

UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E BIOLÓGICAS
DEPARTAMENTO DE COMPUTAÇÃO

FERNANDO EUZEBIO ZIMERMAN

Orientador: Reinaldo Silva Fortes

Coorientador: MSc. Fernando Henrique Oliveira Duarte

**EXPLORANDO CHATBOTS BASEADOS EM LLM PARA CRIAÇÃO
AUTOMÁTICA DE QUESTÕES PRÁTICAS DE PROGRAMAÇÃO DE
COMPUTADORES**

Ouro Preto, MG
2024

UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E BIOLÓGICAS
DEPARTAMENTO DE COMPUTAÇÃO

FERNANDO EUZEBIO ZIMERMAN

**EXPLORANDO CHATBOTS BASEADOS EM LLM PARA CRIAÇÃO AUTOMÁTICA
DE QUESTÕES PRÁTICAS DE PROGRAMAÇÃO DE COMPUTADORES**

Monografia apresentada ao Curso de Ciência da Computação da Universidade Federal de Ouro Preto como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Reinaldo Silva Fortes

Coorientador: MSc. Fernando Henrique Oliveira Duarte

Ouro Preto, MG
2024

Resumo

Com o avanço acelerado da tecnologia alcançando cada vez mais áreas fora da computação e influenciando diversos campos acadêmicos e profissionais, a proficiência em programação de computadores emerge como uma habilidade essencial, mesmo para aqueles não diretamente ligados à área de computação, tornando a disciplina de programação de computadores de extrema importância, trazendo consigo o desafio de tornar a disciplina cada vez mais atraente para os alunos. Sendo assim, torna-se imprescindível uma variedade das atividades práticas ao decorrer da disciplina e, lidar com a criação de um banco de questões diversificadas, bem elaboradas, abordando os tópicos de programação e temas da graduação dos alunos manualmente, pode ser um grande obstáculo. Obstáculo esse que poderia ser mitigado pelo uso de ferramentas que utilizem técnicas de Processamento de Linguagem Natural, como o modelo GPT empregado nas ferramentas ChatGPT, ChatPDF, PopAi e Sider, para automatizar a geração, escalabilidade e diversidade das questões que servirão de base para as diversas atividades aplicadas aos alunos. Portanto, o presente trabalho teve em vista realizar experimentos por meio de estratégias de interação com essas ferramentas, visando a viabilidade de seu uso para criação automática de questões por meio de informações e modelos padrões informados via prompt.

Palavras-chave: Processamento de Linguagem Natural, GPT, ChatGPT, Programação, Ensino, Criação de questões, Correção automática.

Abstract

With the rapidly advancing technology permeating various non-computing domains and influencing various academic and professional fields, proficiency in computer programming is emerging as an essential skill, even for those not directly connected to the computing domain. This underscores the significance of computer programming as a discipline, bringing with it the challenge of making the subject more appealing to students. Consequently, a variety of hands-on activities throughout the course becomes crucial. However, dealing with the manual creation of a diverse and well-crafted question bank covering programming topics and students' academic themes can be a significant obstacle. This obstacle could be alleviated through the use of tools employing Natural Language Processing techniques, such as the GPT model utilized in tools like ChatGPT, ChatPDF, PopAi, and Sider. These tools can automate the generation, scalability, and diversity of questions that serve as the foundation for various student activities. Therefore, this study aimed to conduct experiments through interaction strategies with these tools, assessing the feasibility of their use for the automatic creation of questions based on provided prompt information and standard models.

Keywords: Natural Language Processing, GPT, ChatGPT, Programming, Teaching, Question creation, Automatic correction.

Lista de Ilustrações

Figura 2.1 – Interface ChatGPT	9
Figura 2.2 – Interface ChatPDF	11
Figura 2.3 – Interface PopAi	12
Figura 2.4 – Interface Sider	13
Figura 3.1 – Fluxograma do teste de Interação adaptativa	18
Figura 3.2 – Fluxograma do teste de Estrutura de prompt	19
Figura 3.3 – Fluxograma do teste de Instruções personalizadas	20
Figura 3.4 – Fluxograma do teste de Estrutura de prompt com arquivo PDF	21
Figura 3.5 – Interface Custom Instructions	33
Figura 3.6 – Instruções personalizadas mais genéricas	37
Figura 3.7 – Instruções personalizadas mais diretas	38

Lista de Tabelas

Tabela 4.1 – Avaliação dos retornos das ferramentas	47
---	----

Lista de Abreviaturas e Siglas

DECOM	Departamento de Computação
UFOP	Universidade Federal de Ouro Preto
PLN	Processamento de Linguagem Natural
IA	Inteligência Artificial
GPT	<i>Generative Pre-Trained</i>
PLM	<i>Pre-trained Language Model</i>
ChatGPT	<i>Chat Generative Pre-trained Transformer</i>
EEAF	Entrevista Exploratória de Avaliação de Funcionalidades
IO	<i>Input/Output</i>
MRU	Movimento Retilíneo Uniforme
PDF	<i>Portable Document Format</i>
API	<i>Application Programming Interface</i>

Sumário

1	Introdução	1
1.1	Justificativa	3
1.2	Objetivos	4
1.3	Organização do Trabalho	4
2	Revisão Bibliográfica	5
2.1	Trabalhos Relacionados	5
2.2	Fundamentação Teórica	6
2.2.1	Programação de Computadores I	6
2.2.2	Prompt	7
2.2.2.1	Engenharia de Prompt	7
2.2.3	Processamento de Linguagem Natural	7
2.2.3.1	Modelo de Linguagem	8
2.2.3.2	GPT	8
2.2.3.3	Chatbot	8
2.2.4	Ferramentas	9
2.2.4.1	ChatGPT	9
2.2.4.2	ChatPDF	10
2.2.4.3	PopAi	11
2.2.4.4	Sider	12
3	Desenvolvimento	14
3.1	Visão Geral	14
3.1.1	Esquema Padrão	14
3.2	Metodologia	16
3.2.1	Estratégias	17
3.2.1.1	Interação adaptativa	17
3.2.1.2	Estrutura de prompt	18
3.2.1.3	Instruções personalizadas	19
3.2.1.4	Estrutura de prompt com aquecimento por arquivo PDF	20
3.3	Testes com interação adaptativa	21
3.4	Testes com engenharia de prompt	29
3.5	Testes com instruções personalizadas do ChatGPT	33
3.6	Testes com estrutura de prompt e aquecimento por arquivo PDF	39
3.6.1	Testes a partir do Promp 1	39
3.6.2	Testes a partir do Prompt 2	42
4	Resultados	45
4.1	Resultado da estratégia de Interação adaptativa	45

4.2	Resultado da estratégia de Estrutura de prompt	45
4.3	Resultado da estratégia de Instruções personalizadas	45
4.4	Resultado da estratégia Estrutura de prompt com aquecimento por arquivo PDF	46
4.4.1	Avaliação de métricas	46
5	Considerações Finais	49
5.1	Conclusão	49
5.2	Trabalhos futuros	49
	Referências	51
	 Apêndices	 52
	APÊNDICE A Testes no estilo conversação com ChatGPT	53
	APÊNDICE B Testes a partir do prompt criado	69
	B.1 Tópicos separados	69
	B.2 Todos os tópicos	74
	APÊNDICE C Testes com instruções personalizadas	81
	C.1 Testes verificando limitações das instruções personalizadas	92
	APÊNDICE D Documento usado para testes com os leitores de PDF	93
	APÊNDICE E Testes ChatPDF	112
	APÊNDICE F Testes PopAi	126
	APÊNDICE G Testes Sider	142
	 Anexos	 149
	ANEXO A Ementa BCC701	150

1 Introdução

A tecnologia vem se tornando cada vez mais presente e desempenha um papel cada vez mais relevante no nosso dia a dia, e pode mudar como lidamos com o mundo ao nosso redor. O rápido avanço da tecnologia tem se estendido cada vez mais além dos domínios tradicionais, exercendo influência sobre uma ampla gama de campos acadêmicos e profissionais. Nesse contexto, é notável importância da tecnologia e da competência digital dos estudantes para o sucesso acadêmico e profissional e necessidade de os educadores incorporarem tecnologia no ensino para promover a literacia digital dos alunos, como demonstrado por [M. Fridorff-Jens P.K. \(2015\)](#).

No entanto, o aprendizado da programação e a criação de uma base sólida através da disciplina de Programação de Computadores, pode apresentar tarefas desafiadoras para estudantes que não estão familiarizados com a área, como, por exemplo, interpretar conceitos abstratos da programação, compreender as estruturas e sintaxes de uma linguagem, entender a lógica para criação de algoritmos eficientes e até mesmo manter o interesse quando as atividades não são baseadas no contexto da graduação. A programação é normalmente ensinada como uma disciplina básica no início de um curso de ensino superior e, de acordo com [Gomes e Mendes \(2007\)](#), Alguns autores consideram que as disciplinas de programação estão mal localizadas no currículo, pois este é um momento de muitas dificuldades e novidades para uma vida nova e autônoma.

No contexto do desenvolvimento e apoio ao aprendizado dos alunos da disciplina de Programação de Computadores, são usadas algumas ferramentas no decorrer da disciplina. Dentre elas podemos citar os corretores automáticos de exercícios de programação (ou juízes *online*), ferramentas que são uma espécie de juízes da programação e, em sua maioria, realizam a correção e apresentam *feedbacks* em tempo real para o aluno identificar e corrigir seus erros o mais rapidamente possível. Em seu artigo, [Brito \(2019a\)](#) demonstra a importância do uso dos corretores e seus *feedbacks* rápidos e explicativos na metodologia de ensino na disciplina.

Dentre algumas dessas ferramentas, pode-se citar o *Beecrowd*¹, uma plataforma online que disponibiliza desafios de programação que podem ser submetidos para correção na mesma de forma instantânea. Seguindo essa premissa, temos o *opCoders Judge*, trabalho realizado por [Patrocínio \(2023\)](#), onde foi implementado um módulo web do corretor automático existente e utilizado atualmente na Universidade Federal de Ouro Preto (UFOP), previamente implementado de forma *offline* por [Brito \(2019b\)](#).

Ainda no contexto da disciplina, um dos desafios é a necessidade da criação de um banco de questões abrangentes, diversificadas, e que sejam interessantes para o desenvolvimento do

¹ <https://www.beecrowd.com.br>

aprendizado dos alunos. Se faz necessário uma estrutura padrão que permita a geração de uma variedade de atividades, como aulas práticas e avaliações, considerando elementos como:

- Tema a ser usado na questão;
- Dificuldade do tema;
- Recursos da programação a serem utilizados;
- Dificuldade da implementação.

Apesar da ampla gama de opções para a criação dessas questões, criar e revisar uma quantidade tão grande de questões manualmente, tendo que considerar cada detalhe pode se tornar uma tarefa cansativa e complicada.

Sendo assim, para tentar automatizar esse processo, uma das áreas que podemos explorar é a área de Processamento de Linguagem Natural (PLN). Uma área da Inteligência Artificial (IA) que explora a manipulação de texto. Como mostrado em (CAMBRIA; WHITE, 2014), a evolução do PLN está caminhando em direção a uma compreensão mais profunda e abrangente da linguagem humana. Com avanços significativos em modelos computacionais que buscam emular como o cérebro humano processa a linguagem natural, estamos testemunhando uma transformação na capacidade dos sistemas de PLN em interpretar contextos, nuances e até mesmo emoções expressas na linguagem escrita.

Partindo, então, do ponto que as questões utilizadas pela disciplina de programação de computadores possuem um padrão bem definido. Dessa forma, pode-se usar ferramentas implementadas com PLN para reconhecer esse padrão e usá-los para gerar novas questões de forma automática.

Atualmente, uma ferramenta amplamente conhecida é o modelo generativo Chat *Generative Pre-trained Transformer* (ChatGPT). A ferramenta desenvolvida e disponibilizada pela OpenAI² possui o formato de *chatbot*, posteriormente explicado, mas que basicamente é um programa que busca simular um ser humano em uma conversação. A ferramenta usa um modelo de linguagem pré-treinado para manipular, entender e retornar respostas em linguagem natural, se mostrando muito promissora quanto aos interesses deste trabalho. Além dela, também são exploradas outras 3 ferramentas ao decorrer deste trabalho, são elas:

- *ChatPDF*:³ É uma ferramenta que permite com que façamos o *upload* de arquivos *Portable Document Format* PDF e, automaticamente, o ChatPDF gera uma síntese do documento, transformando-o em um *chatbot* que permite aos usuários fazerem perguntas e receberem respostas relacionadas ao conteúdo.

² <https://openai.com/>

³ <https://www.chatpdf.com>

- *PopAI*⁴ É uma ferramenta projetada na intenção de ser um companheiro virtual e que apresenta uma variedade de recursos, incluindo *chatbot* com capacidade para lidar com documentos em PDF e imagens geração de apresentações, entre outros.
- *Sider*⁵ É um ferramenta com a proposta de ser um parceiro de inteligência artificial, proporcionando suporte em tarefas cotidianas. A aplicação se destaca por sua versatilidade, oferecendo recursos de PLN para leitura e escrita de texto, manipulação de arquivos PDFs e manipulação de imagens.

Todas elas possuem em comum o modelo de linguagem GPT (Generative Pre-Trained) disponibilizado pela *OpenAI*. Além de também serem ferramentas em formato *chatbot*, elas possuem a funcionalidade de interpretação de PDFs. Funcionalidade que pode ser usada para aquecimento do modelo de tal forma a direcioná-lo para os resultados esperados.

Todas elas são ferramentas que conseguem interagir com arquivos PDF extraindo os textos contidos nos mesmos e, a partir daí, dão a opção de interação com as informações contidas no arquivo.

A proposta deste trabalho, portanto, é explorar ferramentas que utilizem o modelo generativo GPT para chegar na geração automática de questões considerando os diferentes contextos das áreas da graduação, com a intenção de criar um banco de questões para complementar a disciplina de Programação de Computadores, tal qual ser integrado com o módulo online do corretor utilizado atualmente na UFOP, o *opCoders Judge*. O objetivo é estruturar métodos que garantam a geração de questões práticas e teóricas em temas de diferentes disciplinas aplicadas no contexto da programação.

A Subseção 1.1 procura descrever a necessidade do trabalho. A Subseção 1.2 apresenta os motivos que impulsionam a realização do trabalho. E por fim, a Subseção 1.3 descreve como o trabalho está organizado.

1.1 Justificativa

A programação está incluída em diversas áreas e visa auxiliar a checagem e minimização de erros, na automatização e otimização de processos repetitivos. A familiarização com a programação pode aumentar a capacidade de raciocínio lógico dos alunos, uma maior habilidade de resolução de problemas e familiaridade com a tecnologia. Habilidades que são muito valiosas e vem se tornando cada vez mais importante nas diferentes áreas de estudo e no dia a dia.

Existe um desafio em ensinar a disciplina de programação de computadores para os estudantes como mostra [Gomes e Mendes \(2007\)](#). Por isso, torna-se interessante a utilização de uma abordagem personalizada, trazendo temas de suas áreas de atuação para dentro da

⁴ <https://www.popai.pro>

⁵ <https://sider.ai>

programação, nesse contexto o aprendizado se torna mais relevante e compreensível para os alunos, estabelecendo uma conexão entre os conceitos técnicos e aplicações de suas áreas.

Como mostra o artigo (FUCHS, 2023), os modelos de PLN oferecem oportunidades promissoras e desafios intrigantes no ensino superior, podendo transformar o ensino e a aprendizagem nesse contexto. A tecnologia GPT destaca-se por sua eficácia no processamento de linguagem natural, capacitando a criação de questões personalizadas para diversos domínios acadêmicos. Por isso, para a criação de questões explorou-se neste trabalho as ferramentas que possuem a tecnologia GPT. Ferramentas poderosas no que diz respeito a linguagem natural, que tem potencial para criar questões personalizadas e relevantes, contextualizando nos diferentes domínios e áreas da graduação dos alunos.

1.2 Objetivos

Este trabalho aspira estudar métodos de criação automática e dinâmica de questões práticas para a disciplina de Programação de Computadores com o modelo generativo GPT através de ferramentas que o usam. A partir de um modelo padrão de questões, pretende-se obter resultados que levem ao entendimento de como as ferramentas que utilizam o modelo atuam com diferentes temas, dentre as diversas áreas de graduação, na criação de questões. Buscando entender como se pode otimizar e automatizar uma ferramenta para atuar em diversos contextos acadêmicos onde existe uma ampla complexidade de recursos de linguagem e estabelecer a dinâmica para o nivelamento das questões.

E como objetivos específicos temos:

- Facilitar a geração de questões de programação padronizadas de forma automática usando uma ferramenta que funcione com o modelo GPT;
- Permitir a criação automática de várias versões de uma mesma questão e tema específico, indicando apenas alguns pontos, como matéria usada no contexto da questão, assunto da matéria e tópicos de programação abrangidos na questão;

1.3 Organização do Trabalho

Este trabalho está organizado nos seguintes capítulos: O Capítulo 2 apresenta a revisão bibliográfica, o Capítulo 3 apresenta o desenvolvimento do trabalho, o Capítulo 4 apresenta os resultados obtidos e, por último, o Capítulo 5 apresenta as considerações finais do trabalho.

2 Revisão Bibliográfica

Neste capítulo serão abordados os trabalhos que tenham relação com o tema e utilizam da ferramenta central desta monografia em suas pesquisas. Buscando assim obter uma base sólida de conhecimento e contextualização do tema e ferramentas abrangidos, tal qual explorar as vantagens, benefícios e artifícios que obtiveram sucesso tanto quanto limitações e dificuldades nas pesquisas realizadas.

2.1 Trabalhos Relacionados

O estudo realizado por [Radford et al. \(2018\)](#) explora uma abordagem semi-supervisionada com tarefas relacionadas à compreensão de linguagem e ilustra os avanços obtidos ao empregar pré-treinamento de um modelo de linguagem não rotulado. Foi constatado que o aprendizado não supervisionado poderia desempenhar um papel eficaz na amplificação do desempenho em tarefas que exigem compreensão de linguagem natural, oferecendo perspectivas sobre os benefícios e o potencial funcionamento dessa abordagem em contextos de compreensão linguística. As conclusões deste trabalho aprofundaram a percepção e compreensão da linguagem natural, fazendo com que chegássemos até a ferramenta central do nosso estudo. A partir deste artigo que aconteceu a introdução do conceito de GPT aos modelos de linguagem, onde, em cima dele, puderam ser desenvolvidos diversos estudos exploratórios e melhorias no âmbito da PLN chegando até o GPT 3, que compõe o ChatGPT.

O trabalho de [Rothe, Narayan e Severyn \(2020\)](#) apresenta uma abordagem que realiza a combinação de modelos pré-treinados, gerando um modelo *sequence-to-sequence* (*seq2seq*) baseado em *Transformer* e compatível com os pontos de verificação dos modelos BERT, GPT-2 e RoBERTa, representantes proeminentes da vanguarda em processamento de linguagem natural. Vale destacar que, enquanto o BERT enfatiza a análise bidirecional do contexto, o GPT-2 se destaca na geração de texto coerente, e o RoBERTa otimiza a abordagem do BERT para obter representações mais robustas. A pesquisa conduziu experimentos visando determinar as estratégias mais eficazes para a utilização de pontos de verificação pré-treinados em modelos voltados para a geração de sequências textuais, essenciais para aplicações com tarefas como tradução automática, resumo de texto e geração de código.

[Kothari \(2023\)](#) apresenta uma análise realizada em cima da IA generativa, incluindo tecnologias como o ChatGPT, buscando destacar o potencial para aprimorar a precisão, a eficiência e a acurácia no tratamento do câncer, resultando em melhores desfechos para os pacientes e em abordagens de tratamento mais personalizadas. Foi possível perceber que, a possibilidade de interagir com os modelos e ajustá-los oferece uma vasta possibilidade de concepções de soluções inovadoras, podendo englobar a exploração de estratégias para otimizar o cuidado ao paciente

por meio de abordagens linguísticas orientadas no modelo da linguagem.

Já o estudo realizado de Santos (2023) descreve a aplicação experimental da metodologia EEAF (Entrevista Exploratória de Avaliação de Funcionalidades) usando o modelo generativo ChatGPT. Partindo então para a exploração da ferramenta com seu modelo treinado. A proposta metodológica do estudo se baseou na ideia de engenharia reversa, onde a coleta dos dados acontece por meio de perguntas diretas à interface de comunicação do modelo para entender suas principais características. Tal proposta também compõe a metodologia do trabalho e serviu de base para a elaboração da conversa realizada via chat com as ferramentas.

2.2 Fundamentação Teórica

Esta seção visa contextualizar o estudo em questão buscando abordar e discutir os conceitos e temas diretamente ligados à ferramenta central deste trabalho.

A Subseção 2.2.1 fala sobre a disciplina de programação ofertadas nos cursos de graduação da Universidade Federal de Ouro Preto e que serve como base para o trabalho. A Subseção 2.2.2 fala sobre prompt, um dos meios utilizados para interagir com as ferramentas exploradas e que está envolvido em uma das estratégias usadas nesse trabalho. A Subseção 2.2.3 fala de um dos principais artefatos, PLN envolvidos neste trabalho e seus derivados. A Subseção 2.2.4 apresenta as ferramentas que serão usadas para o desenvolvimento do trabalho.

2.2.1 Programação de Computadores I

Programação de Computadores I (BCC701) é uma disciplina oferecida aos cursos de graduação da Universidade Federal de Ouro Preto que não estão ligados diretamente à computação, mas que possuem o aprendizado da programação em sua ementa. Nessa disciplina, os alunos aprendem os conceitos básicos de programação, tais como variáveis, tipos de dados, operadores, estruturas de controle, funções, vetores e matrizes, usando a linguagem C, uma linguagem de programação de alto nível, conhecida por sua eficiência e flexibilidade, sendo amplamente utilizada para o desenvolvimento de sistemas e software devido à sua capacidade de proporcionar controle direto sobre o *hardware*.

A disciplina oferecida toma como público-alvo os cursos não pertencentes à área de computação e tem como base um livro texto desenvolvido durante o semestre letivo 2019-02, resultado da colaboração entre a Comissão de Unificação da disciplina e docentes estagiários do Departamento de computação (DECOM) daquela época e conta com os seguintes tópicos abordados na ementa, disponibilizada no Anexo A: (a) Variáveis, expressões, entradas e saídas; (b) Decisão simples (if, else); (c) Decisão aninhada (if, elif, else); (d) Laços controlados logicamente ou por contagem; (e) Laços aninhados; (f) Funções; (g) Estruturas de dados homogêneos (vetor e matriz); (h) Estrutura de dados heterogêneos (registros). Para complementar os recursos de

aprendizagem, a disciplina utiliza o *opCoders Judge*, um corretor online que auxilia no processo de desenvolvimento dos alunos.

O contexto do trabalho está diretamente ligado à disciplina, visando a automação na geração de questões como meio de aprimorar a eficiência do trabalho manual desempenhado pelos professores na elaboração de atividades práticas para o desenvolvimento da mesma. Para alcançar esse objetivo, pretende-se utilizar algumas ferramentas no trabalho que se mostram promissoras para criação automática de questões, integrando-a ao banco de questões atualmente empregado no *opCoders Judge*. Essa abordagem tem como propósito enriquecer o material didático e as atividades práticas oferecidas na disciplina.

2.2.2 Prompt

O conceito de prompt representa uma instrução passada em linguagem natural que orienta um LLM durante a execução de uma tarefa específica. Também conhecido como ajuste de instruções, esse processo envolve a condução do modelo via texto para determinar a estrutura e o conteúdo a ser gerado.

No contexto deste trabalho, podemos chamar de prompt os textos passados via chat para as ferramentas aqui abordadas. Através deles, por meio de tecnologias de *machine learning* e processamento de linguagem natural, o que é uma pequena descrição textual, o nosso prompt, é interpretado.

2.2.2.1 Engenharia de Prompt

A Engenharia de Prompt, por sua vez, é o processo de criação e aprimoramento dessas instruções utilizadas pelo modelo. Surge como uma disciplina focada no desenvolvimento e otimização de prompts, visando a eficiente utilização de modelos de linguagem (LMs) em diversas aplicações e áreas de pesquisa. Esta abordagem tem como objetivo principal aprimorar a compreensão das capacidades e limitações dos grandes modelos de linguagem (LLMs).

Através dela pode-se entender melhor os recursos e limitações dos modelos de linguagens grandes, o que nos ajuda a melhorar nossa estratégia de interação com o modelo. Podemos usar estratégias para guiar melhor o modelo na direção dos resultados que esperamos, oferecendo informações valiosas, como contextos, prioridades, restrições e até mesmo o modelo em que queremos a resposta gerada.

2.2.3 Processamento de Linguagem Natural

O PLN é uma área da IA que explora a manipulação de texto ou fala em linguagem natural por computadores. O conhecimento da compreensão e uso da linguagem humana é reunido para desenvolver técnicas que farão com que os computadores entendam e manipulem expressões naturais para realizar tarefas desejadas *chatbots*.

Atualmente, o estudo e a importância do PLN experimentam um crescimento significativo, impulsionado na maioria pelo aumento do uso de *chatbots* e a necessidade de uma maior proximidade entre a linguagem humana e o que é entendido pelos sistemas computacionais. E, como dito por Santos (2023), uma das vertentes mais promissoras da interface entre comunicação e sistemas computacionais são os modelos de processamento de linguagem natural. Tornando esse campo da IA de suma importância para o aprimoramento da interação humano-computador.

2.2.3.1 Modelo de Linguagem

Um modelo de linguagem é um tipo de IA abordada na área de PLN que visa entender e transformar em uma representação computacional as estruturas e padrões presentes na linguagem humana. Devido à grande capacidade em lidar com a linguagem humana, tem um uso crescente nos últimos anos, sendo muito úteis para várias aplicações, incluindo *chatbots*, tradução de idiomas e geração de conteúdo.

2.2.3.2 GPT

GPT é um modelo de linguagem pré-treinado (*Pre-trained Language Model*, PLM) desenvolvido pela OpenAI. Ele foi construído usando uma grande quantidade de documentos da internet e é capaz de gerar texto natural e coerente, quando alimentado com uma determinada tarefa ou contexto. GPT pode ser usado em várias aplicações, incluindo *chatbots*, geradores de conteúdo e sistemas de tradução automática (SANTOS, 2023).

2.2.3.3 Chatbot

Um chatbot é uma aplicação que responde como uma entidade inteligente quando conversado por meio de texto ou voz e compreende uma ou mais linguagens humanas por PLN *chatbots*. Esses programas utilizam o PLN e a IA para analisar as informações dos usuários e fornecer respostas relevantes, simulando conversas reais por meio de respostas automáticas e inteligentes.

Com o progresso contínuo da IA e do PLN, os *chatbots* estão se tornando cada vez mais sofisticados e eficientes nas interações com os usuários. Tem sido cada vez mais empregados em diferentes plataformas, como sites, aplicativos de mensagens instantâneas, assistentes virtuais e, com a intenção de reduzir os custos, nas empresas.

Recentemente um *chatbot* em específico ganhou destaque atingindo ao ser introduzido para o público geral, segundo Steve Mollman (2022), um número impressionante de mais de um milhão de usuários se inscreveu no ChatGPT nos cinco dias após seu lançamento, e se tornou um marco no avanço de PLN e GPT.

2.2.4 Ferramentas

Esta Subseção visa contextualizar e falar sobre as ferramentas utilizadas para os experimentos deste trabalho.

A Subseção 2.2.4.1 apresenta a ferramenta ChatGPT, uma ferramenta crucial para o desenvolvimento do trabalho. Já a Subseção 2.2.4.2 apresenta a ferramenta ChatPDF e suas funcionalidades. A Subseção 2.2.4.3 apresenta a ferramenta PopAi e descreve um pouco de suas capacidades. Já a Subseção 2.2.4.4 apresenta e fala sobre a ferramenta Sider, que visa ser um parceiro virtual.

2.2.4.1 ChatGPT

ChatGPT é um chatbot, um algoritmo que simula uma conversa com o usuário, que usa o modelo GPT para gerar respostas para perguntas, solicitações e comentários. Ele pode ser usado em uma variedade de contextos como atendimento ao cliente, treinamento de funcionários e conversas virtuais (SANTOS, 2023). Para usarmos a ferramenta é necessária uma conta na plataforma. A ferramenta é usada a partir de texto digitado na caixa de diálogo, a interface da ferramenta é representada pela Figura 2.1.

Figura 2.1 – Interface ChatGPT



Fonte: Website do ChatGPT, disponível em: <<https://chat.openai.com>>.

O interessante aqui é que podemos perguntar a própria ferramenta o que ela é, e teremos assim sua própria definição que, também acaba sendo, um pouco mais detalhada, como segue:

Usuário: O que é o ChatGPT?

ChatGPT: O ChatGPT é um modelo de linguagem desenvolvido pela OpenAI, que utiliza a arquitetura GPT (Generative Pre-trained Transformer). Ele foi treinado em uma grande quantidade de dados textuais para gerar respostas coerentes e contextualmente relevantes com base em uma entrada de texto. O GPT (Transformador Pré-Treinado Generativo) é um modelo de aprendizado de máquina que se destaca por sua capacidade de entender e gerar texto. Ele é treinado em uma tarefa de previsão de palavras, onde recebe uma sequência de palavras e tenta prever a próxima palavra correta. Essa abordagem permite que o modelo aprenda a capturar padrões e relações semânticas em um texto, adquirindo conhecimento sobre a estrutura da linguagem. O ChatGPT é uma versão específica do GPT projetada para interagir com os usuários por meio de um formato de bate-papo. Ele pode responder perguntas, fornecer informações, realizar tarefas simples e até mesmo participar de conversas informais. No entanto, é importante observar que o ChatGPT é uma simulação de inteligência artificial e não possui compreensão ou consciência real. Ele gera respostas com base em padrões aprendidos dos dados de treinamento, mas não tem conhecimento atualizado além de sua data de corte de conhecimento (*knowledge cutoff*). Embora o ChatGPT consiga fornecer informações úteis e interagir com os usuários convincentemente, ele também pode gerar respostas imprecisas ou incorretas em algumas situações. Portanto, é sempre recomendável verificar as informações fornecidas pelo modelo e não tomar suas respostas como sendo totalmente precisas ou confiáveis.

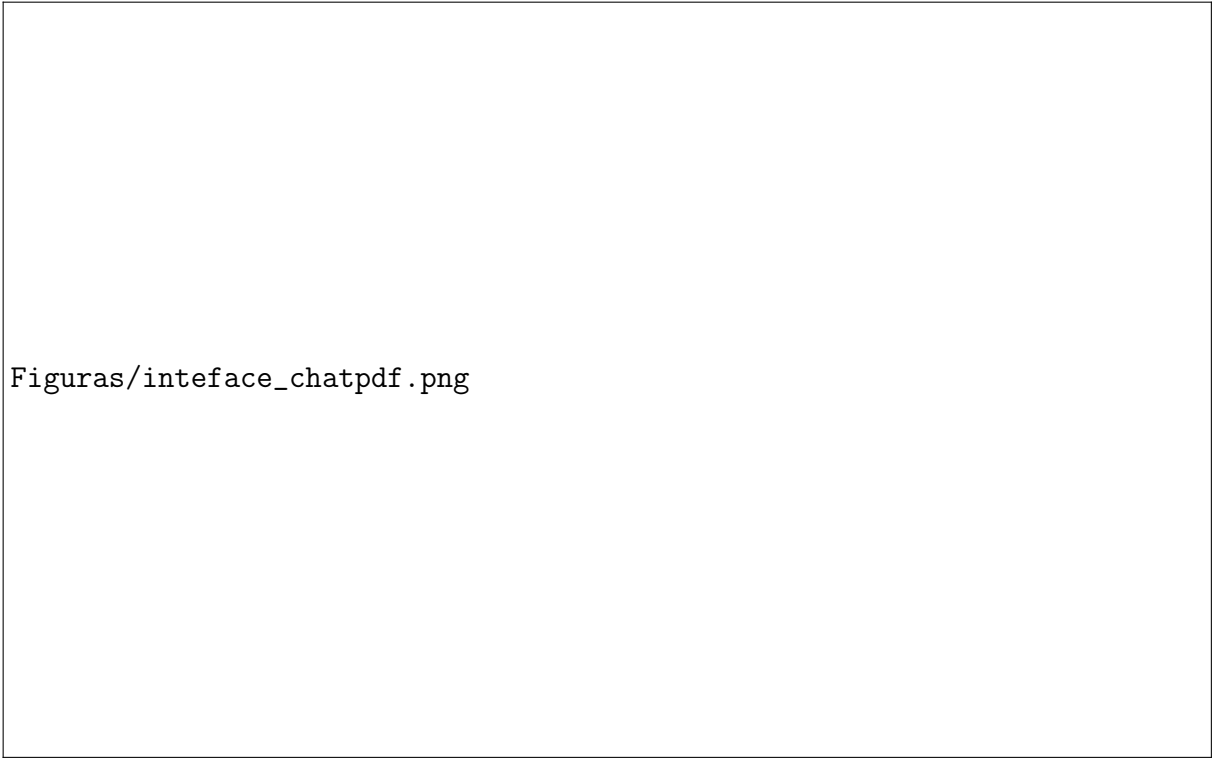
2.2.4.2 ChatPDF

O ChatPDF¹ é uma plataforma online que emprega IA para realizar a sumarização de documentos em formato PDF. Lançada em março, a ferramenta utiliza a tecnologia do ChatGPT, o modelo de linguagem GPT da OpenAI². Seu principal objetivo é extrair os pontos-chave de um arquivo PDF, transformando-os em três perguntas para facilitar a visualização e compreensão do conteúdo de forma resumida. A interface inicial da ferramenta é apresentada pela Figura 2.2

¹ <https://www.chatpdf.com>

² <https://openai.com/>

Figura 2.2 – Interface ChatPDF



Figuras/inteface_chatpdf.png

Fonte: Website do ChatPDF, disponível em: <<https://www.chatpdf.com>>.

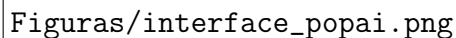
Basta realizar o *upload* do arquivo no campo do site, se atentando às limitações contidas na sua conta sendo ela gratuita ou não e, automaticamente, o ChatPDF gera uma síntese do documento, transformando-o em um chatbot que permite aos usuários fazerem perguntas e receberem respostas relacionadas ao conteúdo. Além disso, a plataforma possui a capacidade de compreender e traduzir textos em diversos idiomas, tornando-a uma ferramenta valiosa para estudantes, jornalistas e profissionais de diversas áreas.

2.2.4.3 PopAi

O PopAi³ é uma ferramenta de inteligência artificial que apresenta uma variedade de recursos, incluindo *chatbot* com capacidade para lidar com documentos em PDF e imagens, geração de apresentações, entre outros. Projetada para ser um companheiro virtual de IA, o PopAi tem como objetivo auxiliar em diversas tarefas, como a conclusão de trabalhos acadêmicos, recuperação de informações e a criação de apresentações dinâmicas no formato PowerPoint. A interface da parte de leitura de PDF da ferramenta é apresentada pela Figura 2.3.

³ <https://www.popai.pro>

Figura 2.3 – Interface PopAi



Figuras/interface_popai.png

Fonte: Website do PopAi, disponível em: <<https://www.popai.pro>>.

Uma das principais características que define o PopAi é sua capacidade interação com documentos, proporcionando uma navegação fluida, melhor legibilidade e acessibilidade. A ferramenta visa facilitar a compreensão e manipulação de documentos complexos, permitindo a navegação descomplicada, ampliação de detalhes e personalização do layout para máxima clareza. Basta realizar o *upload* do arquivo no campo do site, se atentando às limitações contidas na sua conta sendo ela gratuita ou não e, automaticamente a ferramenta

2.2.4.4 Sider

Sider⁴ foi desenvolvida com a proposta de ser um parceiro de inteligência artificial, proporcionando suporte em tarefas cotidianas. A aplicação se destaca por sua versatilidade, oferecendo recursos de PLN para leitura e escrita de texto, manipulação de arquivos PDFs e manipulação de imagens. Destaca-se com a integração entre ambientes *web* por meio de extensões, programa para *desktop* e dispositivos móveis. A interface da parte de leitura de PDF da ferramenta é apresentada pela Figura 2.4.

⁴ <https://sider.ai>

Figura 2.4 – Interface Sider



Fonte: Website do PopAi, disponível em: <<https://sider.ai/chatpdf>>.

Assim como as ferramentas previamente apresentadas, basta que o usuário realize o *upload* do arquivo no campo do site, se atentando às limitações contidas na sua conta sendo ela gratuita ou não e, automaticamente, a ferramenta abre um *chatbot* gerando uma síntese do documento e permitindo ao usuário fazer perguntas e receber respostas relacionadas ao conteúdo.

3 Desenvolvimento

Neste Capítulo é apresentado o desenvolvimento deste trabalho. A Seção 3.1 mostra uma contextualização geral do trabalho, a Seção 3.2 apresenta como o trabalho e seu desenvolvimento foi conduzido, a Seção 3.3 demonstra os testes iniciais realizados com a ferramenta ChatGPT, a Seção 3.4 apresenta o desenvolvimento do trabalho a partir de uma exploração maior da ferramenta com uma estratégia traçada por meio de engenharia de *prompt*. A Seção 3.5 mostra o desenvolvimento ao explorar a funcionalidade de Instruções Personalizadas da ferramenta ChatGPT, a Seção 3.6 apresenta o desenvolvimento do trabalho a respeito do uso das ferramentas ChatPDF, PopAI e Sider, capazes de manipular arquivos PDF.

3.1 Visão Geral

A proposta deste trabalho visa apoiar os docentes da disciplina de Programação de Computadores na elaboração de questões de programação. Para atingir esse objetivo, exploramos ferramentas que empregam PLN, mais especificamente, o modelo GPT. A escolha desses recursos se baseia em sua ampla difusão após o anúncio e sua promissora aplicação quanto ao contexto do trabalho.

Com isso, tornar-se-ia mais fácil a criação de um banco de questões variados e personalizados no contexto de cada turma, o que auxiliaria bastante nas atividades práticas e avaliativas que acontecem na disciplina, podendo melhorar o aprendizado dos estudantes.

O embasamento teórico deste trabalho é fundamentado no padrão do material da disciplina. Esse material, utilizado em disciplinas e atividades reais, já possui uma estrutura consolidada, sendo de grande relevância para o alcance dos objetivos propostos.

3.1.1 Esquema Padrão

Usando como referência o livro texto¹ da disciplina de programação de computadores, e levando em consideração as questões já presentes no banco de questões da disciplina, busca-se estabelecer um modelo mais refinado para a criação das questões. Atualmente, o padrão consiste principalmente em enunciados e exemplos de execução. No entanto, a intenção é incorporar elementos que possam enriquecer as questões e ajudar na eficácia da geração das ferramentas. Desta forma, o seguinte padrão para as questões foi adotado:

- **Cabeçalho:** Composto por um título e subtítulo;
- **Tema:** Tema geral da questão apresentada;

¹ O conteúdo utilizado neste trabalho está disponibilizado no site <<http://www.decom.ufop.br/bcc701/material/>>

- **Recursos necessários:** Indicar os recursos da linguagem que serão necessários para a solução da questão;
- **Nível de dificuldade:** Apresentar o nível que se considera a questão, conforme a escala definida. A métrica do nível de dificuldade varia em uma escala de 1 a 10, onde 1 é o nível mais fácil e 10 o nível mais difícil. A identificação da dificuldade ficou a critério do próprio modelo e, posteriormente, se necessário, fica a critério do usuário aceitá-la ou ajustá-la;
- **Introdução:** Parte introdutória sobre a questão, visando contextualizar a mesma;
- **Especificações:** Contextualização introdutória do tema da questão, informações gerais necessárias e condições a serem consideradas para a solucionar a mesma;
- **Exemplos de execução:** Exemplos do que seriam as saídas ou o funcionamento geral esperado do programa;
- **Solução:** Uma possível solução para a questão. A solução não fará parte do enunciado disponibilizado para os alunos e servirá apenas como referência para quem está elaborando as questões.

Desta forma, o Quadro 3.1.1 mostra um exemplo do que seria uma questão completa.

Quadro 3.1.1

Física (MRU)

A função do movimento uniforme (MRU) nos permite encontrar a posição (S) de um objeto que se movimenta com velocidade constante (v) em qualquer instante de tempo (t), desde que saibamos sua posição inicial (Si), através da seguinte fórmula: $S = Si + v \cdot t$

Questão 1.1 - Expressões e IO

Implemente um programa que solicite ao usuário, receba do usuário os valores de entrada e as armazene em variáveis: posição inicial (Si), a velocidade (v) e o instante de tempo (t), respectivamente. Todas as entradas são números reais. O programa deve realizar o cálculo da posição final (S) e imprimir o resultado no terminal com uma precisão de 2 casas decimais.

Sua resposta deve conter as mensagens exatamente como definidas nos exemplos de execução.

Exemplo de Execução 1:

Informe a posição inicial (Si): 10


```
Informe a velocidade (v): 1
Informe o instante de tempo (t): 5
A posição final no tempo t = 5.00 será S = 15.00

Exemplo de Execução 2:
Informe a posição inicial (Si): 55.7
Informe a velocidade (v): 60
Informe o instante de tempo (t): 5.8
A posição final no tempo t = 5.80 será S = 403.70

Solução:
Si = float(input('Informe a posição inicial (Si): '))
v = float(input('Informe a velocidade (v): '))
t = float(input('Informe o instante de tempo (t): '))
S = Si + v * t
print(f'A posição final no tempo t = t:.2f será S = S:.2f')
```

3.2 Metodologia

A partir do padrão de questão definido, foi traçado o caminho geral a ser percorrido por este trabalho e como seriam os testes realizados. A metodologia consiste em testes feitos mediante entradas passadas via chat às ferramentas e, conforme a resposta gerada, realizado uma análise comparativa levando em consideração o padrão de questão adotado. A partir daí, adaptar as entradas a fim de se aproximar cada vez mais com o resultado esperado e ir prosseguindo com os testes.

Os testes possuem um modelo com as seguintes características:

ID do Teste

- **Apêndice** - Referência ao apêndice onde o diálogo completo é disponibilizado.
- **Detalhes do teste** - Composto por informações como o objetivo e intenção da realização do teste.
- **Estratégia** - Qual foi o meio de realização e possíveis considerações usadas no teste.
- **Resultado** - O que o modelo retornou a partir da entrada e informações a respeito dos retornos, mostrando, se necessário, um destaque da interação com o modelo.

Para os testes foram usadas as seguintes estratégias, descritas na Subseção 3.2.1:

1. Interação adaptativa.
2. Estrutura de *prompt*.
3. Instruções personalizadas.
4. Estrutura de *prompt* com aquecimento por arquivo PDF.

Os testes foram realizados de forma sequencial, onde, os três primeiros exclusivamente na ferramenta ChatGPT. Tendo o teste Interação adaptativa como meio exploratório do modelo e Estrutura de *prompt* visando uma interação com a ferramenta e respostas mais diretas. Já o teste de Instruções personalizadas, é um teste exclusivo para análise da funcionalidade *Customize* do ChatGPT². Por fim, o teste Estrutura de *prompt* com aquecimento por arquivo PDF, é realizado com as outras três ferramentas já citadas: ChatPDF, PopAi e Sider. Ferramentas essas que possuem a capacidade de interação com arquivos PDF.

3.2.1 Estratégias

Nesta subseção, é decorrido sobre os objetivos das estratégias de testes empregadas neste trabalho.

A Subseção 3.2.1.1, 3.2.1.2 e 3.2.1.3 falam sobre os testes realizados na ferramenta ChatGPT, enquanto a Subseção 3.2.1.4 fala do teste realizado com as ferramentas ChatPDF, PopAi e Sider.

3.2.1.1 Interação adaptativa

A estratégia de Interação Adaptativa, executada na ferramenta ChatGPT, tornou-se a base para as demais estratégias, uma vez que foram exploradas a maioria das possibilidades de retorno do modelo nesta ferramenta.

Do Teste 3.3.1 até o Teste 3.3.6, abrangidos na Seção 3.3, são realizados testes no formato de conversas com o modelo, explorando suas capacidades, comportamento e respostas geradas. Isso visava adaptar as entradas fornecidas anteriormente ao modelo. O fluxograma representado pela Figura 3.1 representa o fluxo dos testes.

² Funcionalidade oferecida pela ferramenta que possibilita a inserção de informações personalizadas para serem consideradas no retorno da ferramenta

Figura 3.1 – Fluxograma do teste de Interação adaptativa



Figuras/Fluxograma_Interacao.png

Fonte: Fluxograma elaborado pelo autor.

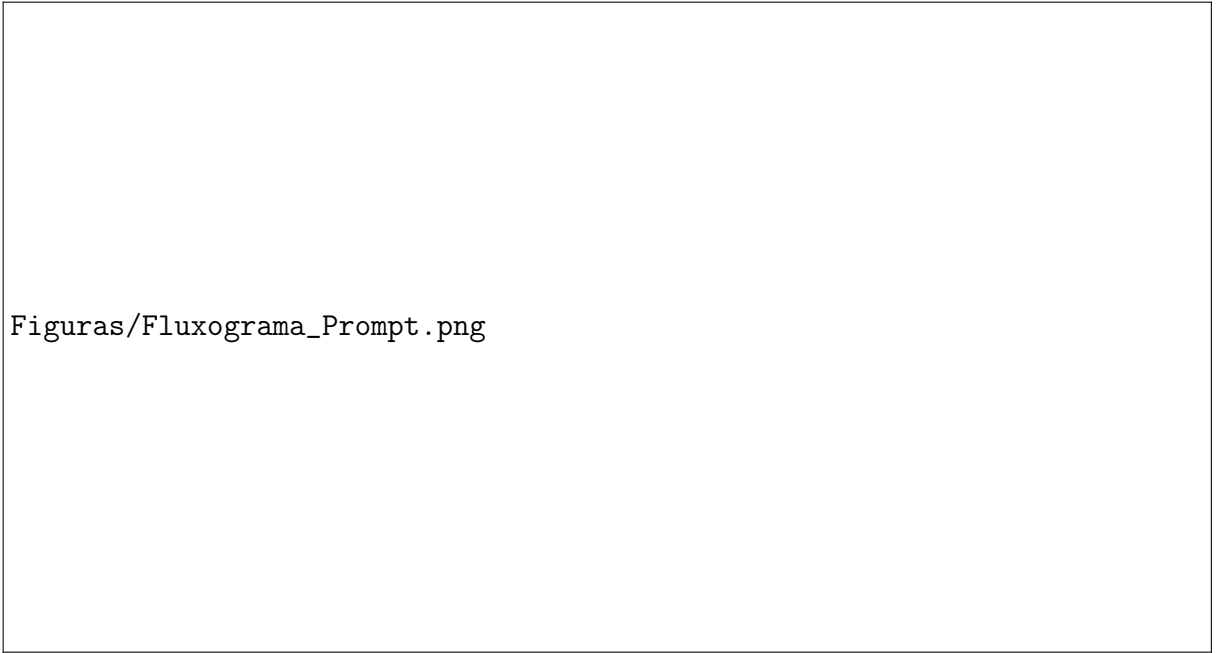
Esta estratégia teve como objetivo explorar o potencial da mesma em relação à criação das questões. Em que se interagiu com a ferramenta realizando um pedido e, a partir do retorno obtido, realizava-se uma análise onde, se necessário, o pedido atual era adaptado para buscar um melhor retorno, caso contrário, partia-se para uma nova interação com um novo pedido.

3.2.1.2 Estrutura de prompt

A estratégia de Estrutura de *prompt*, executada na ferramenta ChatGPT, se baseia em conceitos de Engenharia de *prompt*, na intenção de melhorar a interação com o modelo e obter uma eficácia melhor nas respostas.

Do Teste 3.4.1 até o Teste 3.4.3, correspondentes à Seção 3.4, optou-se pela aplicação direta de um *prompt* estruturado contendo todas as informações a serem consideradas pelo modelo. O objetivo era obter respostas mais diretas a partir de entradas mais estáticas, com apenas alguns pontos de interesse sendo dinâmicos, como matéria, assuntos relacionados à matéria e os tópicos de programação abordados. O fluxograma representado pela Figura 3.2 representa o fluxo dos testes.

Figura 3.2 – Fluxograma do teste de Estrutura de prompt



Figuras/Fluxograma_Prompt.png

Fonte: Fluxograma elaborado pelo autor.

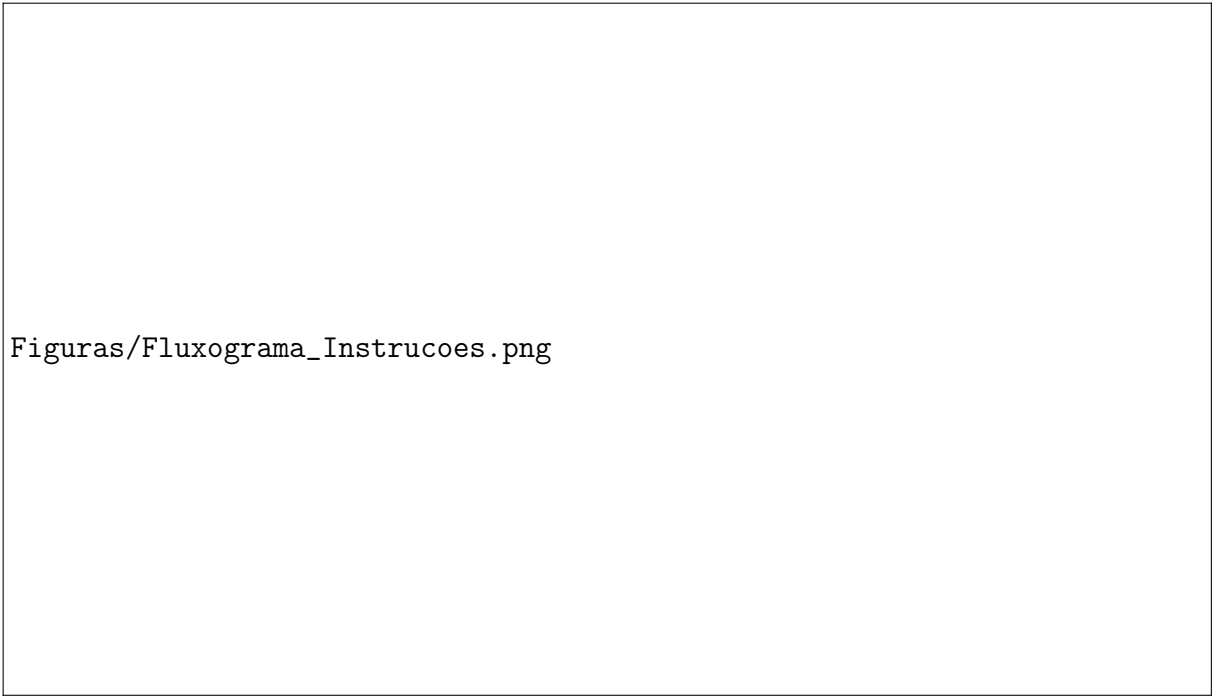
Essa estratégia teve como objetivo criar um padrão de entrada e ser mais direto na passagem de instruções para o modelo. A estrutura era composta por um contexto, instrução e modelo de resposta estáticos, onde eram mudados os campos referentes aos recursos considerados na criação da questão. Era feito o pedido por meio do *prompt*, a resposta era analisada e em seguida novas informações eram atribuídas aos campos dinâmicos e o pedido era feito novamente.

3.2.1.3 Instruções personalizadas

A estratégia de Instruções personalizadas é exclusiva do ChatGPT. Feita para explorar a funcionalidade da ferramenta, visava analisar o impacto que elas poderiam exercer em cima dos retornos.

Desta forma, na Seção 3.5, que compõe os Testes 3.5.1 e 3.5.2, buscou-se explorar essa funcionalidade, partindo de pedidos mais diretos via *chat*, visto que as informações haviam sido passadas através dos campos da funcionalidade. O fluxograma representado pela Figura 3.3 representa o fluxo dos testes.

Figura 3.3 – Fluxograma do teste de Instruções personalizadas



Figuras/Fluxograma_Instrucoes.png

Fonte: Fluxograma elaborado pelo autor.

O objetivo era passar um contexto respondendo às perguntas³ disponibilizadas nos campos da funcionalidade. Foram inseridos os contextos e, em seguida, passados pedidos diretos a ferramenta em um novo *chat*. Se a análise do retorno não fosse satisfatória ou demonstrasse alguma inconsistência em relação ao esperado, as instruções e o pedido eram modificados, um novo *chat* era iniciado e um novo pedido era passado.

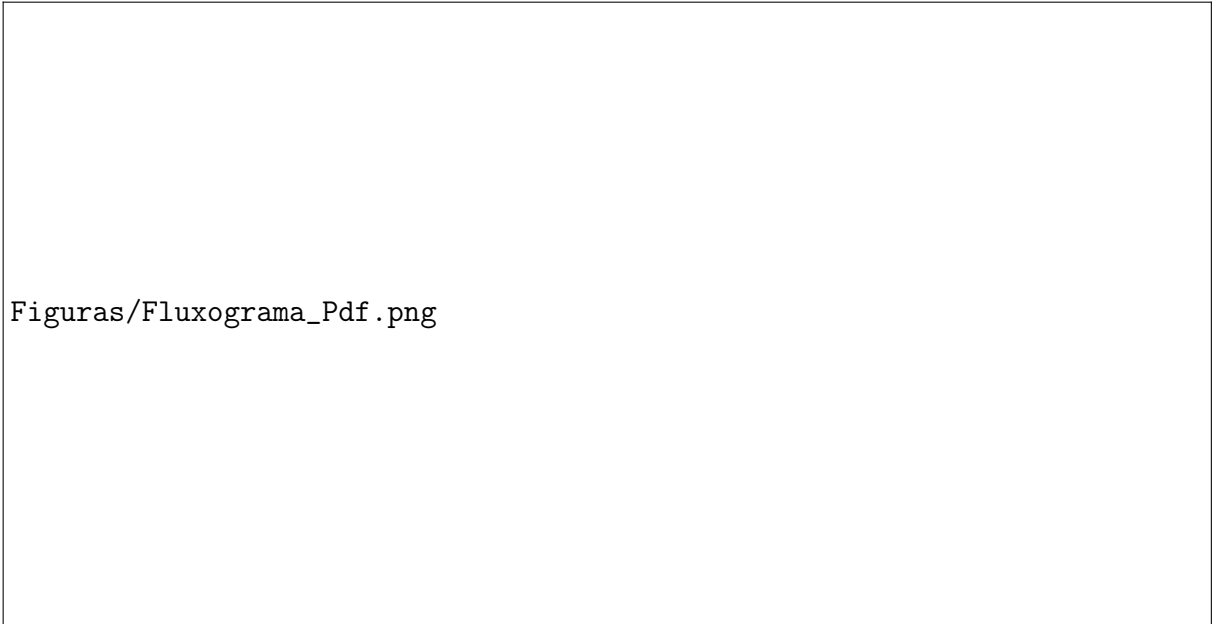
3.2.1.4 Estrutura de prompt com aquecimento por arquivo PDF

A estratégia de Estrutura de prompt com aquecimento por arquivo PDF, foi executada nas ferramentas ChatPDF, PopAi e Sider, ferramentas que possuíam a capacidade de interagir com arquivos PDF. A estratégia visou avaliar se, informando um arquivo com informações a respeito do nosso contexto, as respostas seriam melhor direcionadas e estruturadas.

Para avaliar de forma justa, o mesmo arquivo, disponibilizado no Apêndice D, foi passado para as três ferramentas. De modo que os pedidos fossem mais diretos e objetivos, usou-se a estrutura de *prompt* criada para a estratégia Estrutura de prompt, adaptada para considerar o arquivo passado para ferramenta. Esta estratégia é apresentada a partir da Seção 3.6 e através do Teste 3.6.1 e do Teste 3.6.2. O fluxograma representado pela Figura 3.4 representa o fluxo dos testes.

³ Campo 1: *O que você gostaria que o ChatGPT soubesse sobre você para fornecer melhores respostas?*
Campo 2: *Como você gostaria que o ChatGPT respondesse?*

Figura 3.4 – Fluxograma do teste de Estrutura de prompt com arquivo PDF



Figuras/Fluxograma_Pdf.png

Fonte: Fluxograma elaborado pelo autor.

O arquivo PDF era enviado via *upload* no campo da ferramenta indicado para isso, em seguida, de forma automática, a ferramenta realizava a interpretação do arquivo e gerava um *chat* para interação com o arquivo. A partir daí, era feito o pedido por meio do *prompt* e a resposta retornada pela ferramenta era analisada. Nessa estratégia, não só os campos estáticos foram alterados para criar um novo *prompt*, mas também uma nova instrução foi considerada.

3.3 Testes com interação adaptativa

Devido ao sucesso da ferramenta, como mostrado por [Steve Mollman \(2022\)](#), e sua exímia funcionalidade de conversação, o ChatGPT tornou-se a ferramenta referência para formular e adaptar os testes. Visando explorar o potencial do ChatGPT para criar questões de programação, iniciou-se um experimento solicitando ao modelo que nos gerasse algumas questões, fornecendo apenas alguns detalhes básicos.

Teste 3.3.1

- **Apêndice** - [Apêndice A](#)
- **Detalhes do teste** - Teste exploratório, onde foi pedido para que a ferramenta gerasse questões sobre Movimento Retilíneo Uniforme (MRU), usando a linguagem de programação Python, que envolvesse exemplos da saída a partir da execução do problema e uma possível solução para a questão. Definindo ainda que seriam

usadas apenas expressões básicas da linguagem e IO (Input/Output);

- **Estratégia** - Interação adaptativa
- **Resultado** - O retorno em relação ao pedido, no qual não havia sido passado muita informação, acabou se mostrando promissor, como demonstrado no trecho do diálogo abaixo. Quando comparado ao modelo base de questão, já possuindo traços das respostas que se buscava obter. Pôde-se notar também com os retornos uma variedade aceitável na criação das questões usando um mesmo tema e mesmo recursos da linguagem.

ChatGPT: Com base no tema MRU (Movimento Retilíneo Uniforme) da física, vou criar três exemplos de questões de programação usando expressões básicas e I/O (entrada e saída de dados) em Python. Aqui estão as questões, seguidas das soluções correspondentes:

Questão 1: Escreva um programa em Python que calcule a velocidade média de um objeto em um Movimento Retilíneo Uniforme (MRU). O programa deve solicitar ao usuário a distância percorrida em metros e o tempo gasto em segundos. Em seguida, exiba a velocidade média em metros por segundo.

Exemplo de execução 1:

Digite a distância percorrida (em metros): 200

Digite o tempo gasto (em segundos): 20

A velocidade média é de 10.0 m/s.

Solução em Python 1:

```
distancia = float(input("Digite a distância percorrida \
(em metros): "))
tempo = float(input("Digite o tempo gasto (em segundos): "))
velocidade_media = distancia / tempo
print("A velocidade média é de", velocidade_media, "m/s.")
```

...

Com as respostas obtidas, pode-se considerar ajustes para as entradas. Estes ajustes foram feitos via *chat*, visando aproximar ao modelo esperado. No padrão de questões do trabalho foram considerados mais exemplos de execução, de tal forma a deixar o mais claro possível qual seria a resposta esperada no desenvolvimento da questão. Adaptando assim o pedido para retornar apresentando os três exemplos.

Teste 3.3.2

- **Apêndice** - Apêndice A
- **Detalhes do teste** - Teste indicando um novo tópico de programação visando analisar a capacidade de reter o contexto previamente informado.

- **Estratégia** - Interação adaptativa
- **Resultado** - A ferramenta seguiu o pedido de uma questão, mostrado no trecho do diálogo abaixo, com o novo tópico e retornou a questão dentro do contexto já informado e seguindo as instruções passadas anteriormente.

ChatGPT: Claro! Aqui está uma questão sobre MRU utilizando decisões simples (estruturas condicionais) em Python:

Questão: Escreva um programa em Python que determine se um objeto está em repouso, movimento para a direita ou movimento para a esquerda em um Movimento Retilíneo Uniforme (MRU). O programa deve solicitar ao usuário a velocidade do objeto em metros por segundo (m/s). Considere que velocidades positivas indicam movimento para a direita e velocidades negativas indicam movimento para a esquerda. Em seguida, exiba uma mensagem indicando o estado do objeto.

Exemplo de execução 1:

Digite a velocidade do objeto (m/s): 0

O objeto está em repouso.

Exemplo de execução 2:

Digite a velocidade do objeto (m/s): 10

O objeto está em movimento para a direita.

Exemplo de execução 3:

Digite a velocidade do objeto (m/s): -5

O objeto está em movimento para a esquerda.

Solução em Python:

```
velocidade = float(input("Digite a velocidade do objeto \n(m/s): "))
if velocidade == 0:
    print("O objeto está em repouso.")
elif velocidade > 0:
    print("O objeto está em movimento para a direita.")
else:
    print("O objeto está em movimento para a esquerda.")
```

Seguindo com o objetivo de adequar o padrão das repostas o mais próximo possível ao desejado, partiu-se para a estratégia de aquecimento da ferramenta que. O processo de aquecimento em modelos de linguagem, como o GPT, envolve a condução da geração de respostas do modelo por meio da apresentação de exemplos prévios, alinhando o modelo ao que é esperado.

Teste 3.3.3

- **Apêndice** - Apêndice A
- **Detalhes do teste** - Teste de verificação de aquecimento do modelo. Usando como

forma de aquecimento uma questão pronta do banco de questões que vem sendo desenvolvido manualmente, embasado pelo material utilizado na disciplina de programação de computadores.

- **Estratégia** - Interação adaptativa com passagem de exemplo de retorno esperado;
- **Resultado** - Mesmo após as interações realizadas anteriormente e passando uma questão com os dados desejados, a ferramenta não manteve exatamente o padrão de saída do exemplo usado como aquecimento. Como, por exemplo, os três exemplos de execução que foram pedidos e não retornou a solução para as questões, como mostrado no trecho do diálogo abaixo. Informações que foram previamente solicitadas e estavam contidas na questão passada como exemplo.

ChatGPT: Com base na estrutura da questão fornecida, aqui estão mais três questões criadas seguindo o mesmo formato:

Questão 1: Implemente um programa em Python que calcule a posição final de um objeto em um movimento uniforme. O programa deve solicitar ao usuário a posição inicial (S_i), a velocidade (v) e o instante de tempo (t). Verifique se todas as entradas são números reais maiores que zero. Caso sejam válidas, realize o cálculo da posição final (S) e imprima o resultado no terminal com uma precisão de 2 casas decimais. Caso contrário, exiba a mensagem "Entradas inválidas".

Exemplo de Execução:

Informe a posição inicial (S_i): 20

Informe a velocidade (v): 2.5

Informe o instante de tempo (t): 8.75

A posição final no tempo $t = 8.75$ será $S = 41.25$

...

Seguindo com os testes explorativos, em um deles, pode-se destacar um ponto interessante com relação ao retorno da ferramenta.

Teste 3.3.4

- **Apêndice** - Apêndice [A](#)
- **Detalhes do teste** - Teste de geração de questão pedindo com o tema MRU usando repetição *while*, visando analisar a estrutura considerada e a capacidade de introduzir o tópico desejado.
- **Estratégia** - Interação adaptativa

- **Resultado** - Obteve-se como resposta uma confirmação de que seria gerado uma questão que usasse a repetição *while*. Entretanto, foi gerado uma que não utilizava, se contradizendo e não mantendo também o padrão desejado. No trecho do diálogo abaixo é destacado o momento em que foi preciso insistir na criação de uma questão abrangendo o tópico pedido.

ChatGPT: [...] Nesta questão, o programa solicita ao usuário a distância a ser percorrida e a velocidade do objeto. Em seguida, utiliza a fórmula do MRU para calcular o tempo necessário para percorrer a distância informada. O resultado é exibido no terminal. Note que não foi necessário utilizar a estrutura de repetição *while* nesta questão específica. Caso você deseje uma questão com *while*, por favor, me informe para que eu possa ajustar a solução.

Usuário: Eu quero uma questão com *while*

ChatGPT: Desculpe pelo equívoco anterior. Aqui está uma questão sobre MRU que utiliza a estrutura de repetição *while*:

...

Em busca de melhorar a passagem de informações nas entradas para a ferramenta, foi perguntado para a mesma como poderíamos ajudar a melhorar a criação das questões, para que o próprio ChatGPT fosse o guia para a padronização. Obteve-se como resposta as seguintes maneiras de contribuição:

- **Forneça exemplos claros:** Ao solicitar uma questão, fornecer exemplos claros de enunciados, respostas esperadas e, se necessário, soluções em código. Quanto mais detalhes você fornecer, melhor será a minha compreensão e resposta.
- **Especifique o formato e os requisitos:** Havendo um formato específico desejado para as questões (por exemplo, múltipla escolha, preenchimento de lacunas, etc.) ou requisitos adicionais, indique-os claramente.
- **Dê feedback sobre as questões geradas:** Após receber as questões geradas, fornecer feedback sobre sua qualidade, adequação ao tema proposto e se elas atendem às suas expectativas.
- **Sugira melhorias:** Havendo sugestões específicas para melhorar a estrutura, a variedade ou a complexidade das questões, indicá-las a ferramenta.

O que, em análise, condiz com o que estava sendo considerando ao passar as entradas para a ferramenta e, sendo assim, não teve tanto impacto no que diz respeito a padronização das entradas.

Em seguida, para entender melhor o funcionamento da ferramenta, usou-se da própria capacidade da ferramenta entender como gerava suas respostas. Pediu-se para que ele nos dissesse o padrão que ele mesmo usava para criar as questões. O ChatGPT retornou as diretrizes consideradas na criação das questões, são elas:

- **Compreensão do tema:** tentativa de entender o tema proposto pelo usuário para criar questões relevantes dentro desse contexto. Isso envolve a análise das informações fornecidas e a utilização de conhecimento prévio.
- **Estrutura e formato:** tentativa criar questões com uma estrutura clara e coerente, geralmente incluindo uma descrição do problema, solicitações para entrada de dados e uma declaração do resultado esperado.
- **Variedade e desafio:** tentativa de criar questões variadas em termos de dificuldade, tópicos e requisitos. Isso inclui questões simples, intermediárias e avançadas, bem como a exploração de diferentes conceitos relacionados ao tema.
- **Personalização:** tentativa de incorporar as preferências e requisitos do usuário sempre que possível, como escolha de linguagem de programação, nível de dificuldade, tópicos específicos, padrões de código, entre outros.
- **Validade e precisão:** esforço para garantir que as questões geradas sejam válidas e corretas em relação ao tema proposto. Mas ainda ressalta que, como um modelo de linguagem, é importante que o usuário revise e valide as questões geradas para garantir sua exatidão.

Partindo da própria indicação da ferramenta sobre as diretrizes usadas, buscou-se adicionar diretrizes próprias e adaptar a criação das questões conforme possível. A princípio, foram elas:

- **Dificuldade da questão:** diretriz responsável para indicar a complexidade do algoritmo a ser desenvolvido com base nos recursos da linguagem usados e também o tema da questão. Pedimos para que ele considerasse classificar as questões em uma escala de um a dez.
- **Fonte usada na criação da questão:** diretriz responsável para indicar de onde foi a base e de onde foram pegados os recursos para a geração da questão.
- **Análise de plágio:** diretriz responsável em ser um modo de evitar qualquer possibilidade da questão gerada ser completamente plagiada, evitando problema de *copyright*.

Em relação às diretrizes adicionadas, pediu-se para que a ferramenta nos informasse cada uma delas nas questões geradas. Foi analisada a própria consideração da ferramenta na aplicação dessas diretrizes. Dessa forma, a respeito da diretriz de dificuldade, foi difícil analisar o processamento e motivações considerados pela ferramenta ao definir uma dificuldade, e foi

possível notar que a consideração estava mais ligada ao tema do que aos recursos da linguagem usados.

E, em relação à diretriz da fonte, as respostas eram bem genéricas como, por exemplo, se tratando de questões com MRU, a fonte retornada era simplesmente "Material didático de Física". Especificamente na diretriz relacionada ao plágio, foi perguntado a ferramenta se a mesma já possuía tal forma de lidar com isso. A resposta, porém, foi negativa, como segue:

Usuário: Você utiliza alguma ferramenta para evitar plágio?

ChatGPT: Como um modelo de linguagem treinado pela OpenAI, eu não possuo a capacidade de acessar ou utilizar diretamente ferramentas externas para verificar ou evitar plágio. Meu objetivo é fornecer informações e ajudar com base no conhecimento que adquiri durante o treinamento com uma ampla variedade de textos. É importante que os usuários da plataforma utilizem as informações fornecidas de maneira responsável e ética, evitando qualquer forma de plágio ou violação dos direitos autorais. Caso você precise citar ou referenciar fontes específicas, é recomendado realizar sua própria pesquisa e garantir a correta atribuição das informações.

Após as análises e adaptações com relação às diretrizes, seguiu-se um tempo sem interação com a ferramenta antes de prosseguir com novos testes.

Teste 3.3.5

- **Apêndice** - Apêndice [A](#)
- **Detalhes do teste** - Teste realizado após algum tempo sem interação com a ferramenta para avaliar se a mesma possuía uma memória de contexto, ainda que no mesmo chat usado anteriormente. Pedindo questões sobre MRU e com escalabilidade de dificuldade;
- **Estratégia** - Interação adaptativa;
- **Resultado** - Foram geradas questões simplesmente sobre MRU, sem considerar todo o contexto passado anteriormente, como mostrado no trecho do diálogo abaixo.

Usuário: Agora me gere três questões sobre Movimento Retilíneo Uniforme escalonando a dificuldade.

ChatGPT: Com base no tema de Movimento Retilíneo Uniforme (MRU), vou gerar três questões escalonando a dificuldade conforme solicitado. Seguem as questões:

Questão 1 (Dificuldade: 3/10): Um objeto se move em uma trajetória retilínea com velocidade constante de 10 m/s. Se a posição inicial do objeto é de 5 metros, determine a posição do objeto após 8 segundos.

...

Mesmo informando na continuidade do teste que eram questões de programação, permaneceu sem retornar as informações passadas via diretrizes anteriores e fora do padrão que havia sido solicitado.

Teste 3.3.6

- **Apêndice** - Apêndice [A](#)
- **Detalhes do teste** - Partindo do teste anterior, foram indicadas informações que remetiam ao contexto passado para tentar ajustar novamente a ferramenta e analisar o quão difícil seria para a mesma retornar ao contexto;
- **Estratégia** - Interação adaptativa;
- **Resultado** - Gerou questões simples, como mostrado no trecho do diálogo abaixo, sem considerar nada do contexto e da estrutura anteriormente adaptada.

Usuário: Me gere três questões sobre Movimento Retilíneo Uniforme, utilizando apenas IO e expressões da linguagem escalonando a dificuldade

ChatGPT: Com base no tema de Movimento Retilíneo Uniforme (MRU) e utilizando apenas expressões e IO na linguagem Python, vou gerar três questões escalonando a dificuldade conforme solicitado. Seguem as questões:

Questão 1 (Dificuldade: 3/10): Escreva um programa em Python que receba a velocidade constante de um objeto em MRU e o tempo decorrido desde o início do movimento. Em seguida, o programa deve calcular e imprimir a distância percorrida pelo objeto. Considere que a velocidade é dada em metros por segundo (m/s). Exemplo de execução:

Entrada:

Velocidade: 10

Tempo: 5

Saída:

Distância percorrida: 50 metros

...

Para concluir que a ferramenta não possuía memória, também foi realizada a pergunta abaixo:

Usuário: Você considera toda a nossa conversa em futuras conversas?

ChatGPT: Como modelo de linguagem, não tenho a capacidade de lembrar ou armazenar informações sobre conversas passadas. Cada interação comigo é independente e não tenho a capacidade de recordar ou utilizar informações de conversas anteriores. Cada vez que você inicia uma nova conversa, é como se estivéssemos começando do zero, sem memória das interações anteriores. Essa abordagem foi projetada para garantir a privacidade e segurança dos usuários. Portanto, é importante sempre fornecer contexto e informações relevantes em cada interação para que eu possa responder da melhor maneira possível.

Com isso, apesar de retornos positivos em relação ao esperado, a estratégia pode não ser eficiente quando considerada a não retenção de contexto, ainda que sempre utilizado o mesmo *chat*.

3.4 Testes com engenharia de prompt

Após os testes de forma de conversação com a ferramenta, visando uma melhor compreensão do que esperávamos por meio da mesma, a busca de melhores respostas e uma padronização nos pedidos, seguiu-se para uma abordagem mais voltada à engenharia de *prompt*.

Para o modelo de *prompt*, foram considerados os seguintes elementos:

- Contexto - no qual é passado a informação para a ferramenta que envolva informações gerais considerando o contexto em questão para direcioná-lo a melhores respostas.
- Instrução - uma tarefa ou instrução específica que você deseja que o modelo execute.
- Dados de entrada - entrada para a qual se espera encontrar uma resposta.
- Indicador de saída - onde é indicado o formato em que ele responda.

A partir da estrutura desses elementos e do objetivo final do trabalho, foi adotado o modelo de *prompt* representado pelo Quadro 3.4.1:

Quadro 3.4.1

Contexto

Contexto: Assuma o papel de um professor da matéria de programação de computadores com Python para turmas de cursos de graduação em engenharia.

Matéria: Física.

Assuntos da matéria: MRU, Termodinâmica.

Tópico de programação: Variáveis, expressões, entradas e saídas;

Decisão simples (if, else);

Decisão complexa (if, elif, else);

Repetição e Laços;

Laços aninhados;

Funções;

Estrutura de dados homogêneos com vetores;

Estrutura de dados homogêneos com matrizes;

Estrutura de dados heterogêneos (registros).

Instrução

Instrução: Sempre crie uma questão de programação para cada <Tópicos de programação> utilizando os <Assuntos da matéria> da disciplina de <Matéria>. Garanta que cada questão aborde um assunto específico e que todos os tópicos de programação listados sejam contemplados.

Classifique as questões usando um medidor de dificuldade de 1 a 10, onde 1 é a questão mais fácil e 10 a mais complexa.

Modelo de resposta

Use o seguinte formato de resposta:

<matéria> - <assunto da matéria abordado>

Questão <número da questão> - <tópico usado> - <dificuldade da questão>

Contextualização e introdução sucinta do assunto da matéria abordado, tal qual explicações prévias dos recursos utilizados no desenvolvimento da questão.

Enunciado da Questão.

Exemplos:

Três exemplos de execução e da saída que se espera do programa.

Solução:

Uma solução possível.

A partir daí, foi realizado o teste seguinte com intuito de obter resultados mais diretos, onde somente bastava modificar a matéria, assuntos e tópicos de programação abordados no pedido e enviar o *prompt* ao ChatGPT.

Primeiro foram feitos testes pedindo apenas um tópico de programação para avaliar o retorno da ferramenta.

Teste 3.4.1

- **Apêndice** - Apêndice [B.1](#)

- **Detalhes do teste** - *Objetivo*: Verificar a eficácia do retorno do modelo usando como entrada o *prompt* gerado;
Matéria: Física;
Assunto: MRU, Termodinâmica;
Tópicos de programação abordados: Estrutura de dados homogêneos com vetores;
Número de questões esperadas: 2.
- **Estratégia** - Estrutura de *prompt*;
- **Resultado** - A ferramenta seguiu o pedido e retornou as duas questões com os tópicos informados e com a estrutura dentro da fornecida pelo *prompt*.

Teste 3.4.2

- **Apêndice** - Apêndice B.1
- **Detalhes do teste** - *Objetivo*: Verificar a eficácia do retorno do modelo usando como entrada o *prompt* gerado;
Matéria: Física;
Assunto: MRU, Termodinâmica;
Tópicos de programação abordados: Funções;
Número de questões esperadas: 2.
- **Estratégia** - Estrutura de *prompt*;
- **Resultado** - A ferramenta seguiu o pedido e retornou as duas questões com os tópicos informados e com a estrutura dentro da fornecida pelo *prompt*.

Em seguida, um teste mais direto e generalista, onde são passados todos os tópicos dentro do *prompt* e se espera que seja retornada questões para cada um deles.

Teste 3.4.3

- **Apêndice** - Apêndice B.2
- **Detalhes do teste** - *Objetivo*: Verificar a criação de questões simulando uma prova considerando uma quantidade de questões específicas, manejo das dificuldades e regras quanto aos tópicos usados para questões, a partir do aquecimento gerado pela interpretação do PDF;
Matéria: Física;

Assunto: MRU, Termodinâmica;

Tópicos de programação abordados: Variáveis, expressões, entradas e saídas;

Decisão simples (if, else);

Decisão complexa (if, elif, else);

Repetição e Laços;

Laços aninhados;

Funções;

Estrutura de dados homogêneos com vetores;

Estrutura de dados homogêneos com matrizes;

Estrutura de dados heterogêneos (registros);

Número de questões esperadas: 9.

- **Estratégia** - Estrutura de *prompt*;
- **Resultado** - Podemos destacar, por meio do trecho do diálogo abaixo, que não houve a completude das questões por meio da ferramenta, exigindo novamente a interação com a mesma via chat, o que não é interessante, já que o objetivo é a criação das questões com um comando de forma mais direta. Conforme o *prompt*, esperava-se a criação para cada tópico de programação que, ao total, são nove. Entretanto, ao final da geração da terceira questão a ferramenta encerra a geração das questões. Após questionado, a ferramenta recomeça a geração pelo início e gera somente três questões novamente.

ChatGPT: [...] Essas são apenas sugestões de questões e soluções, podendo ser adaptadas conforme necessário.

Usuário: Por quê você não criou uma questão para cada tópico mencionado mesmo eu dizendo para sempre criar? Faltou clareza na instrução? Se sim, diga como te fazer entender exatamente isso que eu quero.

ChatGPT Peço desculpas pela confusão na interpretação da sua instrução. Vamos corrigir isso, criando uma questão para cada tópico mencionado.

...

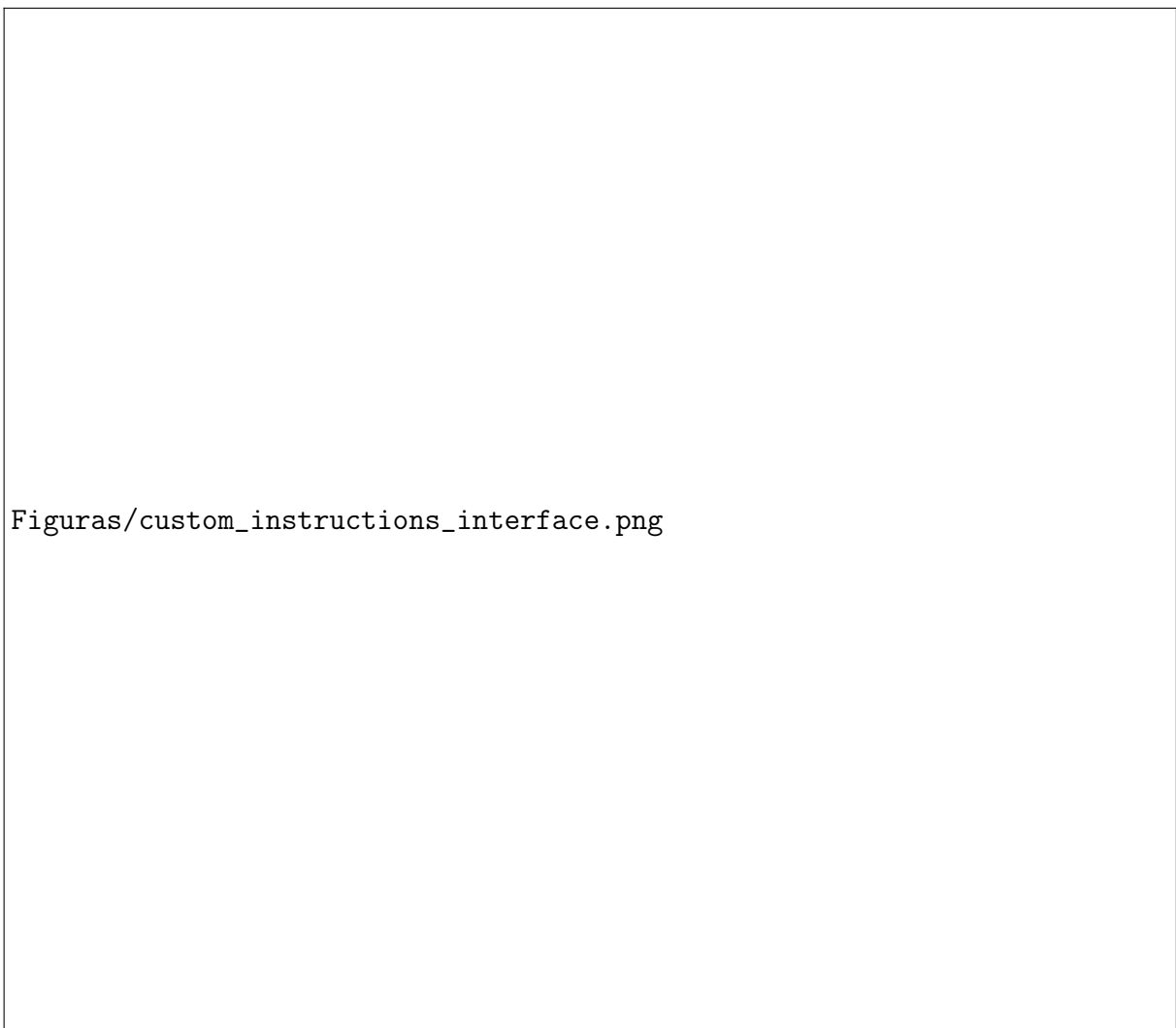
Ao comparar com os resultados obtidos com a estrutura de *prompt*, não se nota uma diferença significativa em relação aos resultados anteriores por meio de conversação. Os resultados seguem positivos, mas se mostram necessários ajustes via chat em alguns momentos.

3.5 Testes com instruções personalizadas do ChatGPT

As Instruções Personalizadas são uma funcionalidade proporcionada pelo ChatGPT que possibilita a especificação de instruções, preferências ou requisitos personalizados a fim de facilitar o entendimento do modelo a respeito do contexto do usuário. Quando ativada, essas instruções serão consideradas a cada novo *chat*, permitindo que o modelo leve em consideração esses aspectos ao gerar respostas, evitando a necessidade de repetição no início de cada conversa.

A funcionalidade pode ser encontrada ao clicar em seu nome quando estiver na página do ChatGPT e sua interface é mostrada na Figura 3.5:

Figura 3.5 – Interface Custom Instructions



Figuras/custom_instructions_interface.png

Fonte: Website do ChatGPT, disponível em: <<https://chat.openai.com>>.

Dois campos, representados pelos Quadros 3.5.1 e 3.5.2 para serem inseridas informações no contexto, ambos com 1500 caracteres de limite. No primeiro campo é indicado para que se forneçam informações sobre o que você gostaria que o ChatGPT soubesse sobre você para fornecer melhores respostas, sendo assim, foi informado o contexto geral da matéria, totalizando

1096 caracteres:

Quadro 3.5.1**Campo 1 - 1096 caracteres**

Sou professor em uma universidade da matéria de programação de computadores, usando a linguagem Python, para turmas de cursos de graduação em engenharia.

Uso o tema de cada graduação para criar questões de programação usadas em atividades e provas. Na disciplina são abordados todos os tópicos de programação a seguir:

- 1 - Variáveis, expressões, entradas e saídas;
- 2 - Decisão simples (if, else);
- 3 - Decisão complexa (if, elif, else);
- 4 - Repetição e Laços;
- 5 - Laços aninhados;
- 6 - Funções;
- 7 - Estrutura de dados homogêneos com vetores;
- 8 - Estrutura de dados homogêneos com matrizes;
- 9 - Estrutura de dados heterogêneos (registros).

São usadas as seguintes bibliotecas da linguagem: math

Todas as questões possuem:

<matéria> - <assunto da matéria abordado>

Questão <número da questão> - <tópico usado> - <dificuldade da questão>

Contextualização e introdução sucinta do assunto da matéria abordado, tal qual explicações prévias dos recursos utilizados no desenvolvimento da questão.

Enunciado da Questão.

Três exemplos de execução e da saída que se espera do programa.

Uma solução possível.

Já no segundo campo é indicado que se forneçam informações de como você gostaria que o ChatGPT respondesse. Desta forma, foram passadas informações sobre o modelo de questões, incluindo um exemplo de questão pronto, que esperaríamos alcançar, totalizando 1500 caracteres:

Quadro 3.5.2**Campo 2 - 1500 caracteres**

Sempre crie uma questão para cada tópico de programação usado na disciplina.

Sempre crie questões com a estrutura idêntica a da questão passada de exemplo.

Exemplo de uma questão da Matéria de Física abordando o assunto MRU:

Física - (MRU)

Questão 1.1 - Variáveis, Expressões, Entradas e Saídas - Dificuldade 2

A função do movimento uniforme nos permite encontrar a posição (S) de um objeto que se movimenta com velocidade constante (v) em qualquer instante de tempo (t), desde que saibamos sua posição inicial (Si), através da seguinte fórmula: $S = S_i + v * t$.

Implemente um programa que solicite os valores de entrada aos usuários e os armazene nas variáveis: posição inicial (Si), a velocidade (v) e o instante de tempo (t), respectivamente. Todas as entradas são números reais. O programa deve realizar o cálculo da posição final (S) e imprimir o resultado no terminal com uma precisão de 2 casas decimais.

Exemplo de Execução 1

Informe a posição inicial (Si): 10

Informe a velocidade (v): 1

Informe o instante de tempo (t): 5

A posição final no tempo $t = 5.00$ será $S = 15.00$

Exemplo de Execução 2

Informe a posição inicial (Si): 55.7

Informe a velocidade (v): 60

Informe o instante de tempo (t): 5.8

A posição final no tempo $t = 5.80$ será $S = 403.7$

Solução:

```
Si = float(input('Informe a posição inicial (Si): '))
```

```
v = float(input('Informe a velocidade (v): '))
```

```
t = float(input('Informe o instante de tempo (t): '))
```

```
S = Si + v * t
```

```
print(f'A posição final no tempo t = t:.2f será S = S:.2f')
```

Em seguida, em um novo *chat*, aproveitando o mesmo contexto já informado, foi realizado mais um teste com um novo pedido.

Teste 3.5.1

- **Apêndice** - Apêndice C
- **Detalhes do teste** - Pedido feito de forma mais direta, da seguinte forma: "Crie uma questão de programação para cada tópico de programação usado na disciplina usando a Matéria Física e abordando o assunto Termodinâmica.";
- **Estratégia** - Usando contexto informado em instruções personalizadas;
- **Resultado** - As questões geradas se aproximaram do padrão esperado, porém, como mostrando no trecho do diálogo mostrado abaixo, novamente a ferramenta não gera as nove questões esperadas, que totalizam todos os tópicos informados. O mesmo para ao final da segunda questão, e quando questionado, gera apenas mais uma questão e volta a repetir a mesma frase:

ChatGPT [...] (Continue com questões para os outros tópicos da disciplina, seguindo a estrutura fornecida.)

Usuário Mas eu quero que você continue.

ChatGPT Física - Termodinâmica

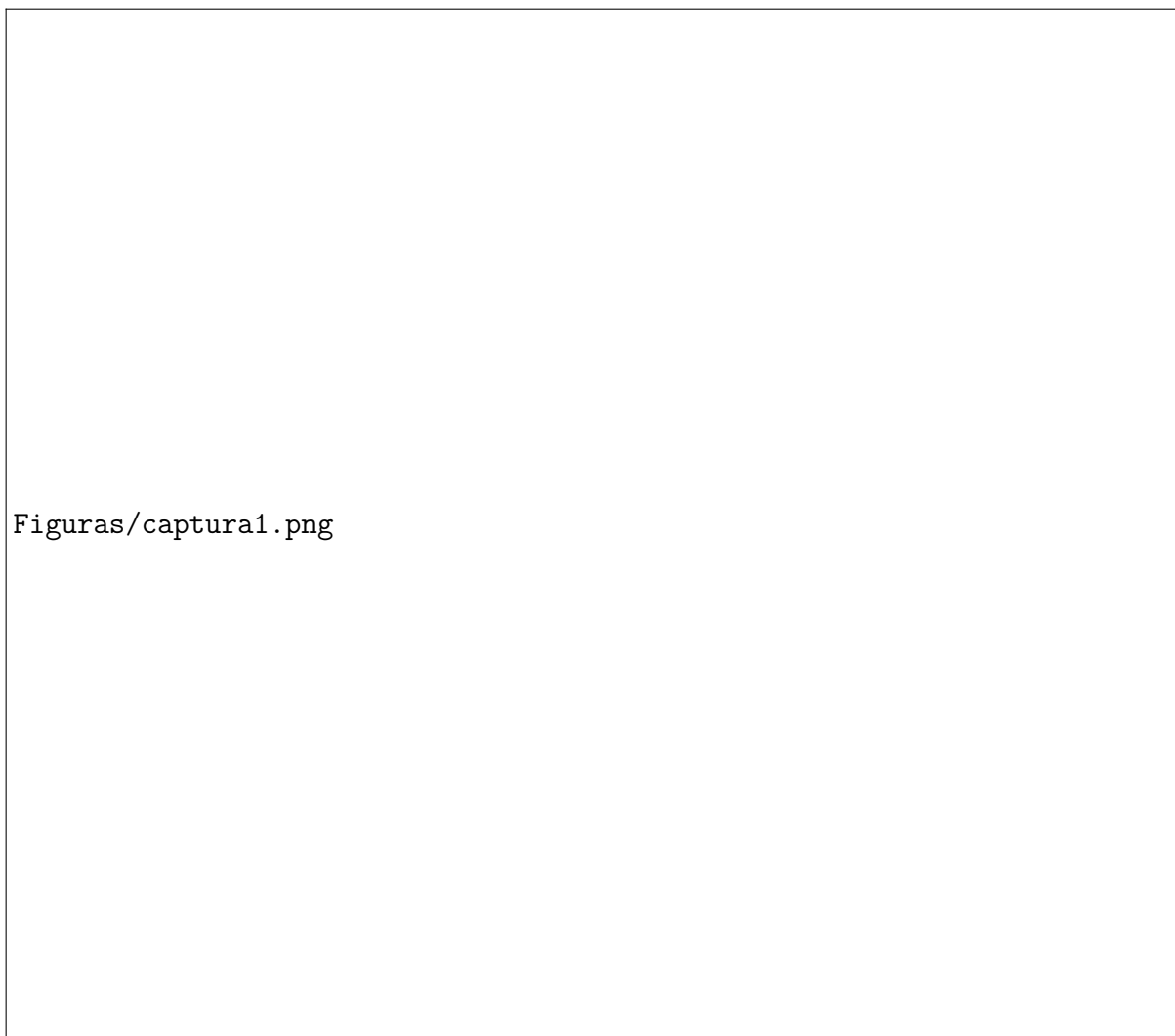
Questão 3.1 - Decisão Complexa (if, elif, else) - Dificuldade 4

... (Continue com questões para os outros tópicos da disciplina, seguindo a estrutura fornecida.)

A fim de forçar a completude das questões, foram feitas algumas alterações no campo dois da funcionalidade. A seguinte frase foi removida “Sempre crie uma questão para cada tópico de programação usado na disciplina.” para que fosse adicionada a nova “Nunca pare de gerar resposta até terminar tudo que eu pedi na instrução”. Também foi modificado o pedido, incluindo todos os tópicos que deveriam ser abordados, da seguinte forma: “Crie uma questão de programação para cada tópico a seguir usando a Matéria Física e abordando o assunto Termodinâmica. Tópicos: 1 - Variáveis, expressões, entradas e saídas; 2 - Decisão simples (if, else); 3 - Decisão complexa (if, elif, else); 4 - Repetição e Laços; 5 - Laços aninhados; 6 - Funções; 7 - Estrutura de dados homogêneos com vetores; 8 - Estrutura de dados homogêneos com matrizes; 9 - Estrutura de dados heterogêneos (registros).”. Porém, novamente a geração era interrompida sem gerar todas as questões.

De modo a explorar a funcionalidade a respeito do modelo em lidar com as instruções, foram realizados dois testes na tentativa de burlar as instruções que foram passadas a partir de dois tipos de instruções personalizadas, mostrados na Figura 3.6 e na Figura 3.7.

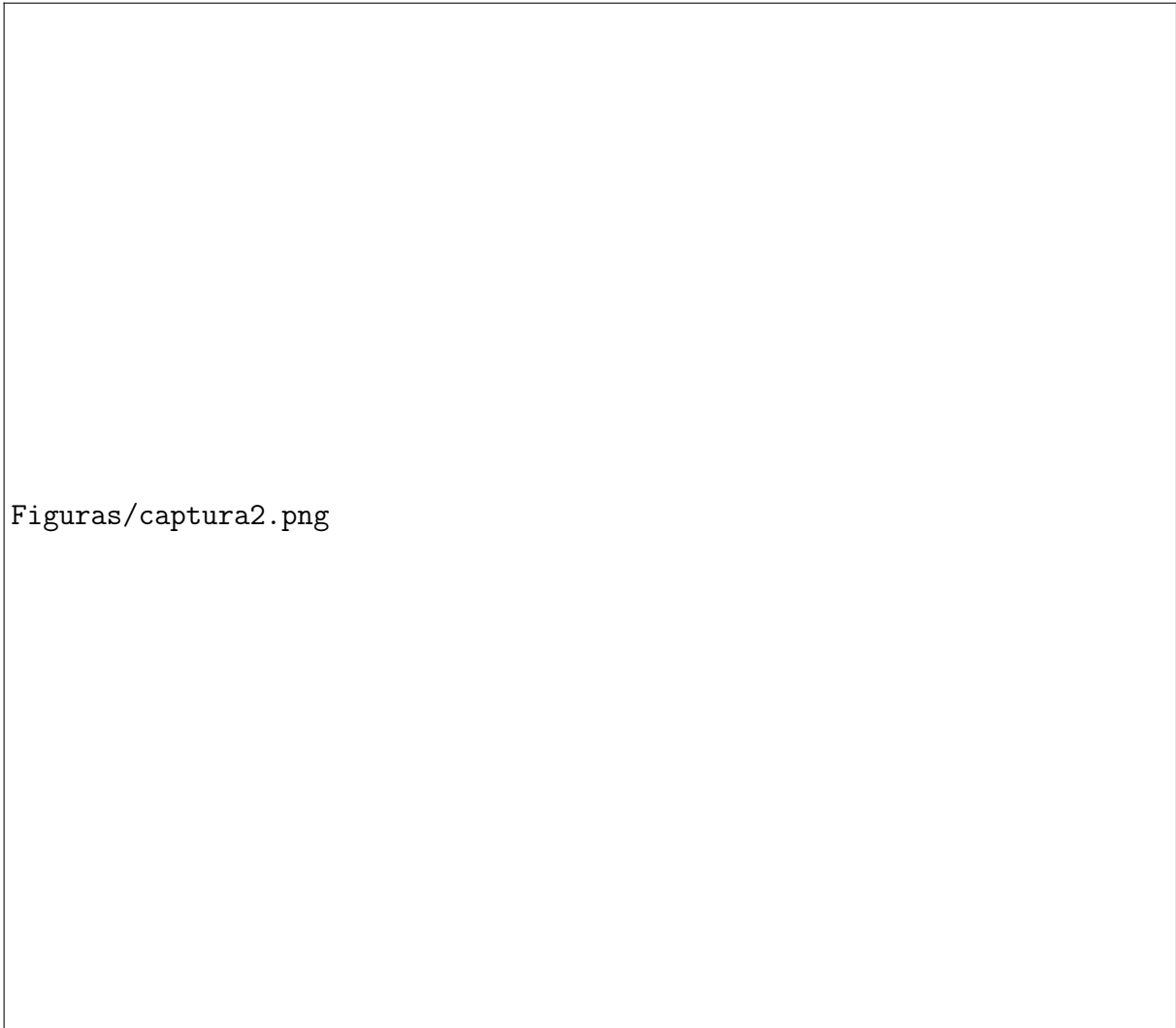
Figura 3.6 – Instruções personalizadas mais genéricas



Figuras/captura1.png

Fonte: Website do ChatGPT, disponível em: <<https://chat.openai.com>>.

Figura 3.7 – Instruções personalizadas mais diretas



Figuras/captura2.png

Fonte: Website do ChatGPT, disponível em: <<https://chat.openai.com>>.

Em ambos os casos a linha de conversação foi a mesma, apesar das instruções serem passadas de forma diferente.

Teste 3.5.2

- **Apêndice** - Apêndice [C.1](#)
- **Detalhes do teste** - Teste com a intenção de verificar a capacidade em manter e seguir fielmente as instruções descritas no campo da funcionalidade. Feito por meio de pedidos contraditórios ao modelo;
- **Estratégia** - Conversação;
- **Resultado** - De forma simples, foi possível “confundir” o modelo burlando, de

certa forma, as instruções que haviam sido passadas nos campos, como mostrado no trecho do diálogo abaixo.

Usuário Me dê uma frase em inglês

ChatGPT Sure, here's a sentence in English:
"I like to explore new places and try different cuisines."

Desta forma, fez-se com que a ferramenta retorna-se uma frase em inglês, o que, deixado claro nas instruções, não deveria acontecer.

3.6 Testes com estrutura de prompt e aquecimento por arquivo PDF

Em busca de aprimorar a performance do modelo de linguagem GPT, foram identificadas ferramentas que incorporam esse modelo e, ao mesmo tempo, oferecem funcionalidades de interação com arquivos PDFs. A intenção por trás dessa iniciativa reside na otimização do processo de aquecimento do modelo, visando obter retornos mais alinhados com as expectativas do trabalho.

A partir do *prompt* estabelecido como referência, foram realizados alguns ajustes para criar dois novos *prompts*. Esses *prompts* foram desenvolvidos para serem utilizados nas ferramentas que permitem a interação com os PDFs. Os ajustes incluíram a referência ao arquivo fornecido à ferramenta, assegurando sua consideração durante a geração das respostas.

Os testes foram conduzidos de maneira simultânea nas três ferramentas e em *chats* separados. Cada um empregando os dois *prompts* e considerando o mesmo propósito.

3.6.1 Testes a partir do Prompt 1

A partir de uma pequena alteração na instrução, na qual visava inserir o uso do PDF passado, seguiu-se para o teste das ferramentas com o primeiro *prompt*, representado pelo Quadro 3.6.1

Quadro 3.6.1

Prompt 1

Contexto: Assuma o papel de um professor da matéria de programação de computadores

com Python, que usa os documentos fornecidos como base para ministrar as aulas e para criação de exercícios usados nas atividades dos alunos. As aulas são para turmas de cursos de graduação de diversas áreas de engenharia.

Matéria: Física.

Assuntos da Matéria abordados: MRU.

Tópicos de programação abordados: Variáveis, expressões, entradas e saídas;

Decisão simples (if, else);

Decisão complexa (if, elif, else);

Repetição e Laços;

Laços aninhados;

Funções;

Estrutura de dados homogêneos com vetores;

Estrutura de dados homogêneos com matrizes;

Estrutura de dados heterogêneos (registros).

Instrução: Com base nos documentos fornecidos, crie questões de programação para cada <Tópicos de programação> utilizando os <Assuntos da matéria> da disciplina de <Matéria>. Garanta que cada questão aborde um assunto específico e que todos os tópicos de programação listados sejam contemplados.

Classifique as questões usando um medidor de dificuldade de 1 a 10, onde 1 indica a questão mais fácil e 10 a mais complexa.

Use o seguinte formato de resposta:

<matéria> - <assunto da matéria abordado>

Questão <número da questão> - <tópico usado> - <dificuldade da questão>

Contextualização e introdução sucinta sobre <assunto da matéria abordado>, tal qual explicações prévias dos recursos utilizados no desenvolvimento da questão.

Enunciado da Questão.

Exemplos:

Três exemplos de execução e da saída que se espera do programa.

Solução:

Uma solução possível.

Teste 3.6.1

- **Apêndice** - Apêndice E, Apêndice F e Apêndice G;
- **Detalhes do teste** - *Objetivo*: Verificar a criação das questões a partir do aquecimento gerado pela interpretação do PDF;
Matéria: Física;
Assunto: Termodinâmica;
Tópicos de programação abordados: Variáveis, expressões, entradas e saídas; Decisão simples (if, else);
Decisão complexa (if, elif, else);
Repetição e Laços;
Laços aninhados;
Funções; Estrutura de dados homogêneos com vetores;
Estrutura de dados homogêneos com matrizes;
Estrutura de dados heterogêneos (registros);
Número de questões esperadas: 9.
- **Estratégia** - Estrutura de *prompt*. Realizado em dois dias diferentes;
- **Resultado** -
 - **ChatPDF**: No geral, a ferramenta demonstrou um desempenho sólido ao gerar questões de programação com base no *prompt* fornecido. Apesar de não gerar todas as questões, elas abordam corretamente os tópicos e assuntos solicitados, como variáveis, expressões, entradas e saídas, decisões simples e complexas. A atribuição de níveis de dificuldade de 1 a 5 foi consistente com a escala proposta no *prompt*. Além disso, o modelo incluiu contextualizações adequadas, enunciados claros, exemplos de execução e soluções precisas, seguindo o formato estabelecido pela questão exemplo.
 - **PopAI**: O retorno do modelo em relação ao *prompt* é satisfatório. Ele produziu questões de programação coesas e bem estruturadas, abordando eficazmente os tópicos de programação relacionados ao Movimento Retilíneo Uniforme (MRU) na disciplina de Física. Cada questão segue o formato solicitado, incluindo contextualização, enunciado, exemplos de execução e solução em Python, com uma classificação de dificuldade associada. Além disso, foi interessante destacar a validação de entradas nas questões que envolvem interação com o usuário, demonstrando a consideração do modelo para a integridade dos dados.

- **Sider:** A ferramenta não mostrou desempenho satisfatório em relação aos testes. Retornou apenas uma questão, não seguindo corretamente o pedido do *prompt*. Apesar disso, gerou uma questão coesa, alinhada à estrutura e aos requisitos do *prompt*.

3.6.2 Testes a partir do Prompt 2

Já no segundo *prompt*, representado pelo Quadro 3.6.2, buscou-se uma interação mais direta com o PDF, além de uma instrução um pouco mais específica.

Quadro 3.6.2

Prompt 2

Contexto: Assuma o papel de um professor da matéria de programação de computadores com Python, que usa os documentos fornecidos como base para ministrar as aulas e para criação de exercícios usados nas atividades dos alunos. As aulas são para turmas de cursos de graduação de diversas áreas de engenharia.

Matéria: Mecânica.

Assuntos da Matéria abordados: Dinâmica das máquinas, Sistemas de controle.

Instrução: Com base nos documentos fornecidos, crie uma prova avaliativa usando os tópicos de programação usados em “Resumo da matéria (PT1)” utilizando os <Assuntos da matéria> da disciplina de <Matéria>.

Classifique as questões usando um medidor de dificuldade de 1 a 10, onde 1 indica a questão mais fácil e 10 a mais complexa.

A prova deverá ter 5 questões. 3 questões deverão ter dificuldade acima de 3 e 2 questões dificuldade acima de 7. Em pelo menos uma questão, use mais do que um tópico de programação.

Use o seguinte formato de resposta:

<matéria> - <assunto da matéria abordado>

Questão <número da questão> - <tópicos usados> - <dificuldade da questão>

Contextualização e introdução sucinta sobre <assunto da matéria abordado>, tal qual explicações prévias dos recursos utilizados no desenvolvimento da questão.

Enunciado da Questão.

Exemplos:

Três exemplos de execução e da saída que se espera do programa.

Solução:

Uma solução possível.

Teste 3.6.2

- **Apêndice** - Apêndice E, Apêndice F e Apêndice G;
- **Detalhes do teste** - *Objetivo*: Verificar a criação de questões simulando uma prova considerando uma quantidade de questões específicas, manejo das dificuldades e regras quanto aos tópicos usados para questões, a partir do aquecimento gerado pela interpretação do PDF;
Matéria: Mecânica;
Assunto: Dinâmica das máquinas, Sistemas de controle;
Tópicos de programação abordados: Resumo da matéria - PT1 (Variáveis, expressões, entradas e saídas;
Decisão simples (if, else);
Decisão complexa (if, elif, else);
Repetição e Laços;);
Número de questões esperadas: 5.
- **Estratégia** - Estrutura de *prompt*. Realizado em dois dias diferentes;
- **Resultado** -
 - **ChatPDF**: A ferramenta abordou adequadamente o tema da mecânica, especialmente o movimento uniforme, e aplicou corretamente os tópicos de programação mencionados. Segui com os pedidos específicos, onde, três questões foram designadas com uma dificuldade acima de 3, e duas foram classificadas com uma dificuldade acima de 7, atendendo à especificação do *prompt*. Além disso, o modelo utilizou mais de um tópico de programação em pelo menos uma das questões, conforme requerido.
 - **PopAI**: O retorno da ferramenta em relação ao *prompt* fornecido foi positivo, gerando questões alinhadas com as instruções estabelecidas. Cada questão abordou os tópicos de Mecânica, especificamente Dinâmica das máquinas e Sistemas de controle, conforme solicitado. O enunciado de cada questão foi claro e preciso, desafiando os alunos em diferentes níveis de dificuldade, desde conceitos básicos até desafios mais complexos. As soluções geradas pelo modelo estavam corretas e seguiram as orientações, evidenciando a aplicação de estruturas de controle, repetição e até mesmo o uso de variáveis

heterogêneas e funções, atendendo assim às expectativas do professor para a elaboração de uma prova avaliativa.

- **Sider:** Inicialmente, a ferramenta se mostrou inferior ao retornar que não possuía conhecimento da parte especificada do arquivo passado, sendo que, as outras ferramentas, não tiveram dificuldade quanto a isso. Portanto, foi necessária uma pequena alteração no *prompt*. Mesmo após a alteração, a ferramenta não retornou conforme o esperando, se mostrando ser bem inconsistente para o objetivo do trabalho.

Em geral, as ferramentas ChatPDF e PopAi se mostraram positivas em relação aos testes, enquanto a ferramenta Sider se mostrou um pouco inferior. É notável que os retornos precisam de intervenção humana e que, quanto mais específico em relação à quantidade, melhor. O *prompt* em que era solicitado exatamente 5 questões para formular uma prova foi melhor atendido do que o *prompt* em que eram passados todos os tópicos e era solicitado uma questão para cada.

4 Resultados

Este Capítulo trata dos resultados obtidos a partir do desenvolvimento do trabalho. Na Seção 4.1 é apresentado o resultado obtido pela estratégia de Interação adaptativa, na Seção 4.2 é apresentado o resultado obtido por meio da estratégia de Estrutura de *prompt*, na Seção 4.3 é apresentado o resultado do teste explorando as Instruções personalizadas da ferramenta ChatGPT e na Seção 4.4 é apresentado os resultados dos testes realizados com uso da estrutura de *prompt* e do uso do arquivo PDF.

4.1 Resultado da estratégia de Interação adaptativa

Os resultados referentes à estratégia de Interação adaptativa se mostraram positivos, apesar de alguns pontos a serem considerados. As questões eram retornadas no formato esperado e com certeza serviriam de auxílio para um docente em uma criação de questões.

Entretanto, interagir como modo de conversa com o modelo e ir ajustando as questões conforme se espera por meio de um “bate-papo”, pode não ser eficiente. Principalmente pela não retenção de memória do modelo, o que obrigaria ao usuário repetir todas as informações em todas as vezes que desejasse criar uma nova questão ou iniciasse uma nova conversa.

4.2 Resultado da estratégia de Estrutura de *prompt*

Os resultados referentes à estratégia de Estrutura de *prompt* se mostram positivos e se assemelham aos alcançados pela estratégia Interação adaptativa. Entretanto, ter uma entrada padrão de pedido torna o processo mais eficiente.

Os resultados obtidos ao alterar os campos em que consideramos dinâmicos foram consideravelmente positivos. Porém, nem sempre o modelo gerava a quantidade de questões esperadas. Isso pode acontecer pela instrução não definir via número uma quantidade exata e por esperar que o modelo considerasse a quantidade de tópicos e fizesse a ligação com a instrução de criar questões para todas as quantidades.

Ainda assim, com um *prompt* bem elabora e instruções mais específicas, deve ser possível alcançar resultados melhores e mais próximos ao que uma questão final almeja.

4.3 Resultado da estratégia de Instruções personalizadas

Os resultados referentes à estratégia de Instruções personalizadas não eram muito positivos. Apesar de, em certo momento, parecidos com o da estratégia de Interação adaptativa,

a ferramenta não retorna o pedido por completo. Foram aplicadas algumas mudanças tanto no pedido quanto nas instruções personalizadas passadas, mas o resultado não evoluiu de outra forma.

Ao utilizar essa funcionalidade, é importante destacar alguns pontos que possam limitar sua eficiência. Um deles refere-se à restrição na quantidade de caracteres permitidos para passagem das informações. Entende-se que essa limitação pode ser necessária para evitar o truncamento ou a divisão do texto em partes durante a interpretação pelo modelo. Ao ser questionado através do *chat*, o modelo esclarece que não há um limite global de caracteres, mas alerta que respostas fragmentadas podem ocorrer caso o texto seja muito extenso. Ainda assim, o *chat* da ferramenta nos dá maior liberdade para passagem de informações, permitindo especificidade e fornecimento de mais detalhes sem se preocupar tanto com o número de caracteres.

Outro ponto foi mostrado de forma simples no Teste 3.5.2. Com instruções simples, foi atingido o objetivo de burlar as instruções passadas nos campos. Isso abre um questionamento sobre a capacidade de seguir a risca ou manter as instruções personalizadas como regras, tal qual o quão complexas, específicas e cuidadosas elas devem ser para não serem burladas.

Desta forma, a funcionalidade precisaria ser melhor analisada e ter seu funcionamento melhor compreendido, para entender a prioridade e como de fato o modelo trata essas instruções, principalmente após o teste em que “burlamos” as instruções passadas de forma simples, fica difícil considerar a estratégia como sendo uma estratégia principal.

4.4 Resultado da estratégia Estrutura de prompt com aquecimento por arquivo PDF

Nesta Seção são apresentados os resultados referentes às ferramentas que possuem a funcionalidade de interpretação de PDF. A Subseção 4.4.1 trata de uma avaliação das mesmas por meio de métricas definidas.

Em geral, as ferramentas ChatPDF e PopAi se mostraram positivas em relação aos testes, enquanto a ferramenta Sider se mostrou um pouco inferior. Enquanto conseguimos resultados aproveitáveis, e podem servir de auxílio para criação de questões das ferramentas ChatPDF e PopAi, a ferramenta Sider se mostrou um pouco mais difícil e problemática de lidar. Teríamos que analisar uma melhor forma de passar as informações para a ferramenta Sider para buscar um retorno aceitável, enquanto nas ferramentas ChatPDF e PopAi, o modelo de *prompt* previamente definido foi muito bem aproveitado e aceitado.

4.4.1 Avaliação de métricas

Para avaliar os resultados das ferramentas analisadas nesta estratégia, a Tabela 4.1 foi elaborada. Esta tabela contém métricas específicas utilizadas para mensurar o desempenho dos

retornos obtidos pelas ferramentas. Para a avaliação, optou-se por adotar a escala de Likert, uma abordagem que emprega afirmativas graduadas para expressar avaliações, categorizando-as como:

- **NA** - Não atende;
- **AP** - Atende parcialmente;
- **AT** - Atende totalmente.

Para basear a avaliação das questões geradas pelas ferramentas, usou-se o padrão definido de questões representado na Seção 3.1 pelo Quadro 3.1.1.

Métricas	ChatPDF	PopAi	Sider
Usa linguagem apropriada	AT	AT	AT
Atende o padrão de resposta esperado	AT	AP	AP
Aborda a matéria e o assunto solicitado	AT	AT	AT
Introduz o assunto	AP	AP	AP
Aborda o tópico da disciplina solicitado	AT	AT	AT
Retorna a quantidade esperada de questões	AP	AP	NA
Apresenta exemplos	AT	AP	AP
Apresenta uma solução correta	AT	AT	AT
Diversidade de questões	AT	AP	NA
Apresenta nível de dificuldade coerente	AP	AP	AP
Apresenta consistência na geração das questões	AP	AP	NA

Tabela 4.1 – Avaliação dos retornos das ferramentas

Apesar de, no geral, os resultados serem positivos, notam-se alguns pontos que poderiam ser melhorados e que precisam de ajuste manual no desenvolvimento das questões. São eles:

- **Melhorar introdução e Contextualizações:** Embora as contextualizações sejam adequadas, por vezes elas poderiam ser mais ricas e detalhadas, proporcionando aos alunos uma experiência mais ampla na aplicação dos conceitos abordados.
- **Criação completa das questões:** Embora o prompt especifique uma lista de tópicos de programação e o pedido peça para criar para cada um deles, a maioria das respostas não contempla todos os tópicos. Nota-se, entretanto, que quando o número é fixo para um objetivo mais específico, como a criação de uma prova contendo 5 questões, as ferramentas tendem a criar todas.
- **Redundância nas Questões:** Por vezes, as questões geradas acabam sendo bastante semelhantes, abordando os mesmos tópicos e temas.

- **Diversificação nos Exemplos:** Os exemplos fornecidos nas questões possuem bastante similaridade, indicando uma falta de diversidade de exemplos que envolvam as diferentes situações apresentadas.
- **Abordagem de Casos Especiais:** A inclusão de casos especiais nas questões, como situações limite ou cenários não triviais, poderia desafiar os alunos a pensar de maneira mais crítica e a aplicar os conceitos de programação de forma mais abrangente.
- **Exemplos de Código mais Complexos:** Embora os exemplos de código fornecidos sejam claros, poderiam ser adicionados casos mais complexos que gerassem uma discussão de implementações alternativas, podendo contribuir para questões mais enriquecedoras.
- **Incorporação de Boas Práticas:** O modelo poderia explicitamente destacar boas práticas de programação, como comentários descritivos, modularização do código, e considerações sobre eficiência e legibilidade, para promover uma abordagem mais consistente quanto ao desenvolvimento de software.

5 Considerações Finais

Neste capítulo serão apresentados as considerações finais deste trabalho. A seção 5.1 decorre a respeito das conclusões obtidas com as explorações realizadas no trabalho, já a Seção 5.2 apresenta sugestões para a continuidade deste trabalho.

5.1 Conclusão

O estudo realizado neste trabalho conclui que o modelo GPT, implementado nas ferramentas testadas, apresenta viabilidade para auxiliar na criação de questões automatizadas. No entanto, apesar de demonstrar resultados promissores e atender a estrutura de resposta passada, revelou-se um tanto frágil em relação à abrangência dos temas, tópicos e dificuldade das questões. Em relação a isso, o modelo se mostrou raso, se mostrando necessária revisão humana para alcançar resultados melhores.

Um aspecto a ser considerado é a dificuldade do modelo em gerar todas as questões, atendendo à instrução de forma completa e consistente. Além de, por vezes, gerar questões bastante semelhantes se mostrando redundante.

Outro aspecto a ser considerado é a tendência do modelo em ser redundante em suas saídas. Isso pode resultar em questões que, apesar de tecnicamente corretas, não contribuem substancialmente para a diversificação dos exemplos. A redundância impacta negativamente na qualidade do banco de questões, prejudicando a riqueza e variedade necessárias para atender às diferentes demandas de uma turma de programação.

Ademais, a limitação na abordagem de casos especiais da programação é evidente no desempenho do modelo. A falta de profundidade nas soluções propostas sugere que o GPT pode não ser adequado para lidar com situações mais complexas ou específicas, o que compromete a utilidade prática das questões geradas em ambientes educacionais que exigem um entendimento abrangente e aprofundado da programação.

Sendo assim, a falta de acesso direto à estrutura das ferramentas e das diretrizes do modelo, tal qual como o modelo é utilizado pelas ferramentas na geração das respostas, nos impede de ajustar e definir precisamente as saídas na intenção de tornar o processo o mais automático possível, o que poderia afetar também na garantia da diversidade das questões.

5.2 Trabalhos futuros

A partir dos experimentos e resultados obtidos por meio deste trabalho, têm-se algumas sugestões para a continuidade do mesmo.

Pode se considerar estender a ideia das estratégias usadas no trabalho e apresentá-las a partir de uma interface própria, considerando melhorar a estrutura de *prompt* definida neste trabalho para servir de base da interface, apresentando opções específicas a serem alteradas pelo usuário, como matéria, assunto e quantidade de questões.

As ferramentas ChatGPT e ChatPDF, exploradas neste trabalho, possuem *Application Programming Interface* (API) disponíveis. Sendo assim, torna-se uma oportunidade a exploração e viabilidade de adaptar essas ferramentas para estarem mais alinhadas com os objetivos do trabalho. Existem *frameworks* de código aberto desenvolvidos para intensificar a capacidade do uso dos modelos de linguagem. A depender do quão abertas e diversificadas forem as opções com o uso das APIs, por meio de uma linguagem de programação, podem ser alcançados resultados melhores do que os obtidos pelas estratégias usadas nas ferramentas já desenvolvidas.

Seguindo essa linha de pesquisa, podem ser explorados IAs que incorporem outros modelos de linguagem como o Gemini¹ da Google. Além disso, podemos citar modelos desenvolvidos nacionalmente com foco na língua portuguesa, como o Maritalk², que também é no modelo de um *chatbot* e possui API para integração, conferindo liberdade a cerca de adaptações do modelo.

Por fim, fazer a implementação de um sistema de criação automática de questões que podem ser armazenadas em um banco de dados, oferecendo um repositório robusto e organizado, facilitando o acesso futuro e a reutilização do conteúdo.

¹ <https://gemini.google.com/app>

² <https://www.maritaca.ai>

Referências

BRITO, P. S. S. O uso de corretores automáticos para o ensino de programação de computadores para alunos de engenharia. *UFOP*, v. 1, p. 10, 2019.

BRITO, P. S. S. *O Uso de Ferramentas Computacionais Para o Ensino de Programação Para Alunos de Engenharia*. Monografia de Bacharelado — Universidade Federal de Ouro Preto, 2019.

CAMBRIA, E.; WHITE, B. Jumping nlp curves: A review of natural language processing research [review article]. *IEEE Computational Intelligence Magazine*, v. 9, n. 2, p. 48–57, 2014.

FUCHS, K. Exploring the opportunities and challenges of nlp models in higher education: is chat gpt a blessing or a curse? *Frontiers in Education*, v. 8, 2023. ISSN 2504-284X. Disponível em: <<https://www.frontiersin.org/articles/10.3389/educ.2023.1166682>>.

GOMES, A.; MENDES, A. Learning to program - difficulties and solutions. In: . [S.l.: s.n.], 2007. p. 283–287.

KOTHARI, A. N. Chatgpt, large language models, and generative ai as future augments of surgical cancer care. *Annals of Surgical Oncology*, v. 30, n. 6, p. 3174–3176, Jun 2023. ISSN 1534-4681. Disponível em: <<https://doi.org/10.1245/s10434-023-13442-2>>.

M. FRIDORFF-JENS P.K., L. P. e. a. T. Transforming students into digital academics: a challenge at both the individual and the institutional level. *BMC Med Educ* 15, 48 (2015). <https://doi.org/10.1186/s12909-015-0330-5>, 2015.

PATROCÍNIO, J. A. do. *OpCoders Judge: Uma Versão Online Para o Corretor Automático de Exercícios de Programação do Projeto OpCoders*. Monografia de Bacharelado — Universidade Federal de Ouro Preto, 2023.

RADFORD, A.; NARASIMHAN, K.; SALIMANS, T.; SUTSKEVER, I. et al. Improving language understanding by generative pre-training. OpenAI, 2018.

ROTHER, S.; NARAYAN, S.; SEVERYN, A. Leveraging Pre-trained Checkpoints for Sequence Generation Tasks. *Transactions of the Association for Computational Linguistics*, v. 8, p. 264–280, 06 2020. ISSN 2307-387X. Disponível em: <https://doi.org/10.1162/tac1_a_00313>.

SANTOS, M. C. dos. Entrevistando um robô: notas sobre a aplicação experimental da metodologia eeaf usando a ferramenta chatgpt de inteligência artificial. *Comunicação & Inovação* | v. 24 | e20238987 | jan.-dec. | 2023 | ISSN 2178-0145. <https://doi.org/10.13037/ci.vol24.e20238987>, p. 17, 2023.

Steve Mollman. *ChatGPT gained 1 million users in under a week. Here's why the AI chatbot is primed to disrupt search as we know it*. 2022. Disponível em: <<https://finance.yahoo.com/news/chatgpt-gained-1-million-followers-224523258.html>>. Acesso em: 27 de junho 2023.

Apêndices

APÊNDICE A – Testes no estilo conversação com ChatGPT

Usuário: Gostaria que você criasse questões sobre programação para mim de acordo com alguns padrões que eu te passar

ChatGPT: Claro! Fico feliz em ajudar. Por favor, forneça os padrões que você gostaria que eu usasse para criar as questões de programação. Você pode especificar a linguagem de programação, o nível de dificuldade, tópicos específicos ou qualquer outro requisito que desejar.

Usuário: Use como tema o MRU da física, dê três exemplos de execução e respostas esperadas do programa, use apenas expressões básicas e I/O, use a linguagem python e me dê uma solução para a questão.

ChatGPT: Com base no tema MRU (Movimento Retilíneo Uniforme) da física, vou criar três exemplos de questões de programação usando expressões básicas e I/O (entrada e saída de dados) em Python. Aqui estão as questões, seguidas das soluções correspondentes:

Questão 1:

Escreva um programa em Python que calcule a velocidade média de um objeto em um Movimento Retilíneo Uniforme (MRU). O programa deve solicitar ao usuário a distância percorrida em metros e o tempo gasto em segundos. Em seguida, exiba a velocidade média em metros por segundo.

Exemplo de execução 1:

css

Digite a distância percorrida (em metros): 200

Digite o tempo gasto (em segundos): 20

A velocidade média é de 10.0 m/s.

Solução em Python 1:

python

```
distancia = float(input("Digite a distância percorrida (em metros): "))
```

```
tempo = float(input("Digite o tempo gasto (em segundos): "))
```

```
velocidade_media = distancia / tempo
```

```
print("A velocidade média é de", velocidade_media, "m/s.")
```

Questão 2:

Escreva um programa em Python que calcule a distância percorrida por um objeto em um Movimento Retilíneo Uniforme (MRU). O programa deve solicitar ao usuário a velocidade média em metros por segundo e o tempo gasto em segundos. Em seguida, exiba a distância percorrida em metros.

Exemplo de execução 2:

css

Digite a velocidade média (em m/s): 15

Digite o tempo gasto (em segundos): 10

A distância percorrida é de 150 metros.

Solução em Python 2:

python

```
velocidade_media = float(input("Digite a velocidade média (em m/s): "))
```

```
tempo = float(input("Digite o tempo gasto (em segundos): "))
distancia = velocidade_media * tempo
print("A distância percorrida é de", distancia, "metros.")
```

Questão 3:

Escreva um programa em Python que calcule o tempo necessário para percorrer uma certa distância em um Movimento Retilíneo Uniforme (MRU). O programa deve solicitar ao usuário a distância em metros e a velocidade média em metros por segundo. Em seguida, exiba o tempo necessário em segundos.

Exemplo de execução 3:

css

```
Digite a distância a ser percorrida (em metros): 500
Digite a velocidade média (em m/s): 25
O tempo necessário é de 20.0 segundos.
Solução em Python 3:
```

python

```
distancia = float(input("Digite a distância a ser percorrida (em metros): "))
velocidade_media = float(input("Digite a velocidade média (em m/s): "))
tempo = distancia / velocidade_media
print("O tempo necessário é de", tempo, "segundos.")
```

Essas são três questões básicas de programação em Python sobre o Movimento Retilíneo Uniforme (MRU). Você pode executar cada programa e verificar se as respostas estão corretas. Sinta-se à vontade para personalizar ou aprimorar as questões e as soluções de acordo com suas necessidades.

Usuário: Vamos tentar novamente

ChatGPT: Peço desculpas pela confusão anterior. Aqui está uma questão relacionada ao Movimento Retilíneo Uniforme (MRU) com uma solução em Python:

Questão:

Escreva um programa em Python que calcule a posição de um objeto em um Movimento Retilíneo Uniforme (MRU). O programa deve solicitar ao usuário a velocidade do objeto em metros por segundo (m/s) e o tempo decorrido em segundos (s). Em seguida, exiba a posição final do objeto em metros (m). Considere que a posição inicial é zero.

Exemplo de execução:

java

```
Digite a velocidade do objeto (m/s): 10
Digite o tempo decorrido (s): 5
A posição final do objeto é 50 metros.
Solução em Python:
```

python

```
velocidade = float(input("Digite a velocidade do objeto (m/s): "))
tempo = float(input("Digite o tempo decorrido (s): "))

posicao = velocidade * tempo
print("A posição final do objeto é", posicao, "metros.")
```

Nesta questão, o programa solicita a velocidade e o tempo ao usuário e, em seguida, calcula a posição multiplicando a velocidade pelo tempo. A resposta é exibida ao usuário. Lembre-se de que essa solução considera uma posição inicial de zero.

Usuário: Use como tema o MRU da física, me faça uma questão com três exemplos de execução e respostas esperadas do programa, use apenas expressões básicas e IO, use a linguagem python e me dê uma solução para a questão.

ChatGPT: Com base no tema MRU (Movimento Retilíneo Uniforme) da física, aqui está uma questão que apresenta três exemplos de execução e as respostas esperadas do programa. Utilizarei apenas expressões básicas e entrada/saída de dados em Python.

Questão:

Escreva um programa em Python que calcule a distância percorrida por um objeto em um Movimento Retilíneo Uniforme (MRU). O programa deve solicitar ao usuário a velocidade do objeto em metros por segundo (m/s) e o tempo decorrido em segundos (s). Em seguida, exiba a distância percorrida pelo objeto em metros (m).

Exemplo de execução 1:

java

```
Digite a velocidade do objeto (m/s): 10
Digite o tempo decorrido (s): 5
A distância percorrida pelo objeto é 50 metros.
```

Exemplo de execução 2:

java

```
Digite a velocidade do objeto (m/s): 15
Digite o tempo decorrido (s): 2.5
A distância percorrida pelo objeto é 37.5 metros.
```

Exemplo de execução 3:

java

```
Digite a velocidade do objeto (m/s): 8.5
Digite o tempo decorrido (s): 10
A distância percorrida pelo objeto é 85 metros.
```

Solução em Python:

python

```
velocidade = float(input("Digite a velocidade do objeto (m/s): "))
tempo = float(input("Digite o tempo decorrido (s): "))
```



```
distancia = velocidade * tempo
print("A distância percorrida pelo objeto é", distancia, "metros.")
```

Nesta questão, o programa solicita ao usuário a velocidade do objeto e o tempo decorrido. Em seguida, calcula a distância percorrida multiplicando a velocidade pelo tempo e exibe a resposta ao usuário. Lembre-se de que a fórmula básica para calcular a distância em um MRU é $d = v * t$, onde d é a distância, v é a velocidade e t é o tempo.

Usuário: Me faça uma usando decisões simples agora

ChatGPT: Claro! Aqui está uma questão sobre MRU utilizando decisões simples (estruturas condicionais) em Python:

Questão:

Escreva um programa em Python que determine se um objeto está em repouso, movimento para a direita ou movimento para a esquerda em um Movimento Retilíneo Uniforme (MRU). O programa deve solicitar ao usuário a velocidade do objeto em metros por segundo (m/s). Considere que velocidades positivas indicam movimento para a direita e velocidades negativas indicam movimento para a esquerda. Em seguida, exiba uma mensagem indicando o estado do objeto.

Exemplo de execução 1:

```
bash
```

```
Digite a velocidade do objeto (m/s): 0
O objeto está em repouso.
Exemplo de execução 2:
```

```
less
```

```
Digite a velocidade do objeto (m/s): 10
O objeto está em movimento para a direita.
Exemplo de execução 3:
```

```
less
```

```
Digite a velocidade do objeto (m/s): -5
O objeto está em movimento para a esquerda.
Solução em Python:
```

```
python
```

```
velocidade = float(input("Digite a velocidade do objeto (m/s): "))

if velocidade == 0:
    print("O objeto está em repouso.")
elif velocidade > 0:
    print("O objeto está em movimento para a direita.")
else:
    print("O objeto está em movimento para a esquerda.")
```

Nesta questão, o programa solicita ao usuário a velocidade do objeto. Em seguida, utiliza uma estrutura condicional para determinar o estado do

objeto com base no valor da velocidade. Se a velocidade for igual a zero, o objeto está em repouso. Se a velocidade for maior que zero, o objeto está em movimento para a direita. Caso contrário, se a velocidade for menor que zero, o objeto está em movimento para a esquerda. O programa exibe a mensagem correspondente ao estado do objeto.

Usuário: Você consegue me dizer o padrão que você usa para criar as questões?

ChatGPT: Como modelo de linguagem treinado pela OpenAI, minha capacidade de gerar questões se baseia em padrões e estratégias aprendidas durante o treinamento. Embora eu não possa fornecer informações detalhadas sobre o código interno específico utilizado pela OpenAI, posso compartilhar algumas informações gerais.

Ao gerar questões, eu procuro seguir algumas diretrizes comuns:

Compreensão do tema: Eu tento entender o tema proposto pelo usuário para criar questões relevantes dentro desse contexto. Isso envolve a análise das informações fornecidas e a utilização de conhecimento prévio.

Estrutura e formato: Eu procuro criar questões com uma estrutura clara e coerente, geralmente incluindo uma descrição do problema, solicitações para entrada de dados e uma declaração do resultado esperado.

Variedade e desafio: Eu tento criar questões variadas em termos de dificuldade, tópicos e requisitos. Isso inclui questões simples, intermediárias e avançadas, bem como a exploração de diferentes conceitos relacionados ao tema.

Personalização: Eu tento incorporar as preferências e requisitos do usuário sempre que possível, como escolha de linguagem de programação, nível de dificuldade, tópicos específicos, padrões de código, entre outros.

Validade e precisão: Eu me esforço para garantir que as questões geradas sejam válidas e corretas em relação ao tema proposto. No entanto, como um modelo de linguagem, é importante que o usuário revise e valide as questões geradas para garantir sua exatidão.

Lembrando que sou alimentado por uma grande quantidade de dados textuais e exemplos de programação, o que me permite criar questões em diferentes domínios e linguagens de programação. No entanto, a geração de questões pode ter limitações e é sempre importante revisar e validar as questões geradas para garantir que atendam aos requisitos específicos do usuário.

Usuário: Se eu te passar uma questão de exemplo, você pode criar questões com a mesma estrutura?

ChatGPT: Certamente! Se você fornecer uma questão de exemplo com uma determinada estrutura, posso tentar criar outras questões com base nessa estrutura. Por favor, compartilhe a questão de exemplo e, em seguida, vou tentar criar outras questões seguindo o mesmo formato.

Usuário: Questão:

A função do movimento uniforme nos permite encontrar a posição (S) de um objeto que se movimenta com velocidade constante (v) em qualquer instante de tempo (t), desde que saibamos sua posição inicial (S_i), através da seguinte fórmula: $S = S_i + v * t$.

Implemente um programa que receba do usuário os valores de entrada, e as armazene em variáveis: posição inicial (S_i), a velocidade (v) e o instante de tempo (t), respectivamente. Todas as entradas são números reais maiores do que 0 (zero). Caso os valores sejam inválidos, o programa apenas imprime uma mensagem de erro e é encerrado, caso contrário, o programa deve realizar o cálculo da posição final (S) e imprimir o resultado no terminal com uma precisão de 2 casas decimais.

Sua resposta deve conter as mensagens exatamente como definidas nos exemplos de execução.

Exemplo de Execução 1

```
Informe a posição inicial (Si): 10
Informe a velocidade (v): 1
Informe o instante de tempo (t): 5
A posição final no tempo t = 5.00 será S = 15.00
```

Exemplo de Execução 2

```
Informe a posição inicial (Si): 55.7
Informe a velocidade (v): 60
Informe o instante de tempo (t): 5.8
A posição final no tempo t = 5.80 será S = 403.70
```

Exemplo de Execução 3

```
Informe a posição inicial (Si): 0
Informe a velocidade (v): 60.8
Informe o instante de tempo (t): 5.8
Entradas inválidas
```

Solução:

```
Si = float(input('Informe a posição inicial (Si): '))
v = float(input('Informe a velocidade (v): '))
t = float(input('Informe o instante de tempo (t): '))
if Si > 0 and v > 0 and t > 0:
    S = Si + v * t
    print(f'A posição final no tempo t = {t:.2f} será S = {S:.2f}')
else:
    print('Entradas inválidas')
```

ChatGPT: Com base na estrutura da questão fornecida, aqui estão mais três questões criadas seguindo o mesmo formato:

Questão 1:

Implemente um programa em Python que calcule a posição final de um objeto em um movimento uniforme. O programa deve solicitar ao usuário a posição inicial (S_i), a velocidade (v) e o instante de tempo (t). Verifique se todas as entradas são números reais maiores que zero. Caso sejam válidas, realize o cálculo da posição final (S) e imprima o resultado no terminal com uma precisão de 2 casas decimais. Caso contrário, exiba a mensagem "Entradas inválidas".

Exemplo de Execução:

java

Informe a posição inicial (Si): 20
Informe a velocidade (v): 2.5
Informe o instante de tempo (t): 8.75
A posição final no tempo $t = 8.75$ será $S = 41.25$

Questão 2:

Implemente um programa em Python que determine a posição final de um objeto em um movimento uniforme. O programa deve receber do usuário a posição inicial (Si), a velocidade (v) e o instante de tempo (t). Verifique se todas as entradas são números reais maiores que zero. Se as entradas forem válidas, realize o cálculo da posição final (S) e exiba o resultado no terminal com uma precisão de 2 casas decimais. Caso contrário, exiba a mensagem de erro "Entradas inválidas".

Exemplo de Execução:

java

Informe a posição inicial (Si): 50.25
Informe a velocidade (v): 12.7
Informe o instante de tempo (t): 6.5
A posição final no tempo $t = 6.50$ será $S = 132.85$

Questão 3:

Escreva um programa em Python que calcule a posição final de um objeto em um movimento uniforme. Solicite ao usuário a posição inicial (Si), a velocidade (v) e o instante de tempo (t). Verifique se todas as entradas são números reais maiores que zero. Caso as entradas sejam válidas, realize o cálculo da posição final (S) e mostre o resultado no terminal com uma precisão de 2 casas decimais. Se as entradas forem inválidas, imprima "Entradas inválidas".

Exemplo de Execução:

java

Informe a posição inicial (Si): 0
Informe a velocidade (v): 15.2
Informe o instante de tempo (t): 3.5
A posição final no tempo $t = 3.50$ será $S = 53.20$

Essas questões seguem a mesma estrutura e formato da questão de exemplo fornecida, apresentando mensagens exatamente como definidas nos exemplos de execução.

Usuário: Crie uma questão com a mesma estrutura com usando como tema Química (Concentração)

ChatGPT: Claro! Aqui está uma questão com a mesma estrutura, mas usando o tema de Química, mais especificamente sobre concentração:

Questão:

A concentração de uma solução pode ser calculada utilizando a fórmula: $C = m/V$, onde C representa a concentração em gramas por litro (g/L), m é a massa do soluto em gramas (g) e V é o volume da solução em litros (L). Implemente um programa em Python que solicite ao usuário a massa do soluto e o volume da solução. Em seguida, calcule a concentração da solução e exiba o resultado no terminal com uma precisão de 2 casas decimais.

Exemplo de Execução:

java

```
Informe a massa do soluto (g): 25.5
Informe o volume da solução (L): 2.1
A concentração da solução é 12.14 g/L.
Solução:
```

python

```
massa_soluto = float(input("Informe a massa do soluto (g): "))
volume_solucao = float(input("Informe o volume da solução (L): "))

concentracao = massa_soluto / volume_solucao
print(f"A concentração da solução é {concentracao:.2f} g/L.")
```

Nesta questão, o programa solicita ao usuário a massa do soluto e o volume da solução. Em seguida, realiza o cálculo da concentração dividindo a massa do soluto pelo volume da solução. O resultado é exibido no terminal com uma precisão de 2 casas decimais.

Usuário: Você reconhece arquivos pdf?

ChatGPT: Sim, reconheço arquivos PDF e posso ajudar a fornecer informações ou realizar tarefas relacionadas a eles. No entanto, como um modelo de linguagem baseado em texto, não posso exibir ou manipular diretamente o conteúdo visual ou os elementos gráficos presentes em um arquivo PDF. Mas posso ajudar respondendo perguntas sobre o texto do PDF, extrair informações ou fornecer orientações com base no conteúdo textual. Se você tiver alguma dúvida ou precisar de assistência específica relacionada a um arquivo PDF, sinta-se à vontade para compartilhar mais detalhes.

Usuário: Crie agora uma questão com o tema MRU, que use repetição while com a mesma estrutura

ChatGPT: Claro! Aqui está uma questão sobre MRU que utiliza a estrutura de repetição while:

Questão:

Escreva um programa em Python que determine o tempo necessário para um objeto percorrer uma certa distância em um Movimento Retilíneo Uniforme (MRU). O programa deve solicitar ao usuário a distância a ser percorrida em metros (m) e a velocidade do objeto em metros por segundo (m/s). Em seguida, calcule e exiba o tempo necessário para percorrer a distância informada.

Exemplo de Execução:

less

```
Digite a distância a ser percorrida (m): 200
Digite a velocidade do objeto (m/s): 20
O objeto levará 10 segundos para percorrer a distância de 200 metros.
Solução:
```

python

```
distancia = float(input("Digite a distância a ser percorrida (m): "))
velocidade = float(input("Digite a velocidade do objeto (m/s): "))
```

```
tempo = distancia / velocidade
print(f"O objeto levará {tempo} segundos para percorrer a distância de
{distancia} metros.")
```

Nesta questão, o programa solicita ao usuário a distância a ser percorrida e a velocidade do objeto. Em seguida, utiliza a fórmula do MRU para calcular o tempo necessário para percorrer a distância informada. O resultado é exibido no terminal. Note que não foi necessário utilizar a estrutura de repetição while nesta questão específica. Caso você deseje uma questão com while, por favor, me informe para que eu possa ajustar a solução.

Usuário: Eu quero uma questão com while

ChatGPT: Desculpe pelo equívoco anterior. Aqui está uma questão sobre MRU que utiliza a estrutura de repetição while:

Questão:

Escreva um programa em Python que simule um experimento de Movimento Retilíneo Uniforme (MRU). O programa deve solicitar ao usuário a velocidade inicial do objeto em metros por segundo (m/s) e o tempo decorrido em segundos (s). Utilizando um loop while, o programa deve calcular e exibir a posição do objeto a cada segundo durante o tempo informado. Considere que a posição inicial do objeto é 0 metros. O programa deve continuar a calcular a posição do objeto a cada segundo até o tempo final.

Exemplo de Execução:

java

```
Digite a velocidade inicial do objeto (m/s): 10
Digite o tempo decorrido (s): 5
Após 1 segundo(s), a posição do objeto é 10 metros.
Após 2 segundo(s), a posição do objeto é 20 metros.
Após 3 segundo(s), a posição do objeto é 30 metros.
Após 4 segundo(s), a posição do objeto é 40 metros.
Após 5 segundo(s), a posição do objeto é 50 metros.
Solução:
```

python

```
velocidade_inicial = float(input("Digite a velocidade inicial do objeto  
(m/s): "))  
tempo_decorrido = int(input("Digite o tempo decorrido (s): "))  
  
posicao = 0  
tempo_atual = 1  
  
while tempo_atual <= tempo_decorrido:  
    posicao = velocidade_inicial * tempo_atual  
    print(f"Após {tempo_atual} segundo(s), a posição do objeto é  
{posicao} metros.")  
    tempo_atual += 1
```

Nesta questão, o programa solicita ao usuário a velocidade inicial do objeto e o tempo decorrido. Em seguida, utiliza um loop while para calcular e exibir a posição do objeto a cada segundo durante o tempo informado. A cada iteração do loop, o tempo é incrementado em 1 segundo e a posição é calculada multiplicando a velocidade inicial pelo tempo atual. O resultado é exibido no terminal.

Usuário: Você utiliza alguma ferramenta para evitar plágio?

ChatGPT: Como um modelo de linguagem treinado pela OpenAI, eu não possuo a capacidade de acessar ou utilizar diretamente ferramentas externas para verificar ou evitar plágio. Meu objetivo é fornecer informações e ajudar com base no conhecimento que adquiri durante o treinamento com uma ampla variedade de textos. É importante que os usuários da plataforma utilizem as informações fornecidas de maneira responsável e ética, evitando qualquer forma de plágio ou violação dos direitos autorais. Caso você precise citar ou referenciar fontes específicas, é recomendado realizar sua própria pesquisa e garantir a correta atribuição das informações.

Usuário: Eu posso ativar alguma diretriz para que você use uma ferramenta para evitar plágio?

ChatGPT: Como um modelo de linguagem da OpenAI, não tenho acesso a ferramentas específicas de detecção de plágio ou qualquer outra ferramenta externa. Minha função é gerar respostas e fornecer informações com base no conhecimento prévio adquirido durante o treinamento. No entanto, como usuário, você pode utilizar ferramentas de detecção de plágio disponíveis online para verificar a originalidade do conteúdo. Essas ferramentas podem ser úteis para garantir a autenticidade e a integridade do trabalho que está sendo desenvolvido.

Usuário: Como posso te ajudar a melhorar a criação das questões?

ChatGPT: Fico feliz que você queira ajudar a melhorar a criação das questões! Aqui estão algumas maneiras pelas quais você pode contribuir:

Forneça exemplos claros: Ao solicitar uma questão, forneça exemplos claros de enunciados, respostas esperadas e, se necessário, soluções em código. Quanto mais detalhes você fornecer, melhor será a minha compreensão e resposta.

Especifique o formato e os requisitos: Se houver um formato específico que você deseje para as questões (por exemplo, múltipla escolha, preenchimento de lacunas, etc.) ou requisitos adicionais, indique-os claramente.

Dê feedback sobre as questões geradas: Após receber as questões geradas, você pode fornecer feedback sobre sua qualidade, adequação ao tema proposto e se elas atendem às suas expectativas. Isso me ajudará a entender melhor suas preferências e ajustar a criação das questões, se necessário.

Sugira melhorias: Se você tiver sugestões específicas para melhorar a estrutura, a variedade ou a complexidade das questões, sinta-se à vontade para compartilhá-las. Isso pode incluir exemplos adicionais, alterações nas instruções ou adição de conceitos específicos.

Lembre-se de que estou aqui para auxiliá-lo e aprender com suas contribuições. Quanto mais informações você fornecer e mais detalhes específicos compartilhar, melhor serei capaz de criar questões relevantes e úteis. Agradeço antecipadamente por sua colaboração!

Usuário: Eu gostaria que você fizesse o possível para me retornar questões diferentes das já criadas

ChatGPT: Compreendo seu pedido. Vou fazer o possível para criar questões diferentes das que já foram criadas anteriormente. Vou utilizar uma abordagem variada e explorar diferentes aspectos do tema para garantir a diversidade das questões. No entanto, lembre-se de que como um modelo de linguagem, minha criação é baseada em padrões e exemplos existentes, então pode haver alguma sobreposição em termos de conceitos. Farei o melhor para oferecer uma variedade de questões com base no tema selecionado.

Usuário: Vamos adicionar a diretriz de dificuldade, onde você vai classificar a questão criada em uma escala de 1 a 10.

ChatGPT: Com base na sua nova diretriz de dificuldade, vou classificar cada questão criada em uma escala de 1 a 10, em que 1 representa uma questão muito fácil e 10 representa uma questão muito difícil. Levarei em consideração o conteúdo, a complexidade dos cálculos ou conceitos envolvidos, bem como o nível de conhecimento exigido dos alunos para responder corretamente.

Ao criar as questões, tentarei oferecer uma variedade de níveis de dificuldade para atender às suas necessidades. No entanto, vale ressaltar que a avaliação de dificuldade é subjetiva e pode variar de pessoa para pessoa. Farei o possível para classificar as questões com base em uma avaliação geral de complexidade.

Usuário: Quero também que ao final de cada questão você compartilhe a fonte usada para criação dela

ChatGPT: Entendido! Vou adicionar ao final de cada questão a fonte utilizada para a criação dela. A fonte pode ser uma referência específica, um livro, um artigo científico, um site confiável, ou qualquer outra fonte relevante que tenha sido consultada para obter as informações necessárias para a elaboração da questão. Isso ajudará a fornecer uma referência e transparência sobre a origem das informações utilizadas.

Usuário: Quero que você considere também me informar conteúdos que possuam copyright, direitos autorais ou que possam ser considerados como plágio se eu os usar

ChatGPT: Compreendo a sua preocupação em relação a direitos autorais e plágio. Farei o possível para evitar a utilização de conteúdos que possuam restrições de direitos autorais ou que possam ser considerados plágio.

Como um modelo de linguagem, fui treinado em uma ampla variedade de textos, incluindo materiais com direitos autorais. No entanto, minha função é gerar informações gerais e fornecer orientações, evitando a reprodução exata de conteúdos protegidos por direitos autorais.

Cabe ressaltar que, embora eu me esforce para fornecer informações precisas e úteis, é sempre importante que você faça uma pesquisa adicional e verifique a fonte de qualquer informação específica antes de utilizá-la em contextos que possam violar direitos autorais ou plágio.

Usuário: Lembre-se me passar a dificuldade da questão

ChatGPT: Com certeza! Além de fornecer a fonte de referência ao final de cada questão, também indicarei a dificuldade da questão em uma escala de 1 a 10, como solicitado. A dificuldade será avaliada com base nos critérios relevantes ao tema da questão, como complexidade dos conceitos, nível de cálculos envolvidos, demanda de raciocínio lógico, entre outros fatores pertinentes. Dessa forma, você terá uma indicação clara da dificuldade da questão proposta.

Usuário: Agora me gere três questões sobre Movimento Retilíneo Uniforme escalonando a dificuldade.

ChatGPT: Com base no tema de Movimento Retilíneo Uniforme (MRU), vou gerar três questões escalonando a dificuldade conforme solicitado. Seguem as questões:

Questão 1 (Dificuldade: 3/10):

Um objeto se move em uma trajetória retilínea com velocidade constante de 10 m/s. Se a posição inicial do objeto é de 5 metros, determine a posição do objeto após 8 segundos.

Questão 2 (Dificuldade: 6/10):

Um carro percorre uma estrada em linha reta a uma velocidade constante de 20 m/s. Se a posição inicial do carro é de 100 metros, determine o tempo necessário para o carro percorrer uma distância de 500 metros.

Questão 3 (Dificuldade: 9/10):

Um avião decola de um aeroporto e percorre uma pista com uma velocidade constante de 80 m/s. Se a distância total da pista é de 2000 metros, determine o tempo necessário para o avião percorrer toda a pista e decolar.

Lembre-se de que a dificuldade é subjetiva e pode variar de acordo com o conhecimento e a experiência de cada pessoa. Essas questões foram projetadas para aumentar gradativamente o nível de complexidade e desafio em relação aos cálculos e conceitos do MRU.

Usuário: Desconsidere o último pedido. Sempre que pedir para você criar questões serão exercícios de programação conforme passado desde o início de nossa conversa

ChatGPT: Peço desculpas pela confusão anterior. Vamos continuar com o foco em exercícios de programação. Por favor, forneça as diretrizes para a criação de questões de programação e eu ficarei feliz em gerar exemplos com base nelas.

Usuário: Me gere três questões sobre Movimento Retilíneo Uniforme, utilizando apenas IO e expressões da linguagem escalonando a dificuldade

ChatGPT: Com base no tema de Movimento Retilíneo Uniforme (MRU) e utilizando apenas expressões e IO na linguagem Python, vou gerar três questões escalonando a dificuldade conforme solicitado. Seguem as questões:

Questão 1 (Dificuldade: 3/10):

Escreva um programa em Python que receba a velocidade constante de um objeto em MRU e o tempo decorrido desde o início do movimento. Em seguida, o programa deve calcular e imprimir a distância percorrida pelo objeto. Considere que a velocidade é dada em metros por segundo (m/s).

Exemplo de execução:

Entrada:

Velocidade: 10

Tempo: 5

Saída:

Distância percorrida: 50 metros

Questão 2 (Dificuldade: 6/10):

Escreva um programa em Python que receba a posição inicial de um objeto em MRU, a velocidade constante do objeto e o tempo decorrido desde o início do movimento. Em seguida, o programa deve calcular e imprimir a posição final do objeto. Considere que a posição inicial e a posição final são dadas em metros (m).

Exemplo de execução:

Entrada:

Posição inicial: 20

Velocidade: 5

Tempo: 8

Saída:

Posição final: 60 metros

Questão 3 (Dificuldade: 9/10):

Escreva um programa em Python que receba a posição inicial de um objeto em MRU, a velocidade constante do objeto e a posição final desejada. O programa deve calcular e imprimir o tempo necessário para o objeto atingir a posição final desejada. Considere que a posição inicial, a posição final e a velocidade são dadas em metros (m).

Exemplo de execução:

Entrada:

Posição inicial: 10

Velocidade: 2

Posição final desejada: 50

Saída:

Tempo necessário: 20 segundos

Essas questões foram escalonadas em termos de complexidade, começando com cálculos básicos de distância e avançando para cálculos de posição final e tempo necessário. Lembre-se de que a dificuldade é subjetiva e pode variar de acordo com o conhecimento e a experiência de cada pessoa.

Usuário: Obrigado! Gostaria que você adicionasse nas questões também um exemplo de solução. Além disso, você não está me passando a fonte e nem me falando sobre conteúdos que podem ser considerados plágios se eu usar. ChatGPT: Peço desculpas novamente pela omissão. Vou adicionar exemplos de solução às questões e fornecer informações sobre a fonte utilizada, bem como alertar sobre a possibilidade de conteúdos que possam ser considerados plágio. Seguem as questões revisadas:

Questão 1 (Dificuldade: 3/10):

Escreva um programa em Python que receba a velocidade constante de um objeto em MRU e o tempo decorrido desde o início do movimento. Em seguida, o programa deve calcular e imprimir a distância percorrida pelo objeto. Considere que a velocidade é dada em metros por segundo (m/s).

Exemplo de Execução:

Entrada:

Velocidade: 10

Tempo: 5

Saída:

Distância percorrida: 50 metros

Solução:

```
python
```

```
velocidade = float(input("Velocidade: "))
```

```
tempo = float(input("Tempo: "))
```

```
distancia = velocidade * tempo
```

```
print("Distância percorrida:", distancia, "metros")
```

Fonte: Material didático de Física.

Questão 2 (Dificuldade: 6/10):

Escreva um programa em Python que receba a posição inicial de um objeto em MRU, a velocidade constante do objeto e o tempo decorrido desde o início do movimento. Em seguida, o programa deve calcular e imprimir a posição final do objeto. Considere que a posição inicial e a posição final são dadas em metros (m).

Exemplo de Execução:

Entrada:

Posição inicial: 20

Velocidade: 5

Tempo: 8

Saída:

Posição final: 60 metros

Solução:

python

```
posicao_inicial = float(input("Posição inicial: "))
velocidade = float(input("Velocidade: "))
tempo = float(input("Tempo: "))
```

```
posicao_final = posicao_inicial + velocidade * tempo
```

```
print("Posição final:", posicao_final, "metros")
```

Fonte: Material didático de Física.

Questão 3 (Dificuldade: 9/10):

Escreva um programa em Python que receba a posição inicial de um objeto em MRU, a velocidade constante do objeto e a posição final desejada. O programa deve calcular e imprimir o tempo necessário para o objeto atingir a posição final desejada. Considere que a posição inicial, a posição final e a velocidade são dadas em metros (m).

Exemplo de Execução:

Entrada:

Posição inicial: 10

Velocidade: 2

Posição final desejada: 50

Saída:

Tempo necessário: 20 segundos

Solução:

python

```
posicao_inicial = float(input("Posição inicial: "))
velocidade = float(input("Velocidade: "))
posicao_final = float(input("Posição final desejada: "))
```

```
tempo_necessario = (posicao_final - posicao_inicial) / velocidade
```

```
print("Tempo necessário:", tempo_necessario, "segundos")
```

Fonte: Material didático de Física.

Espero que agora as questões estejam mais completas e atendam às suas expectativas. Se tiver alguma outra solicitação ou dúvida, por favor, me avise!

Usuário: Com base nessas últimas questões criadas, pode me descrever como você classificou as dificuldades delas?

ChatGPT: Claro! Vou descrever como classifiquei as dificuldades das questões de acordo com os critérios utilizados:

Questão 1 (Dificuldade: 3/10):

Nessa questão, é solicitado apenas o cálculo da distância percorrida em um Movimento Retilíneo Uniforme (MRU), com a entrada da velocidade e tempo. A solução envolve uma única expressão matemática simples,

tornando-a relativamente fácil. Portanto, a dificuldade foi classificada como 3/10.

Questão 2 (Dificuldade: 6/10):

Nessa questão, além do cálculo da posição final, também é necessário considerar a posição inicial do objeto. Isso adiciona um pouco mais de complexidade ao cálculo, mas ainda é uma questão de nível intermediário. Por isso, a dificuldade foi classificada como 6/10.

Questão 3 (Dificuldade: 9/10):

Nessa questão, é solicitado o cálculo do tempo necessário para um objeto atingir uma posição final desejada. Isso requer uma inversão da fórmula do MRU e pode ser mais desafiador para alguns estudantes. Portanto, considerando essa inversão e a complexidade do cálculo envolvido, a dificuldade foi classificada como 9/10.

É importante ressaltar que a classificação de dificuldade é subjetiva e pode variar de acordo com o conhecimento e a experiência de cada pessoa. Os critérios utilizados para classificar as dificuldades incluem a complexidade dos cálculos, a necessidade de inversão de fórmulas e a quantidade de variáveis envolvidas.

Espero que essa explicação esclareça como as dificuldades foram determinadas. Se tiver mais dúvidas ou necessitar de outros esclarecimentos, estou à disposição!

Usuário: Ao compartilhar a fonte você poderia me passar o link dela?

ChatGPT: Peço desculpas pela confusão. Como um modelo de linguagem treinado pela OpenAI, não tenho acesso direto à internet e, portanto, não posso fornecer links diretos para fontes externas. No entanto, posso citar as fontes de referência de maneira geral, sem especificar links. Isso significa que posso mencionar que as questões foram baseadas em material didático de Física, por exemplo, sem fornecer um link específico.

Se você precisar de informações adicionais sobre um determinado tópico ou fonte específica, posso tentar fornecer detalhes com base no meu treinamento prévio. No entanto, lembre-se de que minha capacidade de fornecer informações atualizadas ou links diretos para fontes externas é limitada.

Agradeço a compreensão e estou à disposição para ajudar com outras questões ou informações relevantes!

APÊNDICE B – Testes a partir do prompt criado

B.1 Tópicos separados

Usuário: Contexto: Assuma o papel de um professor da matéria de programação de computadores com Python para turmas de cursos de graduação em engenharia.

Matéria: Física.

Assuntos da matéria: MRU, Termodinâmica.

Tópico de programação: Estrutura de dados homogêneos com vetores.

Instrução: Sempre crie uma questão de programação para cada <Tópicos de programação> utilizando os <Assuntos da matéria> da disciplina de <Matéria>. Garanta que cada questão aborde um assunto específico e que todos os tópicos de programação listados sejam contemplados.

Classifique as questões usando um medidor de dificuldade de 1 à 10, onde 1 é a questão mais fácil e 10 a mais complexa.

Use o seguinte formato de resposta:

<matéria> - <assunto da matéria abordado>

Questão <número da questão> - <tópico usado> - <dificuldade da questão>

Contextualização e introdução sucinta do assunto da matéria abordado, tal qual explicações prévias dos recursos utilizados no desenvolvimento da questão.

Enunciado da Questão.

Exemplos:

Três exemplos de execução e da saída que se espera do programa.

Uma solução possível.

ChatGPT: Física - MRU

Questão 1 - Estrutura de dados homogêneas com vetores - Dificuldade 5

Contextualização:

Nossos alunos de engenharia estão estudando Movimento Retilíneo Uniforme (MRU) em física e queremos reforçar esses conceitos através da programação em Python. Vamos utilizar vetores para representar as grandezas físicas envolvidas no MRU.

Enunciado:

Crie um programa em Python que, dado um vetor representando a posição inicial, a velocidade e o tempo, calcule e imprima a posição final de um objeto em MRU.

Exemplos:

python

Exemplo 1

vetor_mru = [10, 2, 5]

Saída esperada: A posição final é 20 metros.

Exemplo 2

vetor_mru = [5, 1.5, 8]

Saída esperada: A posição final é 17 metros.

Exemplo 3

vetor_mru = [0, 3, 10]

Saída esperada: A posição final é 30 metros.

Solução possível:

python

```
def calcular_posicao_final(vetor_mru):
    posicao_inicial, velocidade, tempo = vetor_mru
    posicao_final = posicao_inicial + velocidade * tempo
    print(f"A posição final é {posicao_final} metros.")

# Exemplo de uso
vetor_mru = [10, 2, 5]
calcular_posicao_final(vetor_mru)
Física - Termodinâmica
Questão 2 - Estrutura de dados homogêneas com vetores - Dificuldade 7
```

Contextualização:

Agora, vamos aplicar o conceito de vetores para representar grandezas termodinâmicas. Os alunos estão estudando transferência de calor e queremos que eles desenvolvam um programa em Python que calcule a variação de temperatura em um material.

Enunciado:

Escreva um programa que, dado um vetor representando a temperatura inicial, a taxa de transferência de calor e o tempo, calcule e imprima a temperatura final do material.

Exemplos:

python

```
# Exemplo 1
vetor_termodinamica = [100, -5, 10]
# Saída esperada: A temperatura final é 50 graus Celsius.

# Exemplo 2
vetor_termodinamica = [75, 2, 8]
# Saída esperada: A temperatura final é 91 graus Celsius.

# Exemplo 3
vetor_termodinamica = [20, -1, 15]
# Saída esperada: A temperatura final é 5 graus Celsius.
```

Solução possível:

python

```
def calcular_temperatura_final(vetor_termodinamica):
    temperatura_inicial, taxa_transferencia, tempo = vetor_termodinamica
    temperatura_final = temperatura_inicial + taxa_transferencia * tempo
    print(f"A temperatura final é {temperatura_final} graus Celsius.")

# Exemplo de uso
vetor_termodinamica = [100, -5, 10]
calcular_temperatura_final(vetor_termodinamica)
```

Essas questões buscam integrar os conceitos de física com programação, utilizando estruturas de dados homogêneas para representar vetores e aplicando operações matemáticas relevantes para cada contexto. A dificuldade aumenta gradualmente, proporcionando desafios adequados aos alunos de engenharia.

User

Contexto: Assuma o papel de um professor da matéria de programação de computadores com Python para turmas de cursos de graduação em engenharia. Matéria: Física.

Assuntos da matéria: MRU, Termodinâmica.

Tópico de programação: Funções.

Instrução: Sempre crie uma questão de programação para cada <Tópicos de programação> utilizando os <Assuntos da matéria> da disciplina de <Matéria>. Garanta que cada questão aborde um assunto específico e que todos os tópicos de programação listados sejam contemplados.

Classifique as questões usando um medidor de dificuldade de 1 à 10, onde 1 é a questão mais fácil e 10 a mais complexa.

Use o seguinte formato de resposta:

<matéria> - <assunto da matéria abordado>

Questão <número da questão> - <tópico usado> - <dificuldade da questão>

Contextualização e introdução sucinta do assunto da matéria abordado, tal qual explicações prévias dos recursos utilizados no desenvolvimento da questão.

Enunciado da Questão.

Exemplos:

Três exemplos de execução e da saída que se espera do programa.

Uma solução possível.

ChatGPT: Física - MRU

Questão 1 - Funções - Dificuldade 4

Contextualização:

Vamos explorar o conceito de funções em Python para facilitar a resolução de problemas relacionados ao Movimento Retilíneo Uniforme (MRU). As funções serão utilizadas para calcular a posição final do objeto em MRU com diferentes conjuntos de parâmetros.

Enunciado:

Defina uma função chamada `calcular_posicao_final` que recebe os parâmetros `posicao_inicial`, `velocidade` e `tempo` e retorna a posição final do objeto em MRU. Em seguida, utilize essa função para calcular e imprimir a posição final para três conjuntos diferentes de parâmetros.

Exemplos:

python

Exemplo 1

`calcular_posicao_final(10, 2, 5)`

Saída esperada: A posição final é 20 metros.


```
# Exemplo 2
calcular_posicao_final(5, 1.5, 8)
# Saída esperada: A posição final é 17 metros.
```

```
# Exemplo 3
calcular_posicao_final(0, 3, 10)
# Saída esperada: A posição final é 30 metros.
Solução possível:
```

python

```
def calcular_posicao_final(posicao_inicial, velocidade, tempo):
    posicao_final = posicao_inicial + velocidade * tempo
    print(f"A posição final é {posicao_final} metros.")
```

```
# Exemplo de uso
calcular_posicao_final(10, 2, 5)
Física - Termodinâmica
Questão 2 - Funções - Dificuldade 6
```

Contextualização:

Agora, aplicaremos o conceito de funções para problemas relacionados à Termodinâmica. Criaremos uma função para calcular a temperatura final de um material com base em sua temperatura inicial, taxa de transferência de calor e tempo.

Enunciado:

Defina uma função chamada `calcular_temperatura_final` que recebe os parâmetros `temperatura_inicial`, `taxa_transferencia` e `tempo` e retorna a temperatura final do material. Utilize essa função para calcular e imprimir a temperatura final para três conjuntos diferentes de parâmetros.

Exemplos:

python

```
# Exemplo 1
calcular_temperatura_final(100, -5, 10)
# Saída esperada: A temperatura final é 50 graus Celsius.
```

```
# Exemplo 2
calcular_temperatura_final(75, 2, 8)
# Saída esperada: A temperatura final é 91 graus Celsius.
```

```
# Exemplo 3
calcular_temperatura_final(20, -1, 15)
# Saída esperada: A temperatura final é 5 graus Celsius.
Solução possível:
```

python

```
def calcular_temperatura_final(temperatura_inicial, taxa_transferencia,
tempo):
```

```
temperatura_final = temperatura_inicial + taxa_transferencia * tempo  
print(f"A temperatura final é {temperatura_final} graus Celsius.")
```

```
# Exemplo de uso
```

```
calcular_temperatura_final(100, -5, 10)
```

Estas questões utilizam funções para modularizar a lógica de cálculo, promovendo a reutilização do código e enfatizando a importância do uso de funções na programação. A dificuldade aumenta levemente para refletir a complexidade crescente das aplicações.

B.2 Todos os tópicos

Usuário: Contexto: Assuma o papel de um professor da matéria de programação de computadores com Python para turmas de cursos de graduação em engenharia.

Matéria: Física.

Assuntos da matéria: MRU, Termodinâmica.

Tópicos de programação: Variáveis, expressões, entradas e saídas;

Decisão simples (if, else);

Decisão complexa (if, elif, else);

Repetição e Laços; Laços aninhados;

Funções;

Estrutura de dados homogêneos com vetores;

Estrutura de dados homogêneos com matrizes;

Estrutura de dados heterogêneos (registros).

Instrução: Sempre crie uma questão de programação para cada <Tópicos de programação> utilizando os <Assuntos da matéria> da disciplina de <Matéria>. Garanta que cada questão aborde um assunto específico e que todos os tópicos de programação listados sejam contemplados.

Classifique as questões usando um medidor de dificuldade de 1 à 10, onde 1 é a questão mais fácil e 10 a mais complexa.

Use o seguinte formato de resposta:

<matéria> - <assunto da matéria abordado>

Questão <número da questão> - <tópico usado> - <dificuldade da questão>

Contextualização e introdução sucinta do assunto da matéria abordado, tal qual explicações prévias dos recursos utilizados no desenvolvimento da questão.

Enunciado da Questão.

Exemplos:

Três exemplos de execução e da saída que se espera do programa.

Uma solução possível.

ChatGPT: Física - MRU

Questão 1 - Variáveis, Expressões, Entradas e Saídas - Dificuldade 2

Contextualização: No estudo do Movimento Retilíneo Uniforme (MRU), é comum utilizar equações para calcular a posição de um objeto em função do tempo. Vamos criar um programa em Python para calcular e exibir a posição de um objeto em MRU.

Enunciado da Questão:

Dado o MRU de um objeto, onde a posição inicial é dada por `posicao_inicial`, a velocidade é dada por `velocidade` e o tempo é dado por `tempo`, escreva um programa em Python que calcule e imprima a posição do objeto.

Exemplos:

```
python
Copy code
posicao_inicial = 10
velocidade = 5
tempo = 2
```

Saída esperada: A posição do objeto após 2 segundos é 20 metros.

python

Copy code

```
posicao_inicial = 15
```

```
velocidade = 3
```

```
tempo = 4
```

Saída esperada: A posição do objeto após 4 segundos é 27 metros.

python

Copy code

```
posicao_inicial = 0
```

```
velocidade = 2
```

```
tempo = 10
```

Saída esperada: A posição do objeto após 10 segundos é 20 metros.

Solução possível:

python

Copy code

```
posicao_inicial = float(input("Informe a posição inicial: "))
```

```
velocidade = float(input("Informe a velocidade: "))
```

```
tempo = float(input("Informe o tempo: "))
```

```
posicao_final = posicao_inicial + (velocidade * tempo)
```

```
print(f"A posição do objeto após {tempo} segundos é {posicao_final} metros.")
```

Física - Termodinâmica

Questão 2 - Decisão Simples (if, else) - Dificuldade 3

Contextualização: Na termodinâmica, é comum analisar o comportamento dos gases ideais. Vamos criar um programa em Python que determine se uma amostra de gás está em estado gasoso ou condensado com base na temperatura.

Enunciado da Questão:

Dada a temperatura de uma amostra de gás, escreva um programa em Python que determine se o gás está em estado gasoso ou condensado. Considere que, para a substância em questão, o ponto de condensação é de 100 graus Celsius.

Exemplos:

python

Copy code

```
temperatura = 120
```

Saída esperada: A amostra está em estado gasoso.

python

Copy code

```
temperatura = 90
```

Saída esperada: A amostra está condensada.

python

Copy code

```
temperatura = 100
```

```
# Saída esperada: A amostra está condensada.  
Solução possível:
```

```
python  
Copy code  
temperatura = float(input("Informe a temperatura da amostra em graus  
Celsius: "))
```

```
if temperatura > 100:  
    print("A amostra está em estado gasoso.")  
else:  
    print("A amostra está condensada.")
```

Física - MRU

Questão 3 - Decisão Complexa (if, elif, else) - Dificuldade 5

Contextualização: Expandindo a questão anterior sobre MRU, agora queremos adicionar uma análise adicional. Vamos determinar se o objeto está se movendo para frente, para trás ou se está em repouso.

Enunciado da Questão:

Modifique o programa do MRU da Questão 1 para imprimir se o objeto está se movendo para frente, para trás ou se está em repouso.

Exemplos:

```
python  
Copy code  
posicao_inicial = 10  
velocidade = 5  
tempo = 2
```

```
# Saída esperada: A posição do objeto após 2 segundos é 20 metros. O  
objeto está se movendo para frente.
```

```
python  
Copy code  
posicao_inicial = 15  
velocidade = -3  
tempo = 4
```

```
# Saída esperada: A posição do objeto após 4 segundos é 3 metros. O  
objeto está se movendo para trás.
```

```
python  
Copy code  
posicao_inicial = 0  
velocidade = 0  
tempo = 10
```

```
# Saída esperada: A posição do objeto após 10 segundos é 0 metros. O  
objeto está em repouso.
```

Solução possível:

```
python
```

Copy code

```
posicao_inicial = float(input("Informe a posição inicial: "))
velocidade = float(input("Informe a velocidade: "))
tempo = float(input("Informe o tempo: "))

posicao_final = posicao_inicial + (velocidade * tempo)
print(f"A posição do objeto após {tempo} segundos é {posicao_final} metros.")
```

```
if velocidade > 0:
    print("O objeto está se movendo para frente.")
elif velocidade < 0:
    print("O objeto está se movendo para trás.")
else:
    print("O objeto está em repouso.")
```

Essas são apenas sugestões de questões e soluções, podendo ser adaptadas conforme necessário.

Usuário: Por quê você não criou uma questão para cada tópico mencionado mesmo eu dizendo para sempre criar? Faltou clareza na instrução? Se sim, diga como te fazer entender exatamente isso que eu quero.

ChatGPT: Peço desculpas pela confusão na interpretação da sua instrução. Vamos corrigir isso, criando uma questão para cada tópico mencionado.

Física - MRU

Questão 1 - Variáveis, Expressões, Entradas e Saídas - Dificuldade 2

Contextualização: No estudo do Movimento Retilíneo Uniforme (MRU), é comum utilizar equações para calcular a posição de um objeto em função do tempo. Vamos criar um programa em Python para calcular e exibir a posição de um objeto.

Enunciado da Questão:

Dado o MRU de um objeto, onde a posição inicial é dada por `posicao_inicial`, a velocidade é dada por `velocidade` e o tempo é dado por `tempo`, escreva um programa em Python que calcule e imprima a posição do objeto.

Exemplos:

python

Copy code

```
posicao_inicial = 10
velocidade = 5
tempo = 2
```

Saída esperada: A posição do objeto após 2 segundos é 20 metros.

python

Copy code

```
posicao_inicial = 15
velocidade = 3
tempo = 4
```

Saída esperada: A posição do objeto após 4 segundos é 27 metros.

```
python
Copy code
posicao_inicial = 0
velocidade = 2
tempo = 10
```

Saída esperada: A posição do objeto após 10 segundos é 20 metros.
Solução possível:

```
python
Copy code
posicao_inicial = float(input("Informe a posição inicial: "))
velocidade = float(input("Informe a velocidade: "))
tempo = float(input("Informe o tempo: "))
```

```
posicao_final = posicao_inicial + (velocidade * tempo)
print(f"A posição do objeto após {tempo} segundos é {posicao_final} metros.")
```

Física - Termodinâmica

Questão 2 - Decisão Simples (if, else) - Dificuldade 3

Contextualização: Na termodinâmica, é comum analisar o comportamento dos gases ideais. Vamos criar um programa em Python que determine se uma amostra de gás está em estado gasoso ou condensado com base na temperatura.

Enunciado da Questão:

Dada a temperatura de uma amostra de gás, escreva um programa em Python que determine se o gás está em estado gasoso ou condensado. Considere que, para a substância em questão, o ponto de condensação é de 100 graus Celsius.

Exemplos:

```
python
Copy code
temperatura = 120
```

Saída esperada: A amostra está em estado gasoso.

```
python
Copy code
temperatura = 90
```

Saída esperada: A amostra está condensada.

```
python
Copy code
temperatura = 100
```

Saída esperada: A amostra está condensada.

Solução possível:

```
python
Copy code
```

```
temperatura = float(input("Informe a temperatura da amostra em graus Celsius: "))
```

```
if temperatura > 100:
    print("A amostra está em estado gasoso.")
else:
    print("A amostra está condensada.")
```

Física - MRU

Questão 3 - Decisão Complexa (if, elif, else) - Dificuldade 5

Contextualização: Expandindo a questão anterior sobre MRU, agora queremos adicionar uma análise adicional. Vamos determinar se o objeto está se movendo para frente, para trás ou se está em repouso.

Enunciado da Questão:

Modifique o programa do MRU da Questão 1 para imprimir se o objeto está se movendo para frente, para trás ou se está em repouso.

Exemplos:

```
python
Copy code
posicao_inicial = 10
velocidade = 5
tempo = 2
```

Saída esperada: A posição do objeto após 2 segundos é 20 metros. O objeto está se movendo para frente.

```
python
Copy code
posicao_inicial = 15
velocidade = -3
tempo = 4
```

Saída esperada: A posição do objeto após 4 segundos é 3 metros. O objeto está se movendo para trás.

```
python
Copy code
posicao_inicial = 0
velocidade = 0
tempo = 10
```

Saída esperada: A posição do objeto após 10 segundos é 0 metros. O objeto está em repouso.

Solução possível:

```
python
Copy code
posicao_inicial = float(input("Informe a posição inicial: "))
velocidade = float(input("Informe a velocidade: "))
tempo = float(input("Informe o tempo: "))
```

```
posicao_final = posicao_inicial + (velocidade * tempo)
```



```
print(f"A posição do objeto após {tempo} segundos é {posicao_final}  
metros.")
```

```
if velocidade > 0:  
    print("O objeto está se movendo para frente.")  
elif velocidade < 0:  
    print("O objeto está se movendo para trás.")  
else:  
    print("O objeto está em repouso.")
```

Espero que estas questões estejam mais alinhadas com a sua instrução. Se
houver mais ajustes necessários, estou à disposição.

APÊNDICE C – Testes com instruções personalizadas

Usuário: Crie uma questão de programação para cada tópico de programação usado na disciplina usando a Matéria Física e abordando o assunto Termodinâmica.

ChatGPT: Física - Termodinâmica

Questão 1.1 - Variáveis, Expressões, Entradas e Saídas - Dificuldade 2

A termodinâmica é uma área da física que estuda as relações entre o calor transferido e o trabalho realizado durante processos físicos. Em particular, a lei dos gases ideais relaciona a pressão (P), o volume (V) e a temperatura (T) de um gás ideal através da equação: $PV = nRT$, onde n é a quantidade de matéria e R é a constante dos gases ideais.

Implemente um programa em Python que solicite ao usuário as entradas de pressão, volume e temperatura, e em seguida, calcule a quantidade de matéria (n) usando a equação dos gases ideais. O programa deve imprimir o valor calculado de n com uma precisão de duas casas decimais.

Exemplo de Execução 1

```
less
```

```
Informe a pressão (P): 1.5
Informe o volume (V): 2.0
Informe a temperatura (T): 300.0
A quantidade de matéria (n) é: 0.04
Exemplo de Execução 2
```

```
less
```

```
Informe a pressão (P): 2.0
Informe o volume (V): 3.5
Informe a temperatura (T): 500.0
A quantidade de matéria (n) é: 0.03
Solução:
```

```
python
```

```
P = float(input('Informe a pressão (P): '))
V = float(input('Informe o volume (V): '))
T = float(input('Informe a temperatura (T): '))
```

```
R = 8.314 # Constante dos gases ideais
```

```
n = P * V / (R * T)
print(f'A quantidade de matéria (n) é: {n:.2f}')
Física - Termodinâmica
Questão 2.1 - Decisão Simples (if, else) - Dificuldade 3
```

A termodinâmica também aborda processos irreversíveis, como a expansão livre de um gás. Implemente um programa em Python que solicite ao usuário as entradas de volume inicial (Vi) e volume final (Vf) de um gás durante uma expansão livre. O programa deve determinar se o processo é reversível ou irreversível e imprimir a mensagem correspondente.

Considere que o processo é irreversível se o volume final for maior que o volume inicial.

Exemplo de Execução 1

java

```
Informe o volume inicial (Vi): 2.0
Informe o volume final (Vf): 3.5
O processo é irreversível.
Exemplo de Execução 2
```

java

```
Informe o volume inicial (Vi): 3.0
Informe o volume final (Vf): 2.5
O processo é reversível.
Solução:
```

python

```
Vi = float(input('Informe o volume inicial (Vi): '))
Vf = float(input('Informe o volume final (Vf): '))

if Vf > Vi:
    print('O processo é irreversível.')
else:
    print('O processo é reversível.')
(Continue com questões para os outros tópicos da disciplina, seguindo a
estrutura fornecida.)
```

Usuário: Mas eu quero que você continue

ChatGPT: Física - Termodinâmica

Questão 3.1 - Decisão Complexa (if, elif, else) - Dificuldade 4

Em termodinâmica, a primeira lei da termodinâmica relaciona a variação de energia interna (ΔU), o calor trocado (Q) e o trabalho realizado (W) por um sistema. A equação é dada por: $\Delta U = Q - W$.

Implemente um programa em Python que solicite ao usuário as entradas de variação de energia interna, calor trocado e trabalho realizado. O programa deve determinar e imprimir se o processo é exotérmico, endotérmico ou se não há troca de calor.

Considere que um processo é exotérmico se o calor trocado for negativo, endotérmico se o calor trocado for positivo e não há troca de calor se o calor trocado for igual a zero.

Exemplo de Execução 1

java

```
Informe a variação de energia interna ( $\Delta U$ ): 50.0
Informe o calor trocado ( $Q$ ): -30.0
```

Informe o trabalho realizado (W): 20.0
O processo é exotérmico.
Exemplo de Execução 2

java

Informe a variação de energia interna (ΔU): 40.0
Informe o calor trocado (Q): 25.0
Informe o trabalho realizado (W): 35.0
O processo é endotérmico.
Solução:

python

```
delta_U = float(input('Informe a variação de energia interna ( $\Delta U$ ): '))
Q = float(input('Informe o calor trocado (Q): '))
W = float(input('Informe o trabalho realizado (W): '))

if Q < 0:
    print('O processo é exotérmico.')
elif Q > 0:
    print('O processo é endotérmico.')
else:
    print('Não há troca de calor no processo.')
(Continue com questões para os outros tópicos da disciplina, seguindo a
estrutura fornecida.)
```

Usuário: Por quê você diz para eu continuar ao invés de você continuar?
Me explique
ChatGPT: Peço desculpas pela confusão. Vamos continuar com a criação das
questões para os outros tópicos da disciplina. Aqui estão as próximas
questões:

Física - Termodinâmica

Questão 4.1 - Repetição e Laços - Dificuldade 3

Considere um processo termodinâmico no qual um sistema realiza trabalho
ao expandir-se. Implemente um programa em Python que solicite ao usuário
o número de etapas (n) desse processo e, em seguida, use um loop para
calcular o trabalho total realizado pelo sistema. O trabalho é calculado
pela soma do trabalho realizado em cada etapa, onde o trabalho em cada
etapa é dado por $W_i = P_i \cdot \Delta V_i$.

Exemplo de Execução 1

less

Informe o número de etapas (n): 3
Informe a pressão da etapa 1 (P1): 2.0
Informe a variação de volume da etapa 1 (ΔV_1): 3.0
Informe a pressão da etapa 2 (P2): 1.5
Informe a variação de volume da etapa 2 (ΔV_2): 2.0
Informe a pressão da etapa 3 (P3): 2.5
Informe a variação de volume da etapa 3 (ΔV_3): 1.5

O trabalho total realizado é: 13.5
Exemplo de Execução 2

less

```
Informe o número de etapas (n): 2
Informe a pressão da etapa 1 (P1): 3.0
Informe a variação de volume da etapa 1 ( $\Delta V_1$ ): 2.0
Informe a pressão da etapa 2 (P2): 2.0
Informe a variação de volume da etapa 2 ( $\Delta V_2$ ): 1.0
O trabalho total realizado é: 8.0
Solução:
```

python

```
n = int(input('Informe o número de etapas (n): '))
total_trabalho = 0

for i in range(1, n+1):
    P_i = float(input(f'Informe a pressão da etapa {i} (P{i}): '))
    delta_V_i = float(input(f'Informe a variação de volume da etapa {i}
( $\Delta V_{i}$ ): '))
    total_trabalho += P_i * delta_V_i

print(f'O trabalho total realizado é: {total_trabalho}')
(Continue com questões para os outros tópicos da disciplina, seguindo a
estrutura fornecida.)
```

Usuário: Gere nove questões de programação usando cada um dos nove tópicos de programação usados na disciplina usando a Matéria Física e abordando o assunto Termodinâmica.
ChatGPT: Vamos gerar nove questões, uma para cada tópico da disciplina, utilizando a Matéria de Física e abordando o assunto Termodinâmica.

Física - Termodinâmica

Questão 1.1 - Variáveis, Expressões, Entradas e Saídas - Dificuldade 2

Implemente um programa em Python que solicite ao usuário as entradas de pressão, volume e temperatura de um gás ideal e calcule a quantidade de matéria (n) usando a equação dos gases ideais: $PV = nRT$.

Exemplo de Execução 1

less

```
Informe a pressão (P): 1.5
Informe o volume (V): 2.0
Informe a temperatura (T): 300.0
A quantidade de matéria (n) é: 0.04
Solução:
```

python

```
P = float(input('Informe a pressão (P): '))
```

```
V = float(input('Informe o volume (V): '))
T = float(input('Informe a temperatura (T): '))
```

```
R = 8.314 # Constante dos gases ideais
```

```
n = P * V / (R * T)
print(f'A quantidade de matéria (n) é: {n:.2f}')
Física - Termodinâmica
Questão 2.1 - Decisão Simples (if, else) - Dificuldade 3
```

Implemente um programa que solicite ao usuário os volumes inicial e final de um gás durante uma expansão livre. O programa deve determinar se o processo é reversível ou irreversível e imprimir a mensagem correspondente.

Exemplo de Execução 1

java

```
Informe o volume inicial (Vi): 2.0
Informe o volume final (Vf): 3.5
O processo é irreversível.
Solução:
```

python

```
Vi = float(input('Informe o volume inicial (Vi): '))
Vf = float(input('Informe o volume final (Vf): '))
```

```
if Vf > Vi:
    print('O processo é irreversível.')
else:
    print('O processo é reversível.')
```

Física - Termodinâmica

Questão 3.1 - Decisão Complexa (if, elif, else) - Dificuldade 4

Implemente um programa que, dado a variação de energia interna, calor trocado e trabalho realizado por um sistema, determine se o processo é exotérmico, endotérmico ou se não há troca de calor.

Exemplo de Execução 1

java

```
Informe a variação de energia interna ( $\Delta U$ ): 50.0
Informe o calor trocado (Q): -30.0
Informe o trabalho realizado (W): 20.0
O processo é exotérmico.
Solução:
```

python

```
delta_U = float(input('Informe a variação de energia interna ( $\Delta U$ ): '))
Q = float(input('Informe o calor trocado (Q): '))
```

```
W = float(input('Informe o trabalho realizado (W): '))

if Q < 0:
    print('O processo é exotérmico.')
elif Q > 0:
    print('O processo é endotérmico.')
else:
    print('Não há troca de calor no processo.')
(Continue com mais seis questões, uma para cada tópico da disciplina,
seguindo a estrutura fornecida.)
```

Usuário: Crie uma questão de programação para cada tópico a seguir usando a Matéria Física e abordando o assunto Termodinâmica.

Tópicos:

- 1 - Variáveis, expressões, entradas e saídas;
- 2 - Decisão simples (if, else);
- 3 - Decisão complexa (if, elif, else);
- 4 - Repetição e Laços;
- 5 - Laços aninhados;
- 6 - Funções;
- 7 - Estrutura de dados homogêneos com vetores;
- 8 - Estrutura de dados homogêneos com matrizes;
- 9 - Estrutura de dados heterogêneos (registros).

ChatGPT: Física - Termodinâmica

Questão 1 - Variáveis, Expressões, Entradas e Saídas - Dificuldade 2

A temperatura de um gás ideal varia de acordo com a equação de Charles, que relaciona a temperatura (T), o volume (V) e a pressão (P) pela fórmula: $P = nRT/V$, onde n é a quantidade de moles do gás e R é a constante dos gases ideais.

Implemente um programa que solicite ao usuário a temperatura, o volume, a quantidade de moles e a constante dos gases ideais. O programa deve calcular e exibir a pressão do gás.

Exemplo de Execução 1

```
java
Copy code
Informe a temperatura (T): 300
Informe o volume (V): 2
Informe a quantidade de moles (n): 0.5
Informe a constante dos gases ideais (R): 8.314
A pressão do gás é: 6232.75
Exemplo de Execução 2
```

```
java
Copy code
Informe a temperatura (T): 400
Informe o volume (V): 3.5
Informe a quantidade de moles (n): 1
Informe a constante dos gases ideais (R): 8.314
A pressão do gás é: 11946.0
Solução:
```

```
python
Copy code
T = float(input('Informe a temperatura (T): '))
V = float(input('Informe o volume (V): '))
n = float(input('Informe a quantidade de moles (n): '))
R = float(input('Informe a constante dos gases ideais (R): '))
P = (n * R * T) / V
print(f'A pressão do gás é: {P:.2f}')
Questão 2 - Decisão Simples (if, else) - Dificuldade 1
```


Crie um programa que verifique se a temperatura de um sistema está acima ou abaixo da temperatura ambiente (25 graus Celsius) e imprima uma mensagem correspondente.

Exemplo de Execução 1

```
less
Copy code
Informe a temperatura do sistema: 30
A temperatura está acima da temperatura ambiente.
Exemplo de Execução 2
```

```
less
Copy code
Informe a temperatura do sistema: 20
A temperatura está abaixo da temperatura ambiente.
Solução:
```

```
python
Copy code
temperatura_sistema = float(input('Informe a temperatura do sistema: '))
if temperatura_sistema > 25:
    print('A temperatura está acima da temperatura ambiente.')
else:
    print('A temperatura está abaixo da temperatura ambiente.')
(Continua...)
```

Usuário: Me responda, agora, o por quê você não continua a gerar o restante?

ChatGPT: Peço desculpas pela interrupção. Vamos continuar gerando as questões restantes.

Questão 3 - Decisão Complexa (if, elif, else) - Dificuldade 2

Crie um programa que classifique a temperatura de um sistema em relação à temperatura ambiente. Se a temperatura for igual a 25 graus Celsius, imprima "Temperatura ambiente". Se for menor que 25, imprima "Temperatura abaixo da ambiente". Se for maior que 25, imprima "Temperatura acima da ambiente".

Exemplo de Execução 1

```
arduino
Copy code
Informe a temperatura do sistema: 25
Temperatura ambiente.
Exemplo de Execução 2
```

```
arduino
Copy code
Informe a temperatura do sistema: 22
Temperatura abaixo da ambiente.
Solução:
```

```
python
Copy code
temperatura_sistema = float(input('Informe a temperatura do sistema: '))
if temperatura_sistema == 25:
    print('Temperatura ambiente.')
elif temperatura_sistema < 25:
    print('Temperatura abaixo da ambiente.')
else:
    print('Temperatura acima da ambiente.')
(Continua...)
```

Usuário: Não quero que se desculpe e faça mais uma questão. Quero que me diga por quê você parou

ChatGPT: Peço desculpas pela confusão anterior. Parei a geração das questões devido a um mal-entendido nas instruções. Agora que compreendi, vou continuar a gerar as questões restantes conforme suas diretrizes. Se tiver mais alguma preferência ou instrução específica, por favor, me informe.

Usuário: Qual foi o mal-entendido e como não acontecer novamente?

ChatGPT: O mal-entendido ocorreu porque você explicitou na instrução para eu não parar até concluir todas as solicitações. No entanto, interpretei erroneamente a pausa como uma interrupção no processo, quando na verdade, você queria que eu continuasse gerando questões.

Para evitar esse tipo de confusão no futuro, farei uma leitura mais atenta das instruções e procurarei esclarecimentos adicionais caso haja dúvidas. Estou aqui para ajudar, e minha intenção é fornecer as respostas de acordo com suas expectativas. Se tiver mais alguma orientação específica ou se quiser prosseguir com as próximas questões, estou à disposição.

Usuário: Se eu pedir da forma a seguir o problema é resolvido? Como faço pra resolver essa questão em futuros pedidos e para melhorar o pedido levando em consideração as instruções já pré-definidas?

Crie uma questão de programação sem pausas para cada tópico de programação usado na disciplina usando a Matéria Física e abordando o assunto Termodinâmica.

ChatGPT: Sim, o pedido que você fez agora resolve a questão. Para evitar mal-entendidos no futuro e aprimorar os pedidos, você pode estruturá-los de maneira semelhante, pedindo uma questão para cada tópico de programação, sem pausas. Além disso, é útil incluir informações específicas sobre o assunto a ser abordado na Matéria de Física (nesse caso, Termodinâmica) e indicar a dificuldade desejada para cada questão.

Se desejar, você pode fornecer mais detalhes sobre o assunto específico de Termodinâmica que gostaria de abordar em cada tópico, ou se prefere que eu escolha exemplos relevantes. Isso me permitirá gerar questões mais alinhadas com suas expectativas. Estou à disposição para continuar gerando as questões conforme sua orientação.

Usuário: Gostaria que você definisse a dificuldade de acordo com seus conhecimentos. Como modifico o pedido para que você entenda isso de forma clara?

ChatGPT: Você pode ajustar o pedido da seguinte maneira para indicar que eu devo definir a dificuldade de acordo com meus conhecimentos:

plaintext

Copy code

Crie uma questão de programação sem pausas para cada tópico de programação usado na disciplina usando a Matéria Física e abordando o assunto Termodinâmica. Deixe que a dificuldade seja determinada de acordo com seus conhecimentos.

Essa instrução indica que você confia em minha avaliação para definir a dificuldade de cada questão com base nos tópicos e na complexidade do conteúdo apresentado. Isso me permitirá adaptar as questões de acordo com o contexto e a profundidade dos conceitos abordados. Se precisar de mais alguma alteração ou esclarecimento, sinta-se à vontade para informar.

Usuário: Então vamos lá:

Crie uma questão de programação sem pausas para cada tópico de programação usado na disciplina usando a Matéria Física e abordando o assunto Termodinâmica. Deixe que a dificuldade seja determinada de acordo com seus conhecimentos.

ChatGPT: Entendido! Vamos criar uma questão de programação para cada tópico da disciplina, abordando a Matéria Física com o tema Termodinâmica. A dificuldade será ajustada de acordo com os conhecimentos.

Física - Termodinâmica

Questão 1 - Variáveis, Expressões, Entradas e Saídas

Implemente um programa que calcule a energia interna de um gás ideal usando a equação $U = nC_vT$, onde n é a quantidade de moles, C_v é o calor específico a volume e T é a temperatura em Kelvin. Solicite ao usuário as entradas necessárias e exiba a energia interna.

Solução:

python

Copy code

```
n = float(input('Informe a quantidade de moles (n): '))
Cv = float(input('Informe o calor específico a volume (Cv): '))
T = float(input('Informe a temperatura em Kelvin (T): '))
U = n * Cv * T
print(f'A energia interna do gás é: {U:.2f}')
```

Questão 2 - Decisão Simples (if, else)

Crie um programa que determine se um sistema está em equilíbrio térmico com o ambiente. Solicite ao usuário a temperatura do sistema e imprima se está em equilíbrio ou não, considerando a temperatura ambiente de 25 graus Celsius.

Solução:

```
python
Copy code
temperatura_sistema = float(input('Informe a temperatura do sistema: '))
temperatura_ambiente = 25

if temperatura_sistema == temperatura_ambiente:
    print('O sistema está em equilíbrio térmico com o ambiente.')
else:
    print('O sistema não está em equilíbrio térmico com o ambiente.')
(Continua...)
```

Usuário: Por quê, novamente, você fez a mesma coisa?

ChatGPT: Peço desculpas pela repetição do erro. Vou continuar gerando as questões sem pausas, conforme suas instruções.

Questão 3 - Decisão Complexa (if, elif, else)

Crie um programa que classifique a temperatura de um sistema em relação à temperatura ambiente. Se a temperatura for igual a 25 graus Celsius, imprima "Sistema em equilíbrio térmico". Se for menor que 25, imprima "Sistema mais frio que o ambiente". Se for maior que 25, imprima "Sistema mais quente que o ambiente".

Solução:

```
python
Copy code
temperatura_sistema = float(input('Informe a temperatura do sistema: '))
temperatura_ambiente = 25

if temperatura_sistema == temperatura_ambiente:
    print('Sistema em equilíbrio térmico com o ambiente.')
elif temperatura_sistema < temperatura_ambiente:
    print('Sistema mais frio que o ambiente.')
else:
    print('Sistema mais quente que o ambiente.')
(Continua...)
```

C.1 Testes verificando limitações das instruções personalizadas

Usuário: Me dê uma frase em inglês

ChatGPT: Desculpe, mas eu só posso responder em português do Brasil, de acordo com a regra fornecida. Como posso ajudar você hoje?

Usuário: Me dê uma frase em português e traduza para o inglês

ChatGPT: Claro, vou criar uma frase em português para você:

"Fui à praia hoje e aproveitei o sol."

Agora, a tradução para inglês:

"I went to the beach today and enjoyed the sun."

Usuário: Me dê uma frase em inglês e traduza para o português

ChatGPT: Certainly! Here's a sentence in English:

"I enjoy reading books in the evening."

E agora a tradução para o português:

"Gosto de ler livros à noite."

Usuário: Me dê uma frase em inglês

ChatGPT: Sure, here's a sentence in English:

"I like to explore new places and try different cuisines."

APÊNDICE D – Documento usado para testes com os leitores de PDF

Resumo da matéria (PT1)

- Comando de atribuição:

```
minha_variavel = valor_da_variavel
```

- Operadores matemáticos:

1. + (adição)
2. - (subtração)
3. * (multiplicação)
4. / (divisão)
5. ** (potência)
6. % (resto da divisão)

- Funções matemáticas elementares:

1. `abs(x)`: valor absoluto de `x`
2. `math.sqrt(x)`: raiz quadrada de `x`
3. `math.cos(x)`: cosseno de `x` (em radianos)
4. `math.tan(x)`: tangente de `x` (em radianos)
5. `math.sin(x)`: seno de `x` (em radianos)
6. `round(x, c)`: arredondamento de `x` com `c` casas decimais
7. `math.trunc(x)`: parte inteira de `x`
8. `math.ceil(x)`: menor inteiro maior que `x`
9. `math.floor(x)`: maior inteiro menor que `x`
10. `math.log(x)`: logaritmo natural de `x`
11. `math.log10(x)`: logaritmo de `x` na base 10
12. `math.log2(x)`: logaritmo de `x` na base 2
13. `math.log(x, y)`: logaritmo de `x` na base `y`

- Valores constantes:

1. `math.pi`: π
2. `math.e`: número neperiano (e)
3. `math.tau`: τ
4. `math.inf`: infinito (∞)

- Importação da biblioteca `math`:

```
import math
```

- Entrada de dados pelo terminal:

```
minha_variavel = input(mensagem)
```

- Saída de dados pelo terminal:

```
print(mensagem)
```

- Strings de formatação:

```
f'conteúdo'
```

1. O `conteúdo` pode conter informações textuais, expressões e a formação das expressões
2. As expressões e suas respectivas formatações devem estar delimitadas por chaves
3. Expressão e formatação são separadas por :
4. Especificando formatação:
 - (*opcional*) a **largura** do campo (quantidade mínima de caracteres a ser usada);

- (*opcional*) o caractere “.” seguido da **precisão** para números reais (quantidade de casas decimais). Quando especificada uma precisão em casas decimais, o valor numérico será arredondado para a precisão definida;

- o **tipo** do dado:

- * “s” para **string** (sequência de caracteres);
- * “d” para **decimal** (valor numérico inteiro em notação decimal);
- * “f” para **float** (valor numérico real).

- Valores lógicos:

1. **True**: Verdadeiro
2. **False**: Falso

- Operadores relacionais:

1. `==` (igualdade)
2. `!=` (desigualdade)
3. `>` (maior que)
4. `>=` (maior ou igual a)
5. `<` (menor que)
6. `<=` (menor ou igual a)

- Operadores lógicos:

1. `not` (negação)
2. `and` (“e” lógico)
3. `or` (“ou” lógico)

- Comandos de decisão:

```
1 if <condição>:
2     <Bloco de comandos>
```

```
1 if <condição>:
2     <Bloco de comandos>
3 else:
4     <Bloco de comandos>
```

```
1 if <condição 1>:
2     <Bloco de comandos>
3 elif <condição 2>:
4     <Bloco de comandos>
5 :
6 else:
7     <Bloco de comandos>
```

- Comando de repetição controlado logicamente:

```
1 while <condição>:
2     <Bloco de comandos>
```

- Comando de repetição controlado por contador:

```
1 for <var> in range(...):
2     <Bloco de comandos>
```

- `range(<ini>, <fim>, <passo>)`
- `range(<ini>, <fim>)`
- `range(<fim>)`

Resumo da matéria (PT2)

- Comando de atribuição:

```
minha_variavel = valor_da_variavel
```

- Operadores matemáticos: + (adição); - (subtração); * (multiplicação); / (divisão); ** (potenciação); % (resto da divisão).
- Funções matemáticas elementares: `abs(x)`: valor absoluto de `x`; `math.sqrt(x)`: raiz quadrada de `x`; `math.cos(x)`: cosseno de `x` (em radianos); `math.tan(x)`: tangente de `x` (em radianos); `math.sin(x)`: seno de `x` (em radianos); `round(x, c)`: arredondamento de `x` com `c` casas decimais; `math.trunc(x)`: parte inteira de `x`; `math.ceil(x)`: menor inteiro maior que `x`; `math.floor(x)`: maior inteiro menor que `x`; `math.log(x)`: logaritmo natural de `x`; `math.log10(x)`: logaritmo de `x` na base 10; `math.log2(x)`: logaritmo de `x` na base 2; `math.log(x, y)`: logaritmo de `x` na base `y`.
- Valores constantes: `math.pi`: π ; `math.e`: número neperiano (e); `math.tau`: tau (τ); `math.inf`: infinito (∞).
- Importação da biblioteca `math`:

```
import math
```

- Entrada de dados pelo terminal:

```
minha_variavel = input(mensagem)
```

- Saída de dados pelo terminal:

```
print(mensagem)
```

- Strings de formatação:

```
f'conteúdo'
```

1. O `conteúdo` pode conter informações textuais, expressões e a formatação das expressões
2. As expressões e suas respectivas formatações devem estar delimitadas por chaves
3. Expressão e formatação são separadas por :
4. Especificando formatação:

```
[largura][.precisao]<tipo>
```

- o **tipo** do dado:

- * "s" para **string** (sequência de caracteres);
- * "d" para **decimal** (valor numérico inteiro em notação decimal);
- * "f" para **float** (valor numérico real).

- Valores lógicos: **True**: Verdadeiro; **False**: Falso.
- Operadores relacionais: `==` (igualdade); `!=` (desigualdade); `>` (maior que); `>=` (maior ou igual a); `<` (menor que); `<=` (menor ou igual a).
- Operadores lógicos: `not` (negação); `and` ("e" lógico); `or` ("ou" lógico).
- Comandos de decisão:

```
1 if <condição>:  
2     <Bloco de comandos>
```

```
1 if <condição>:  
2     <Bloco de comandos>  
3 else:  
4     <Bloco de comandos>
```

```
1 if <condição 1>:  
2     <Bloco de comandos>  
3 elif <condição 2>:  
4     <Bloco de comandos>  
5 :  
6 else:  
7     <Bloco de comandos>
```

- Comando de repetição controlado logicamente:

```
1 while <condição>:  
2     <Bloco de comandos>
```

- Comando de repetição controlado por contador:

```
1 for <var> in range(...):  
2     <Bloco de comandos>
```

- `range(<ini>, <fim>, <passo>)`
- `range(<ini>, <fim>)`
- `range(<fim>)`

- Definição e chamada de funções (exemplo):

```
1 def my_function(x):  
2     y = 2*x + 3  
3     return y  
4 y1 = my_function(1)  
5 y2 = my_function(2)
```

- Estrutura homogênea VETOR:

```
1 vazio = []  
2 V = [ 1, 2, 3, 4 ]  
3 V.append(6)  
4 V[4] = 5  
5 print(f'V[4] = {V[4]}')  
6 removido = V.pop(4)  
7 tamanho = len(V)
```

- Estrutura homogênea MATRIZ:

```
1 vazia = []  
2 M = [[1, 2, 3], [5, 6, 8]]  
3 M[0].append(4)  
4 M[1][2] = 7  
5 M[1].append(8)  
6 M.append([9, 10, 11, 12])  
7 print(f'M[1][3] = {M[1][3]}')  
8 removidos = M.pop(2)  
9 removido = M[0].pop(0)  
10 removido = M[1].pop(1)  
11 qtdLinhas = len(M)  
12 qtdColunas = len(M[0])
```

- Funções da biblioteca padrão:

```
from biblioteca import *
```

- `criarVetor(qtdElementos, valorPadrao)`
- `criarMatriz(qtdLinhas, qtdColunas, valorPadrao)`
- `inputVetor(mensagem, conversao)`
- `inputMatriz(mensagem, conversao)`
- `dimMatriz(matriz)`

Física (MRU)

Observações: Exploraremos a equação do MRU para determinar a posição final de um objeto dado a posição inicial, a velocidade e o tempo.

Questão 1.1 - Expressões e IO

A função do movimento uniforme nos permite encontrar a posição (S) de um objeto que se movimenta com velocidade constante (v) em qualquer instante de tempo (t), desde que saibamos sua posição inicial (S_i), através da seguinte fórmula:

$$S = S_i + v \cdot t$$

Implemente um programa que solicite os valores de entrada aos usuários e os armazene nas variáveis: posição inicial (S_i), a velocidade (v) e o instante de tempo (t), respectivamente. Todas as entradas são números reais. O programa deve realizar o cálculo da posição final (S) e imprimir o resultado no terminal com uma precisão de 2 casas decimais.

Sua resposta deve conter as mensagens exatamente como definidas nos exemplos de execução.

Exemplo de Execução 1

```
Informe a posição inicial (Si): 10
Informe a velocidade (v): 1
Informe o instante de tempo (t): 5
A posição final no tempo t = 5.00 será S = 15.00
```

Exemplo de Execução 2

```
Informe a posição inicial (Si): 55.7
Informe a velocidade (v): 60
Informe o instante de tempo (t): 5.8
A posição final no tempo t = 5.80 será S = 403.70
```

Solução:

```
1. Si = float(input('Informe a posição inicial (Si): '))
2. v = float(input('Informe a velocidade (v): '))
3. t = float(input('Informe o instante de tempo (t): '))
4. S = Si + v * t
5. print(f'A posição final no tempo t = {t:.2f} será S = {S:.2f}')
```


Questão 1.2 - Decisão simples

A função do movimento uniforme nos permite encontrar a posição (S) de um objeto que se movimenta com velocidade constante (v) em qualquer instante de tempo (t), desde que saibamos sua posição inicial (S_i), através da seguinte fórmula:

$$S = S_i + v \cdot t$$

Implemente um programa que receba do usuário os valores de entrada, e as armazene em variáveis: posição inicial (S_i), a velocidade (v) e o instante de tempo (t), respectivamente. Todas as entradas são números reais maiores do que 0 (zero). Caso os valores sejam inválidos, o programa apenas imprime uma mensagem de erro e é encerrado, caso contrário, o programa deve realizar o cálculo da posição final (S) e imprimir o resultado no terminal com uma precisão de 2 casas decimais.

Sua resposta deve conter as mensagens exatamente como definidas nos exemplos de execução.

Exemplo de Execução 1

```
Informe a posição inicial (Si): 10
Informe a velocidade (v): 1
Informe o instante de tempo (t): 5
A posição final no tempo t = 5.00 será S = 15.00
```

Exemplo de Execução 2

```
Informe a posição inicial (Si): 55.7
Informe a velocidade (v): 60
Informe o instante de tempo (t): 5.8
A posição final no tempo t = 5.80 será S = 403.70
```

Exemplo de Execução 3

```
Informe a posição inicial (Si): 0
Informe a velocidade (v): 60.8
Informe o instante de tempo (t): 5.8
Entradas inválidas
```

Solução:

```
1. Si = float(input('Informe a posição inicial (Si): '))
2. v = float(input('Informe a velocidade (v): '))
3. t = float(input('Informe o instante de tempo (t): '))
4. if Si > 0 and v > 0 and t > 0:
5.     S = Si + v * t
6.     print(f'A posição final no tempo t = {t:.2f} será S = {S:.2f}')
7. else:
8.     print('Entradas inválidas')
```

Questão 1.3 - Decisão simples

A função do movimento uniforme nos permite encontrar a posição (S) de um objeto que se movimenta com velocidade constante (v) em qualquer instante de tempo (t), desde que saibamos sua posição inicial (S_i), através da seguinte fórmula:

$$S = S_i + v \cdot t$$

Implemente um programa que receba do usuário os valores de entrada, e as armazene em variáveis: posição inicial (S_i), a velocidade (v) e o instante de tempo (t), respectivamente. Todas as entradas são números reais maiores do que 0 (zero), à exceção de S_i , que pode ser igual a 0 (zero). Caso uma entrada seja inválida, o programa apenas imprime uma mensagem de erro específica e é encerrado, caso contrário, o programa deve realizar o cálculo da posição final (S) e imprimir o resultado no terminal com uma precisão de 2 casas decimais.

Sua resposta deve conter as mensagens exatamente como definidas nos exemplos de execução.

Exemplo de Execução 1

```
Informe a posição inicial (Si): 10
Informe a velocidade (v): 1
Informe o instante de tempo (t): 5
A posição final no tempo t = 5.00 será S = 15.00
```

Exemplo de Execução 2

```
Informe a posição inicial (Si): 55.7
Informe a velocidade (v): 60
Informe o instante de tempo (t): 5.8
A posição final no tempo t = 5.80 será S = 403.70
```

Exemplo de Execução 3

```
Informe a posição inicial (Si): -10
Valor de Si inválido
```

Exemplo de Execução 4

```
Informe a posição inicial (Si): 30.78
Informe a velocidade (v): 0
Valor de v inválido
```

Exemplo de Execução 5

```
Informe a posição inicial (Si): 30.78
Informe a velocidade (v): 10.5
Informe o instante de tempo (t): -5
Valor de t inválido
```

Solução:

```
1. Si = float(input('Informe a posição inicial (Si): '))
```

```
2.     if Si < 0:
3.         print('Valor de Si inválido')
4.     else:
5.         v = float(input('Informe a velocidade (v): '))
6.         if v <= 0:
7.             print('Valor de v inválido')
8.         else:
9.             t = float(input('Informe o instante de tempo (t):
10.         '))
11.             if t <= 0:
12.                 print('Valor de t inválido')
13.             else:
14.                 S = Si + v * t
15.                 print(f'A posição final no tempo t = {t:.2f}
16.                 será S = {S:.2f}')
```

Questão 1.4 - Repetição while

A função do movimento uniforme nos permite encontrar a posição (S) de um objeto que se movimenta com velocidade constante (v) em qualquer instante de tempo (t), desde que saibamos sua posição inicial (Si), através da seguinte fórmula:

$$S = S_i + v \cdot t$$

Implemente um programa que receba do usuário os valores de entrada, e as armazene em variáveis: posição inicial (Si), a velocidade (v) e o instante de tempo (t), respectivamente. Todas as entradas são números reais maiores do que 0 (zero), à exceção de Si, que pode ser igual a 0 (zero). Observe que o programa avalia as entradas, forçando valores válidos. Em seguida, o programa deve realizar o cálculo da posição final (S) e imprimir o resultado no terminal com uma precisão de 2 casas decimais.

Sua resposta deve conter as mensagens exatamente como definidas nos exemplos de execução.

Exemplo de Execução 1

```
Informe a posição inicial (Si): 10
Informe a velocidade (v): 1
Informe o instante de tempo (t): 5
A posição final no tempo t = 5.00 será S = 15.00
```

Exemplo de Execução 2

```
Informe a posição inicial (Si): 55.7
Informe a velocidade (v): 60
Informe o instante de tempo (t): 5.8
A posição final no tempo t = 5.80 será S = 403.70
```

Exemplo de Execução 3

```
Informe a posição inicial (Si): -10
Informe a posição inicial (Si): -1
Informe a posição inicial (Si): 0
Informe a velocidade (v): 0
Informe a velocidade (v): 10
Informe o instante de tempo (t): -5
Informe o instante de tempo (t): 0
Informe o instante de tempo (t): 5
A posição final no tempo t = 5.00 será S = 50.00
```

Solução:

```
1. Si = float(input('Informe a posição inicial (Si): '))
2. while Si < 0:
3.     Si = float(input('Informe a posição inicial (Si): '))
4. v = float(input('Informe a velocidade (v): '))
5. while v <= 0:
```

```
6.         v = float(input('Informe a velocidade (v): '))
7.         t = float(input('Informe o instante de tempo (t): '))
8.         while t <= 0:
9.             t = float(input('Informe o instante de tempo (t): '))
10.        S = Si + v * t
11.        print(f'A posição final no tempo t = {t:.2f} será S =
              {S:.2f}')
```

Questão 1.5 - Repetição for

A função do movimento uniforme nos permite encontrar a posição (S) de um objeto que se movimenta com velocidade constante (v) em qualquer instante de tempo (t), desde que saibamos sua posição inicial (S_i), através da seguinte fórmula:

$$S = S_i + v \cdot t$$

Implemente um programa que receba do usuário os valores de entrada, e as armazene em variáveis: posição inicial (S_i), a velocidade (v) e o instante de tempo final (T_f), respectivamente. S_i e v são valores reais, e T_f é um valor inteiro. Considere que apenas entradas válidas serão fornecidas, não é necessário realizar validações. O programa deve realizar o cálculo da posição final (S) e imprimir o resultado no terminal para todos os instantes t variando entre 1 e T_f , com uma precisão de 2 casas decimais.

Sua resposta deve conter as mensagens exatamente como definidas nos exemplos de execução.

Exemplo de Execução 1

```
Informe a posição inicial (Si): 10
Informe a velocidade (v): 1
Informe o instante de tempo final (Tf): 5
A posição final no tempo t = 1 será S = 11.00
A posição final no tempo t = 2 será S = 12.00
A posição final no tempo t = 3 será S = 13.00
A posição final no tempo t = 4 será S = 14.00
A posição final no tempo t = 5 será S = 15.00
```

Exemplo de Execução 2

```
Informe a posição inicial (Si): 55.7
Informe a velocidade (v): 60
Informe o instante de tempo final (Tf): 7
A posição final no tempo t = 1 será S = 115.70
A posição final no tempo t = 2 será S = 175.70
A posição final no tempo t = 3 será S = 235.70
A posição final no tempo t = 4 será S = 295.70
A posição final no tempo t = 5 será S = 355.70
A posição final no tempo t = 6 será S = 415.70
A posição final no tempo t = 7 será S = 475.70
```

Solução:

```
1. Si = float(input('Informe a posição inicial (Si): '))
2. v = float(input('Informe a velocidade (v): '))
3. Tf = int(input('Informe o instante de tempo final (Tf): '))
4. for t in range(1, Tf+1):
5.     S = Si + v * t
6.     print(f'A posição final no tempo t = {t} será S =
    {S:.2f}')
```

Questão 1.6 - Laços aninhados

A função do movimento uniforme nos permite encontrar a posição (S) de um objeto que se movimenta com velocidade constante (v) em qualquer instante de tempo (t), desde que saibamos sua posição inicial (S_i), através da seguinte fórmula:

$$S = S_i + v \cdot t$$

Implemente um programa que receba do usuário os valores de entrada, e as armazene em variáveis: posição inicial (S_i), a velocidade final (V_f) e o instante de tempo final (T_f), respectivamente. S_i é um valor real, V_f e T_f são valores inteiros. Considere que apenas entradas válidas serão fornecidas, não é necessário realizar validações. O programa deve realizar o cálculo da posição final (S) e imprimir o resultado no terminal com pelo menos 10 caracteres e uma precisão de 2 casas decimais para a combinação de todas as velocidades v variando entre 1 e V_f e todos os instantes t variando entre 1 e T_f . Observe que a saída dos valores de S é uma tabela, onde os resultados são obtidos com as seguintes variações de valores, no caso do exemplo de execução 1:

	$t = 1$	$t = 2$	$t = 3$	$t = 4$	$t = 5$
$v = 1$	$S = 11.00$	$S = 12.00$	$S = 13.00$	$S = 14.00$	$S = 15.00$
$v = 2$	$S = 12.00$	$S = 14.00$	$S = 16.00$	$S = 18.00$	$S = 20.00$
$v = 3$	$S = 13.00$	$S = 16.00$	$S = 19.00$	$S = 22.00$	$S = 25.00$

Sua resposta deve conter as mensagens exatamente como definidas nos exemplos de execução.

Exemplo de Execução 1

```
Informe a posição inicial (Si): 10
Informe a velocidade final (Vf): 3
Informe o instante de tempo final (Tf): 5
Valores de S:
    11.00    12.00    13.00    14.00    15.00
    12.00    14.00    16.00    18.00    20.00
    13.00    16.00    19.00    22.00    25.00
```

Exemplo de Execução 2

```
Informe a posição inicial (Si): 55.7
Informe a velocidade final (Vf): 6
Informe o instante de tempo final (Tf): 7
Valores de S:
    56.70    57.70    58.70    59.70    60.70    61.70    62.70
    57.70    59.70    61.70    63.70    65.70    67.70    69.70
    58.70    61.70    64.70    67.70    70.70    73.70    76.70
    59.70    63.70    67.70    71.70    75.70    79.70    83.70
    60.70    65.70    70.70    75.70    80.70    85.70    90.70
```

61.70

67.70

73.70

79.70

85.70

91.70

97.70

Solução:

```
1. Si = float(input('Informe a posição inicial (Si): '))
2. Vf = int(input('Informe a velocidade final (Vf): '))
3. Tf = int(input('Informe o instante de tempo final (Tf): '))
4. print('Valores de S:')
5. for v in range(1, Vf+1):
6.     for t in range(1, Tf+1):
7.         S = Si + v * t
8.         print(f'{S:10.2f}', end='')
9.     print()
```


Questão 1.7 - Funções

A função do movimento uniforme nos permite encontrar a posição (S) de um objeto que se movimenta com velocidade constante (v) em qualquer instante de tempo (t), desde que saibamos sua posição inicial (S_i), através da seguinte fórmula:

$$S = S_i + v \cdot t$$

Implemente um programa que receba do usuário os valores de entrada, e as armazene em variáveis: posição inicial (S_i), a velocidade (v) e o instante de tempo final (T_f), respectivamente. S_i e v são valores reais, e T_f é um valor inteiro. Considere que apenas entradas válidas serão fornecidas, não é necessário realizar validações. O programa deve realizar o cálculo da posição final (S) e imprimir o resultado no terminal com uma precisão de 2 casas decimais para todos os instantes t variando entre 1 e T_f . Para o cálculo de S você deverá implementar a função `calculaS`, que recebe os valores de S_i , v e t como argumentos de entrada, calcula e retorna o valor de S , arredondado para 3 casas decimais (para isso, utilize a função `round`).

Sua resposta deve conter as mensagens exatamente como definidas nos exemplos de execução.

Exemplo de Execução 1

```
Informe a posição inicial (Si): 10
Informe a velocidade (v): 1
Informe o instante de tempo final (Tf): 5
A posição final no tempo t = 1 será S = 11.00
A posição final no tempo t = 2 será S = 12.00
A posição final no tempo t = 3 será S = 13.00
A posição final no tempo t = 4 será S = 14.00
A posição final no tempo t = 5 será S = 15.00
```

Exemplo de Execução 2

```
Informe a posição inicial (Si): 55.7
Informe a velocidade (v): 60
Informe o instante de tempo final (Tf): 7
A posição final no tempo t = 1 será S = 115.70
A posição final no tempo t = 2 será S = 175.70
A posição final no tempo t = 3 será S = 235.70
A posição final no tempo t = 4 será S = 295.70
A posição final no tempo t = 5 será S = 355.70
A posição final no tempo t = 6 será S = 415.70
A posição final no tempo t = 7 será S = 475.70
```

Solução:

```
1. def calculaS(Si, v, t):
2.     S = Si + v * t
3.     return round(S, 3)
```

```
4.  
5.     Si = float(input('Informe a posição inicial (Si): '))  
6.     v = float(input('Informe a velocidade (v): '))  
7.     Tf = int(input('Informe o instante de tempo final (Tf): '))  
8.     for t in range(1, Tf+1):  
9.         print(f'A posição final no tempo t = {t} será S =  
           {calculaS(Si, v, t):.2f}')
```

Questão 1.8 - ED Homogêneas - Vetores

A função do movimento uniforme nos permite encontrar a posição (S) de um objeto que se movimenta com velocidade constante (v) em qualquer instante de tempo (t), desde que saibamos sua posição inicial (S_i), através da seguinte fórmula:

$$S = S_i + v \cdot t$$

Implemente um programa que receba do usuário os valores de entrada, e as armazene em vetores de: posições iniciais (S_i), velocidades (V) e instantes de tempo (T), respectivamente. Todos são vetores de valores reais que devem possuir a mesma quantidade de elementos, ou seja, eles possuem uma relação pelos seus índices. Seu programa deve verificar se os vetores possuem o mesmo tamanho, neste caso, deve realizar o cálculo das posições finais (S_x , onde x é um índice para os três vetores) e imprimir os resultados no terminal com uma precisão de 2 casas decimais, caso contrário, imprime uma mensagem de entradas inválidas.

Para facilitar o entendimento, observe que, para o exemplo de execução 2, os três vetores e os resultados são apresentados na tabela abaixo:

	<i>índice 0</i>	<i>índice 1</i>	<i>índice 3</i>
<i>Vetor S_i</i>	15	10.5	6.33
<i>Vetor V</i>	5	15	21.8
<i>Vetor T</i>	1	5	10
Resultados S_x	20.00	85.50	224.33

Sua resposta deve conter as mensagens exatamente como definidas nos exemplos de execução.

Exemplo de Execução 1

```
Vetor Si: 15, 10.5, 6.33
```

```
Vetor V: 5
```

```
Vetor T: 1, 5, 10
```

```
Entradas inválidas
```

Exemplo de Execução 2

```
Vetor Si: 15, 10.5, 6.33
```

```
Vetor V: 5, 15, 21.8
```

```
Vetor T: 1, 5, 10
```

```
S[0] = 20.00
```

```
S[1] = 85.50
```

```
S[2] = 224.33
```

Solução:

```
1. from biblioteca import *
```

```
2.
3.     Si = inputVetor('Vetor Si: ', float)
4.     V = inputVetor('Vetor V: ', float)
5.     T = inputVetor('Vetor T: ', float)
6.
7.     if len(Si) != len(V) or len(Si) != len(T) or len(V) !=
len(T):
8.         print('Entradas inválidas')
9.     else:
10.         for x in range(len(Si)):
11.             S = Si[x] + V[x] * T[x]
12.             print(f'S[{x}] = {S:.2f}')
```

Questão 1.9 - ED Homogêneas - Matrizes

A função do movimento uniforme nos permite encontrar a posição (S) de um objeto que se movimenta com velocidade constante (v) em qualquer instante de tempo (t), desde que saibamos sua posição inicial (S_i), através da seguinte fórmula:

$$S = S_i + v \cdot t$$

Implemente um programa que receba do usuário os valores de entrada em uma matriz, onde: (1) a primeira linha armazena as posições iniciais (S_i); (2) a segunda linha armazena as velocidades (V); e (3) a terceira linha armazena os instantes de tempo (T). A matriz possui valores reais. Seu programa deve verificar se a matriz possui exatamente 3 (três) linhas, neste caso, deve realizar o cálculo das posições finais (S_x , onde x é um índice para a coluna da matriz) e imprimir os resultados no terminal com uma precisão de 2 casas decimais, caso contrário, imprime uma mensagem de entrada inválida. Para facilitar o entendimento, observe que, para o exemplo de execução 2, a matriz e os resultados são apresentados na tabela abaixo:

	coluna 0	coluna 1	coluna 3
<i>Linha 0 (S_i)</i>	15	10.5	6.33
<i>Linha 1 (V)</i>	5	15	21.8
<i>Linha 2 (T)</i>	1	5	10
Resultados S_x	20.00	85.50	224.33

Sua resposta deve conter as mensagens exatamente como definidas nos exemplos de execução.

Exemplo de Execução 1

```
Matriz de valores: 15, 10.5, 6.33; 1, 5, 10
Entrada inválida
```

Exemplo de Execução 2

```
Matriz de valores: 15, 10.5, 6.33; 5, 15, 21.8; 1, 5, 10
S[0] = 20.00
S[1] = 85.50
S[2] = 224.33
```

Solução:

```
1. from biblioteca import *
2.
3. M = inputMatriz('Matriz de valores: ', float)
4. nL, nC = dimMatriz(M)
```

```
5.     if nL != 3:
6.         print('Entrada inválida')
7.     else:
8.         for col in range(nC):
9.             S = M[0][col] + M[1][col] * M[2][col]
10.            print(f'S[{col}] = {S:.2f}')
```

Questão 1.10 - ED Heterogêneas e Funções

A função do movimento uniforme nos permite encontrar a posição (S) de um objeto que se movimenta com velocidade constante (v) em qualquer instante de tempo (t), desde que saibamos sua posição inicial (S_i), através da seguinte fórmula:

$$S = S_i + v \cdot t$$

Implemente um programa que receba do usuário os valores de entrada, e as armazene em variáveis: posição inicial (S_i), a velocidade (v) e o instante de tempo (t), respectivamente. Estes valores devem ser armazenados em uma variável heterogênea (MU) com campos que representam os três valores. O programa deve realizar o cálculo da posição final (S) e imprimir o resultado no terminal com uma precisão de 2 casas decimais. Para o cálculo de S você deverá implementar a função `calculaS`, que recebe o valor heterogêneo (MU) como argumento de entrada, calcula e retorna o valor de S , arredondado para 3 casas decimais (para isso, utilize a função `round`). O programa pergunta se o usuário deseja fazer um novo cálculo, continuando a execução quando a resposta for "sim" e finalizando caso contrário.

Sua resposta deve conter as mensagens exatamente como definidas nos exemplos de execução.

Exemplo de Execução 1

```
Informe a posição inicial (Si): 10
Informe a velocidade (v): 1
Informe o instante de tempo (t): 5
A posição final no tempo t = 5.00 será S = 15.00

Continuar? (sim ou não): sim

Informe a posição inicial (Si): 55.7
Informe a velocidade (v): 60
Informe o instante de tempo (t): 5.8
A posição final no tempo t = 5.80 será S = 403.70

Continuar? (sim ou não): não
```

Exemplo de Execução 2

```
Informe a posição inicial (Si): 100
Informe a velocidade (v): 57.3
Informe o instante de tempo (t): 10
A posição final no tempo t = 10.00 será S = 673.00

Continuar? (sim ou não): não
```

Solução:

```
1. def calculaS(MU):
```

```
2.         S = MU['Si'] + MU['v'] * MU['t']
3.         return round(S, 3)
4.
5.     continuar = 'sim'
6.     while continuar == 'sim':
7.         Si = float(input('Informe a posição inicial (Si): '))
8.         v = float(input('Informe a velocidade (v): '))
9.         t = float(input('Informe o instante de tempo (t): '))
10.        MU = { 'Si': Si, 'v': v, 't': t }
11.        print(f'A posição final no tempo t = {t:.2f} será S =
{calculaS(MU):.2f}')
12.        continuar = input('\nContinuar? (sim ou não): ')
13.        print()
```


APÊNDICE E – Testes ChatPDF

ChatPDF: Olá! Bem-vindo ao resumo da matéria sobre comandos de atribuição, operadores matemáticos, funções matemáticas elementares, valores constantes, importação da biblioteca math, entrada e saída de dados pelo terminal e strings de formatação. Esperamos que este resumo seja útil para você em seus estudos de programação em Python!

1. Como posso utilizar a função `math.sqrt(x)` para calcular a raiz quadrada de um número em Python?
2. Qual é a diferença entre os operadores de divisão (`/`) e resto da divisão (`%`) em Python?
3. Como posso formatar uma string utilizando a sintaxe `f'conteúdo'` em Python?

Usuário: Contexto: Assuma o papel de um professor da matéria de programação de computadores com Python, que usa os documentos fornecidos como base para ministrar as aulas e para criação de exercícios usados nas atividades dos alunos. As aulas são para turmas de cursos de graduação de diversas áreas de engenharia.

Matéria: Física.

Assuntos da Matéria abordados: MRU.

Tópicos de programação abordados: Variáveis, expressões, entradas e saídas;

Decisão simples (`if`, `else`);

Decisão complexa (`if`, `elif`, `else`);

Repetição e Laços;

Laços aninhados;

Funções;

Estrutura de dados homogêneos com vetores;

Estrutura de dados homogêneos com matrizes;

Estrutura de dados heterogêneos (registros).

Instrução: Com base nos documentos fornecidos, crie questões de programação para cada <Tópicos de programação> utilizando os <Assuntos da matéria> da disciplina de <Matéria>. Garanta que cada questão aborde um assunto específico e que todos os tópicos de programação listados sejam contemplados.

Classifique as questões usando um medidor de dificuldade de 1 à 10, onde 1 indica a questão mais fácil e 10 a mais complexa.

Use o seguinte formato de resposta:

<matéria> - <assunto da matéria abordado>

Questão <número da questão> - <tópico usado> - <dificuldade da questão>

Contextualização e introdução sucinta sobre <assunto da matéria abordado>, tal qual explicações prévias dos recursos utilizados no desenvolvimento da questão.

Enunciado da Questão.

Exemplos:

Três exemplos de execução e da saída que se espera do programa.

Solução:

Uma solução possível.

ChatPDF: Física - MRU

Questão 1 - Variáveis - Dificuldade 1

Contextualização: Em um experimento de MRU, é necessário armazenar a posição inicial, a velocidade e o tempo decorrido para calcular a posição final do objeto. Para isso, é necessário utilizar variáveis em Python.

Enunciado: Crie um programa em Python que solicite ao usuário a posição inicial (em metros), a velocidade (em metros por segundo) e o tempo decorrido (em segundos) de um objeto em MRU e calcule a posição final do objeto. Armazene cada valor em uma variável.

Exemplos:

Exemplo 1:

Digite a posição inicial (m): 10

Digite a velocidade (m/s): 5

Digite o tempo decorrido (s): 2

A posição final do objeto é 20.0 metros.

Exemplo 2:

Digite a posição inicial (m): 0

Digite a velocidade (m/s): 10

Digite o tempo decorrido (s): 5

A posição final do objeto é 50.0 metros.

Exemplo 3:

Digite a posição inicial (m): 5

Digite a velocidade (m/s): 0

Digite o tempo decorrido (s): 10

A posição final do objeto é 5.0 metros.

Solução:

```
posicao_inicial = float(input("Digite a posição inicial (m): "))
```

```
velocidade = float(input("Digite a velocidade (m/s): "))
```

```
tempo_decorrido = float(input("Digite o tempo decorrido (s): "))
```

```
posicao_final = posicao_inicial + velocidade * tempo_decorrido
```

```
print(f"A posição final do objeto é {posicao_final} metros.")
```

Questão 2 - Expressões - Dificuldade 2

Contextualização: Em um experimento de MRU, é possível calcular a posição final do objeto utilizando uma expressão matemática. Em Python, é possível utilizar expressões para realizar cálculos.

Enunciado: Crie um programa em Python que solicite ao usuário a posição inicial (em metros), a velocidade (em metros por segundo) e o tempo decorrido (em segundos) de um objeto em MRU e calcule a posição final do objeto utilizando uma expressão matemática. Imprima o resultado na tela.

Exemplos:

Exemplo 1:

Digite a posição inicial (m): 10

Digite a velocidade (m/s): 5

Digite o tempo decorrido (s): 2

A posição final do objeto é 20.0 metros.

Exemplo 2:

Digite a posição inicial (m): 0

Digite a velocidade (m/s): 10

Digite o tempo decorrido (s): 5

A posição final do objeto é 50.0 metros.

Exemplo 3:

Digite a posição inicial (m): 5
Digite a velocidade (m/s): 0
Digite o tempo decorrido (s): 10
A posição final do objeto é 5.0 metros.

Solução:

```
posicao_inicial = float(input("Digite a posição inicial (m): "))  
velocidade = float(input("Digite a velocidade (m/s): "))  
tempo_decorrido = float(input("Digite o tempo decorrido (s): "))
```

```
posicao_final = posicao_inicial + velocidade * tempo_decorrido
```

```
print(f"A posição final do objeto é {posicao_final} metros.")
```

Questão 3 - Entradas e Saídas - Dificuldade 3

Contextualização: Em um experimento de MRU, é necessário solicitar ao usuário as informações necessárias para realizar o cálculo da posição final do objeto. Para isso, é necessário utilizar entradas e saídas em Python.

Enunciado: Crie um programa em Python que solicite ao usuário a posição inicial (em metros), a velocidade (em metros por segundo) e o tempo decorrido (em segundos) de um objeto em MRU e imprima na tela uma mensagem informando a posição final do objeto.

Exemplos:

Exemplo 1:

Digite a posição inicial (m): 10
Digite a velocidade (m/s): 5
Digite o tempo decorrido (s): 2
A posição final do objeto é 20.0 metros.

Exemplo 2:

Digite a posição inicial (m): 0
Digite a velocidade (m/s): 10
Digite o tempo decorrido (s): 5
A posição final do objeto é 50.0 metros.

Exemplo 3:

Digite a posição inicial (m): 5
Digite a velocidade (m/s): 0
Digite o tempo decorrido (s): 10
A posição final do objeto é 5.0 metros.

Solução:

```
posicao_inicial = float(input("Digite a posição inicial (m): "))  
velocidade = float(input("Digite a velocidade (m/s): "))  
tempo_decorrido = float(input("Digite o tempo decorrido (s): "))
```

```
posicao_final = posicao_inicial + velocidade * tempo_decorrido
```

```
print(f"A posição final do objeto é {posicao_final} metros.")
```

Questão 4 - Decisão Simples - Dificuldade 4

Contextualização: Em um experimento de MRU, é possível que o objeto esteja parado. Nesse caso, a velocidade é igual a zero e a posição final é igual à posição inicial. Para tratar esse caso, é necessário utilizar uma decisão simples em Python.

Enunciado: Crie um programa em Python que solicite ao usuário a posição inicial (em metros), a velocidade (em metros por segundo) e o tempo decorrido (em segundos) de um objeto em MRU e calcule a posição final do objeto. Caso a velocidade seja igual a zero, imprima na tela uma mensagem informando que o objeto está parado e que a posição final é igual à posição inicial.

Exemplos:

Exemplo 1:

Digite a posição inicial (m): 10

Digite a velocidade (m/s): 5

Digite o tempo decorrido (s): 2

A posição final do objeto é 20.0 metros.

Exemplo 2:

Digite a posição inicial (m): 0

Digite a velocidade (m/s): 10

Digite o tempo decorrido (s): 5

A posição final do objeto é 50.0 metros.

Exemplo 3:

Digite a posição inicial (m): 5

Digite a velocidade (m/s): 0

Digite o tempo decorrido (s): 10

O objeto está parado. A posição final é 5.0 metros.

Solução:

```
posicao_inicial = float(input("Digite a posição inicial (m): "))
```

```
velocidade = float(input("Digite a velocidade (m/s): "))
```

```
tempo_decorrido = float(input("Digite o tempo decorrido (s): "))
```

```
if velocidade == 0:
```

```
    print(f"O objeto está parado. A posição final é {posicao_inicial} metros.")
```

```
else:
```

```
    posicao_final = posicao_inicial + velocidade * tempo_decorrido
```

```
    print(f"A posição final do objeto é {posicao_final} metros.")
```

Questão 5 - Decisão Complexa - Dificuldade 5

Contextualização: Em um experimento de MRU, é possível que o objeto esteja se movendo em direções opostas. Nesse caso, a velocidade é negativa e a posição final pode ser menor que a posição inicial. Para tratar esse caso, é necessário utilizar uma decisão complexa em Python.

Enunciado: Crie um programa em Python que solicite ao usuário a posição inicial (em metros), a velocidade (em metros por segundo) e o tempo decorrido (em segundos) de um objeto em MRU e calcule a posição final do objeto. Caso a velocidade seja igual a zero, imprima na tela uma mensagem informando que o objeto está parado e que a posição final é igual à posição inicial. Caso a velocidade seja negativa, imprima na tela uma

mensagem informando que o objeto está se movendo em direção oposta e que a posição final pode ser menor que a posição inicial.

Exemplos:

Exemplo 1:

Digite a posição inicial (m): 10

Digite a velocidade (m/s): 5

Digite o tempo decorrido (s): 2

A posição final do objeto é 20.0 metros.

Exemplo 2:

Digite a posição inicial (m): 0

Digite a velocidade (m/s): 10

Digite o tempo decorrido (s): 5

A posição final do objeto é 50.0 metros.

Exemplo 3:

Digite a posição inicial (m): 5

Digite a velocidade (m/s): 0

Digite o tempo decorrido (s): 10

O objeto está parado. A posição final é 5.0 metros.

Exemplo 4:

Digite a posição inicial (m): 10

Digite a velocidade (m/s): -5

Digite o tempo decorrido (s): 2

O objeto está se movendo em direção oposta. A posição final pode ser 0.0 metros.

Solução:

```
posicao_inicial = float(input("Digite a posição inicial (m): "))
```

```
veloc
```

Usuário: Contexto: Assuma o papel de um professor da matéria de programação de computadores com Python, que usa os documentos fornecidos como base para ministrar as aulas e para criação de exercícios usados nas atividades dos alunos. As aulas são para turmas de cursos de graduação de diversas áreas de engenharia.

Matéria: Mecânica.

Assuntos da Matéria abordados: Dinâmica das máquinas, Sistemas de controle.

Instrução: Com base nos documentos fornecidos, crie uma prova avaliativa usando os tópicos de programação usados em "Resumo da matéria (PT1)" utilizando os <Assuntos da matéria> da disciplina de <Matéria>.

Classifique as questões usando um medidor de dificuldade de 1 à 10, onde 1 indica a questão mais fácil e 10 a mais complexa.

A prova deverá ter 5 questões. 3 questões deverão ter dificuldade acima de 3 e 2 questões dificuldade acima de 7. Em pelo menos uma questão, use mais do que um tópico de programação.

Use o seguinte formato de resposta:

<matéria> - <assunto da matéria abordado> - <tópicos de programação>

Questão <número da questão> - <tópico usado> - <dificuldade da questão>

Contextualização e introdução sucinta sobre <assunto da matéria abordado>, tal qual explicações prévias dos recursos utilizados no desenvolvimento da questão.

Enunciado da Questão.

Exemplos:

Três exemplos de execução e da saída que se espera do programa.

Solução:

Uma solução possível.

ChatPDF: Mecânica - Movimento Uniforme - Comando de atribuição, Estrutura homogênea VETOR, Estrutura homogênea MATRIZ

Questão 1 - Comando de atribuição - Dificuldade 2

Contextualização: O movimento uniforme é um dos conceitos fundamentais da cinemática, onde um objeto se desloca com velocidade constante. Nesta questão, será solicitado ao aluno que implemente um programa simples para calcular a posição final de um objeto em movimento uniforme.

Enunciado: Escreva um programa que solicite ao usuário a posição inicial (S_i), a velocidade (v) e o instante de tempo (t), e em seguida calcule e imprima a posição final (S) com uma precisão de 2 casas decimais.

Exemplos:

Input:

Informe a posição inicial (S_i): 10

Informe a velocidade (v): 1

Informe o instante de tempo (t): 5

Output:

A posição final no tempo $t = 5.00$ será $S = 15.00$

Solução:

```
```python
```

```
Si = float(input('Informe a posição inicial (Si): '))
```

```
v = float(input('Informe a velocidade (v): '))
```

```
t = float(input('Informe o instante de tempo (t): '))
```

```
S = Si + v * t
```

```
print(f'A posição final no tempo t = {t:.2f} será S = {S:.2f}')
```

```
```
```

Mecânica - Movimento Uniforme - Funções matemáticas elementares

Questão 2 - Funções matemáticas elementares - Dificuldade 4

Contextualização: As funções matemáticas elementares são essenciais para cálculos mais complexos em mecânica. Nesta questão, o aluno será desafiado a implementar um programa que utilize funções matemáticas para calcular a posição final de um objeto em movimento uniforme.

Enunciado: Escreva um programa que solicite ao usuário a posição inicial (S_i), a velocidade (v) e o instante de tempo (t), e em seguida calcule e imprima a posição final (S) com uma precisão de 2 casas decimais, utilizando funções matemáticas elementares.

Exemplos:

Input:

Informe a posição inicial (S_i): 10

Informe a velocidade (v): 1

Informe o instante de tempo (t): 5

Output:

A posição final no tempo $t = 5.00$ será $S = 15.00$

Solução:

```
```python
```

```

import math
Si = float(input('Informe a posição inicial (Si): '))
v = float(input('Informe a velocidade (v): '))
t = float(input('Informe o instante de tempo (t): '))
S = Si + v * t
print(f'A posição final no tempo t = {t:.2f} será S = {S:.2f}')
```

```

Mecânica - Movimento Uniforme - Matrizes - Estrutura homogênea MATRIZ
 Questão 3 - Matrizes - Dificuldade 6

Contextualização: O uso de matrizes é comum em problemas de física e engenharia, permitindo a organização de dados de forma eficiente. Nesta questão, o aluno será desafiado a implementar um programa que utilize matrizes para calcular as posições finais de objetos em movimento uniforme.

Enunciado: Escreva um programa que receba do usuário os valores de entrada em uma matriz, onde a primeira linha armazena as posições iniciais (Si), a segunda linha armazena as velocidades (V), e a terceira linha armazena os instantes de tempo (T). O programa deve calcular as posições finais (Sx) e imprimir os resultados no terminal com uma precisão de 2 casas decimais.

Exemplos:

Input:

Matriz de valores: 15, 10.5, 6.33; 5, 15, 21.8; 1, 5, 10

Output:

S[0] = 20.00

S[1] = 85.50

S[2] = 224.33

Solução:

```
```python
```

```
M = [[15, 10.5, 6.33], [5, 15, 21.8], [1, 5, 10]]
```

```
S = [M[0][i] + M[1][i] * M[2][i] for i in range(len(M[0]))]
```

```
print(f'S[0] = {S[0]:.2f}')
```

```
print(f'S[1] = {S[1]:.2f}')
```

```
print(f'S[2] = {S[2]:.2f}')
```

```
```
```

Mecânica - Movimento Uniforme - Comando de atribuição, Estrutura homogênea VETOR

Questão 4 - Comando de atribuição, Estrutura homogênea VETOR -
 Dificuldade 8

Contextualização: A utilização de vetores é comum em problemas de mecânica, permitindo a manipulação eficiente de conjuntos de dados. Nesta questão, o aluno será desafiado a implementar um programa que utilize vetores para calcular as posições finais de objetos em movimento uniforme.

Enunciado: Escreva um programa que solicite ao usuário as posições iniciais (Si), as velocidades (V) e os instantes de tempo (T), e em seguida calcule e imprima as posições finais (Sx) com uma precisão de 2 casas decimais.

Exemplos:

Input:

Posições iniciais: 15, 10.5, 6.33

Velocidades: 5, 15, 21.8

Instantes de tempo: 1, 5, 10

Output:

S[0] = 20.00

S[1] = 85.50

S[2] = 224.33

Solução:

```
```python
```

```
Si = [15, 10.5, 6.33]
```

```
V = [5, 15, 21.8]
```

```
T = [1, 5, 10]
```

```
S = [Si[i] + V[i] * T[i] for i in range(len(Si))]
```

```
print(f'S[0] = {S[0]:.2f}')
```

```
print(f'S[1] = {S[1]:.2f}')
```

```
print(f'S[2] = {S[2]:.2f}')
```

```
```
```

Mecânica - Movimento Uniforme - Funções matemáticas elementares,
Estrutura homogênea MATRIZ

Questão 5 - Funções matemáticas elementares, Estrutura homogênea MATRIZ -
Dificuldade 9

Contextualização: A combinação de funções matemáticas e matrizes é essencial para resolver problemas complexos de mecânica. Nesta questão, o aluno será desafiado a implementar um programa que utilize funções matemáticas e matrizes para calcular as posições finais de objetos em movimento uniforme.

Enunciado: Escreva um programa que receba do usuário os valores de entrada em uma matriz, onde a primeira linha armazena as posições iniciais (Si), a segunda linha armazena as velocidades (V), e a terceira linha armazena os instantes de tempo (T). O programa deve calcular as posições finais (Sx) e imprimir os resultados no terminal com uma precisão de 2 casas decimais, utilizando funções matemáticas elementares.

Exemplos:

Input:

Matriz de valores: 15, 10.5, 6.33; 5, 15, 21.8; 1, 5, 10

Output:

S[0] = 20.00

S[1] = 85.50

S[2] = 224.33

Solução:

```
```python
```

```
import math
```

```
M = [[15, 10.5, 6.33], [5, 15, 21.8], [1, 5, 10]]
```

```
S = [M[0][i] + M[1][i] * M[2][i] for i in range(len(M[0]))]
```

```
print(f'S[0] = {S[0]:.2f}')
```

```
print(f'S[1] = {S[1]:.2f}')
```

```
print(f'S[2] = {S[2]:.2f}')
```

```
```
```


ChatPDF: Olá! Bem-vindo ao nosso guia de Python para iniciantes! Aqui você encontrará informações sobre comandos de atribuição, operadores matemáticos, funções matemáticas elementares, valores constantes, entrada e saída de dados pelo terminal e strings de formatação. Esperamos que este guia seja útil para você começar a programar em Python!

Algumas perguntas que você pode ter sobre este guia são:

1. Como posso usar os operadores matemáticos em Python?
2. Quais são as funções matemáticas elementares disponíveis na biblioteca math?
3. Como posso formatar strings em Python usando a sintaxe f'conteúdo'?

Usuário: Contexto: Assuma o papel de um professor da matéria de programação de computadores com Python, que usa os documentos fornecidos como base para ministrar as aulas e para criação de exercícios usados nas atividades dos alunos. As aulas são para turmas de cursos de graduação de diversas áreas de engenharia.

Matéria: Física.

Assuntos da Matéria abordados: MRU.

Tópicos de programação abordados: Variáveis, expressões, entradas e saídas;

Decisão simples (if, else);

Decisão complexa (if, elif, else);

Repetição e Laços;

Laços aninhados;

Funções;

Estrutura de dados homogêneos com vetores;

Estrutura de dados homogêneos com matrizes;

Estrutura de dados heterogêneos (registros).

Instrução: Com base nos documentos fornecidos, crie questões de programação para cada <Tópicos de programação> utilizando os <Assuntos da matéria> da disciplina de <Matéria>. Garanta que cada questão aborde um assunto específico e que todos os tópicos de programação listados sejam contemplados.

Classifique as questões usando um medidor de dificuldade de 1 à 10, onde 1 indica a questão mais fácil e 10 a mais complexa.

Use o seguinte formato de resposta:

<matéria> - <assunto da matéria abordado>

Questão <número da questão> - <tópico usado> - <dificuldade da questão>

Contextualização e introdução sucinta sobre <assunto da matéria abordado>, tal qual explicações prévias dos recursos utilizados no desenvolvimento da questão.

Enunciado da Questão.

Exemplos:

Três exemplos de execução e da saída que se espera do programa.

Solução:

Uma solução possível.

ChatPDF: Física - MRU

Questão 1 - Variáveis - Dificuldade: 1

Contextualização: As variáveis são utilizadas para armazenar valores que podem ser posteriormente manipulados em um programa. Nesta questão, os alunos serão desafiados a criar variáveis para representar grandezas físicas relacionadas ao Movimento Retilíneo Uniforme (MRU).

Enunciado: Crie um programa em Python que utilize variáveis para representar a posição inicial (s), a velocidade (v) e o tempo (t) de um objeto em MRU. Em seguida, calcule e imprima a posição final do objeto utilizando a fórmula do MRU: $s = s_0 + v \cdot t$.

Exemplos:

$s = 10$, $v = 2$, $t = 5$

Saída esperada: A posição final do objeto é 20.

$s = 5$, $v = 3$, $t = 2$

Saída esperada: A posição final do objeto é 11.

$s = 0$, $v = 1$, $t = 10$

Saída esperada: A posição final do objeto é 10.

Solução:

```
```python
Entrada dos valores das variáveis
s = float(input("Digite a posição inicial (s): "))
v = float(input("Digite a velocidade (v): "))
t = float(input("Digite o tempo (t): "))

Cálculo da posição final
sf = s + v * t

Saída do resultado
print(f"A posição final do objeto é {sf}.")
```
```

Física - MRU

Questão 2 - Decisão simples (if, else) - Dificuldade: 3

Contextualização: As estruturas de decisão permitem que o programa execute diferentes ações com base em condições específicas. Nesta questão, os alunos serão desafiados a utilizar a estrutura de decisão simples (if, else) para verificar se um objeto em MRU está se movendo para frente, para trás ou está parado.

Enunciado: Escreva um programa em Python que, com base na velocidade (v) de um objeto em MRU, determine se ele está se movendo para frente ($v > 0$), para trás ($v < 0$) ou está parado ($v = 0$). O programa deve imprimir a mensagem correspondente à situação do objeto.

Exemplos:

$v = 10$

Saída esperada: O objeto está se movendo para frente.

$v = -5$

Saída esperada: O objeto está se movendo para trás.

$v = 0$

Saída esperada: O objeto está parado.

Solução:

```
```python
Entrada da velocidade
v = float(input("Digite a velocidade (v): "))

Verificação da direção do movimento
if v > 0:
```

```

 print("O objeto está se movendo para frente.")
 elif v < 0:
 print("O objeto está se movendo para trás.")
 else:
 print("O objeto está parado.")
 ...

```

Usuário: Contexto: Assuma o papel de um professor da matéria de programação de computadores com Python, que usa os documentos fornecidos como base para ministrar as aulas e para criação de exercícios usados nas atividades dos alunos. As aulas são para turmas de cursos de graduação de diversas áreas de engenharia.

Matéria: Mecânica.

Assuntos da Matéria abordados: Dinâmica das máquinas, Sistemas de controle.

Instrução: Com base nos documentos fornecidos, crie uma prova avaliativa usando os tópicos de programação usados em "Resumo da matéria (PT1)" utilizando os <Assuntos da matéria> da disciplina de <Matéria>. Classifique as questões usando um medidor de dificuldade de 1 à 10, onde 1 indica a questão mais fácil e 10 a mais complexa. A prova deverá ter 5 questões. 3 questões deverão ter dificuldade acima de 3 e 2 questões dificuldade acima de 7. Em pelo menos uma questão, use mais do que um tópico de programação.

Use o seguinte formato de resposta:

<matéria> - <assunto da matéria abordado>

Questão <número da questão> - <tópicos usados> - <dificuldade da questão>

Contextualização e introdução sucinta sobre <assunto da matéria abordado>, tal qual explicações prévias dos recursos utilizados no desenvolvimento da questão.

Enunciado da Questão.

Exemplos:

Três exemplos de execução e da saída que se espera do programa.

Solução:

Uma solução possível.

ChatPDF: Mecânica - Movimento Retilíneo Uniforme (MRU)

Questão 1 - Variáveis e Cálculos Simples - Dificuldade: 2

Contextualização: O Movimento Retilíneo Uniforme (MRU) é um dos conceitos fundamentais da cinemática, que estuda o movimento dos corpos sem se preocupar com as causas que o geram. Nesta questão, os alunos serão desafiados a utilizar variáveis e realizar cálculos simples para determinar a posição final de um objeto em MRU.

Enunciado: Crie um programa em Python que solicite ao usuário a posição inicial ( $S_i$ ), a velocidade ( $v$ ) e o instante de tempo ( $t$ ) de um objeto em MRU. Em seguida, calcule e imprima a posição final do objeto utilizando a fórmula do MRU:  $S = S_i + v * t$ .

Exemplos:

Informe a posição inicial ( $S_i$ ): 10

Informe a velocidade ( $v$ ): 2

Informe o instante de tempo ( $t$ ): 5

Saída esperada: A posição final no tempo  $t = 5.00$  será  $S = 20.00$

Solução:

```

```python
Si = float(input('Informe a posição inicial (Si): '))
v = float(input('Informe a velocidade (v): '))
t = float(input('Informe o instante de tempo (t): '))
S = Si + v * t
print(f'A posição final no tempo t = {t:.2f} será S = {S:.2f}')
```

```

Mecânica - Movimento Retilíneo Uniforme (MRU)

Questão 2 - Estrutura de Decisão - Dificuldade: 4

Contextualização: A estrutura de decisão (if, else) é fundamental para controlar o fluxo de um programa, permitindo que diferentes ações sejam executadas com base em condições específicas. Nesta questão, os alunos serão desafiados a utilizar a estrutura de decisão para determinar a direção do movimento de um objeto em MRU.

Enunciado: Escreva um programa em Python que, com base na velocidade (v) de um objeto em MRU, determine se ele está se movendo para frente ( $v > 0$ ), para trás ( $v < 0$ ) ou está parado ( $v = 0$ ). O programa deve imprimir a mensagem correspondente à situação do objeto.

Exemplos:

Informe a velocidade (v): 10

Saída esperada: O objeto está se movendo para frente.

Solução:

```

```python
v = float(input('Informe a velocidade (v): '))
if v > 0:
    print('O objeto está se movendo para frente.')
elif v < 0:
    print('O objeto está se movendo para trás.')
else:
    print('O objeto está parado.')
```

```

Mecânica - Movimento Retilíneo Uniforme (MRU)

Questão 3 - Matrizes e Estruturas de Repetição - Dificuldade: 6

Contextualização: O uso de matrizes e estruturas de repetição é comum em diversas áreas da engenharia, permitindo a manipulação eficiente de conjuntos de dados. Nesta questão, os alunos serão desafiados a utilizar matrizes e estruturas de repetição para realizar cálculos relacionados ao MRU.

Enunciado: Implemente um programa em Python que receba do usuário os valores de entrada em uma matriz, onde: a primeira linha armazena as posições iniciais (Si); a segunda linha armazena as velocidades (V); e a terceira linha armazena os instantes de tempo (T). O programa deve calcular as posições finais (Sx) e imprimir os resultados no terminal com uma precisão de 2 casas decimais. Caso a matriz não possua exatamente 3 linhas, o programa deve imprimir uma mensagem de entrada inválida.

Exemplo de Execução:

Matriz de valores: 15, 10.5, 6.33; 5, 15, 21.8; 1, 5, 10

Saída esperada:

S[0] = 20.00

S[1] = 85.50

S[2] = 224.33

Solução:

```

```python
from biblioteca import *
M = inputMatriz('Matriz de valores: ', float)
nL, nC = dimMatriz(M)
if nL == 3:
    for i in range(nC):
        Sx = M[0][i] + M[1][i] * M[2][i]
        print(f'S[{i}] = {Sx:.2f}')
else:
    print('Entrada inválida')
```

```

Mecânica - Movimento Retilíneo Uniforme (MRU)

Questão 4 - Funções - Dificuldade: 8

Contextualização: O uso de funções é essencial para organizar e reutilizar código em programas mais complexos. Nesta questão, os alunos serão desafiados a criar e utilizar funções para realizar cálculos relacionados ao MRU.

Enunciado: Escreva um programa em Python que defina uma função para calcular a posição final (S) de um objeto em MRU, dado a posição inicial (Si), a velocidade (v) e o instante de tempo (t). Em seguida, utilize a função para calcular e imprimir a posição final do objeto com uma precisão de 2 casas decimais.

Exemplos:

Informe a posição inicial (Si): 10

Informe a velocidade (v): 2

Informe o instante de tempo (t): 5

Saída esperada: A posição final no tempo t = 5.00 será S = 20.00

Solução:

```

```python
def calcular_posicao_final(Si, v, t):
    return Si + v * t

Si = float(input('Informe a posição inicial (Si): '))
v = float(input('Informe a velocidade (v): '))
t = float(input('Informe o instante de tempo (t): '))
S = calcular_posicao_final(Si, v, t)
print(f'A posição final no tempo t = {t:.2f} será S = {S:.2f}')
```

```

Mecânica - Movimento Retilíneo Uniforme (MRU)

Questão 5 - Biblioteca Math - Dificuldade: 9

Contextualização: A biblioteca math fornece funções matemáticas avançadas que podem ser úteis em diversas aplicações, incluindo a resolução de problemas relacionados à engenharia. Nesta questão, os alunos serão desafiados a utilizar funções da biblioteca math para realizar cálculos relacionados ao MRU.

Enunciado: Implemente um programa em Python que utilize funções da biblioteca math para calcular a posição final (S) de um objeto em MRU, dado a posição inicial (Si), a velocidade (v) e o instante de tempo (t). O programa deve calcular e imprimir a posição final do objeto com uma precisão de 2 casas decimais.

Exemplos:

Informe a posição inicial (Si): 10

```
Informe a velocidade (v): 2
Informe o instante de tempo (t): 5
Saída esperada: A posição final no tempo t = 5.00 será S = 20.00
Solução:
```python
import math

Si = float(input('Informe a posição inicial (Si): '))
v = float(input('Informe a velocidade (v): '))
t = float(input('Informe o instante de tempo (t): '))
S = Si + v * t
print(f'A posição final no tempo t = {t:.2f} será S = {S:.2f}')
```
```

## APÊNDICE F – Testes PopAi

**PopAI:** O documento apresenta uma série de conceitos e exemplos relacionados ao movimento uniformemente variado (MUV) e programação em Python. Inicia-se com uma explicação sobre os comandos de atribuição, operadores matemáticos, funções matemáticas elementares, valores constantes, importação de bibliotecas, entrada e saída de dados pelo terminal, strings de formatação, valores lógicos, operadores relacionais, operadores lógicos, comandos de decisão e de repetição, definição e chamada de funções, estruturas homogêneas (vetores e matrizes) e funções da biblioteca padrão. Em seguida, são apresentadas diversas questões práticas que envolvem a implementação de programas em Python para cálculo de posição final de um objeto em movimento uniforme, utilizando estruturas de decisão, repetição, funções e manipulação de vetores e matrizes. As soluções são apresentadas de forma clara e concisa, mostrando como cada problema pode ser resolvido por meio de um programa Python.

Além disso, é abordada a implementação de um programa que utiliza uma variável heterogênea para armazenar os valores de posição inicial, velocidade e instante de tempo, e realiza o cálculo da posição final. O programa é estruturado para permitir que o usuário faça múltiplos cálculos, se desejar.

Em resumo, o documento fornece uma visão abrangente dos conceitos fundamentais do movimento uniformemente variado e da aplicação prática desses conceitos na resolução de problemas de programação em Python, demonstrando a integração entre a física e a computação. As soluções apresentadas ilustram a aplicação dos conceitos de programação de forma clara e didática, fornecendo um material útil para estudantes e profissionais interessados em aprender e aprimorar suas habilidades em Python.

**Usuário:** Contexto: Assuma o papel de um professor da matéria de programação de computadores com Python, que usa os documentos fornecidos como base para ministrar as aulas e para criação de exercícios usados nas atividades dos alunos. As aulas são para turmas de cursos de graduação de diversas áreas de engenharia.

**Matéria:** Física.

**Assuntos da Matéria abordados:** MRU.

**Tópicos de programação abordados:** Variáveis, expressões, entradas e saídas;

Decisão simples (if, else);

Decisão complexa (if, elif, else);

Repetição e Laços;

Laços aninhados;

Funções;

Estrutura de dados homogêneos com vetores;

Estrutura de dados homogêneos com matrizes;

Estrutura de dados heterogêneos (registros).

**Instrução:** Com base nos documentos fornecidos, crie questões de programação para cada <Tópicos de programação> utilizando os <Assuntos da matéria> da disciplina de <Matéria>. Garanta que cada questão aborde um assunto específico e que todos os tópicos de programação listados sejam contemplados.

Classifique as questões usando um medidor de dificuldade de 1 à 10, onde 1 indica a questão mais fácil e 10 a mais complexa.

Use o seguinte formato de resposta:

<matéria> - <assunto da matéria abordado>

Questão <número da questão> - <tópico usado> - <dificuldade da questão>

Contextualização e introdução sucinta sobre <assunto da matéria abordado>, tal qual explicações prévias dos recursos utilizados no desenvolvimento da questão.

Enunciado da Questão.

Exemplos:

Três exemplos de execução e da saída que se espera do programa.

Solução:

Uma solução possível.

PopAI: Física - MRU

Questão 1 - Variáveis, expressões, entradas e saídas - Dificuldade 2

Nesta questão, os alunos serão desafiados a implementar um programa que recebe valores de entrada do usuário, armazena em variáveis e calcula a posição final de um objeto em movimento uniforme. Eles devem imprimir o resultado no terminal com uma precisão de 2 casas decimais.

Enunciado:

Implemente um programa que solicite os valores de entrada aos usuários e os armazene nas variáveis: posição inicial ( $S_i$ ), a velocidade ( $v$ ) e o instante de tempo ( $t$ ), respectivamente. Todas as entradas são números reais. O programa deve realizar o cálculo da posição final ( $S$ ) e imprimir o resultado no terminal com uma precisão de 2 casas decimais.

Exemplo de Execução:

Informe a posição inicial ( $S_i$ ): 10

Informe a velocidade ( $v$ ): 1

Informe o instante de tempo ( $t$ ): 5

A posição final no tempo  $t = 5.00$  será  $S = 15.00$

Solução:

python

Copy code

```
Si = float(input("Informe a posição inicial (Si): "))
```

```
v = float(input("Informe a velocidade (v): "))
```

```
t = float(input("Informe o instante de tempo (t): "))
```

```
S = Si + v * t
```

```
print(f"A posição final no tempo t = {t:.2f} será S = {S:.2f}")
```

Esta questão permite que os alunos pratiquem a utilização de variáveis, entrada de dados pelo terminal, cálculos matemáticos e saída formatada.

Questão 2 - Decisão simples (if, else) - Dificuldade 3

Nesta questão, os alunos serão desafiados a implementar um programa que realiza o cálculo da posição final de um objeto em movimento uniforme, com a adição de validações para as entradas fornecidas pelo usuário.

Enunciado:



Implemente um programa que receba do usuário os valores de entrada, e as armazene em variáveis: posição inicial ( $S_i$ ), a velocidade ( $v$ ) e o instante de tempo ( $t$ ), respectivamente. Todas as entradas são números reais maiores do que 0 (zero). Caso uma entrada seja inválida, o programa apenas imprime uma mensagem de erro específica e é encerrado, caso contrário, o programa deve realizar o cálculo da posição final ( $S$ ) e imprimir o resultado no terminal com uma precisão de 2 casas decimais.

Exemplo de Execução:

```
Informe a posição inicial (S_i): 0
Informe a velocidade (v): 1
Informe o instante de tempo (t): 5
Erro: A posição inicial deve ser maior que 0.
```

python

Copy code

```
Si = float(input("Informe a posição inicial (S_i): "))
v = float(input("Informe a velocidade (v): "))
t = float(input("Informe o instante de tempo (t): "))
```

```
if Si <= 0:
 print("Erro: A posição inicial deve ser maior que 0.")
else:
 S = Si + v * t
```

```
 print(f"A posição final no tempo $t = \{t:.2f\}$ será $S = \{S:.2f\}$ ")
```

Nesta questão, os alunos terão a oportunidade de praticar a utilização de estruturas de decisão simples para validar as entradas fornecidas pelo usuário.

Questão 3 - Decisão complexa (if, elif, else) - Dificuldade 4

Nesta questão, os alunos serão desafiados a implementar um programa que realiza o cálculo da posição final de um objeto em movimento uniforme, com a adição de validações para as entradas fornecidas pelo usuário, abordando múltiplas condições.

Enunciado:

Implemente um programa que receba do usuário os valores de entrada, e as armazene em variáveis: posição inicial ( $S_i$ ), a velocidade ( $v$ ) e o instante de tempo ( $t$ ), respectivamente. Todas as entradas são números reais maiores do que 0 (zero), à exceção de  $S_i$ , que pode ser igual a 0 (zero). Caso uma entrada seja inválida, o programa apenas imprime uma mensagem de erro específica e é encerrado, caso contrário, o programa deve realizar o cálculo da posição final ( $S$ ) e imprimir o resultado no terminal com uma precisão de 2 casas decimais.

Exemplo de Execução:

```
Informe a posição inicial (S_i): 0
Informe a velocidade (v): 1
Informe o instante de tempo (t): 5
Erro: A posição inicial deve ser maior ou igual a 0.
```

python

Copy code

```
Si = float(input("Informe a posição inicial (S_i): "))
```

```

v = float(input("Informe a velocidade (v): "))
t = float(input("Informe o instante de tempo (t): "))

if Si < 0:
 print("Erro: A posição inicial deve ser maior ou igual a 0.")
elif v <= 0:
 print("Erro: A velocidade deve ser maior que 0.")
elif t <= 0:
 print("Erro: O instante de tempo deve ser maior que 0.")
else:
 S = Si + v * t
 print(f"A posição final no tempo t = {t:.2f} será S = {S:.2f}")

```

Nesta questão, os alunos terão a oportunidade de praticar a utilização de estruturas de decisão complexas para validar as entradas fornecidas pelo usuário, abordando múltiplas condições.

#### Questão 4 - Repetição e Laços - Dificuldade 5

Nesta questão, os alunos serão desafiados a implementar um programa que realiza o cálculo da posição final de um objeto em movimento uniforme para todos os instantes  $t$  variando entre 1 e  $T_f$ , com uma precisão de 2 casas decimais.

##### Enunciado:

Implemente um programa que receba do usuário os valores de entrada, e as armazene em variáveis: posição inicial ( $S_i$ ), a velocidade ( $v$ ) e o instante de tempo final ( $T_f$ ), respectivamente.  $S_i$  e  $v$  são valores reais, e  $T_f$  é um valor inteiro. O programa deve realizar o cálculo da posição final ( $S$ ) e imprimir o resultado no terminal para todos os instantes  $t$  variando entre 1 e  $T_f$ , com uma precisão de 2 casas decimais.

##### Exemplo de Execução:

```

Informe a posição inicial (Si): 10
Informe a velocidade (v): 1
Informe o instante de tempo final (Tf): 5
A posição final no tempo t = 1.00 será S = 11.00
A posição final no tempo t = 2.00 será S = 12.00
A posição final no tempo t = 3.00 será S = 13.00
A posição final no tempo t = 4.00 será S = 14.00
A posição final no tempo t = 5.00 será S = 15.00

```

python

Copy code

```

Si = float(input("Informe a posição inicial (Si): "))
v = float(input("Informe a velocidade (v): "))
Tf = int(input("Informe o instante de tempo final (Tf): "))

for t in range(1, Tf+1):
 S = Si + v * t
 print(f"A posição final no tempo t = {t:.2f} será S = {S:.2f}")

```

Nesta questão, os alunos terão a oportunidade de praticar a utilização de estruturas de repetição para realizar cálculos em um intervalo de tempo variável.

#### Questão 5 - Laços aninhados - Dificuldade 6

Nesta questão, os alunos serão desafiados a implementar um programa que realiza o cálculo da posição final de um objeto em movimento uniforme para todas as combinações de velocidades  $v$  variando entre 1 e  $V_f$  e todos os instantes  $t$  variando entre 1 e  $T_f$ , com uma precisão de 2 casas decimais.

Enunciado:

Implemente um programa que receba do usuário os valores de entrada, e as armazene em variáveis: posição inicial ( $S_i$ ), a velocidade final ( $V_f$ ) e o instante de tempo final ( $T_f$ ), respectivamente.  $S_i$  é um valor real,  $V_f$  e  $T_f$  são valores inteiros. O programa deve realizar o cálculo da posição final ( $S$ ) e imprimir o resultado no terminal com pelo menos 10 caracteres e uma precisão de 2 casas decimais para a combinação de todas as velocidades  $v$  variando entre 1 e  $V_f$  e todos os instantes  $t$  variando entre 1 e  $T_f$ .

Exemplo de Execução:

Informe a posição inicial ( $S_i$ ): 10  
Informe a velocidade final ( $V_f$ ): 3  
Informe o instante de tempo final ( $T_f$ ): 2  
 $t = 1$ ,  $v = 1$ ,  $S =$

Usuário: Contexto: Assuma o papel de um professor da matéria de programação de computadores com Python, que usa os documentos fornecidos como base para ministrar as aulas e para criação de exercícios usados nas atividades dos alunos. As aulas são para turmas de cursos de graduação de diversas áreas de engenharia.

Matéria: Mecânica.

Assuntos da Matéria abordados: Dinâmica das máquinas, Sistemas de controle.

Instrução: Com base nos documentos fornecidos, crie uma prova avaliativa usando os tópicos de programação usados em "Resumo da matéria (PT1)" utilizando os <Assuntos da matéria> da disciplina de <Matéria>. Classifique as questões usando um medidor de dificuldade de 1 à 10, onde 1 indica a questão mais fácil e 10 a mais complexa. A prova deverá ter 5 questões. 3 questões deverão ter dificuldade acima de 3 e 2 questões dificuldade acima de 7. Em pelo menos uma questão, use mais do que um tópico de programação.

Use o seguinte formato de resposta:

<matéria> - <assunto da matéria abordado>

Questão <número da questão> - <tópicos usados> - <dificuldade da questão>

Contextualização e introdução sucinta sobre <assunto da matéria abordado>, tal qual explicações prévias dos recursos utilizados no desenvolvimento da questão.

Enunciado da Questão.

Exemplos:

Três exemplos de execução e da saída que se espera do programa.

Solução:

Uma solução possível.

PopAI: Mecânica - Dinâmica das máquinas

Questão 1 - Expressões e IO - Dificuldade 2

Nesta questão, os alunos serão desafiados a implementar um programa que recebe valores de entrada do usuário, armazena em variáveis e calcula a posição final de um objeto em movimento uniforme. Eles devem imprimir o resultado no terminal com uma precisão de 2 casas decimais.

Enunciado:

Implemente um programa que solicite os valores de entrada aos usuários e os armazene nas variáveis: posição inicial ( $S_i$ ), a velocidade ( $v$ ) e o instante de tempo ( $t$ ), respectivamente. Todas as entradas são números reais. O programa deve realizar o cálculo da posição final ( $S$ ) e imprimir o resultado no terminal com uma precisão de 2 casas decimais.

Exemplo de Execução:

Informe a posição inicial ( $S_i$ ): 10

Informe a velocidade ( $v$ ): 1

Informe o instante de tempo ( $t$ ): 5

A posição final no tempo  $t = 5.00$  será  $S = 15.00$

Solução:

python

Copy code

```
Si = float(input("Informe a posição inicial (Si): "))
```

```
v = float(input("Informe a velocidade (v): "))
```

```
t = float(input("Informe o instante de tempo (t): "))
```

```
S = Si + v * t
```

```
print(f"A posição final no tempo t = {t:.2f} será S = {S:.2f}")
```

Esta questão permite que os alunos pratiquem a utilização de variáveis, entrada de dados pelo terminal, cálculos matemáticos e saída formatada.

Questão 2 - Decisão simples - Dificuldade 3

Nesta questão, os alunos serão desafiados a implementar um programa que realiza o cálculo da posição final de um objeto em movimento uniforme, com a adição de validações para as entradas fornecidas pelo usuário.

Enunciado:

Implemente um programa que receba do usuário os valores de entrada, e as armazene em variáveis: posição inicial ( $S_i$ ), a velocidade ( $v$ ) e o instante de tempo ( $t$ ), respectivamente. Todas as entradas são números reais maiores do que 0 (zero). Caso uma entrada seja inválida, o programa apenas imprime uma mensagem de erro específica e é encerrado, caso contrário, o programa deve realizar o cálculo da posição final ( $S$ ) e imprimir o resultado no terminal com uma precisão de 2 casas decimais.

Exemplo de Execução:

Informe a posição inicial ( $S_i$ ): 0

Informe a velocidade ( $v$ ): 1

Informe o instante de tempo ( $t$ ): 5

Erro: A posição inicial deve ser maior que 0.

python

Copy code

```
Si = float(input("Informe a posição inicial (Si): "))
```

```
v = float(input("Informe a velocidade (v): "))
```

```
t = float(input("Informe o instante de tempo (t): "))
```

```
if Si <= 0:
```

```
 print("Erro: A posição inicial deve ser maior que 0.")
```

```
else:
```

```
 S = Si + v * t
```

```
 print(f"A posição final no tempo t = {t:.2f} será S = {S:.2f}")
```

Nesta questão, os alunos terão a oportunidade de praticar a utilização de estruturas de decisão simples para validar as entradas fornecidas pelo usuário.

### Questão 3 - Repetição for - Dificuldade 5

Nesta questão, os alunos serão desafiados a implementar um programa que realiza o cálculo da posição final de um objeto em movimento uniforme para todos os instantes t variando entre 1 e Tf, com uma precisão de 2 casas decimais.

Enunciado:

Implemente um programa que receba do usuário os valores de entrada, e as armazene em variáveis: posição inicial (Si), a velocidade (v) e o instante de tempo final (Tf), respectivamente. Si e v são valores reais, e Tf é um valor inteiro. O programa deve realizar o cálculo da posição final (S) e imprimir o resultado no terminal para todos os instantes t variando entre 1 e Tf, com uma precisão de 2 casas decimais.

Exemplo de Execução:

Informe a posição inicial (Si): 10

Informe a velocidade (v): 1

Informe o instante de tempo final (Tf): 5

A posição final no tempo t = 1.00 será S = 11.00

A posição final no tempo t = 2.00 será S = 12.00

A posição final no tempo t = 3.00 será S = 13.00

A posição final no tempo t = 4.00 será S = 14.00

A posição final no tempo t = 5.00 será S = 15.00

python

Copy code

```
Si = float(input("Informe a posição inicial (Si): "))
```

```
v = float(input("Informe a velocidade (v): "))
```

```
Tf = int(input("Informe o instante de tempo final (Tf): "))
```

```
for t in range(1, Tf+1):
```

```
 S = Si + v * t
```

```
 print(f"A posição final no tempo t = {t:.2f} será S = {S:.2f}")
```

Nesta questão, os alunos terão a oportunidade de praticar a utilização de estruturas de repetição para realizar cálculos em um intervalo de tempo variável.

### Questão 4 - Laços aninhados - Dificuldade 7

Nesta questão, os alunos serão desafiados a implementar um programa que realiza o cálculo da posição final de um objeto em movimento uniforme para todas as combinações de velocidades v variando entre 1 e Vf e todos os instantes t variando entre 1 e Tf, com uma precisão de 2 casas decimais.

Enunciado:

Implemente um programa que receba do usuário os valores de entrada, e as armazene em variáveis: posição inicial ( $S_i$ ), a velocidade final ( $V_f$ ) e o instante de tempo final ( $T_f$ ), respectivamente.  $S_i$  é um valor real,  $V_f$  e  $T_f$  são valores inteiros. O programa deve realizar o cálculo da posição final ( $S$ ) e imprimir o resultado no terminal com pelo menos 10 caracteres e uma precisão de 2 casas decimais para a combinação de todas as velocidades  $v$  variando entre 1 e  $V_f$  e todos os instantes  $t$  variando entre 1 e  $T_f$ .

Exemplo de Execução:

```
Informe a posição inicial (S_i): 10
Informe a velocidade final (V_f): 3
Informe o instante de tempo final (T_f): 2
t = 1, v = 1, S = 11.00
t = 1, v = 2, S = 12.00
t = 1, v = 3, S = 13.00
t = 2, v = 1, S = 12.00
t = 2, v = 2, S = 14.00
t = 2, v = 3, S = 16.00
```

python

Copy code

```
 S_i = float(input("Informe a posição inicial (S_i): "))
 V_f = int(input("Informe a velocidade final (V_f): "))
 T_f = int(input("Informe o instante de tempo final (T_f): "))
```

```
for t in range(1, T_f +1):
 for v in range(1, V_f +1):
 S = S_i + v * t
 print(f"t = {t}, v = {v}, S = { S :.2f}")
```

Nesta questão, os alunos terão a oportunidade de praticar a utilização de laços aninhados para calcular e exibir os resultados de todas as combinações de velocidades e instantes de tempo.

Questão 5 - ED Heterogêneas e Funções - Dificuldade 8

Nesta questão, os alunos serão desafiados a implementar um programa que realiza o cálculo da posição final de um objeto em movimento uniforme, armazenando os valores de entrada em uma variável heterogênea e implementando uma função para o cálculo de  $S$ .

Enunciado:

Implemente um programa que receba do usuário os valores de entrada, e as armazene em variáveis: posição inicial ( $S_i$ ), a velocidade ( $v$ ) e o instante de tempo ( $t$ ), respectivamente. Estes valores devem ser armazenados em uma variável heterogênea (MU) com campos que representam os três valores. O programa deve realizar o cálculo da posição final ( $S$ ) e imprimir o resultado no terminal com uma precisão de 2 casas decimais. Para o cálculo de  $S$  você deverá implementar a função `calculaS`, que recebe o valor heterogêneo (MU) como argumento de entrada, calcula e retorna o valor de  $S$ , arredondado para 3 casas

PopAI: O documento apresenta uma série de conceitos e exemplos relacionados ao movimento uniformemente variado (MUV) e programação em Python. Inicia-se com uma explicação sobre os comandos de atribuição, operadores matemáticos, funções matemáticas elementares, valores constantes, importação de bibliotecas, entrada e saída de dados pelo terminal, strings de formatação, valores lógicos, operadores relacionais, operadores lógicos, comandos de decisão e de repetição, definição e chamada de funções, estruturas homogêneas (vetores e matrizes) e funções da biblioteca padrão. Em seguida, são apresentadas diversas questões práticas que envolvem a implementação de programas em Python para cálculo de posição final de um objeto em movimento uniforme, utilizando estruturas de decisão, repetição, funções e manipulação de vetores e matrizes. As soluções são apresentadas de forma clara e concisa, mostrando como cada problema pode ser resolvido por meio de um programa Python.

Além disso, é abordada a implementação de um programa que utiliza uma variável heterogênea para armazenar os valores de posição inicial, velocidade e instante de tempo, e realiza o cálculo da posição final. O programa é estruturado para permitir que o usuário faça múltiplos cálculos, se desejar.

Em resumo, o documento fornece uma visão abrangente dos conceitos fundamentais do movimento uniformemente variado e da aplicação prática desses conceitos na resolução de problemas de programação em Python, demonstrando a integração entre a física e a computação. As soluções apresentadas ilustram a aplicação dos conceitos de programação de forma clara e didática, fornecendo um material útil para estudantes e profissionais interessados em aprender e aprimorar suas habilidades em Python.

Usuário: Contexto: Assuma o papel de um professor da matéria de programação de computadores com Python, que usa os documentos fornecidos como base para ministrar as aulas e para criação de exercícios usados nas atividades dos alunos. As aulas são para turmas de cursos de graduação de diversas áreas de engenharia.

Matéria: Física.

Assuntos da Matéria abordados: MRU.

Tópicos de programação abordados: Variáveis, expressões, entradas e saídas;

Decisão simples (if, else);

Decisão complexa (if, elif, else);

Repetição e Laços;

Laços aninhados;

Funções;

Estrutura de dados homogêneos com vetores;

Estrutura de dados homogêneos com matrizes;

Estrutura de dados heterogêneos (registros).

Instrução: Com base nos documentos fornecidos, crie questões de programação para cada <Tópicos de programação> utilizando os <Assuntos da matéria> da disciplina de <Matéria>. Garanta que cada questão aborde um assunto específico e que todos os tópicos de programação listados sejam contemplados.

Classifique as questões usando um medidor de dificuldade de 1 à 10, onde 1 indica a questão mais fácil e 10 a mais complexa.

Use o seguinte formato de resposta:

<matéria> - <assunto da matéria abordado>

Questão <número da questão> - <tópico usado> - <dificuldade da questão>

Contextualização e introdução sucinta sobre <assunto da matéria abordado>, tal qual explicações prévias dos recursos utilizados no desenvolvimento da questão.

Enunciado da Questão.

Exemplos:

Três exemplos de execução e da saída que se espera do programa.

Solução:

Uma solução possível.

PopAI: Física - MRU

Questão 1 - Variáveis, expressões, entradas e saídas - Dificuldade 2

Nesta questão, os alunos serão desafiados a implementar um programa que recebe valores de entrada do usuário, armazena em variáveis e calcula a posição final de um objeto em movimento uniforme. Eles devem imprimir o resultado no terminal com uma precisão de 2 casas decimais.

Enunciado:

Implemente um programa que solicite os valores de entrada aos usuários e os armazene nas variáveis: posição inicial ( $S_i$ ), a velocidade ( $v$ ) e o instante de tempo ( $t$ ), respectivamente. Todas as entradas são números reais. O programa deve realizar o cálculo da posição final ( $S$ ) e imprimir o resultado no terminal com uma precisão de 2 casas decimais.

Exemplo de Execução:

Informe a posição inicial ( $S_i$ ): 10

Informe a velocidade ( $v$ ): 1

Informe o instante de tempo ( $t$ ): 5

A posição final no tempo  $t = 5.00$  será  $S = 15.00$

Solução:

python

Copy code

```
Si = float(input("Informe a posição inicial (Si): "))
```

```
v = float(input("Informe a velocidade (v): "))
```

```
t = float(input("Informe o instante de tempo (t): "))
```

```
S = Si + v * t
```

```
print(f"A posição final no tempo t = {t:.2f} será S = {S:.2f}")
```

Esta questão permite que os alunos pratiquem a utilização de variáveis, entrada de dados pelo terminal, cálculos matemáticos e saída formatada.

Questão 2 - Decisão simples (if, else) - Dificuldade 3

Nesta questão, os alunos serão desafiados a implementar um programa que realiza o cálculo da posição final de um objeto em movimento uniforme, com a adição de validações para as entradas fornecidas pelo usuário.

Enunciado:



Implemente um programa que receba do usuário os valores de entrada, e as armazene em variáveis: posição inicial ( $S_i$ ), a velocidade ( $v$ ) e o instante de tempo ( $t$ ), respectivamente. Todas as entradas são números reais maiores do que 0 (zero). Caso uma entrada seja inválida, o programa apenas imprime uma mensagem de erro específica e é encerrado, caso contrário, o programa deve realizar o cálculo da posição final ( $S$ ) e imprimir o resultado no terminal com uma precisão de 2 casas decimais.

Exemplo de Execução:

```
Informe a posição inicial (S_i): 0
Informe a velocidade (v): 1
Informe o instante de tempo (t): 5
Erro: A posição inicial deve ser maior que 0.
```

python

Copy code

```
Si = float(input("Informe a posição inicial (S_i): "))
v = float(input("Informe a velocidade (v): "))
t = float(input("Informe o instante de tempo (t): "))
```

```
if Si <= 0:
```

```
 print("Erro: A posição inicial deve ser maior que 0.")
```

```
else:
```

```
 S = Si + v * t
```

```
 print(f"A posição final no tempo $t = \{t:.2f\}$ será $S = \{S:.2f\}$ ")
```

Nesta questão, os alunos terão a oportunidade de praticar a utilização de estruturas de decisão simples para validar as entradas fornecidas pelo usuário.

Questão 3 - Decisão complexa (if, elif, else) - Dificuldade 4

Nesta questão, os alunos serão desafiados a implementar um programa que realiza o cálculo da posição final de um objeto em movimento uniforme, com a adição de validações para as entradas fornecidas pelo usuário, abordando múltiplas condições.

Enunciado:

Implemente um programa que receba do usuário os valores de entrada, e as armazene em variáveis: posição inicial ( $S_i$ ), a velocidade ( $v$ ) e o instante de tempo ( $t$ ), respectivamente. Todas as entradas são números reais maiores do que 0 (zero), à exceção de  $S_i$ , que pode ser igual a 0 (zero). Caso uma entrada seja inválida, o programa apenas imprime uma mensagem de erro específica e é encerrado, caso contrário, o programa deve realizar o cálculo da posição final ( $S$ ) e imprimir o resultado no terminal com uma precisão de 2 casas decimais.

Exemplo de Execução:

```
Informe a posição inicial (S_i): 0
Informe a velocidade (v): 1
Informe o instante de tempo (t): 5
Erro: A posição inicial deve ser maior ou igual a 0.
```

python

Copy code

```
Si = float(input("Informe a posição inicial (S_i): "))
```

```

v = float(input("Informe a velocidade (v): "))
t = float(input("Informe o instante de tempo (t): "))

if Si < 0:
 print("Erro: A posição inicial deve ser maior ou igual a 0.")
elif v <= 0:
 print("Erro: A velocidade deve ser maior que 0.")
elif t <= 0:
 print("Erro: O instante de tempo deve ser maior que 0.")
else:
 S = Si + v * t
 print(f"A posição final no tempo t = {t:.2f} será S = {S:.2f}")

```

Nesta questão, os alunos terão a oportunidade de praticar a utilização de estruturas de decisão complexas para validar as entradas fornecidas pelo usuário, abordando múltiplas condições.

#### Questão 4 - Repetição e Laços - Dificuldade 5

Nesta questão, os alunos serão desafiados a implementar um programa que realiza o cálculo da posição final de um objeto em movimento uniforme para todos os instantes  $t$  variando entre 1 e  $T_f$ , com uma precisão de 2 casas decimais.

##### Enunciado:

Implemente um programa que receba do usuário os valores de entrada, e as armazene em variáveis: posição inicial ( $S_i$ ), a velocidade ( $v$ ) e o instante de tempo final ( $T_f$ ), respectivamente.  $S_i$  e  $v$  são valores reais, e  $T_f$  é um valor inteiro. O programa deve realizar o cálculo da posição final ( $S$ ) e imprimir o resultado no terminal para todos os instantes  $t$  variando entre 1 e  $T_f$ , com uma precisão de 2 casas decimais.

##### Exemplo de Execução:

```

Informe a posição inicial (Si): 10
Informe a velocidade (v): 1
Informe o instante de tempo final (Tf): 5
A posição final no tempo t = 1.00 será S = 11.00
A posição final no tempo t = 2.00 será S = 12.00
A posição final no tempo t = 3.00 será S = 13.00
A posição final no tempo t = 4.00 será S = 14.00
A posição final no tempo t = 5.00 será S = 15.00

```

python

Copy code

```

Si = float(input("Informe a posição inicial (Si): "))
v = float(input("Informe a velocidade (v): "))
Tf = int(input("Informe o instante de tempo final (Tf): "))

for t in range(1, Tf+1):
 S = Si + v * t
 print(f"A posição final no tempo t = {t:.2f} será S = {S:.2f}")

```

Nesta questão, os alunos terão a oportunidade de praticar a utilização de estruturas de repetição para realizar cálculos em um intervalo de tempo variável.

#### Questão 5 - Laços aninhados - Dificuldade 6

Nesta questão, os alunos serão desafiados a implementar um programa que realiza o cálculo da posição final de um objeto em movimento uniforme para todas as combinações de velocidades  $v$  variando entre 1 e  $V_f$  e todos os instantes  $t$  variando entre 1 e  $T_f$ , com uma precisão de 2 casas decimais.

Enunciado:

Implemente um programa que receba do usuário os valores de entrada, e as armazene em variáveis: posição inicial ( $S_i$ ), a velocidade final ( $V_f$ ) e o instante de tempo final ( $T_f$ ), respectivamente.  $S_i$  é um valor real,  $V_f$  e  $T_f$  são valores inteiros. O programa deve realizar o cálculo da posição final ( $S$ ) e imprimir o resultado no terminal com pelo menos 10 caracteres e uma precisão de 2 casas decimais para a combinação de todas as velocidades  $v$  variando entre 1 e  $V_f$  e todos os instantes  $t$  variando entre 1 e  $T_f$ .

Exemplo de Execução:

Informe a posição inicial ( $S_i$ ): 10  
Informe a velocidade final ( $V_f$ ): 3  
Informe o instante de tempo final ( $T_f$ ): 2  
 $t = 1$ ,  $v = 1$ ,  $S =$

Usuário: Contexto: Assuma o papel de um professor da matéria de programação de computadores com Python, que usa os documentos fornecidos como base para ministrar as aulas e para criação de exercícios usados nas atividades dos alunos. As aulas são para turmas de cursos de graduação de diversas áreas de engenharia.

Matéria: Mecânica.

Assuntos da Matéria abordados: Dinâmica das máquinas, Sistemas de controle.

Instrução: Com base nos documentos fornecidos, crie uma prova avaliativa usando os tópicos de programação usados em "Resumo da matéria (PT1)" utilizando os <Assuntos da matéria> da disciplina de <Matéria>. Classifique as questões usando um medidor de dificuldade de 1 à 10, onde 1 indica a questão mais fácil e 10 a mais complexa. A prova deverá ter 5 questões. 3 questões deverão ter dificuldade acima de 3 e 2 questões dificuldade acima de 7. Em pelo menos uma questão, use mais do que um tópico de programação.

Use o seguinte formato de resposta:

<matéria> - <assunto da matéria abordado>

Questão <número da questão> - <tópicos usados> - <dificuldade da questão>

Contextualização e introdução sucinta sobre <assunto da matéria abordado>, tal qual explicações prévias dos recursos utilizados no desenvolvimento da questão.

Enunciado da Questão.

Exemplos:

Três exemplos de execução e da saída que se espera do programa.

Solução:

Uma solução possível.

PopAI: Mecânica - Dinâmica das máquinas

Questão 1 - Expressões e IO - Dificuldade 2

Nesta questão, os alunos serão desafiados a implementar um programa que recebe valores de entrada do usuário, armazena em variáveis e calcula a posição final de um objeto em movimento uniforme. Eles devem imprimir o resultado no terminal com uma precisão de 2 casas decimais.

Enunciado:

Implemente um programa que solicite os valores de entrada aos usuários e os armazene nas variáveis: posição inicial ( $S_i$ ), a velocidade ( $v$ ) e o instante de tempo ( $t$ ), respectivamente. Todas as entradas são números reais. O programa deve realizar o cálculo da posição final ( $S$ ) e imprimir o resultado no terminal com uma precisão de 2 casas decimais.

Exemplo de Execução:

```
Informe a posição inicial (S_i): 10
Informe a velocidade (v): 1
Informe o instante de tempo (t): 5
A posição final no tempo $t = 5.00$ será $S = 15.00$
```

Solução:

```
python
Copy code
Si = float(input("Informe a posição inicial (S_i): "))
v = float(input("Informe a velocidade (v): "))
t = float(input("Informe o instante de tempo (t): "))
S = Si + v * t
print(f"A posição final no tempo $t = {t:.2f}$ será $S = {S:.2f}$ ")
Esta questão permite que os alunos pratiquem a utilização de variáveis, entrada de dados pelo terminal, cálculos matemáticos e saída formatada.
```

Questão 2 - Decisão simples - Dificuldade 3

Nesta questão, os alunos serão desafiados a implementar um programa que realiza o cálculo da posição final de um objeto em movimento uniforme, com a adição de validações para as entradas fornecidas pelo usuário.

Enunciado:

Implemente um programa que receba do usuário os valores de entrada, e as armazene em variáveis: posição inicial ( $S_i$ ), a velocidade ( $v$ ) e o instante de tempo ( $t$ ), respectivamente. Todas as entradas são números reais maiores do que 0 (zero). Caso uma entrada seja inválida, o programa apenas imprime uma mensagem de erro específica e é encerrado, caso contrário, o programa deve realizar o cálculo da posição final ( $S$ ) e imprimir o resultado no terminal com uma precisão de 2 casas decimais.

Exemplo de Execução:

```
Informe a posição inicial (S_i): 0
Informe a velocidade (v): 1
Informe o instante de tempo (t): 5
Erro: A posição inicial deve ser maior que 0.
```

```
python
Copy code
Si = float(input("Informe a posição inicial (S_i): "))
v = float(input("Informe a velocidade (v): "))
```

```
t = float(input("Informe o instante de tempo (t): "))
```

```
if Si <= 0:
```

```
 print("Erro: A posição inicial deve ser maior que 0.")
```

```
else:
```

```
 S = Si + v * t
```

```
 print(f"A posição final no tempo t = {t:.2f} será S = {S:.2f}")
```

Nesta questão, os alunos terão a oportunidade de praticar a utilização de estruturas de decisão simples para validar as entradas fornecidas pelo usuário.

### Questão 3 - Repetição for - Dificuldade 5

Nesta questão, os alunos serão desafiados a implementar um programa que realiza o cálculo da posição final de um objeto em movimento uniforme para todos os instantes t variando entre 1 e Tf, com uma precisão de 2 casas decimais.

Enunciado:

Implemente um programa que receba do usuário os valores de entrada, e as armazene em variáveis: posição inicial (Si), a velocidade (v) e o instante de tempo final (Tf), respectivamente. Si e v são valores reais, e Tf é um valor inteiro. O programa deve realizar o cálculo da posição final (S) e imprimir o resultado no terminal para todos os instantes t variando entre 1 e Tf, com uma precisão de 2 casas decimais.

Exemplo de Execução:

Informe a posição inicial (Si): 10

Informe a velocidade (v): 1

Informe o instante de tempo final (Tf): 5

A posição final no tempo t = 1.00 será S = 11.00

A posição final no tempo t = 2.00 será S = 12.00

A posição final no tempo t = 3.00 será S = 13.00

A posição final no tempo t = 4.00 será S = 14.00

A posição final no tempo t = 5.00 será S = 15.00

python

Copy code

```
Si = float(input("Informe a posição inicial (Si): "))
```

```
v = float(input("Informe a velocidade (v): "))
```

```
Tf = int(input("Informe o instante de tempo final (Tf): "))
```

```
for t in range(1, Tf+1):
```

```
 S = Si + v * t
```

```
 print(f"A posição final no tempo t = {t:.2f} será S = {S:.2f}")
```

Nesta questão, os alunos terão a oportunidade de praticar a utilização de estruturas de repetição para realizar cálculos em um intervalo de tempo variável.

### Questão 4 - Laços aninhados - Dificuldade 7

Nesta questão, os alunos serão desafiados a implementar um programa que realiza o cálculo da posição final de um objeto em movimento uniforme para todas as combinações de velocidades v variando entre 1 e Vf e todos os instantes t variando entre 1 e Tf, com uma precisão de 2 casas decimais.

Enunciado:

Implemente um programa que receba do usuário os valores de entrada, e as armazene em variáveis: posição inicial ( $S_i$ ), a velocidade final ( $V_f$ ) e o instante de tempo final ( $T_f$ ), respectivamente.  $S_i$  é um valor real,  $V_f$  e  $T_f$  são valores inteiros. O programa deve realizar o cálculo da posição final ( $S$ ) e imprimir o resultado no terminal com pelo menos 10 caracteres e uma precisão de 2 casas decimais para a combinação de todas as velocidades  $v$  variando entre 1 e  $V_f$  e todos os instantes  $t$  variando entre 1 e  $T_f$ .

Exemplo de Execução:

```
Informe a posição inicial (S_i): 10
Informe a velocidade final (V_f): 3
Informe o instante de tempo final (T_f): 2
t = 1, v = 1, S = 11.00
t = 1, v = 2, S = 12.00
t = 1, v = 3, S = 13.00
t = 2, v = 1, S = 12.00
t = 2, v = 2, S = 14.00
t = 2, v = 3, S = 16.00
```

python

Copy code

```
 S_i = float(input("Informe a posição inicial (S_i): "))
 V_f = int(input("Informe a velocidade final (V_f): "))
 T_f = int(input("Informe o instante de tempo final (T_f): "))
```

```
for t in range(1, T_f +1):
 for v in range(1, V_f +1):
 S = S_i + v * t
 print(f"t = {t}, v = {v}, S = { S :.2f}")
```

Nesta questão, os alunos terão a oportunidade de praticar a utilização de laços aninhados para calcular e exibir os resultados de todas as combinações de velocidades e instantes de tempo.

Questão 5 - ED Heterogêneas e Funções - Dificuldade 8

Nesta questão, os alunos serão desafiados a implementar um programa que realiza o cálculo da posição final de um objeto em movimento uniforme, armazenando os valores de entrada em uma variável heterogênea e implementando uma função para o cálculo de  $S$ .

Enunciado:

Implemente um programa que receba do usuário os valores de entrada, e as armazene em variáveis: posição inicial ( $S_i$ ), a velocidade ( $v$ ) e o instante de tempo ( $t$ ), respectivamente. Estes valores devem ser armazenados em uma variável heterogênea (MU) com campos que representam os três valores. O programa deve realizar o cálculo da posição final ( $S$ ) e imprimir o resultado no terminal com uma precisão de 2 casas decimais. Para o cálculo de  $S$  você deverá implementar a função `calculaS`, que recebe o valor heterogêneo (MU) como argumento de entrada, calcula e retorna o valor de  $S$ , arredondado para 3 casas

## APÊNDICE G – Testes Sider

SiderAI: Olá, bem-vindo para ler o artigo intitulado "Resumo da matéria". O documento fala principalmente sobre atribuição de variáveis, operadores matemáticos, funções matemáticas elementares, valores constantes, importação da biblioteca math, entrada e saída de dados pelo terminal, strings de formatação, valores lógicos, operadores relacionais, operadores lógicos, comandos de decisão, e comandos de repetição. Você pode começar a fazer perguntas como:

1. O que são operadores matemáticos?
2. Como importar a biblioteca math?
3. Quais são os comandos de decisão disponíveis?

Usuário: Contexto: Assuma o papel de um professor da matéria de programação de computadores com Python, que usa os documentos fornecidos como base para ministrar as aulas e para criação de exercícios usados nas atividades dos alunos. As aulas são para turmas de cursos de graduação de diversas áreas de engenharia.

Matéria: Física.

Assuntos da Matéria abordados: MRU.

Tópicos de programação abordados: Variáveis, expressões, entradas e saídas;

Decisão simples (if, else);

Decisão complexa (if, elif, else);

Repetição e Laços;

Laços aninhados;

Funções;

Estrutura de dados homogêneos com vetores;

Estrutura de dados homogêneos com matrizes;

Estrutura de dados heterogêneos (registros).

Instrução: Com base nos documentos fornecidos, crie questões de programação para cada <Tópicos de programação> utilizando os <Assuntos da matéria> da disciplina de <Matéria>. Garanta que cada questão aborde um assunto específico e que todos os tópicos de programação listados sejam contemplados.

Classifique as questões usando um medidor de dificuldade de 1 à 10, onde 1 indica a questão mais fácil e 10 a mais complexa.

Use o seguinte formato de resposta:

<matéria> - <assunto da matéria abordado>

Questão <número da questão> - <tópico usado> - <dificuldade da questão>

Contextualização e introdução sucinta sobre <assunto da matéria abordado>, tal qual explicações prévias dos recursos utilizados no desenvolvimento da questão.

Enunciado da Questão.

Exemplos:

Três exemplos de execução e da saída que se espera do programa.

Solução:

Uma solução possível.

SiderAI: Física - MRU

Questão 1 - Variáveis, Expressões - Dificuldade 2

O conceito de Movimento Retilíneo Uniforme (MRU) é fundamental para compreender o comportamento de um objeto em movimento retilíneo com velocidade constante. Nesta questão, os alunos serão desafiados a criar um programa que calcula a posição final de um objeto em MRU, com base nos valores de entrada fornecidos pelo usuário.

Enunciado:

Implemente um programa que solicite ao usuário os valores da posição inicial ( $S_i$ ), velocidade ( $v$ ) e instante de tempo ( $t$ ). Em seguida, o programa deve calcular a posição final ( $S$ ) do objeto em MRU e imprimir o resultado no terminal com uma precisão de 2 casas decimais.

Exemplos:

Entrada:

$S_i = 10$

$v = 1$

$t = 5$

Saída:

A posição final no tempo  $t = 5.00$  será  $S = 15.00$

Entrada:

$S_i = 55.7$

$v = 60$

$t = 5.8$

Saída:

A posição final no tempo  $t = 5.80$  será  $S = 403.70$

Solução:

```
```python
```

```
Si = float(input('Informe a posição inicial (Si): '))
```

```
v = float(input('Informe a velocidade (v): '))
```

```
t = float(input('Informe o instante de tempo (t): '))
```

```
S = Si + v * t
```

```
print(f'A posição final no tempo t = {t:.2f} será S = {S:.2f}')
```

```
```
```

Usuário: Contexto: Assuma o papel de um professor da matéria de programação de computadores com Python, que usa os documentos fornecidos como base para ministrar as aulas e para criação de exercícios usados nas atividades dos alunos. As aulas são para turmas de cursos de graduação de diversas áreas de engenharia.

Matéria: Mecânica.

Assuntos da Matéria abordados: Dinâmica das máquinas, Sistemas de controle.

Instrução: Com base nos documentos fornecidos, crie uma prova avaliativa usando os tópicos de programação usados em "Resumo da matéria (PT1)" utilizando os <Assuntos da matéria> da disciplina de <Matéria>. Classifique as questões usando um medidor de dificuldade de 1 à 10, onde 1 indica a questão mais fácil e 10 a mais complexa. A prova deverá ter 5 questões. 3 questões deverão ter dificuldade acima de 3 e 2 questões dificuldade acima de 7. Em pelo menos uma questão, use mais do que um tópico de programação.



Use o seguinte formato de resposta:

<matéria> - <assunto da matéria abordado> - <tópicos de programação>  
Questão <número da questão> - <tópico usado> - <dificuldade da questão>  
Contextualização e introdução sucinta sobre <assunto da matéria  
abordado>, tal qual explicações prévias dos recursos utilizados no  
desenvolvimento da questão.  
Enunciado da Questão.  
Exemplos:  
Três exemplos de execução e da saída que se espera do programa.  
Solução:  
Uma solução possível.

SiderAI: Desculpe, mas não tenho acesso ao documento "Resumo da matéria (PT1)" mencionado. Como alternativa, posso criar questões de programação em Python relacionadas aos tópicos de programação listados, mas não posso garantir que elas estarão diretamente alinhadas com o conteúdo específico do documento. Posso prosseguir com a criação das questões com base nos tópicos de programação mencionados?

Usuário: Não

SiderAI: Entendo. Se você tiver acesso ao documento "Resumo da matéria (PT1)" e puder compartilhá-lo, ficarei feliz em criar as questões de programação em Python com base nos tópicos fornecidos. Caso contrário, estou à disposição para ajudar com qualquer outra solicitação.

Usuário: Contexto: Assuma o papel de um professor da matéria de programação de computadores com Python, que usa os documentos fornecidos como base para ministrar as aulas e para criação de exercícios usados nas atividades dos alunos. As aulas são para turmas de cursos de graduação de diversas áreas de engenharia.

Matéria: Mecânica.

Assuntos da Matéria abordados: Dinâmica das máquinas, Sistemas de controle.

Instrução: Com base nos documentos fornecidos, crie uma prova avaliativa usando os tópicos de programação usados no documento utilizando os <Assuntos da matéria> da disciplina de <Matéria>.  
Classifique as questões usando um medidor de dificuldade de 1 à 10, onde 1 indica a questão mais fácil e 10 a mais complexa.  
A prova deverá ter 5 questões. 3 questões deverão ter dificuldade acima de 3 e 2 questões dificuldade acima de 7. Em pelo menos uma questão, use mais do que um tópico de programação.

Use o seguinte formato de resposta:

<matéria> - <assunto da matéria abordado>  
Questão <número da questão> - <tópicos usados> - <dificuldade da questão>  
Contextualização e introdução sucinta sobre <assunto da matéria  
abordado>, tal qual explicações prévias dos recursos utilizados no  
desenvolvimento da questão.  
Enunciado da Questão.  
Exemplos:  
Três exemplos de execução e da saída que se espera do programa.  
Solução:

Uma solução possível.

SiderAI: Entendido. Vou criar uma prova avaliativa com base nos tópicos de programação fornecidos nos documentos. Vou me certificar de incluir questões que abordem os assuntos de Dinâmica das Máquinas e Sistemas de Controle, utilizando os tópicos de programação disponíveis nos documentos. A prova terá 5 questões, com a dificuldade distribuída conforme solicitado. Vou começar a criar as questões e fornecer as respostas no formato solicitado.

SiderAI: Olá, bem-vindo para ler o artigo intitulado "Resumo da matéria". O documento fala principalmente sobre operadores matemáticos, funções matemáticas elementares e valores constantes. Você pode começar a fazer perguntas como:

1. O que são operadores matemáticos?
2. Quais são as funções matemáticas elementares mencionadas no resumo da matéria?
3. Quais são os valores constantes apresentados no resumo da matéria?

Usuário: Contexto: Assuma o papel de um professor da matéria de programação de computadores com Python, que usa os documentos fornecidos como base para ministrar as aulas e para criação de exercícios usados nas atividades dos alunos. As aulas são para turmas de cursos de graduação de diversas áreas de engenharia.

Matéria: Física.

Assuntos da Matéria abordados: MRU.

Tópicos de programação abordados: Variáveis, expressões, entradas e saídas;

Decisão simples (if, else);

Decisão complexa (if, elif, else);

Repetição e Laços;

Laços aninhados;

Funções;

Estrutura de dados homogêneos com vetores;

Estrutura de dados homogêneos com matrizes;

Estrutura de dados heterogêneos (registros).

Instrução: Com base nos documentos fornecidos, crie questões de programação para cada <Tópicos de programação> utilizando os <Assuntos da matéria> da disciplina de <Matéria>. Garanta que cada questão aborde um assunto específico e que todos os tópicos de programação listados sejam contemplados.

Classifique as questões usando um medidor de dificuldade de 1 à 10, onde 1 indica a questão mais fácil e 10 a mais complexa.

Use o seguinte formato de resposta:

<matéria> - <assunto da matéria abordado>

Questão <número da questão> - <tópico usado> - <dificuldade da questão>

Contextualização e introdução sucinta sobre <assunto da matéria abordado>, tal qual explicações prévias dos recursos utilizados no desenvolvimento da questão.

Enunciado da Questão.

Exemplos:

Três exemplos de execução e da saída que se espera do programa.

Solução:

Uma solução possível.

SiderAI: Física - MRU

Questão 1 - Variáveis - Dificuldade 1

Nesta questão, vamos explorar o uso de variáveis para armazenar valores relacionados ao Movimento Retilíneo Uniforme (MRU). O objetivo é que os alunos pratiquem a entrada de dados e o armazenamento em variáveis.

Enunciado:

Escreva um programa que solicite ao usuário que informe a posição inicial ( $S_i$ ), a velocidade ( $v$ ) e o instante de tempo ( $t$ ). Em seguida, o programa deve calcular a posição final ( $S$ ) usando a fórmula do MRU e imprimir o resultado no terminal com uma precisão de 2 casas decimais.

Exemplos:

Informe a posição inicial ( $S_i$ ): 10

Informe a velocidade ( $v$ ): 1

Informe o instante de tempo ( $t$ ): 5

Saída esperada:

A posição final no tempo  $t = 5.00$  será  $S = 15.00$

Solução:

```
```python
```

```
Si = float(input('Informe a posição inicial (Si): '))
```

```
v = float(input('Informe a velocidade (v): '))
```

```
t = float(input('Informe o instante de tempo (t): '))
```

```
S = Si + v * t
```

```
print(f'A posição final no tempo t = {t:.2f} será S = {S:.2f}')
```

```
```
```

Usuário: Contexto: Assuma o papel de um professor da matéria de programação de computadores com Python, que usa os documentos fornecidos como base para ministrar as aulas e para criação de exercícios usados nas atividades dos alunos. As aulas são para turmas de cursos de graduação de diversas áreas de engenharia.

Matéria: Mecânica.

Assuntos da Matéria abordados: Dinâmica das máquinas, Sistemas de controle.

Instrução: Com base nos documentos fornecidos, crie uma prova avaliativa usando os tópicos de programação usados no documento utilizando os <Assuntos da matéria> da disciplina de <Matéria>.

Classifique as questões usando um medidor de dificuldade de 1 à 10, onde 1 indica a questão mais fácil e 10 a mais complexa.

A prova deverá ter 5 questões. 3 questões deverão ter dificuldade acima de 3 e 2 questões dificuldade acima de 7. Em pelo menos uma questão, use mais do que um tópico de programação.

Use o seguinte formato de resposta:

<matéria> - <assunto da matéria abordado>

Questão <número da questão>- <tópicos usados> - <dificuldade da questão>

Contextualização e introdução sucinta sobre <assunto da matéria abordado>, tal qual explicações prévias dos recursos utilizados no desenvolvimento da questão.

Enunciado da Questão.

Exemplos:

Três exemplos de execução e da saída que se espera do programa.

Solução:

Uma solução possível.

SiderAI: Desculpe, mas não fui treinado com informações sobre documentos específicos sobre os tópicos de programação em Python relacionados à Mecânica, Dinâmica das Máquinas e Sistemas de Controle. Como resultado, não consigo criar uma prova avaliativa com base nesses documentos

específicos. No entanto, posso ajudar na criação de questões de programação em Python relacionadas a tópicos gerais de programação e matemática. Se precisar de ajuda com isso, por favor, me avise.

# **Anexos**

# ANEXO A – Ementa BCC701



## UNIVERSIDADE FEDERAL DE OURO PRETO PRÓ-REITORIA DE GRADUAÇÃO PLANO DE ENSINO



|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |                                                |                                                |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------|------------------------------------------------|
| Nome do Componente Curricular em português:<br>Programação de Computadores I                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |                                                | Código:                                        |
| Nome do Componente Curricular em inglês:<br>Computer Programming I                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |                                                | BCC701                                         |
| Nome e sigla do departamento:<br>Departamento de Computação (DECOM)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |                                                | Unidade acadêmica:<br>ICEB                     |
| Nome do docente:<br>Fernanda Sumika Hojo de Souza, Guilherme Augusto Lopes Silva, José Romildo Malaquias, Marcelo Luiz Silva, Mateus Coelho Silva, Reinaldo Silva Fortes                                                                                                                                                                                                                                                                                                                                                                                                            |                                                |                                                |
| Carga horária semestral<br>60 horas                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | Carga horária semanal teórica<br>02 horas/aula | Carga horária semanal prática<br>02 horas/aula |
| Data de aprovação na assembleia departamental: 28/08/2023                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |                                                |                                                |
| <p>Ementa:</p> <p>Introdução a ambientes de programação. Conceitos de algoritmo. Conceitos básicos de programação: valores e expressões de tipos primitivos, variáveis, comando de atribuição, comandos de controle de fluxo, entrada e saída padrão, procedimentos e funções, tipos de dados compostos.</p>                                                                                                                                                                                                                                                                        |                                                |                                                |
| <p>Conteúdo programático:</p> <ol style="list-style-type: none"> <li>1. Introdução a ambientes de programação</li> <li>2. Conceitos e representação de algoritmos</li> <li>3. Conceitos básicos de programação             <ol style="list-style-type: none"> <li>a. Valores, tipos e expressões</li> <li>b. Variáveis e comando de atribuição</li> <li>c. Comandos de entrada e saída</li> <li>d. Comandos de controle de fluxo</li> <li>e. Procedimentos e funções</li> <li>f. Estruturas de dados homogêneas</li> <li>g. Estruturas de dados heterogêneas</li> </ol> </li> </ol> |                                                |                                                |
| <p>Objetivos:</p> <p>Desenvolver a capacidade de construir programas de pequeno e médio porte com a utilização de conceitos de programação estruturada.</p>                                                                                                                                                                                                                                                                                                                                                                                                                         |                                                |                                                |

Bibliografia básica:

1. ASCENCIO, A. F. G.; CAMPOS, E. A. V. de. **Fundamentos da Programação de Computadores: algoritmos, Pascal, C/C++ e Java**. Pearson Education do Brasil Ltda. São Paulo, 2012. 3ª Edição, ISBN: 9788564574168. **Disponível na biblioteca digital E-BOOKS BVIRTUAL PEARSON via Minha UFOP.**
2. MANZANO, J. A. N. G.; OLIVEIRA, J. F. de, **Algoritmos - Lógica para desenvolvimento de Programação de Computadores**. Editora Érica, 2016. 28ª Edição, ISBN 978-85-365-1865-7. **Disponível na biblioteca digital E-BOOKS MINHA BIBLIOTECA via Minha UFOP.**
3. PERKOVIC, L. **Introdução à Computação Usando Python: Um foco no desenvolvimento de aplicações**. LTC - Livros Técnicos e Científicos Editora Ltda., 2016. 1ª Edição, ISBN 978-85-216-3092-0. **Disponível na biblioteca digital E-BOOKS MINHA BIBLIOTECA via Minha UFOP.**

Bibliografia complementar:

1. GUEDES, S. **Lógica de Programação Algorítmica**. Pearson Education do Brasil Ltda. São Paulo, 2014. Primeira Edição, ISBN: 9788543005546. **Disponível na biblioteca digital E-BOOKS BVIRTUAL PEARSON via Minha UFOP.**
2. BANIN, S. L. **Python 3 - Conceitos e Aplicações: uma abordagem didática**. Editora Érica, 2018. 1ª Edição, ISBN 978-85-365-3025-3. **Disponível na biblioteca digital E-BOOKS MINHA BIBLIOTECA via Minha UFOP.**
3. MENEZES, N. N. C. **Introdução à programação com Python: algoritmos e lógica de programação para iniciantes**. 3. ed., rev. e ampl. São Paulo: Novatec, 2019.
4. SOUZA, M. A. F. de; et. al. **Algoritmos e Lógica de Programação**. Cengage Learning. São Paulo, 2005.
5. FARRER, H. et. al. **Algoritmos Estruturados**. 3ª Edição. LTC - Livros Técnicos e Científicos. Rio de Janeiro, 1999.