Plano de desenvolvimento da disciplina

Publicado em 15/02/2022

Instituto de Computação — Universidade de Campinas

Algoritmos e Programação de Computadores

Primeiro Semestre de 2022 – MC10W

Prof. *Lehilton Lelis Chaves Pedrosa*

Monitores Darwin, Diogo, Samuel (PED) e Paulo, Augusto (PAD)

Introdução

Este é o plano de desenvolvimento da disciplina e um guia de estudos. Leia-o com atenção e consulte este documento durante todo o semestre. Também, sempre acompanhe os avisos na página da disciplina.

Objetivos

Ao final do curso, @ estudante deverá ser capaz de:

- descrever problemas computacionais, computabilidade e limitações;
- descrever organização básica de computadores e de sistemas de software;
- utilizar estruturas de dados elementares: variável, lista, matriz
- implementar e testar programas escritos em Python.
- escrever algoritmos iterativos e recursivos;

Pré-requisitos

A disciplina não tem pré-requisitos formais. O foco será em problemas computacionais e algoritmos, então ter tido uma experiência anterior com programação pode ajudar a desenvolver as atividades ministradas, mas não substitui nem é necessário para o bom andamento da disciplina.

Ao início do curso, é recomendável que @ estudante seja capaz de:

- desenvolver e executar um cronograma de estudo contínuo;
- interpretar e descrever problemas e situações textualmente;

- consultar, organizar e resumir fontes de pesquisa diversas (bibliotecas e internet).

Atividades

Aula expositiva

As aulas ocorrerão presencialmente. O professor disponibilizará conteúdo em texto ou vídeo com antecedência, que deve ser estudado **antes** das aulas correspondentes. Durante o horário das aulas haverá revisão e discussão dos conteúdos e exercícios e @s estudantes devem participar levantando questões, dúvidas, sugestões, ou possivelmente resolvendo os problemas solicitados. A qualquer momento, estudantes podem tirar dúvidas com o professor.

Listas de exercícios para fixação

Serão propostos diversas listas de exercícios para fixação para serem realizados durante e após as aulas. O conteúdo das listas é considerado parte integrante do curso. Há tanto questões teóricas quanto práticas. Implementar os programas solicitados nas questões serve como um exercício mais simples antes das tarefas de programação. Como as questões são abertas, não há gabarito. Esses exercícios para fixação não serão corrigidos e não é obrigatório realizar todos eles.

Recomenda-se tentar fazer esses exercícios tanto individualmente quanto em grupo com não mais do **quatro** estudantes. Se desejarem resolver as atividades em grupo, sugere-se criar um repositório no Gitlab, onde podem compartilhar códigos e discutir entre vocês.

Tarefas

Haverá diversas tarefas individuais que servirão de avaliação da disciplina. O número previsto de tarefas é 10 e pode mudar a depender do andamento da turma. Cada tarefa é um conjunto de um ou mais exercícios de programação ou trabalhos mais elaborados, que deverão ser implementados na linguagem de programação Python. Todas as tarefas devem ser desenvolvidas em um repositório de controle de versões Git criado pelo professor, utilizando a infraestrutura do IC https://gitlab.ic.unicamp.br/. Para criar uma conta, logue-se no sistema utilizando a senha fornecida pelo IC. Cada estudante terá um repositório de nome `raXXXXXX`, em que `XXXXXX` corresponde ao número de RA.

A sequência das atividades será a seguinte:

1. O enunciado da tarefa é publicado após ou simultaneamente os conteúdos correspondentes serem ministrados.

2. Um conjunto de arquivos auxiliares da tarefa será inserido automaticamente no repositório pessoal em um diretório de nome `tarefaNN`, onde `NN` é o número da tarefa. Esses arquivos auxiliares podem ser acessados bastando executar o comando `git pull` na cópia de trabalho pessoal.

3. @ estudante realiza os vários exercícios da tarefa, alterando ou incluindo novos arquivos no repositório correspondente. À medida em que cada pequeno passo for realizado, deve-se fazer um `git commit` com uma mensagem adequada. É **obrigatório** que todo o desenvolvimento da tarefa seja registrado utilizando o controle de versões, isso é, **não** escreva suas tarefas e copie de outros locais e faça commit mesmo dos programas não prontos, parcialmente escritos. Não serão aceitas tarefas que não tenham histórico de commits ou submetidas por outros meios.

4. A cada pequena parte implementada, deve-se testar o programa. Para isso, pode-se executar em um terminal `python3 testar.py`, o que executará um script de teste fornecido juntamente com os arquivos auxiliares. Além desses, pode ser necessário criar outros testes.

5. Terminada a hora de trabalho, @ estudante executa o comando `git push`. Isso enviará as alterações para o repositório remoto. Além disso, ativará um script de correção, que testará automaticamente a tarefa (utilizando os mesmos arquivos auxiliares e/ou outros testes). O resultado dessa correção ficará anotada na planilha de notas disponibilizada para cada aluno.

As tarefas serão corrigidas pelo sistema de correção automática e por um monitor. Uma vez terminada a tarefa e após ela ter passado nos testes automáticos, deve-se **solicitar a correção ao monitor** clicando-se no botão correspondente da interface de notas (com exceção das tarefas que forem marcadas para correção automática apenas). Algumas tarefas determinadas (especificadas no enunciado), deverão ser *apresentadas pessoalmente* a um monitor presencialmente ou por meio de videoconferência de maneira sucinta nos horários de atendimento disponíveis. Além disso, @s estudantes são encorajad@s a mostrar e conversar sobre suas dúvidas e seus códigos com os monitores, via chat e videoconferência, sempre que desejarem nos canais de atendimento.

A cada tarefa será atribuído um conceito com os seguintes valores e significados:

- **A** (valor 10): a tarefa foi executada satisfatoriamente; deve-se prosseguir à proxima atividade;
- **B** (valor 8): os objetivos mínimos foram alcançados, mas há questões pontuais que podem ser melhoradas; deve-se prosseguir à proxima atividade;
- **C** (valor 6): nem todos os objetivos mínimos foram alcançados; a tarefa é considerada aprovada, mas recomenda-se corrigi-la e submeter novamente;
- **D** (valor 0): a tarefa não foi realizada, ou os objetivos não foram satisfeitos; é obrigatório refazer a tarefa.

Prazos

A disciplina adotará um método de avaliação contínua e individualizado. Assim, as tarefas serão divulgadas continuamente, de acordo com os conteúdos ministrados, mas cada um@ poderá levar mais tempo ou menos tempo para executá-las, de acordo com seu aprendizado sobre o conteúdo. As regras são as seguintes:

- Cada tarefa terá um prazo recomendado de pelo menos uma semana a partir da publicação. Se uma tarefa for entregue depois do prazo recomendado, será **descontado 10%** da nota correspondente.
- Todas as tarefas são **obrigatórias** e devem ser entregues até **14/7/2022**.
- Uma tarefá só será corrigida após as tarefas anteriores tiverem sido realizadas corretamente (tiverem recebido conceitos A, B ou C).

É responsabilidade de cada um@ organizar e dividir o tempo para realizar cada atividade. É proibido acumular tarefas deliberadamente: estudantes com tarefas atrasadas que não tiverem nenhuma atividade no repositório por duas semanas deverão apresentar justificativa ao professor. Caso um estudante não responda as mensagens de aviso do professor, será presumida a desistência da disciplina e a correção de suas tarefas será bloqueada. Recomenda-se fortemente que se realize a maior parte dos exercícios possível durante as aulas de laboratório e não se deixem tarefas pendentes acumuladas.

Avaliação

A nota da disciplina será a média ponderada de duas partes

1. A média aritmética *M* das notas de todas as tarefas, valendo de 0 a 10.
2. A nota de participação *P* do estudante atribuída pessoalmente pelo professor, valendo de 0 a 10.

Serão considerados aprovados os estudantes que tiverem pelo menos 75% de presença e obtido conceito pelo menos C em todas as tarefas. Nesse caso, a nota de aproveitamento será ***A=0,7M+0,3P***. Do contrário, a nota de aproveitamento do semestre será ***A=mínimo{4,M}***.

Nota de participação

Para calcular a nota de participação serão levadas em conta tanto a participação nas aulas online quanto a realização das demais atividades. Isso será monitorado considerando atividade regular no repositório git bem como a participação nas aulas, no chat de discussão e no atendimento. Para garantir a nota máxima de participação, deve-se:

- Manter atividade regular no repositório git, realizando inclusões e alterações de pelo menos **200** linhas de código no repositório em cada semana (até 7 pontos). Vocês podem incluir no repositório trechos de códigos das **listas de exercícios para fixação** realizados em grupo, mas é obrigatório fazer um comentário indicando as pessoas com quem colaborou.
- Participar continuamente das aulas e/ou dos canais de chat, compartilhando dúvidas, respondendo perguntas ou fazendo comentários pertinentes sobre a disciplina (até 3 pontos).

Exame

Aquel@ estudante com pelo menos 75% de presença e $A \geq 2,5$ que não tiver conseguido completar todas as tarefas até o prazo poderá realizar exame. Como exame, o professor solicitará a realização de um subconjunto das tarefas com prazo de pelo menos uma semana e atribuirá nota *E*. O prazo para a entrega das tarefas do exame é 26/7/2022. Para estudantes que fizerem o exame, a nota final será ***mínimo{5,(A+E)/2}***.

Fraude

Em caso de fraude (plágio, atestado falso, assinar lista por colegas, abandonar aula após assinar, usar bibliotecas não permitidas, copiar quaisquer trechos da internet sem autorização expressa, mostrar ou distribuir laboratório de programação individual, cola independentemente de origem, consulta a material proibido etc.), os envolvidos serão reprovados com nota 0 e será registrada a ocorrência no histórico escolar. Fique atento:

- Cada um@ é responsável por manter seguros os arquivos de seu repositório. Não compartilhe sua senha nem deixe cópias de arquivos em

computadores compartilhados. **Todas** as tarefas são individuais.
- É proibido utilizar ajuda de terceiros (amigos, professores particulares, etc.) para realizar as tarefas sem autorização expressa do professor da disciplina.
- É permitido compartilhar e discutir sobre trechos de código apenas dos exercícios para fixação. Sempre que utilizar trechos de código de outras pessoas ou de outras fontes, deve-se indicar isso explicitamente.
- É permitido tirar dúvidas e discutir a respeito de tarefas de programação com colegas e monitores. Mas toda discussão deve ser feita em canais abertos no Discord e não é permitido compartilhar algoritmos para resolver a tarefa (em português, código ou pseudocódigo) nem ideias prontas de solução etc.
- O material disponibilizado e os trabalhos apresentados nas avaliações são para uso exclusivo da disciplina. Não compartilhem seus trabalhos mesmo após o término da disciplina. Se desejar publicar algum trabalho que realizou, converse antes com o professor.
- Cláusula de arrependimento: se você cometer qualquer tipo de fraude, mas reconhecê-la **antes** de ser comunicad@, então o professor irá relaxar as ações acima por ações locais.

Conteúdo e bibliografia

A disciplina não seguirá nenhum livro específico. Serão utilizadas diversas fontes bibliográficas, a depender do conteúdo. A parte introdutória da disciplina abordará conceitos fundamentais para algoritmos e será baseada principalmente nos capítulos introdutórios das referências [1] e [2]. Essas referências estão em inglês, mas as principais definições serão resumidas em português na página da disciplina.

1. D. Harel. *Algorithmics*, Addison Wesley, 3ª edição, 2004.
2. D. Harel. *Computers Ltd.: what they really can't do*, Oxford Univ. Press, 2000.

As fases seguintes abordarão as principais construções da linguagem de programação Python e serão baseadas principalmente no tutorial disponibilizado na documentação oficial do Python. O tutorial original é escrito em inglês [3], mas a versão traduzida é atualizada e igualmente recomendada [4].

3. *The Python Tutorial*, https://docs.python.org/3/tutorial/
4. *O Tutorial Python*, https://docs.python.org/pt-br/3/tutorial/

Alguns conteúdos e definições mais formais serão extraídos da referência [5], particularmente a respeito de teste e práticas de programação. Também, vários

exemplos e exercícios podem ser extraídos da referência [6], embora não abordaremos alguns assuntos mais avançados desse livro.

5. Guttag, John V. *Introduction to computation and programming using Python*, Mit Press, 2013.
6. Sedgewick, Robert, Kevin Wayne, and Robert Dondero. *Introduction to programming in Python: An interdisciplinary approach*, Addison-Wesley Professional, 2015.

Além das referências anteriores, pesquisar com cuidado na internet também é um bom jeito de aprender. Se você é proficiente em inglês e gosta de realizar atividades online, então pode ser divertido seguir o livro interativo [7], mas ele segue uma ordem um pouco diferente da adotada na disciplina. Existe uma tradução em português desse livro [8] disponibilizada pelo IME/USP, mas ela não está tão atualizada e contém alguns problemas nos códigos interativos.

7. Miller, B. et al. *How to Think Like a Computer Scientist: Interactive Edition*, https://runestone.academy/ns/books/published/thinkcspy/index.html
8. Miller, B. et al. *Aprendendo com Python: Edição interativa*, https://panda.ime.usp.br/pensepy/static/pensepy/index.html

Nesta disciplina, usaremos principalmente lápis e papel para simular e entender trechos de código, mas algumas vezes pode ser conveniente utilizar alguma ferramenta interativa, como [9]. Ferramentas como essa devem ser utilizadas apenas no início do curso para trechos muito curtos de código; com o decorrer do tempo, utilizaremos ferramentas de teste e depuração.

9. Philip J. Guo. *Online Python Tutor: Embeddable Web-Based Program Visualization for CS Education*. In: SIGCSE, 2013. http://pythontutor.com/visualize.html

Material didático

A página da disciplina conterá um material de apoio, incluindo uma breve discussão sobre cada unidade, algumas definições e exemplos utilizados em aula. Esse material **não** é uma referência completa do conteúdo e existe apenas para auxiliar o professor durante as aulas expositivas ou, depois, para relembrar o que foi discutido em sala. Para o estudo, é recomendado que @ estudante, principalmente, pratique programação resolvendo e implementando os exercícios propostos e, além disso, busque e estude exemplos de código-fonte nos capítulos correspondentes dos livros-textos. A página também inclui alguns tutoriais para instalar o ambiente, configurar um editor, utilizar git e depurar

programas. Esses tutoriais foram gentilmente criados por monitores de diversos semestres.

Aulas e atendimento

As aulas expositivas com o professor serão **presenciais** na sala 352 no IC 3,5 às terças das 21 às 23h e às quintas das 19 às 21h. A sala tem capacidade apenas para 54 alunos; sendo assim, os estudantes que chegarem depois serão encaminhados para a sala 353, que terá a aula transmitida.

As aulas de laboratório ocorrerão na sala 300 no IC 3. Nessas aulas, os monitores irão esclarecer o enunciado das tarefas e tirar outras dúvidas, além de fornecer o atendimento.

O atendimento e comunicação entre estudantes, professor e monitores serão feitos via Discord. Cada estudante receberá um link por e-mail com o convite para se cadastrar no servidor. Você pode utilizar uma conta de Discord pessoal existente, mas deve alterar seu apelido visível no servidor da disciplina para seu nome completo e número de RA (e.g., `Fulano da Silva (123456)`). Se preferir, também pode criar uma conta nova com e-mail institucional.

É recomendável utilizar o Discord a partir do computador (utilizando o programa para desktop ou o site), mas também é possível baixar e instalar o aplicativo de celular disponível.

Todos podem escrever e ler as mensagens nas salas de atendimento e vocês são **fortemente encorajados** tanto a fazer perguntas quanto a responder dúvidas de colegas. O professor e os monitores acompanharão as conversas e tentarão responder sempre que possível. Pelo menos um monitor ficará disponível no chat nos horários de atendimento marcados. A tabela com os horários de antedimento estará disponível no Discord e pode sofrer alterações de acordo com a demanda de estudantes e disponibilidade dos monitores.

Observações importantes:

1. Observe os horários de atendimento com atenção e siga as regras descritas no servidor do Discord.

2. Para correção de tarefas que peçam apresentação, compareça a um atendimento (presencial ou virtual) com um monitor PED.

3. O professor responderá dúvidas gerais durante as aulas após a parte inicial com a exposição do conteúdo. Para conversar pessoalmente e individualmente com o professor, você deve solicitar um horário ao

professor via Discord com antecedência, que poderá atendê-lo ou na sala do professor (sala 12), ou via videoconferência, a depender da disponibilidade.

(horários de atendimento ainda serão definidos)