1
18	13/07	Strings
19	15/07	Strings
20	20/07	Estruturas de dados I
21	21/07	Estruturas de dados I
22	22/07	Exercícios
23	27/07	Estruturas de dados II
24	29/07	Estruturas de dados II
25	03/08	Arquivos
26	05/08	Arquivos
27	10/08	Revisão + Prova 2
28	11/08	Revisão + Prova 2
29	12/08	Prova 2
30	16/08	Prazo máximo para entrega de atividades