
Python para IA: do zero ao primeiro chatbot

Asimov Academy

ASIMOV

Conteúdo

01. Boas vindas	6
O Panorama Atual da Inteligência Artificial	6
Benefícios de Acessar a Segunda Camada	6
Por que Python?	7
Superando Obstáculos	7
O Projeto Prático	7
Conclusão	8
02. Ensino Baseado em Projetos	9
Desafios da Metodologia Tradicional	9
O Impacto do Tempo no Aprendizado	9
Eficiência e Aplicação Prática	10
Ensino Aplicado à Prática	10
Conclusão	10
03. O Python como uma ferramenta	11
A Importância da Programação	11
Programação Desenvolve Habilidades Cognitivas	11
O Verdadeiro Potencial das IAs	11
Por Que Escolher Python?	11
Pontos Fortes do Python	12
Python como Ferramenta	12
04. Por que aprender a aprender	13
O Impacto de um Aprendizado Eficiente	13
Crítica ao Ensino Tradicional	13
Um Novo Modelo de Educação	14
Benefícios de Aprender a Aprender	14
05. O que, como e por quê?	15
O Círculo Dourado: Pensar de Dentro para Fora	15
Definição de Motivações	15
Conceitos Fundamentais sobre Motivação	16
Resumo dos Conceitos Principais	16

06. Como nosso cérebro aprende?	17
O Cérebro e o Processo de Aprendizagem	17
Os Dois Modos de Funcionamento do Cérebro	17
Memória de Trabalho vs. Memória de Longo Prazo	18
Os Ciclos da Aprendizagem	18
Como Aproveitar Melhor as Capacidades do Cérebro	18
A Importância de se Divertir ao Aprender	19
07. Hábitos e Procrastinação	20
O que são hábitos?	20
As quatro etapas da formação de um hábito	20
A relação entre hábitos e procrastinação	20
Como evitar a procrastinação e formar bons hábitos?	21
Conclusão	21
08. As principais técnicas de Aprendizado	22
Fundamentos do Aprendizado Eficiente	22
Técnica Pomodoro	22
Recall Ativo	22
Prática Deliberada	23
Intercalação de Estudos (Interleaving)	23
Constância	24
Conclusão	24
09. Como aprender a programar	25
Desenvolva a Mentalidade de Programador	25
Buscando Soluções	25
Aprendendo com Programadores Mais Experientes	26
Estudando Códigos Prontos	26
Leitura de Documentação	26
Criando Projetos para Aprender na Prática	27
Modificando Projetos Existentes	27
Compartilhando Seus Projetos	27
Conclusão	27
10. Apresentação do projeto	28
11. Interagindo com o usuário – Variáveis e tipos de dados	29
Trabalhando no Google Colab	29

Compreendendo Variáveis	30
Tipos de Dados	30
Entrada de Dados e Formatação de Strings	30
Conclusão da Aula	31
12. Iterações repetidas – Loops e condicionais	32
Gerenciando Sessões no Google Collaboratory	32
Construindo o Loop	32
Operadores de Comparação	32
Lidando com Erros	33
Validação de Entrada	33
Estruturas Condicionais	33
Criando o Loop no Chatbot	34
Conclusão	34
13. Armazenando interações – Listas, dicionários e funções	35
Conceitos Fundamentais Revisados	35
Estruturas de Dados	35
Listas	36
Dicionários	36
Funções	37
Implementação no Chatbot	37
Conclusão	37
14. Aumentando o poder do seu script com bibliotecas	38
O que são Bibliotecas em Python?	38
Bibliotecas Padrão	38
Instalando Novas Bibliotecas	38
Principais Bibliotecas a Considerar	39
Funcionalidades do LangChain	39
Conclusão	39
15. Utilizando LangChain para Acessar o Llama	40
O que são Bibliotecas?	40
Acesso ao Modelo Llama 3	40
Criando a API Key	40
Configurando o Ambiente	40
Configurando o Modelo de Linguagem	41
Realizando uma Chamada ao Modelo	41

Conclusão	42
16. Criando seu primeiro ChatBot	43
Estrutura do Chatbot	43
Configuração Inicial	43
Prompt Templates e Chains	43
O que é um Prompt?	43
O que é um Prompt Template?	44
Criando o Chatbot	44
Explicação do Código	45
Exemplo de Execução	46
Conclusão	46
17. Dando acesso a sites para o nosso bot	48
A importância do acesso externo	48
Configuração Inicial	48
Lógica e funcionamento	49
Código finalizado	50
Conclusão	52
18. Acessando vídeos de Youtube e PDFs	53
Preparação e Instalação das Bibliotecas	53
Extraindo Dados do YouTube	53
Unificando o Conteúdo	54
Criando o Template e Invocando o Bot	54
Extraindo Dados de Arquivos PDF	55
Montando o Google Drive	55
Definindo o Loader do PDF	56
Unificando o Conteúdo do PDF	56
Criando o Template e Invocando o Bot para o PDF	56
Conclusão	57
19. Criando a estrutura final do nosso ChatBot	58
Fluxo do Projeto	59
Exemplo de Loop de Seleção	59
Funções para Carregamento de Documentos	60
Organização do Código	60
Conclusão	61

20. Finalizando o projeto	62
Implementação dos Document Loaders	62
Carregamento de Sites	62
Carregamento de PDFs	63
Carregamento de Vídeos do YouTube	63
Mudanças na função resposta_bot	64
Passagem do Documento para o Chatbot	64
Exemplo do código final	64
Conclusão	66
21. O que fazer agora	67
Direcionamentos para Aprofundamento	67
Aprofundamento em Aplicações de IA	67
Desenvolvimento de Agents Customizados	67
Fundamentos de Data Science e Machine Learning	67
Criação de Dashboards e Aplicações Web	68
Análise e Tratamento de Dados	68
Considerações Finais	68

01. Boas vindas

Se você chegou até aqui, é porque pensa diferente. Você quer se destacar da multidão e ir além do pensamento comum. A combinação entre a programação Python e a Inteligência Artificial permitirá exatamente isso.

O Panorama Atual da Inteligência Artificial

Hoje, quando falamos em Inteligência Artificial, vemos uma infinidade de soluções, modelos, sites e plataformas diferentes. Muitas pessoas e empresas estão correndo atrás de informações, tentando descobrir qual modelo é “melhor”. No entanto, na prática, pouco valor está sendo gerado.

Essas pessoas fazem parte da **primeira camada**: utilizam IA de forma superficial, limitando-se a soluções criadas por terceiros.

Mas **você quer ir além**. Com Python, você acessa a **segunda camada**, onde pode usar as IAs da maneira que fizer mais sentido para você. Você terá acesso irrestrito ao poder dessas tecnologias para criar sistemas personalizados e integrá-los com qualquer aplicação que desejar.

Imagine poder:

- Automatizar a organização de arquivos no seu computador;
- Fazer com que o ChatGPT leia e responda seus e-mails automaticamente;
- Criar um sistema que responda a perguntas com base em dados de planilhas ou bancos de dados;
- Enviar informações diretamente para o seu WhatsApp sem precisar acessar a interface da IA.

Isso é possível combinando IA e Python!

Claro, aprender a programar exige esforço. Escrever um prompt para o ChatGPT é mais fácil do que construir um sistema. No entanto, o retorno dessa habilidade é imenso. No final deste curso, você verá que programar não é tão difícil quanto parece.

Benefícios de Acessar a Segunda Camada

Os benefícios de aprender a programar e acessar essa segunda camada vão muito além da Inteligência Artificial. Aqui estão **dois grandes ganhos**:

1. **Diferenciação Profissional:** Enquanto todos ao seu redor gastam horas utilizando plataformas manuais, você poderá automatizar essas tarefas e obter resultados instantaneamente.

2. **Capacidade de Resolver Problemas:** A programação ensina a estruturar o pensamento de forma lógica. Isso significa que, além de criar automações e sistemas, você se tornará uma pessoa mais analítica e eficiente na resolução de problemas, mesmo fora da programação.

Alguém que pensa diferente e domina a lógica de programação certamente será um profissional de destaque, independentemente da área de atuação.

Por que Python?

A linguagem escolhida para desbloquear o poder da IA é **Python**. Entre os motivos para essa escolha estão:

- É uma das linguagens mais utilizadas no mundo;
- Possui uma sintaxe simples e intuitiva, facilitando o aprendizado;
- Oferece uma enorme variedade de bibliotecas para IA, automação, análise de dados e muito mais;
- Permite resultados práticos rapidamente, mesmo para quem nunca programou antes.

Python é a melhor escolha para quem deseja utilizar a programação como ferramenta, independentemente da profissão. Seja você contador, advogado, médico ou engenheiro, essa linguagem será útil para você!

Superando Obstáculos

Aprender a programar não é simples. Muitas pessoas falham nesse processo porque não enxergam o potencial da programação e, conseqüentemente, perdem a motivação diante das dificuldades.

Nosso objetivo é evitar que isso aconteça com você. Por isso, o aprendizado aqui será baseado em **um projeto prático e real de Inteligência Artificial**.

O Projeto Prático

Vamos criar um **chatbot para responder dúvidas**. Esse chatbot poderá ser aplicado em diversas situações:

- Responder perguntas frequentes de clientes;
- Auxiliar professores no atendimento a alunos;
- Automatizar respostas internas dentro de empresas.

E o melhor: faremos isso em poucas horas!

Nosso objetivo é mostrar que a programação é poderosa e acessível. Em pouco tempo, você conseguirá criar soluções úteis para o seu dia a dia, aumentar sua produtividade no trabalho e abrir novas oportunidades profissionais.

Conclusão

A programação muda vidas e pode mudar a sua também!

Vamos juntos nessa jornada!

02. Ensino Baseado em Projetos

A metodologia de ensino baseada em projetos tem se mostrado mais eficiente do que abordagens tradicionais. Essa estratégia permite que o aprendizado ocorra de forma prática, à medida que os conceitos são aplicados diretamente em situações reais.

Ao invés de apresentar os conceitos de programação de forma isolada, o aprendizado se desenvolve conforme a necessidade do projeto em construção. Essa abordagem proporciona uma compreensão mais clara da aplicabilidade dos conceitos, tornando o processo mais envolvente e eficiente.

A linguagem Python permite atuar em diversas áreas, como automação, web scraping, análise de dados, desenvolvimento de aplicações web e inteligência artificial. Dessa forma, os projetos propostos contemplam diferentes aplicações práticas da linguagem.

Desafios da Metodologia Tradicional

O modelo tradicional de ensino, que aborda cada conceito separadamente antes da aplicação prática, pode gerar desmotivação e dificultar a retenção do conteúdo. Muitas vezes, o aprendizado ocorre sem uma visão clara de como os elementos se combinam para formar soluções completas, o que pode levar ao abandono do estudo antes da conclusão.

Dados indicam que a taxa de conclusão em cursos online convencionais é relativamente baixa, o que reforça a necessidade de abordagens mais dinâmicas e aplicáveis.

O Impacto do Tempo no Aprendizado

O tempo é um recurso valioso e deve ser aproveitado de maneira eficiente no processo de aprendizagem. Métodos que não promovem a retenção efetiva do conhecimento resultam em desperdício de esforço e tempo.

Para ilustrar esse impacto, pode-se utilizar um exemplo numérico em Python:

```
avg_duration = 30 # Duração média de cada vídeo em minutos
num_videos = 23 # Número total de vídeos no curso
init_views = 8000000 # Número de visualizações no primeiro vídeo
final_views = 180000 # Número de visualizações no último vídeo

step_views = (final_views - init_views) / (num_videos - 1)
video_views_prox = [i for i in range(init_views, final_views - 1, int(step_views))]
minutes_wasted = [(v - final_views) * avg_duration for v in video_views_prox]
years_wasted = [v / (60 * 24 * 365) for v in minutes_wasted]
sum(years_wasted)
```

Esse cálculo estima a quantidade de tempo despendida por aqueles que iniciam um curso e não o concluem. Estratégias que aumentam a eficiência do aprendizado minimizam esse desperdício e promovem melhor aproveitamento do tempo disponível.

Eficiência e Aplicação Prática

A programação é uma ferramenta que melhora a eficiência tanto no aspecto cognitivo, ao fortalecer a capacidade de resolução de problemas, quanto na otimização de tarefas repetitivas. Com o uso adequado da automação, é possível reduzir significativamente o tempo gasto em processos manuais, permitindo foco em atividades mais estratégicas e criativas.

Ensino Aplicado à Prática

A abordagem baseada em projetos possibilita um aprendizado mais dinâmico e eficaz. Os conceitos são introduzidos de acordo com a necessidade para o desenvolvimento das aplicações, evitando a fragmentação do conhecimento e promovendo uma compreensão integrada.

Para aqueles que desejam aprofundar-se nos fundamentos da linguagem Python, há materiais complementares que cobrem desde conceitos básicos até temas avançados.

Conclusão

A adoção de metodologias que favorecem a aplicação prática do conhecimento melhora a retenção de informações e o engajamento no aprendizado. Dessa forma, o ensino baseado em projetos se mostra uma abordagem eficaz para garantir a assimilação dos conceitos e incentivar a continuidade do estudo. A jornada de aprendizado avança com a exploração dos próximos conteúdos.

03. O Python como uma ferramenta

Este material é voltado para aqueles que têm interesse em Inteligência Artificial, especialmente na utilização de modelos de linguagem, como o ChatGPT, e desejam aprender a programar em Python. Não há necessidade de conhecimento prévio, apenas um computador e disposição para aprender.

A Importância da Programação

Diante do avanço das IAs e da facilidade de acesso a essas tecnologias, pode surgir a dúvida: por que aprender a programar? A programação não apenas facilita diversas tarefas e cria novas oportunidades, mas também oferece benefícios fundamentais:

Programação Desenvolve Habilidades Cognitivas

A prática da programação fortalece duas grandes capacidades:

- **Resolução de problemas:** Cada trecho de código exige a solução de um problema específico, aprimorando a capacidade analítica e lógica.
- **Aprendizado contínuo:** A tecnologia está em constante evolução, e o contato com a programação estimula a habilidade de aprender conceitos novos de forma eficiente.

O Verdadeiro Potencial das IAs

O verdadeiro poder dos modelos de linguagem é explorado plenamente apenas por meio da programação. A utilização de interfaces prontas oferece apenas uma fração das possibilidades que essas ferramentas possuem. Com programação, é possível automatizar processos, integrar modelos de IA a diferentes sistemas e expandir suas funcionalidades.

Por exemplo, ao invés de simplesmente gerar textos com um modelo como o ChatGPT, um programador pode criar um sistema que produza e publique centenas de artigos automaticamente. Outro exemplo prático é o uso do modelo Whisper para transcrição de áudio: ao combinar Python com esse modelo, é possível desenvolver um sistema que automatiza todo o processo de geração de legendas para vídeos, reduzindo significativamente o tempo e o esforço manual necessários.

Por Que Escolher Python?

Entre as diversas linguagens de programação disponíveis, Python se destaca por algumas razões específicas:

Pontos Fortes do Python

- **Facilidade de aprendizado:** Python possui uma sintaxe simples e intuitiva, tornando-se uma das linguagens mais acessíveis para iniciantes.
- **Versatilidade:** É amplamente utilizado em automação, análise de dados, inteligência artificial, desenvolvimento web e muitas outras áreas.
- **Relevância para Inteligência Artificial:** Python é a principal linguagem utilizada para Machine Learning, IA e análise de dados, sendo amplamente adotada tanto no meio acadêmico quanto na indústria.

Quando Python Não é a Melhor Opção Para aqueles que buscam uma carreira no desenvolvimento web de larga escala ou na criação de sites complexos, existem opções mais adequadas, como JavaScript combinado com frameworks como React ou Angular.

Python como Ferramenta

Python se destaca como uma excelente linguagem para aqueles que desejam utilizar a programação como meio para atingir objetivos específicos, sem necessariamente seguir carreira como desenvolvedor. Seu uso proporciona maior eficiência e abre novas possibilidades para otimização de processos e integração com Inteligência Artificial.

04. Por que aprender a aprender

Diferenciar-se exige um caminho distinto. No aprendizado da programação, é comum focar diretamente nos conteúdos técnicos, mas essa abordagem pode ser um equívoco.

Lógica, resolução de problemas, adaptação a novos paradigmas e tecnologias em constante evolução são desafios inerentes à programação. Esses elementos não fazem parte do cotidiano da maioria das pessoas, e, sem uma preparação adequada, a frustração pode se tornar um obstáculo significativo. Superar esses desafios exige não apenas habilidades técnicas, mas também um preparo cognitivo e humano.

Este módulo tem como objetivo fornecer ferramentas essenciais para desenvolver a capacidade de aprender de maneira eficiente, facilitando a assimilação de conteúdos complexos e promovendo um aprendizado contínuo e estruturado.

O Impacto de um Aprendizado Eficiente

A habilidade de aprender de forma eficaz proporciona segurança. Mesmo diante de um novo desafio, a confiança na capacidade de adquirir conhecimento permite enfrentar obstáculos e crescer com cada experiência.

Compreender os mecanismos do aprendizado humano possibilita a utilização de técnicas simples e eficazes para a fixação do conhecimento. Aplicar esses métodos pode tornar o processo mais prático, produtivo e satisfatório.

Crítica ao Ensino Tradicional

A habilidade de aprender a aprender deveria ser um conteúdo obrigatório nas escolas. No entanto, essa competência raramente é abordada, mesmo em ambientes acadêmicos avançados. Grande parte da vida escolar é dedicada ao aprendizado de diversas disciplinas, mas poucas vezes se discute como aprender de maneira eficaz.

A metodologia educacional tradicional muitas vezes pressupõe que o aluno deve simplesmente absorver o conteúdo, sem um direcionamento sobre como otimizar esse processo. Isso pode tornar a experiência ineficiente e desmotivadora, resultando na perda de potenciais talentos e dificultando a retenção do conhecimento.

Um Novo Modelo de Educação

É fundamental adotar uma abordagem pedagógica que valorize não apenas a transmissão de conhecimento, mas também o desenvolvimento das capacidades cognitivas dos alunos. Um ensino eficiente deve considerar tanto os aspectos técnicos quanto os humanos, despertando o interesse e o potencial de cada indivíduo.

Benefícios de Aprender a Aprender

Dominar essa habilidade não apenas acelera o aprendizado de conceitos técnicos, mas também contribui para o desenvolvimento pessoal. Entre os benefícios estão:

- Maior capacidade de concentração e retenção de informações;
- Aumento da motivação e autoconfiança;
- Melhoria na resolução de problemas e tomada de decisões;
- Desenvolvimento de uma mentalidade de aprendizado contínuo.

Compreender e aplicar estratégias para aprender melhor possibilita a adaptação a novos desafios e a maximização do potencial individual. A partir desse conhecimento, torna-se possível aprimorar a maneira de absorver informações e alcançar um desenvolvimento mais sólido e consistente.

05. O que, como e por quê?

Por que algumas pessoas conseguem aprender a programar enquanto outras falham? Seria uma questão de inteligência, memória ou QI? A experiência no ensino da programação demonstra que não.

Pessoas com acesso a recursos, tempo livre e materiais de qualidade frequentemente enfrentam dificuldades para aprender a programar, enquanto outras, em condições adversas, prosperam na área. O fator determinante não está nas condições externas, mas sim nas **motivações** individuais.

A motivação é o combustível para superar desafios, sejam eles relacionados à programação ou a qualquer outra área da vida. A tomada de decisões baseada em motivações sólidas aumenta significativamente a persistência e o sucesso na jornada de aprendizado.

O Círculo Dourado: Pensar de Dentro para Fora

Simon Sinek propôs o conceito do **Círculo Dourado**, que ilustra um processo de tomada de decisão mais eficaz. Esse modelo é composto por três níveis concêntricos:

1. **Por que?** (motivação fundamental)
2. **Como?** (estratégia para alcançar o objetivo)
3. **O que?** (ação concreta a ser realizada)

A abordagem mais comum é iniciar pelo “o que” e seguir para o “como” e “por que”. No entanto, para garantir decisões mais alinhadas com objetivos de longo prazo, recomenda-se inverter essa ordem: começar pelo “por que”, seguir para o “como” e, por fim, definir o “o que”.

Definição de Motivações

Antes de iniciar a aprendizagem da programação, é essencial compreender as próprias motivações. Alguns questionamentos podem auxiliar nesse processo:

- O que motiva as atividades diárias?
- Quais são os objetivos de longo prazo?
- Quais valores e princípios sustentam essas motivações?

Motivações profundas, enraizadas em valores como compromisso, responsabilidade e generosidade, são mais eficazes do que razões superficiais ou voltadas exclusivamente para resultados.

Conceitos Fundamentais sobre Motivação

1. Motivações vs. Resultados

Muitas vezes, há confusão entre motivação e resultado. “Quero ganhar mais dinheiro” ou “Quero trocar de emprego” são resultados, não motivações. Para identificar a verdadeira motivação, deve-se questionar constantemente o “por quê” até chegar a uma causa mais profunda, como a busca por estabilidade, realização pessoal ou contribuição para a família e a sociedade.

2. Motivações Intrínsecas vs. Extrínsecas

Motivações podem ser classificadas como intrínsecas ou extrínsecas:

- **Intrínsecas:** derivam de valores internos e são sustentáveis a longo prazo.
- **Extrínsecas:** são influenciadas por fatores externos, como expectativas alheias, e tendem a ser mais frágeis.

É fundamental validar as influências externas para garantir que as motivações sejam genuínas e alinhadas aos próprios valores.

3. Motivações Fortes vs. Fracas

Motivações voltadas para o benefício próprio geralmente são mais fracas do que aquelas baseadas no compromisso com outras pessoas. Compromissos altruístas, que envolvem responsabilidades além do interesse individual, costumam gerar maior perseverança e dedicação.

Resumo dos Conceitos Principais

- Tomar decisões de dentro para fora, começando pelo **porquê**.
- Motivações sólidas são essenciais para manter a constância no aprendizado.
- Para identificar as verdadeiras motivações, é necessário diferenciar:
 - **Motivações e resultados:** as motivações estão na origem das ações, não nas consequências.
 - **Motivações intrínsecas e extrínsecas:** é importante que as motivações sejam genuínas e não apenas reflexos de influências externas.
 - **Motivações fracas e fortes:** motivações ligadas ao compromisso com outras pessoas costumam ser mais poderosas e duradouras.

A compreensão e a definição clara das próprias motivações são fatores determinantes para o sucesso no aprendizado da programação. Motivação forte e bem fundamentada proporciona energia para superar desafios e alcançar objetivos com maior persistência e entusiasmo.

06. Como nosso cérebro aprende?

O ser humano possui uma das mais complexas máquinas desenvolvidas pela natureza: o cérebro. Seus limites ainda são desconhecidos, e ele é capaz de feitos extraordinários, impulsionando nossa evolução e permitindo expressar emoções, criatividade e aprendizado. No entanto, muitas vezes não damos a ele a atenção e o tempo necessários para seu pleno desenvolvimento.

Neste capítulo, vamos explorar algumas características fundamentais do nosso cérebro e como elas influenciam diretamente o aprendizado. O conteúdo é baseado em pesquisas recentes da neurociência, com referência ao livro “Aprendendo a Aprender” e ao curso de mesmo nome oferecido no Coursera.

O Cérebro e o Processo de Aprendizagem

Nosso cérebro é composto por bilhões de neurônios, que são as células responsáveis pelo processamento e transmissão de informação. A comunicação entre neurônios ocorre por meio de impulsos elétricos chamados sinapses. O fortalecimento dessas conexões é a base do aprendizado: quanto mais exercitamos um conhecimento ou habilidade, mais forte se torna essa conexão neural.

Podemos comparar esse processo com abrir uma trilha na floresta. No início, é difícil e trabalhoso, pois estamos desbravando um caminho novo. Mas, com a repetição, a trilha se torna mais fácil de percorrer, podendo se transformar em uma estrada bem estabelecida. O mesmo acontece com nosso cérebro ao aprender: a repetição torna a conexão neural mais forte e automática.

Os Dois Modos de Funcionamento do Cérebro

Nosso cérebro opera em dois modos distintos: o **modo foco** e o **modo difuso**.

- **Modo Foco:** é o estado de atenção ativa, no qual concentramos nossa energia para aprender algo novo ou resolver um problema. É uma fase essencial para absorver conhecimento de forma estruturada e racional.
- **Modo Difuso:** ocorre quando relaxamos e deixamos nossa mente divagar. É nesse estado que costumamos ter insights criativos e consolidamos informações na memória de longo prazo.

Uma analogia útil é comparar o aprendizado com o treinamento físico: no modo foco, estamos exercitando um músculo; no modo difuso, estamos permitindo que o corpo se recupere e cresça. Ambos são essenciais para o progresso.

Memória de Trabalho vs. Memória de Longo Prazo

Nosso cérebro armazena informações de duas maneiras:

- **Memória de Trabalho:** é como uma pequena gaveta bagunçada com espaço limitado. Para adicionar um novo item, algo precisa ser retirado. Se sobrecarregarmos essa memória, sentimos a sensação de “cabeça cheia”.
- **Memória de Longo Prazo:** é um grande galpão de armazenamento. Quanto mais utilizamos uma informação, mais acessível ela se torna dentro desse galpão, facilitando sua recuperação.

Uma boa técnica de estudo deve permitir que a memória de trabalho seja usada de forma eficiente, mantendo apenas informações essenciais e transferindo conhecimentos importantes para a memória de longo prazo.

Os Ciclos da Aprendizagem

O aprendizado não ocorre de forma linear. No início, podemos sentir grande progresso, mas, em determinado momento, nos deparamos com uma barreira e sentimos que desaprendemos o que já sabíamos. É comum enfrentar altos e baixos nesse processo.

Essa oscilação é natural e ocorre porque nosso cérebro está consolidando novas conexões neurais. Como abrir uma trilha na mata, há momentos em que precisamos recuar para encontrar um caminho melhor. O importante é persistir, entender que essa fase é parte do processo e permitir momentos de descanso para o modo difuso atuar.

Como Aproveitar Melhor as Capacidades do Cérebro

Com base no que aprendemos sobre o funcionamento do cérebro, podemos estruturar estratégias eficazes para potencializar o aprendizado:

1. **Alterne entre o modo foco e o modo difuso:** evite longos períodos de estudo sem pausas. Momentos de descanso ajudam na fixação do aprendizado.
2. **Otimize sua memória de trabalho:** anote informações importantes para liberar espaço mental e evitar sobrecarga.
3. **Revise periodicamente:** relembrar informações faz com que fiquem mais acessíveis na memória de longo prazo.
4. **Pratique ativamente:** ao invés de apenas ler ou assistir a aulas, exercite o aprendizado com exercícios e projetos práticos.

5. **Seja paciente com os ciclos de aprendizado:** dificuldades e momentos de esquecimento fazem parte do processo. Persistência e paciência são essenciais.

A Importância de se Divertir ao Aprender

Por fim, uma das percepções mais importantes é que aprender deve ser uma experiência prazerosa. Encare desafios com entusiasmo e curiosidade. Se divirta ao aprender, dê risada dos erros e veja-os como oportunidades de crescimento. O aprendizado é um privilégio e deve ser encarado com leveza e gratidão.

Lembre-se: **aprender é bom!** Quanto mais você se diverte no processo, mais natural e duradouro será seu aprendizado.

07. Hábitos e Procrastinação

Compreender como os hábitos funcionam e como estão ligados à procrastinação é essencial para otimizar nosso aprendizado e produtividade. Agora que já possuímos um entendimento básico sobre o funcionamento do nosso cérebro e sobre os processos que envolvem a aprendizagem, podemos explorar aspectos mais práticos do nosso dia a dia. Um desses aspectos é o poder dos hábitos.

Nosso cotidiano é fortemente baseado em hábitos. Saber formá-los conscientemente pode ser uma ferramenta poderosa para o sucesso.

O que são hábitos?

Os hábitos são ações que realizamos de forma automática, sem demandar esforço cognitivo. Mesmo tarefas complexas podem se tornar hábitos, pois, após muitas repetições, o cérebro as “decora”. Um bom exemplo disso é dirigir um carro. No início, exige atenção e esforço, mas com o tempo, torna-se algo natural, quase sem necessidade de pensar. Isso acontece porque o hábito está formado.

Nosso cérebro sempre buscará economizar energia, pois, apesar de pequeno, ele consome cerca de 30% de toda a energia do corpo. Os hábitos são um mecanismo de eficiência que permitem essa economia. No entanto, cabe a nós definir se eles serão positivos ou negativos em nossa rotina.

As quatro etapas da formação de um hábito

Os hábitos seguem um ciclo de quatro etapas:

1. **Estímulo:** É o gatilho que desencadeia o hábito. Por exemplo, a vibração do celular pode levar automaticamente a pegá-lo e navegar nas redes sociais.
2. **Rotina:** É a ação repetida, como checar o Instagram, acessar o YouTube ou ler um site de notícias.
3. **Recompensa:** Toda rotina de hábito gera algum tipo de prazer, por menor que seja. Isso reforça o hábito e faz com que ele se repita no futuro.
4. **Crença:** Para que um hábito se estabeleça, é necessário acreditar que ele é possível e importante. Caso contrário, o cérebro tende a sabotar a tentativa de mudança.

A relação entre hábitos e procrastinação

A procrastinação ocorre quando nos deparamos com uma tarefa difícil que exige grande esforço mental. O cérebro, tentando evitar esse desconforto, busca atividades mais fáceis e prazerosas, caindo no

piloto automático de hábitos que já estão estabelecidos. Ou seja, procrastinar é um reflexo natural do cérebro para evitar o desconforto imediato.

Estudos da neurociência revelam que, ao enfrentarmos uma tarefa difícil, o cérebro ativa áreas associadas à dor física. No entanto, assim que começamos a tarefa, essa sensação de desconforto desaparece. Isso significa que a parte mais difícil é sempre começar.

Como evitar a procrastinação e formar bons hábitos?

1. **Torne-se consciente dos seus hábitos:** Identifique quais são os gatilhos que te levam à procrastinação e tente reduzi-los. Se o celular é uma distração, mantenha-o longe durante o estudo.
2. **Crie um ambiente propício:** Elimine distrações como redes sociais, notificações e e-mails durante períodos de concentração.
3. **Adote bons hábitos:** Exercícios físicos, por exemplo, são fundamentais para a produção de novos neurônios e podem melhorar sua capacidade de aprendizado.
4. **Associe hábitos a recompensas:** Torne a experiência prazerosa. Se gosta de estar com amigos, estude em grupo. Se aprecia o contato com a natureza, leve seu material para um parque.
5. **Cuide do sono:** Durante o descanso, o cérebro processa e organiza as informações aprendidas ao longo do dia. Dormir mal compromete tanto a fixação quanto a absorção de novos conteúdos.

Conclusão

A procrastinação é um dos maiores inimigos do aprendizado. Para combatê-la, é essencial entender como os hábitos funcionam e como utilizá-los a nosso favor. Criar um ambiente livre de distrações, incorporar hábitos produtivos e recompensar-se de maneira adequada são estratégias fundamentais para manter a disciplina e a consistência no estudo.

A mudança de hábitos não ocorre de um dia para o outro, mas com paciência, persistência e estratégias corretas, é possível transformar sua rotina e alcançar melhores resultados. O segredo é começar!

08. As principais técnicas de Aprendizado

Neste capítulo, abordaremos técnicas práticas de aprendizado que podem ser aplicadas no seu dia a dia para otimizar seus estudos. No entanto, é essencial que haja um compromisso real com a aplicação dessas técnicas. Apenas consumir o conteúdo sem testar sua eficácia na prática será um desperdício de tempo. Portanto, experimente cada método, avalie os resultados e adapte-os às suas necessidades.

Fundamentos do Aprendizado Eficiente

O aprendizado eficiente não depende apenas de dedicação, mas de estratégias que otimizem a absorção e retenção de informações. Algumas abordagens comprovadas cientificamente podem tornar o processo mais produtivo e menos desgastante. Vamos explorar algumas dessas técnicas e como aplicá-las no seu cotidiano.

Técnica Pomodoro

A técnica Pomodoro é uma das mais famosas técnicas de estudo do mundo, pois se baseia na neurociência do foco e da produtividade. O método consiste em dividir o tempo de estudo ou trabalho em blocos de 25 minutos, chamados “pomodoros”, seguidos de pausas de 5 minutos.

Por que funciona?

- O estudo é tratado como um processo, não um resultado.
- A procrastinação é reduzida, pois o cérebro se sente menos sobrecarregado.
- As pausas permitem que o cérebro alterne entre os modos focado e difuso, facilitando a retenção do conteúdo.

Durante a pausa, dê uma pequena recompensa ao seu cérebro para reforçar o hábito. Após quatro ciclos de Pomodoro, faça um intervalo maior, de 15 a 30 minutos.

Recall Ativo

A técnica de “recall ativo” consiste em lembrar ativamente o que foi estudado, em vez de apenas reler ou sublinhar o material.

Como aplicar?

1. Ao final de cada aula, pegue um papel e escreva tudo o que você lembrar do conteúdo.

2. Não consulte suas anotações. O objetivo é fortalecer sua memória.
3. Depois, compare com o material e veja o que ficou de fora.

O estudo ativo gera um envolvimento mais profundo com a matéria, ajudando na formação de memórias de longo prazo.

Prática Deliberada

A prática deliberada é uma estratégia de aprendizado que envolve a criação de pequenos desafios para testar seu conhecimento.

Como aplicar?

- Após estudar um tópico, formule perguntas como:
 - O que é um hábito?
 - O que é procrastinação?
 - Qual a relação entre os dois?
- Responda sem consultar o material.
- Revise suas respostas e ajuste o que for necessário.

Esse método ativa o recall ativo e reforça a compreensão do conteúdo.

Intercalação de Estudos (Interleaving)

Muitas pessoas estudam um tópico de cada vez até dominá-lo completamente. Entretanto, pesquisas mostram que alternar entre diferentes tópicos melhora o aprendizado, pois ensina o cérebro a escolher a estratégia certa para cada situação.

Exemplo prático:

- Se você está aprendendo a programar, em vez de estudar apenas loops por um dia inteiro, misture exercícios de condições, funções e estruturas de dados ao longo da semana.
- Isso melhora a capacidade de aplicar conceitos em situações diferentes.

Constância

O aprendizado é um processo fisiológico que requer tempo e prática. Assim como na academia, onde o treino constante traz resultados, no estudo, a regularidade é essencial para fortalecer as conexões neurais.

- É melhor estudar 1 hora por dia durante 6 meses do que estudar 10 horas seguidas e depois ficar dias sem praticar.
- As memórias se consolidam durante o sono, portanto, é essencial manter uma rotina de descanso adequada.

Conclusão

Se você seguir essas técnicas, perceberá uma melhora significativa na sua capacidade de aprendizado e retenção de informação. Lembre-se:

1. **Use o Pomodoro** para criar um ambiente produtivo.
2. **Pratique o Recall Ativo** para reforçar suas memórias.
3. **Utilize a Prática Deliberada** para testar seus conhecimentos.
4. **Intercale seus estudos** para aprender a aplicar os conceitos.
5. **Mantenha a constância** para garantir um aprendizado duradouro.

Com esses métodos, você estará mais preparado para enfrentar desafios acadêmicos e profissionais de forma eficiente e estruturada.

09. Como aprender a programar

Aprender a programar é, antes de tudo, aprender a resolver problemas. A programação não é apenas sobre escrever código, mas sim desenvolver um raciocínio lógico e investigativo.

Sou um programador autodidata, e grande parte dos professores da Asimov também são. Em uma área dinâmica como a programação, a habilidade de aprender sozinho é essencial. Novas tecnologias, paradigmas, linguagens e modelos de IA surgem constantemente, tornando impossível depender apenas de cursos e professores para acompanhar todas as mudanças.

Na Asimov, temos o compromisso de ajudar vocês a atingirem seus objetivos. Nosso objetivo não é que vocês dependam dos professores, mas sim que aprendam a aprender. Dessa forma, vocês terão autonomia para continuar evoluindo mesmo após o curso.

Para isso, vamos compartilhar alguns conceitos e práticas fundamentais que ajudarão vocês a desenvolver essa mentalidade autodidata.

Desenvolva a Mentalidade de Programador

A programação é, essencialmente, a arte de resolver problemas. Um bom programador tem a mentalidade de que **todo problema tem uma solução – basta encontrá-la**.

Buscando Soluções

Google: seu melhor amigo O Google é a ferramenta mais poderosa para programadores. A maioria das dúvidas e erros que você enfrentará já foram resolvidos por alguém antes. Saber como pesquisar corretamente é uma habilidade essencial.

Dicas para uma busca eficiente:

- Para dúvidas conceituais, consulte a documentação oficial da linguagem que está estudando.
- Para erros específicos, o Stack Overflow é a melhor fonte – quase todo erro que você encontrar já tem uma resposta lá.
- Para tutoriais e explicações mais detalhadas, o blog da Asimov e artigos especializados podem ser ótimas referências.

Uso Inteligente do ChatGPT e Outras IAs Ferramentas como ChatGPT são úteis para explicar conceitos, corrigir erros e sugerir melhorias em códigos. Mas é importante usá-las para **aprender**, não apenas para obter respostas prontas.

Se você pedir para uma IA corrigir um erro, tente entender o motivo da correção. Isso fará com que você evolua muito mais rápido.

Participação em Comunidades A programação não precisa ser uma jornada solitária. Existem comunidades onde programadores compartilham conhecimento e se ajudam.

- A Asimov tem uma **comunidade exclusiva para alunos**, onde você pode interagir com colegas e tirar dúvidas.
- Outras comunidades ativas estão no **Discord, Telegram, Stack Overflow e Reddit**.
- Compartilhar dúvidas e projetos com outras pessoas acelera o aprendizado e permite que você conheça diferentes abordagens para resolver problemas.

Aproveite o suporte da Asimov Os alunos da Asimov têm acesso a um suporte qualificado, onde os professores respondem dúvidas com explicações detalhadas. Além disso, eles oferecem dicas de carreira e orientação personalizada para o seu aprendizado.

Aprendendo com Programadores Mais Experientes

Se você quer se tornar um bom programador, deve estudar como os programadores mais experientes estruturam seus códigos e resolvem problemas.

Estudando Códigos Prontos

- Os alunos da Asimov têm acesso aos códigos-fonte dos projetos dos professores. Estude esses códigos e tente entender por que cada decisão foi tomada.
- Antes de assistir às explicações, analise os códigos sozinho e tente encontrar padrões.
- Explore o **GitHub**, o maior repositório de código aberto do mundo. Lá, você pode ver como outros programadores escrevem seus códigos e aprender com projetos reais.

Leitura de Documentação

Bibliotecas e frameworks ampliam as possibilidades de uma linguagem. Para usá-los da melhor forma, é essencial ler suas documentações oficiais.

Por exemplo, a biblioteca **Streamlit**, muito usada para criar Web Apps, tem uma documentação completa que ensina a utilizar suas funcionalidades corretamente. Esse hábito tornará você mais independente ao programar.

Criando Projetos para Aprender na Prática

A melhor maneira de aprender programação é aplicando os conhecimentos em projetos reais.

Modificando Projetos Existentes

- Pegue um projeto pronto e **adicione novas funcionalidades**.
- Mude a estrutura e os estilos para personalizá-lo.
- Transforme um código genérico em algo útil para você.

Isso torna o aprendizado mais envolvente e significativo.

Compartilhando Seus Projetos

Mesmo que seu projeto pareça simples, ele pode ajudar outras pessoas.

- Publique no **GitHub** para que outras pessoas possam ver e dar feedback.
- Compartilhe na **comunidade da Asimov** para aprender com sugestões dos colegas.

A programação é uma jornada contínua, e ensinar o que você aprendeu reforça ainda mais o seu conhecimento.

Conclusão

Se você deseja se tornar um programador independente e capaz de resolver qualquer problema, siga estas estratégias:

1. **Desenvolva a mentalidade de programador** → Todo problema tem solução, basta encontrá-la.
2. **Saiba buscar informações** → Use o Google, Stack Overflow e outras fontes confiáveis.
3. **Aprenda com programadores experientes** → Leia códigos prontos e documentações.
4. **Crie projetos práticos** → A melhor forma de aprender é programando.
5. **Compartilhe e participe da comunidade** → Interaja com outros programadores para acelerar seu aprendizado.

Com essas técnicas, você terá autonomia para continuar evoluindo na programação, independentemente das mudanças na tecnologia.

10. Apresentação do projeto

Este projeto tem como objetivo criar um chatbot inteligente capaz de interagir com diferentes tipos de dados, como sites, vídeos do YouTube e documentos em PDF. Utilizando a biblioteca LangChain e modelos de linguagem avançados, o chatbot é projetado para responder perguntas de forma automatizada, extraindo informações relevantes dos conteúdos fornecidos.

O funcionamento do chatbot é simples e intuitivo. O usuário pode escolher entre três opções de interação: conversar com um site, com um PDF ou com um vídeo do YouTube. Após a seleção, o chatbot carrega o conteúdo escolhido e o utiliza como base para responder às perguntas do usuário. Por exemplo, se o usuário optar por interagir com um site, o chatbot extrai o texto da página e o utiliza para gerar respostas contextualizadas. O mesmo acontece com PDFs e vídeos do YouTube, onde o chatbot transcreve o conteúdo e o utiliza para fornecer respostas precisas.

O chatbot, chamado Asimo, foi projetado para ser amigável e eficiente. Ele utiliza um modelo de linguagem poderoso para processar as informações e gerar respostas que fazem sentido com base no contexto fornecido. Além disso, o projeto é altamente modular, permitindo que novos tipos de dados e funcionalidades sejam adicionados no futuro.

Este projeto é um exemplo prático de como a inteligência artificial pode ser aplicada para automatizar tarefas e melhorar a interação com informações digitais. Ele demonstra o poder do Python e de bibliotecas como LangChain para criar soluções inovadoras e acessíveis, mesmo para quem está começando no mundo da programação e da IA.

11. Interagindo com o usuário – Variáveis e tipos de dados

Vamos iniciar a nossa jornada com o código em Python. Se você é iniciante, não se preocupe, pois aqui aprenderemos desde os primeiros passos. Mesmo aqueles que já têm alguma experiência certamente encontrarão valor neste conteúdo. A programação é uma habilidade em constante evolução e sempre podemos aprender algo novo.

Antes de começarmos, é importante que você faça o download dos arquivos do curso, que incluem um arquivo zip contendo os materiais necessários. Dentro do arquivo, você encontrará pastas com os arquivos que utilizaremos, além de aulas completas que estarão disponíveis ao final do curso. É fundamental ter acesso a esses arquivos para que você possa verificar o conteúdo, mas sempre que possível, deve tentar refazer os códigos por conta própria, pois a prática é essencial para consolidar o aprendizado.

Trabalhando no Google Colab

Vamos usar o Google Colaboratory, uma ferramenta muito útil para escrever e executar código em Python. No Colab, você encontrará células de texto e células de código. As células de código são identificadas por um ícone de “play”, que permite executá-las. Para adicionar uma nova célula, você pode escolher entre criar uma de código ou uma de texto, conforme a necessidade.

Uma vantagem de usar o Google Colab é que ele permite que nós usemos o Python sem precisar instalá-lo localmente. Isso pode ser bem útil para pessoas que por algum motivo não possam instalar o python em sua máquina.

Além disso, o Colab tb permite a integração com o google drive o que pode facilitar a execução de algumas tarefas. ### Exemplo de Interação com o Usuário

Começaremos a desenvolver um chatbot simples. Para isso, escreveremos um pequeno trecho de código que pedirá o nome do usuário e o cumprimentará. Para rodar o código, você deve usar `Ctrl + Enter`.

```
nome = input("Digite seu nome para começarmos: ")
print(f"Olá {nome}, o que você gostaria de saber?")
```

Nesse exemplo, utilizamos a função `input()`, que permite capturar informações do usuário, e `print()`, que exibe mensagens na tela. Assim, conseguimos interagir com o usuário, armazenando suas respostas para uso posterior.

Compreendendo Variáveis

Um conceito fundamental que precisamos entender é o de variáveis, que são formas de armazenar dados na memória do computador. Por exemplo:

```
nome_pessoa = "Adriano"
idade_pessoa = 32
altura_pessoa = 1.74
```

Essas variáveis funcionam como rótulos para os dados que estão sendo armazenados e são essenciais para o desenvolvimento do nosso programa.

Tipos de Dados

No Python, os tipos de dados categorizam os valores que podemos usar em um programa. Os principais tipos incluem:

- **Integers (int):** Números inteiros.
- **Floats (float):** Números com casas decimais.
- **Strings (str):** Texto, que deve ser delimitado por aspas.

Exemplo de Tipos de Dados

```
idade = 32                # int
altura = 1.74             # float
nome = "Adriano"          # str
```

Com esses dados, também podemos realizar operações aritméticas. Aqui estão alguns exemplos:

```
idade = 32
print(idade ** 2)  #Exponenciação
print(idade / 2)   # Divisão
```

Entrada de Dados e Formatação de Strings

Ao capturarmos dados do usuário com a função `input()`, podemos tornar as interações mais dinâmicas:

```
idade = input("Qual é a sua idade? ")
print(f"Sua idade é {idade}.")
```

A formatação de strings pode ser feita com f-strings, incorporando variáveis diretamente na string:

```
idade = 32 print(f"Sua idade é {idade}.")
```

Conclusão da Aula

Ao final desta aula, cobrimos como interagir com o usuário, armazenar informações e apresentar resultados. Essa base nos permitirá construir projetos cada vez mais complexos. Na próxima aula, continuaremos a desenvolver nosso chatbot e explorar novos conceitos. Vamos em frente!

12. Iterações repetidas – Loops e condicionais

Agora que vocês já têm uma boa base de programação, estamos prontos para evoluir e dar mais um passo em nossa jornada. Nesta segunda aula, vamos discutir um conceito importante: as estruturas de repetição. Em nosso projeto, que envolve a construção de um chatbot, precisaremos interagir com o usuário um número indefinido de vezes. A chave para isso é utilizar loops, que nos permitem repetir um bloco de código sem necessidade de reescrevê-lo várias vezes.

Gerenciando Sessões no Google Collaboratory

Antes de entrarmos nos loops, é importante saber que o Google Collaboratory tem um limite de sessões abertas ao mesmo tempo. Se você abrir muitas páginas, pode receber uma mensagem para fechar algumas delas. Para evitar esse problema, gerencie suas sessões fechando as que não está mais utilizando.

Construindo o Loop

O que vamos aprender nesta aula é como criar um loop usando a estrutura `while`. Ao começarmos, façamos uma pequena interação, onde primeiro pediremos ao usuário para digitar seu nome e fazer uma pergunta.

Por exemplo:

```
nome = input("Digite seu nome: ")
pergunta = input(f"Olá {nome}, como posso ajudar? ")
```

A resposta do bot será desenvolvida posteriormente, mas já podemos observar que, ao interagir, o usuário pode fazer diversas perguntas. Para que o usuário possa sair dessa iteração, precisaremos de uma condição de saída.

Operadores de Comparação

Os operadores de comparação são essenciais para determinar quando devemos sair de um loop. Em Python, temos diversos operadores, como:

- **Igual a (==)**
- **Diferente de (!=)**
- **Maior que (>)**
- **Menor que (<)**
- **Maior ou igual (>=)**

- **Menor ou igual (<=)**

Por exemplo, para verificar se o usuário deseja sair do loop, podemos usar um input:

```
pergunta = input("Digite 'x' se você quiser sair: ")
```

Lembre-se de que o valor retornado pelo `input` é sempre uma string, então precisamos compará-lo adequadamente.

Lidando com Erros

Um bom programador reconhece que cometer erros faz parte do aprendizado. Ao digitar um valor não definido, como `x` sem aspas, você receberá um erro. A mensagem “NameError” indica que a variável `x` não está definida. O importante é aprender a interpretar esses erros e corrigi-los, entendendo a lógica por trás do código.

Validação de Entrada

Para garantir que o código funcione corretamente, também podemos usar o método `.lower()` para tratar a entrada do usuário. Desse modo:

```
if pergunta.lower() == 'x':
```

Isso permitirá que independentemente do uso de letra maiúscula ou minúscula, o programa reconhecerá o comando de saída.

Estruturas Condicionais

Agora, vamos introduzir as estruturas condicionais, começando pelo `if`. O `if` é utilizado quando desejamos criar bifurcações em nosso código. Por exemplo, em um cenário onde maiores de idade têm acesso a uma balada e menores de idade não, poderíamos escrever:

```
if idade >= 18:
    print("Você pode entrar na balada.")
else:
    print("Desculpe, você precisa ir embora.")
```

A indentação é essencial em Python; ela determina quais linhas de código pertencem ao bloco do `if` ou do `else`.

Criando o Loop no Chatbot

Vamos implementar o loop em nosso chatbot. A ideia é que o bot continue perguntando até que o usuário decida sair. Começaremos com um loop `while`, que executo indefinidamente.

```
while True:
    pergunta = input("Digite sua pergunta (ou 'x' para sair): ")
    if pergunta.lower() == 'x':
        print("Muito obrigado por usar o AsimoBot!")
        break
    # Aqui você pode inserir a lógica para responder a pergunta
```

Esse código permite que o chatbot interaja continuamente até que o usuário digite x, momento em que o loop é encerrado usando a instrução `break`.

Conclusão

Nesta aula, aprendemos sobre operadores de comparação e a construção de estruturas condicionais. Além disso, introduzimos o conceito de loops, especialmente o loop `while`, que é fundamental para a interação contínua em nosso chatbot. É importante praticar e não depender apenas da memória. Tentem refazer todos os códigos por conta própria. Essa prática é o que realmente solidificará seu aprendizado. Nos vemos na próxima aula!

13. Armazenando interações – Listas, dicionários e funções

Neste capítulo, serão abordados conceitos fundamentais para armazenar interações em Python, focando em duas estruturas de dados essenciais: listas e dicionários. O objetivo é permitir que um chatbot armazene e gerencie o histórico das conversas com o usuário, preservando as interações durante o tempo de execução.

Conceitos Fundamentais Revisados

Antes de entrar nas novas estruturas, é importante relembrar os conceitos fundamentais aprendidos nas aulas anteriores:

1. **Variáveis:** Estruturas que armazenam valores, podendo ser informados pelo usuário ou definidos no código.
2. **Tipos de Dados:** Os principais tipos incluem:
 - Strings: dados de texto.
 - Integers (ints): números inteiros.
 - Floats: números decimais.
 - Booleans: valores verdadeiros ou falsos.
3. **Interação com o Usuário:**
 - `print()`: exibe informações ao usuário.
 - `input()`: captura informações do usuário.
4. **Estruturas de Repetição:** Utilização do `while` para executar um bloco de código repetidamente enquanto uma condição for verdadeira. Para sair do loop, utiliza-se a palavra-chave `break`.
5. **Estruturas Condicionais:** O uso do `if` para bifurcação do fluxo do código, acompanhado por operadores de comparação, como igual a (`==`) ou diferente de (`!=`).

Estruturas de Dados

As listas e dicionários serão utilizados para armazenar interações do usuário. Essas estruturas são fundamentais para manter um histórico de conversas em um chatbot.

Listas

Em Python, uma lista é uma coleção ordenada de itens que pode armazenar múltiplos valores em uma única variável. A sintaxe para criar uma lista é:

```
mensagens = []
```

Os elementos são adicionados a uma lista mediante o método `append()`:

```
mensagens.append("Olá")
mensagens.append("Tudo bem?")
```

Para acessar os itens da lista, utiliza-se o índice, que começa em 0:

```
print(mensagens[0]) # Imprime "Olá"
```

O método `pop()` permite remover itens da lista. Para remover o último item:

```
mensagens.pop(-1)
```

Além disso, é possível realizar slicing em listas para acessar subconjuntos de elementos:

```
print(mensagens[0:2]) # Retorna os primeiros dois itens
```

Dicionários

Dicionários são estruturas que armazenam pares de chave e valor, permitindo acessar rapidamente os dados associados a uma chave específica. A sintaxe de um dicionário é:

```
pessoas = {
    "Maria": 30,
    "João": 25,
    "Ana": 28
}
```

Para acessar um valor de um dicionário, utiliza-se a chave correspondente:

```
print(pessoas["Maria"]) # Imprime 30
```

A adição de novos pares chave-valor é realizada da seguinte forma:

```
pessoas["Adriano"] = 32
```

Para remover um item do dicionário, utiliza-se a palavra-chave `del`:

```
del pessoas["Adriano"]
```

Funções

As funções são blocos de código reutilizáveis que executam tarefas específicas. Para criar uma função, utiliza-se a palavra-chave `def`. Um exemplo é a função para somar dois valores:

```
def somar(a, b):  
    return a + b
```

Para chamar a função e armazenar o resultado:

```
resultado = somar(1, 2)  
print(resultado) # Imprime 3
```

Implementação no Chatbot

Para integrar os conceitos de listas e dicionários ao projeto do chatbot, cria-se uma lista vazia para armazenar as mensagens trocadas com o usuário. O código pode se estruturar da seguinte forma:

```
mensagens = []  
while True:  
    pergunta = input("Digite sua pergunta (ou 'x' para sair): ")  
    if pergunta.lower() == 'x':  
        break  
    mensagens.append({"role": "user", "content": pergunta})
```

Para armazenar as respostas do bot, a função de resposta pode ser definida, embora neste estágio ela retorne um valor padrão.

Conclusão

Nesta aula, foram abordadas as estruturas de dados listas e dicionários, além de funções como blocos de código reutilizáveis. Essas estruturas permitirão ao chatbot armazenar e gerenciar o histórico das interações com o usuário. A próxima etapa incluirá o aprimoramento do bot, fornecendo respostas mais elaboradas e personalizadas.

14. Aumentando o poder do seu script com bibliotecas

Neste capítulo, será abordado o conceito de bibliotecas em Python e como elas podem expandir as capacidades de um script. Após dominar os fundamentos da programação, o conhecimento sobre bibliotecas permitirá o uso de ferramentas adicionais que facilitam e potencializam o desenvolvimento de aplicações.

O que são Bibliotecas em Python?

As bibliotecas em Python podem ser comparadas a caixas de ferramentas repletas de ferramentas prontas para uso. Elas economizam tempo e esforço, pois permitem que programadores utilizem funcionalidades já desenvolvidas por outros, em vez de criar tudo do zero. Qualquer pessoa pode criar uma biblioteca, disponibilizando seus scripts para a comunidade.

Python é uma linguagem de código aberto, o que permite que desenvolvedores contribuam com novas bibliotecas e funcionalidades. Isso fomenta uma rica ecossistema de bibliotecas, que podem ser utilizadas em diversos projetos.

Bibliotecas Padrão

Python já inclui diversas bibliotecas na sua biblioteca padrão, que podem ser utilizadas diretamente sem necessidade de instalação adicional. Exemplos incluem:

- **Math:** para operações matemáticas.
- **Random:** para geração de números aleatórios.
- **Datetime:** para manipulação de datas e horários.

Para usar uma biblioteca padrão, basta importá-la no seu código:

```
import math
```

Com a biblioteca `math`, é possível, por exemplo, calcular a raiz quadrada:

```
resultado = math.sqrt(25) # Retorna 5.0
```

Instalando Novas Bibliotecas

Quando se deseja utilizar bibliotecas que não estão incluídas nas bibliotecas padrão, é necessário instalá-las. Algumas bibliotecas populares, como `Pandas` e `Matplotlib`, são amplamente usadas para análise e visualização de dados. Para instalar uma biblioteca, utiliza-se o gerenciador de pacotes `pip`. O comando é o seguinte:

```
pip install nome_da_biblioteca
```

Esse comando deve ser executado em um terminal ou dentro de um notebook Jupyter, precedido pelo símbolo `!` se estiver em um ambiente de notebook:

```
!pip install nome_da_biblioteca
```

Principais Bibliotecas a Considerar

Neste curso, a biblioteca principal que será utilizada é a **LangChain**. Esta biblioteca é projetada para simplificar a construção de aplicativos que utilizam modelos de linguagem avançados, como o Chat-GPT. O LangChain permite a conexão com diversos modelos e APIs, funcionando como um facilitador na construção de aplicações.

Funcionalidades do LangChain

O LangChain possui recursos que permitem:

- **Criação de Chains:** Estruturas que facilitam a organização e interação entre diferentes componentes da aplicação.
- **Desenvolvimento de Agents:** Framework para a criação de agentes com diversas capacidades e ferramentas.

Um curso específico sobre a criação de agents com LangChain está disponível na plataforma e se propõe a aprofundar esses conceitos.

Conclusão

Com a compreensão do que são bibliotecas e como utilizá-las, será possível expandir significativamente a funcionalidade dos scripts desenvolvidos. A partir de agora, as bibliotecas serão um componente essencial em todos os projetos, especialmente a LangChain, que proporcionará ferramentas poderosas para a construção de aplicações de inteligência artificial. Na próxima aula, será iniciado o aprofundamento no conteúdo do LangChain.

15. Utilizando LangChain para Acessar o Llama

Neste capítulo, será abordado o acesso a um modelo de linguagem através da programação Python, utilizando a biblioteca LangChain para interagir com o modelo Llama 3 da Meta. A partir de agora, as interações serão realizadas por meio de código, eliminando a necessidade de interfaces padrão.

O que são Bibliotecas?

Bibliotecas em Python são comparáveis a caixas de ferramentas, contendo funcionalidades prontas para uso. Elas permitem que programadores utilizem recursos já desenvolvidos por outros, facilitando o acesso a novas funcionalidades sem a necessidade de construir cada ferramenta desde o início.

O LangChain, que será utilizado nesta aula, é esse facilitador, permitindo acesso a diversos modelos de linguagem, como o Llama 3, sem complicações. O acesso a modelos como os da OpenAI, Hugging Face e Google também é possível através desta biblioteca.

Acesso ao Modelo Llama 3

Para acessar o Llama 3, é necessário criar uma conta na plataforma Grok, que fornece a infraestrutura para rodar modelos de linguagem. Após o login, deve-se gerar uma API Key, que servirá como um identificador de acesso ao modelo.

Criando a API Key

1. Acesse a seção de API Keys na plataforma Grok.
2. Clique em “Create API Key” e forneça um nome para a chave.
3. Copie o valor gerado, pois ele será utilizado para autenticação.

Configurando o Ambiente

Durante o desenvolvimento do nosso projeto, utilizaremos a plataforma google cloud. E inicialmente, antes de mais nada, será necessário que nós venhamos a importar as bibliotecas que estamos usando até o momento. Note que é necessário fazer essa importação antes de qualquer outra coisa.



Figure 1: Instalando as bibliotecas necessárias

Carregue a sua api key em uma variável. Por exemplo:

```
api_key = "sua_api_key_aqui"
```

Para que o script reconheça a API Key, utilize a biblioteca os para criar uma variável de ambiente. O código é:

```
import os
os.environ['GROQ_API_KEY'] = api_key
```

Configurando o Modelo de Linguagem

Crie uma instância do modelo Llama 3, especificando qual modelo utilizar. O código é:

```
from langchain_groq import ChatGroq
chat = ChatGroq(model='llama-3.3-70b-versatile')
```

Realizando uma Chamada ao Modelo

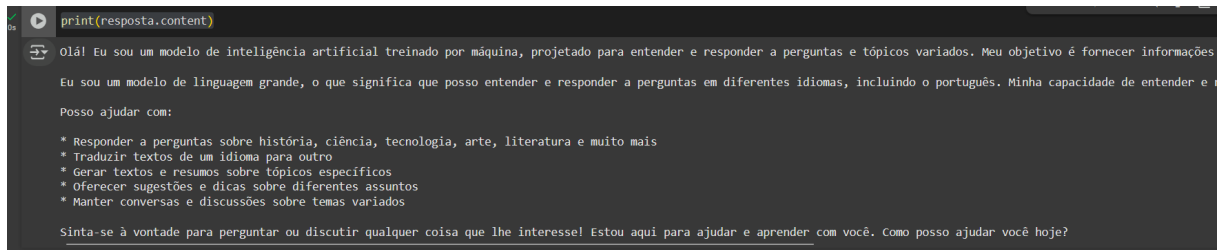
Agora, é possível interagir com o modelo, enviando perguntas e recebendo respostas. O código para enviar uma pergunta é:

```
resposta = chat.invoke('Olá, modelo! Quem é você?')
```

Em seguida, print a variável resposta para ver o resultado:

```
print(resposta.content)
```

Dessa forma, o modelo poderá retornar uma resposta, e o acesso a um modelo de linguagem se torna simples e direto através de poucas linhas de código.



```
print(resposta.content)

Olá! Eu sou um modelo de inteligência artificial treinado por máquina, projetado para entender e responder a perguntas e tópicos variados. Meu objetivo é fornecer informações e ajudar você em suas tarefas.

Eu sou um modelo de linguagem grande, o que significa que posso entender e responder a perguntas em diferentes idiomas, incluindo o português. Minha capacidade de entender e responder a perguntas é ampla.

Posso ajudar com:

* Responder a perguntas sobre história, ciência, tecnologia, arte, literatura e muito mais
* Traduzir textos de um idioma para outro
* Gerar textos e resumos sobre tópicos específicos
* Oferecer sugestões e dicas sobre diferentes assuntos
* Manter conversas e discussões sobre temas variados

Sinta-se à vontade para perguntar ou discutir qualquer coisa que lhe interesse! Estou aqui para ajudar e aprender com você. Como posso ajudar você hoje?
```

Figure 2: Instalando as bibliotecas necessárias

Conclusão

Nesta aula, foi abordada a utilização do LangChain para acessar o modelo Llama 3. A capacidade de interagir com modelos de linguagem por meio de código abre novas possibilidades para desenvolvedor. A próxima aula continuará a explorar o potencial de aplicações desenvolvidas com esses modelos.

16. Criando seu primeiro ChatBot

Neste capítulo, será abordado o processo de criação de um chatbot utilizando Python. Ao final, será possível interagir com um modelo de linguagem de maneira semelhante ao ChatGPT, estabelecendo uma estrutura básica para conversação.

Estrutura do Chatbot

O chatbot será construído com um código que permitirá a comunicação fluida entre o usuário e o modelo. O bot iniciará a conversa e responderá às perguntas de forma interativa.

Configuração Inicial

É necessário instalar algumas bibliotecas antes de começar, e cada capítulo poderá aumentar a lista de bibliotecas utilizadas. No início da célula, deve-se realizar a instalação das bibliotecas necessárias:

```
!pip install langchain
!pip install langchain-groq
```

Após rodar essa célula, as bibliotecas estarão disponíveis para uso.

Prompt Templates e Chains

Dentre os conceitos fundamentais do LangChain, dois se destacam: **Prompt Templates** e **Chains**. Um **Prompt Template** é uma estrutura que permite reutilizar prompts semelhantes, facilitando a interação com o modelo.

O que é um Prompt?

Um **prompt** é uma instrução ou solicitação enviada a um modelo de linguagem, servindo como entrada para gerar uma resposta. Ele consiste no texto que define o que se espera que o modelo faça. Por exemplo, um prompt pode ser uma pergunta ou uma frase que inicia uma conversa. O modelo de linguagem, ao receber esse texto, utiliza-o para produzir uma saída relevante.

Exemplo de Prompt:

```
prompt = "Qual é a capital da França?"
```

O que é um Prompt Template?

Um **prompt template** é uma estrutura de dados que facilita a criação de prompts reutilizáveis. Muitas vezes, é comum que certos prompts compartilhem uma estrutura semelhante, mudando apenas algumas informações específicas. Um prompt template permite que se defina uma estrutura básica e, posteriormente, substitua variáveis dentro desse template conforme a necessidade.

Por exemplo, se a intenção é criar um prompt que traduza expressões para diferentes idiomas, pode-se definir um template como este:

```
prompt_template = "Traduza a expressão '{expressao}' para a língua '{lingua}'."
```

Nesse caso, '{expressao}' e '{lingua}' são variáveis que podem ser substituídas durante a execução do código.

A criação de um prompt template facilita a manutenção e a organização do código ao permitir que o modelo seja facilmente alimentado com diferentes entradas, sem a necessidade de reescrever a estrutura do prompt a cada interação.

Criando o Chatbot

A criação do chatbot envolve a combinação de prompts e a utilização do modelo de linguagem. O código seguinte ilustra como estruturar o chatbot:

```
import os
from langchain_groq import ChatGroq
from langchain.prompts import ChatPromptTemplate

api_key = 'sua_api_key_aqui' # Substitua pela sua API Key
os.environ['GROQ_API_KEY'] = api_key
chat = ChatGroq(model='llama-3.3-70b-versatile')

def resposta_bot(mensagens):
    mensagens_modelo = [('system', 'Você é um assistente amigável chamado Asimov')]
    mensagens_modelo += mensagens
    template = ChatPromptTemplate.from_messages(mensagens_modelo)
    chain = template | chat
    return chain.invoke({}).content

print('Bem-vindo ao AsimoBot')

mensagens = []
while True:
    pergunta = input('Usuário: ')
    if pergunta.lower() == 'x':
        break
    mensagens.append(('user', pergunta))
```

```
resposta = resposta_bot(mensagens)
mensagens.append(('assistant', resposta))
print(f'Bot: {resposta}')

print('Muito obrigado por usar o AsimoBot')
print(mensagens)
```

Explicação do Código

- **Importação das Bibliotecas:**

```
import os
from langchain_grok import ChatGrok
from langchain.prompts import ChatPromptTemplate
```

Nesta parte, as bibliotecas necessárias são importadas. A biblioteca `os` é usada para definir variáveis de ambiente, enquanto `langchain_grok` fornece funcionalidades para acessar o modelo de linguagem, e `ChatPromptTemplate` facilita a criação de prompts.

- **Configuração da API Key:**

```
api_key = 'sua_api_key_aqui' # Substitua pela sua API Key
os.environ['GROQ_API_KEY'] = api_key
```

É definida a variável `api_key` com a chave de acesso à API. Esta chave é armazenada como uma variável de ambiente, permitindo identificação ao acessar o modelo.

- **Inicialização do Chat:**

```
chat = ChatGrok(model='llama-3.3-70b-versatile')
```

Uma instância do modelo de linguagem Llama 3 é criada.

- **Definição da Função para Obter Respostas do Bot:**

```
def resposta_bot(mensagens):
    mensagens_modelo = [('system', 'Você é um assistente amigável chamado Asimov')]
    mensagens_modelo += mensagens
    template = ChatPromptTemplate.from_messages(mensagens_modelo)
    chain = template | chat
    return chain.invoke({}).content
```

A função `resposta_bot` recebe uma lista de mensagens. Ela inicia a conversa adicionando uma mensagem do sistema que define o chatbot como “Asimov”. O `ChatPromptTemplate` é criado usando as mensagens acumuladas, e a função invoca o modelo, retornando a resposta.

- **Interação com o Usuário:**

```
print('Bem-vindo ao AsimoBot')

mensagens = []
while True:
    pergunta = input('Usuário: ')
    if pergunta.lower() == 'x':
        break
    mensagens.append(('user', pergunta))
    resposta = resposta_bot(mensagens)
    mensagens.append(('assistant', resposta))
    print(f'Bot: {resposta}')
```

Um loop permite que o usuário digite suas perguntas. Se o usuário digitar 'x', o loop é encerrado. As mensagens do usuário e do bot são armazenadas na lista mensagens.

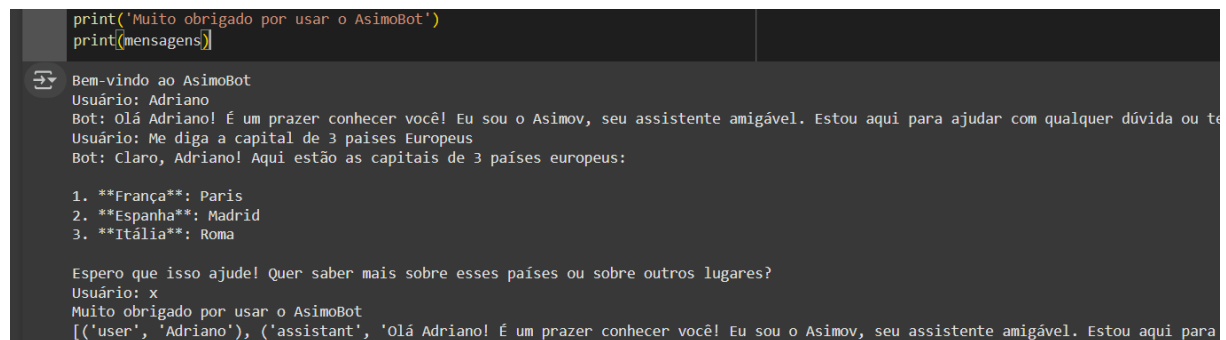
• Finalização da Interação:

```
print('Muito obrigado por usar o AsimoBot')
print(mensagens)
```

Ao final, uma mensagem de agradecimento é exibida, assim como o histórico da conversa armazenado.

Exemplo de Execução

Ao executar o chatbot, o usuário é saudado e tem a oportunidade de fazer perguntas. Um exemplo de interação poderia ser:



```
print('Muito obrigado por usar o AsimoBot')
print(mensagens)
```

Bem-vindo ao AsimoBot
Usuário: Adriano
Bot: Olá Adriano! É um prazer conhecer você! Eu sou o Asimov, seu assistente amigável. Estou aqui para ajudar com qualquer dúvida ou te
Usuário: Me diga a capital de 3 países Europeus
Bot: Claro, Adriano! Aqui estão as capitais de 3 países europeus:

1. **França**: Paris
2. **Espanha**: Madrid
3. **Itália**: Roma

Espero que isso ajude! Quer saber mais sobre esses países ou sobre outros lugares?
Usuário: x
Muito obrigado por usar o AsimoBot
[('user', 'Adriano'), ('assistant', 'Olá Adriano! É um prazer conhecer você! Eu sou o Asimov, seu assistente amigável. Estou aqui para

Figure 3: Instalando as bibliotecas necessárias

Conclusão

Nesta aula, foi possível construir um chatbot básico utilizando o LangChain e o modelo Llama 3. O uso de Prompt Templates e a estrutura do chatbot permitem uma interatividade fluida com o modelo de

linguagem. Este projeto estabelece uma base sólida para futuras implementações e novas funcionalidades. Na próxima aula, serão apresentadas novas ferramentas e técnicas para aprimorar ainda mais o chatbot.

17. Dando acesso a sites para o nosso bot

Neste capítulo, será abordada a integração de ferramentas do LangChain que permitem ao chatbot acessar informações externas. Esta capacidade é essencial para enriquecer as respostas do modelo de linguagem e tornar o chatbot mais útil e informativo.

O foco será em como utilizar um carregador de documentos para fazer scraping de um site, permitindo que o modelo tenha acesso a dados atualizados e específicos que não estão disponíveis na sua base de conhecimento padrão.

A importância do acesso externo

O acesso a informações externas é fundamental, pois os modelos de linguagem, como o LLaMA ou os desenvolvidos pela OpenAI, não conseguem compreender o contexto específico ou detalhes únicos do site de uma empresa, por exemplo, a menos que esses dados sejam fornecidos. O **web scraping** é uma técnica que nos permite extrair dados de websites automaticamente. Isso se torna uma ferramenta poderosa, uma vez que possibilita que os modelos acessem informações atualizadas e específicas, enriquecendo sua base de conhecimento e melhorando a relevância das respostas que fornecem.

Utilizando o **LangChain**, podemos realizar o scraping de dados de maneira simplificada, capturando informações que variam desde textos em sites até dados estruturados em planilhas online. Este processo não apenas proporciona uma forma eficiente de coletar informações, mas também permite que os modelos de linguagem ofereçam respostas mais precisas e contextualizadas, fundamentadas em dados recentes e específicos de diferentes fontes.

Configuração Inicial

Para começar, precisamos ter as bibliotecas necessárias instaladas em nosso ambiente de desenvolvimento. O **LangChain**, entre outras funcionalidades, oferece ferramentas para realizar scraping diretamente de páginas web. A instalação pode ser feita através do comando:

```
!pip install langchain
!pip install langchain-groq
!pip install langchain-community
```

Uma vez que temos o **LangChain** instalado, podemos importar o módulo que nos permitirá acessar os dados da web com facilidade. A classe `WebBaseLoader` é uma das principais ferramentas para essa tarefa. Além dela, devemos importar também os outros módulos que são importantes para o funcionamento do nosso chatbot do capítulo passado:

```
from langchain.document_loaders import WebBaseLoader
import os
from langchain_groq import ChatGroq
from langchain.prompts import ChatPromptTemplate
```

Lógica e funcionamento

Com o módulo importado, estaremos prontos para realizar o scraping. Vamos supor que queremos acessar o site da **Asimov**. A seguir, definimos a URL e utilizamos o WebBaseLoader para extrair as informações:

```
# URL do site que queremos acessar
url = "https://asimov.academy"

# Criando o loader e realizando o scraping
loader = WebBaseLoader(url)
lista_documentos = loader.load()

# Exibindo a lista de documentos capturados
print(lista_documentos)
```

Após essa execução, a variável `lista_documentos` conterá todos os documentos extraídos da página, permitindo que tenhamos acesso a um conjunto de informações organizadas.

Agora que conseguimos capturar os dados, o próximo passo é processá-los e unificá-los em uma única string. Isso facilita a utilização das informações em interações ou quando criamos respostas contextuais. Podemos fazer isso facilmente com um loop em Python:

```
# Inicializando uma string vazia para armazenar o conteúdo
documento = ""

# Iterando sobre a lista de documentos para unir o conteúdo
for doc in lista_documentos:
    documento += doc.page_content
```

Assim, ao final do loop, teremos todos os conteúdos dos documentos reunidos na variável `documento`, prontos para serem utilizados.

Depois de unir as informações, podemos criar um template de chat que utilizará esses dados. O **LangChain** permite a criação de prompts de maneira fácil e eficiente, onde podemos especificar diferentes papéis para o modelo. Abaixo, definimos um template para um assistente que pode responder perguntas baseadas nas informações capturadas:

```
# Criando o template do prompt
template = ChatPromptTemplate.from_messages([

    {"role": "system", "content": "Você é um assistente chamado Asimov e acessa informações
    ↪ fornecidas para responder perguntas."},
```

```
    {"role": "user", "content": "Com base nas informações extraídas do site, quais são as  
↪ trilhas disponíveis na Asimov?\n\n{documentos_informados}"}  
])
```

Além disso, criaremos uma **cadeia de execução**, também chamada de **pipeline**. No LangChain, um **pipeline** é um fluxo automatizado onde a saída de um componente é passada diretamente como entrada para outro.

No nosso caso, utilizamos o operador `|` para conectar o **template** ao **modelo de IA**, garantindo que o prompt formatado seja enviado corretamente:

```
chain = template | chat
```

Com o template pronto, podemos agora invocar o chat e obter respostas contextuais. É aqui que a integração dos dados realmente brilha, permitindo que o assistente utilize as informações mais recentes do site para responder a perguntas. O código para a invocação do chat pode ser assim:

```
# Chamando o template e capturando a resposta  
resposta = chain.invoke({  
    "documentos_informados": documento,  
    "input": "Quais as trilhas disponíveis na Asimov?"  
})
```

```
# Exibindo a resposta do assistente  
print(resposta.content)
```

Dessa forma, conseguimos utilizar o conteúdo extraído da web para enriquecer as respostas do assistente. A habilidade de realizar web scraping e integrar dados externos é uma poderosa ferramenta que adiciona um valor significativo aos modelos de linguagem, permitindo respostas que são não apenas informativas, mas também altamente relevantes e atualizadas.

Código finalizado

A seguir, segue um exemplo do código completo, contendo também etapas que foram executadas nas aulas anteriores para facilitar o entendimento:

```
# Importação dos módulos  
import os  
from langchain_groq import ChatGroq  
from langchain.prompts import ChatPromptTemplate  
from langchain.document_loaders import WebBaseLoader  
  
# Definir a chave da API (substitua 'SUA_CHAVE_AQUI' pela chave correta)  
os.environ['GROQ_API_KEY'] = 'SUA_CHAVE_AQUI'  
  
# Inicializar o modelo de chat  
chat = ChatGroq(model='llama-3.3-70b-versatile')
```

```
# URL do site que queremos acessar
url = "https://asimov.academy"

# Criando o loader e realizando o scraping
loader = WebBaseLoader(url)
lista_documentos = loader.load()

# Exibindo a lista de documentos capturados
print(lista_documentos)

# Inicializando uma string vazia para armazenar o conteúdo
documento = ""

# Iterando sobre a lista de documentos para unir o conteúdo
for doc in lista_documentos:
    documento += doc.page_content

# Criando o template do prompt
template = ChatPromptTemplate.from_messages([
    {"role": "system", "content": "Você é um assistente chamado Asimov e acessa informações"},
    ↪ {"role": "system", "content": "fornecidas para responder perguntas."},
    {"role": "user", "content": "Com base nas informações extraídas do site, quais são as"},
    ↪ {"role": "user", "content": "trilhas disponíveis na Asimov?\n\n{documentos_informados}"}
])

# Criando a cadeia de execução (pipeline)
chain = template | chat

# Chamando o template e capturando a resposta
resposta = chain.invoke({
    "documentos_informados": documento,
    "input": "Quais as trilhas disponíveis na Asimov?"
})

# Exibindo a resposta do assistente
print(resposta.content)
```

A saída será algo parecido com isso:

```
... De acordo com as informações fornecidas, as trilhas disponíveis na Asimov são:

1. **Trilha Aplicações IA com Python**: Aprenda a criar aplicações de Inteligência Artificial com Python.
2. **Trilha Dashboards Interativos com Python**: Crie dashboards interativos e dinâmicos com Python.
3. **Trilha Python Office**: Aprenda a criar automações com Python do zero.
4. **Trilha Visão Computacional**: Aprenda a criar sistemas inteligentes com imagens e vídeos.
5. **Trilha Data Science e Machine Learning**: Domine a ciência de dados usando Python e Machine Learning.
6. **Trilha Análise e Visualização de Dados**: Analise dados profissionalmente usando Python.
7. **Trilha Trading Quantitativo**: Construa estratégias de trading usando o Método Científico.

Essas trilhas são oferecidas pela Asimov, uma plataforma de ensino de programação em Python que visa ajudar os alunos a desenvolver habili
```

Figure 4: saída do código

Conclusão

Em resumo, este capítulo mostrou como realizar o scraping de uma página web utilizando o **LangChain**, capturar dados e utilizá-los para construir um assistente que fornece respostas engajadas e contextualizadas. Na próxima aula, vamos explorar como trabalhar com dados de formatos como PDF e YouTube, ampliando ainda mais a nossa capacidade de analisar e extrair informações de diferentes fontes.

18. Acessando vídeos de Youtube e PDFs

Neste capítulo, daremos continuidade ao nosso curso. Nosso bot já tem a capacidade de acessar dados de sites, e agora vamos expandir essa funcionalidade para extrair informações do YouTube e de arquivos PDF. Com esses novos recursos, nosso assistente ficará ainda mais rico em dados, possibilitando respostas mais precisas e contextuais.

Ao integrar dados externos, nosso bot passa a ter acesso a conteúdos atualizados e específicos, que não estão disponíveis na base de conhecimento padrão dos modelos de linguagem. Nesta aula, veremos:

- **Acesso a vídeos do YouTube:** Extraindo as transcrições automáticas.
- **Acesso a arquivos PDF:** Integrando documentos armazenados, por exemplo, no Google Drive.

Preparação e Instalação das Bibliotecas

Antes de começar, certifique-se de que todas as bibliotecas necessárias estejam instaladas. Assim como no capítulo anterior, precisamos das bibliotecas do LangChain. Além delas, desta vez utilizaremos duas novas bibliotecas: a **youtube-transcript-api**, que permite carregar transcrições de vídeos do YouTube, e a **PyPDF2**, que será usada para carregar e manipular arquivos PDFs.

```
!pip install langchain
!pip install langchain-groq
!pip install langchain-community
!pip install youtube_transcript_api==0.6.2
!pip install pypdf==5.0.0
```

Em seguida, fazemos as importações necessárias e configuramos a chave de API para o ChatGroq:

```
import os
from langchain_groq import ChatGroq
from langchain.prompts import ChatPromptTemplate
from langchain.document_loaders import WebBaseLoader

# Definir a chave da API (substitua 'SUA_CHAVE_AQUI' pela sua chave correta)
os.environ['GROQ_API_KEY'] = 'SUA_CHAVE_AQUI'

# Inicializar o modelo de chat (estamos usando o Yama 3.1 com 70 bilhões de parâmetros)
chat = ChatGroq(model='llama-3.3-70b-versatile')
```

Extraindo Dados do YouTube

Para extrair informações de vídeos do YouTube, utilizaremos um loader específico que pega as transcrições automáticas. **Atenção:** É importante informar o idioma (neste caso, PT para português), pois

o loader utilizará as transcrições geradas automaticamente pelo YouTube.

Suponha que você queira utilizar um vídeo do Professor Rodrigo, aqui da Asimov, que aborda temas de programação. Primeiro, copie a URL do vídeo e, em seguida, crie o loader:

```
from langchain_community.document_loaders import YoutubeLoader

url = 'https://www.youtube.com/watch?v=nLopLwffsLI&ab_channel=AsimovAcademy'
loader = YoutubeLoader.from_youtube_url(
    url,
    language=['pt']
)
lista_documentos = loader.load()
```

Unificando o Conteúdo

Cada documento na lista possui um atributo `page_content` que contém o texto extraído. Agora, vamos unificar esse conteúdo em uma única string:

```
documento = ''
for doc in lista_documentos:
    documento = documento + doc.page_content
print(documento)
```

Você pode utilizar um `print(documento)` para conferir se o conteúdo foi extraído corretamente. Essa parte é fundamental para garantir que o modelo receba o contexto completo do vídeo.

Criando o Template e Invocando o Bot

Crie o template do prompt que será enviado para o modelo, indicando que as informações do YouTube devem ser consideradas na resposta:

```
template = ChatPromptTemplate.from_messages([
    ('system', 'Você é um assistente amigável que possui as seguintes informações para formular  
↪ uma resposta: {informacoes}'),
    ('user', '{input}')
])
```

Utilize, então, o mesmo mecanismo de pipeline que já aprendemos:

```
chain_youtube = template | chat

resposta = chain_youtube.invoke({'informacoes': documento, 'input': 'Aprender a programar é  
↪ difícil?'})

print(resposta.content)
```

O resultado, será algo semelhante a isso:

```
chain_youtube = template | chat
resposta = chain_youtube.invoke({'informacoes': documento, 'input': 'Aprender a programar é difícil?'})
print(resposta.content)
```

De acordo com o texto, aprender a programar pode ser difícil para algumas pessoas, especialmente aquelas que não têm uma predisposição natural. Ele também afirma que a lógica de programação é a habilidade mais importante que uma pessoa precisa ter para de fato conseguir utilizar o computador. Portanto, a resposta para a pergunta é que aprender a programar pode ser difícil para algumas pessoas, mas com as ferramentas certas e o tempo certo, é possível.

Figure 5: saída do código

Dica: Se desejar testar outras perguntas, basta alterar o valor do `input` na invocação.

Extraindo Dados de Arquivos PDF

Agora vamos integrar informações a partir de arquivos PDF. Neste exemplo, usaremos um PDF que contém o roteiro de uma viagem ao Egito, armazenado no Google Drive.

Montando o Google Drive

Para acessar arquivos do seu Google Drive, você precisa montá-lo no seu ambiente. No Google Colab, por exemplo, você vai clicar em “arquivos” e em seguida na opção “montar o drive”. Lembre-se que o drive deve estar montado para que o script consiga localizar o arquivo.

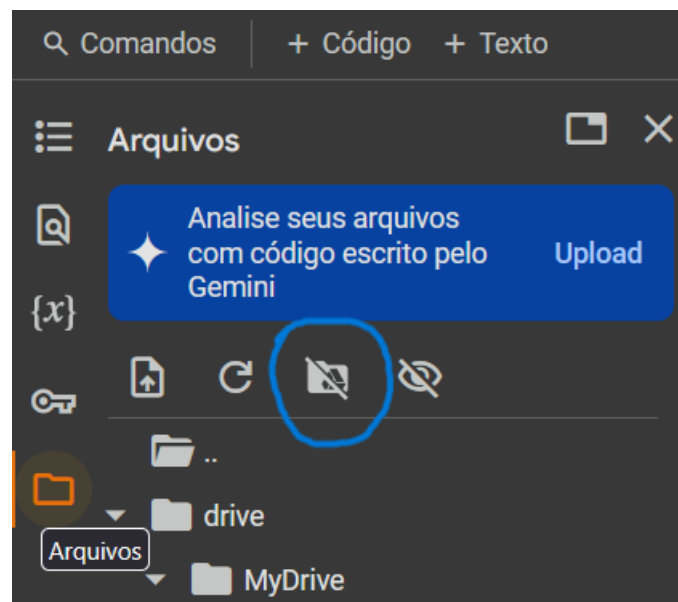


Figure 6: montando drive

Agora, faça o upload do arquivo e em seguida clique com o botão direito sobre ele e selecione a opção “copiar caminho”

Armazenaremos esse caminho em uma variável chamada “caminho”. Ficará mais ou menos assim:

```
caminho = "/content/drive/MyDrive colab/RoteiroViagemEgito.pdf"
```

Definindo o Loader do PDF

Utilize o loader específico para PDFs (neste exemplo, usaremos o PyPDFLoader):

```
from langchain_community.document_loaders import PyPDFLoader
loader = PyPDFLoader(caminho)

lista_documentos = loader.load()
```

Unificando o Conteúdo do PDF

Semelhante a lógica que usamos no webloader e com o YoutubeLoader, vamos unir todas as páginas do PDF em uma única string:

```
documento = ''
for doc in lista_documentos:
    documento = documento + doc.page_content
```

Criando o Template e Invocando o Bot para o PDF

Crie o template para que o modelo utilize as informações do PDF ao responder perguntas:

```
template = ChatPromptTemplate.from_messages([

    ('system', 'Você é um assistente amigável que possui as seguintes informações para formular
    ↪ uma resposta: {informacoes}'),

    ('user', '{input}'))

])
```

Agora, crie a cadeia de execução e invoque o modelo:

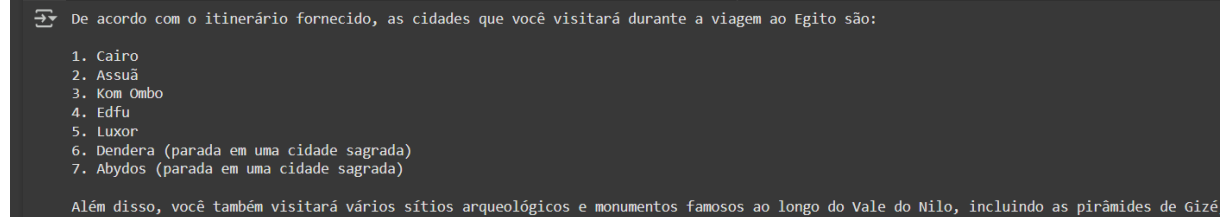
```
chain_youtube = template | chat

resposta = chain_youtube.invoke({'informacoes': documento, 'input': 'Por quais cidades
    ↪ passaremos na viagem?'})

print(resposta.content)
```

O resultado, será algo semelhante a isso:

```
resposta = chain_youtube.invoke({'informacoes': documento, 'input': 'Por quais cidades passaremos na viagem?'})
print(resposta.content)
```



De acordo com o itinerário fornecido, as cidades que você visitará durante a viagem ao Egito são:

1. Cairo
2. Assuã
3. Kom Ombo
4. Edfu
5. Luxor
6. Dendera (parada em uma cidade sagrada)
7. Abydos (parada em uma cidade sagrada)

Além disso, você também visitará vários sítios arqueológicos e monumentos famosos ao longo do Vale do Nilo, incluindo as pirâmides de Gizé

Figure 7: saída do código

Conclusão

Neste capítulo, aprendemos a expandir as capacidades do nosso bot para acessar dados externos de duas novas fontes:

- **YouTube:** Utilizando o YouTubeLoader para extrair transcrições automáticas (lembre-se de especificar o idioma).
- **PDFs:** Utilizando o PyPDFLoader para acessar documentos armazenados no Google Drive.

Com esses recursos, seu assistente passa a oferecer respostas mais ricas e contextualizadas, integrando dados atualizados de diferentes fontes. Essa abordagem abre caminho para a construção de aplicações ainda mais robustas e úteis.

19. Criando a estrutura final do nosso ChatBot

Nesse capítulo, daremos continuidade ao nosso projeto de chatbot. O objetivo é criar uma estrutura que permita ao usuário escolher qual fonte de dados utilizar como base para as respostas do bot. Poderá ser um site, um PDF ou um vídeo do YouTube. Em duas etapas, iremos:

- Preparar uma “casca” com toda a estrutura necessária.
- Finalizar a implementação do carregamento dos documentos.

Dica: Em programação, é importante reutilizar o código já desenvolvido. Copie ou crie funções para evitar reescrever o mesmo trecho várias vezes.

Usaremos como base, aquele chatbot básico que construímos no capítulo 16:

```
import os
from langchain_groq import ChatGroq
from langchain.prompts import ChatPromptTemplate

api_key = 'sua_api_key_aqui' # Substitua pela sua API Key
os.environ['GROQ_API_KEY'] = api_key
chat = ChatGroq(model='llama-3.3-70b-versatile')

def resposta_bot(mensagens):
    mensagens_modelo = [('system', 'Você é um assistente amigável chamado Asimov')]
    mensagens_modelo += mensagens
    template = ChatPromptTemplate.from_messages(mensagens_modelo)
    chain = template | chat
    return chain.invoke({}).content

print('Bem-vindo ao AsimoBot')

mensagens = []
while True:
    pergunta = input('Usuário: ')
    if pergunta.lower() == 'x':
        break
    mensagens.append(('user', pergunta))
    resposta = resposta_bot(mensagens)
    mensagens.append(('assistant', resposta))
    print(f'Bot: {resposta}')

print('Muito obrigado por usar o AsimoBot')
print(mensagens)
```

Fluxo do Projeto

Antes de iniciar as alterações no loop de conversa entre o usuário e o bot, precisamos apresentar uma forma de seleção de conteúdo. O fluxo será o seguinte:

1. Exibir opções para o usuário:

- Digitar **1** para conversar com um **site**.
- Digitar **2** para conversar com um **PDF**.
- Digitar **3** para conversar com um **vídeo do YouTube**.

2. Validar a entrada:

O sistema deverá continuar solicitando a entrada até que o usuário digite um valor válido (apenas “1”, “2” ou “3”).

3. Chamada das funções de carregamento:

Dependendo da escolha, serão chamadas funções específicas para:

- Carregar o conteúdo de um site.
- Carregar um PDF.
- Carregar um vídeo do YouTube.

4. Integração com o chatbot:

Após carregar o documento, este será utilizado para fornecer respostas ao usuário.

Exemplo de Loop de Seleção

A seguir, um exemplo de como implementar o loop de seleção utilizando a função `input` e condicionais:

```
texto_selecao = '''
Digite 1 se você quiser conversar com um site
Digite 2 se você quiser conversar com um pdf
Digite 3 se você quiser conversar com um vídeo de youtube
'''

while True:
    selecao = input(texto_selecao)
    if selecao == '1':
        documento = carrega_site()
        break
    if selecao == '2':
        documento = carrega_pdf()
        break
    if selecao == '3':
        documento = carrega_youtube()
```

```
break
print('Digite um valor entre 1 e 3')
```

Observação:

Lembre-se que o retorno da função `input` é sempre uma string. Portanto, ao comparar valores, utilize as aspas (ex.: "1", "2", "3").

Funções para Carregamento de Documentos

Para manter o código organizado, crie funções específicas para carregar cada tipo de documento. Seguem exemplos:

```
def carrega_site():
    url_site = input('Digite a url do site: ')
    documento = ''
    return documento

def carrega_pdf():
    documento = ''
    return documento

def carrega_youtube():
    url_youtube = input('Digite a url do vídeo: ')
    documento = ''
    return documento
```

Organização do Código

A estrutura geral do projeto está dividida em três partes:

1. Configuração inicial e importações:

Instalação de bibliotecas necessárias (lembre-se de executar as células na ordem correta).

2. Funções e estruturas auxiliares:

- Funções de carregamento dos documentos.
- Configuração da *language chain* para o chatbot.

3. Loop de conversa com o usuário:

- Primeiro, o loop de seleção para escolher o tipo de documento.
- Depois, o loop principal de conversa, que utiliza o documento carregado para responder às perguntas

Conclusão

No próximo capítulo, iremos finalizar a implementação do carregamento dos documentos e integrar completamente essa funcionalidade ao nosso chatbot.

20. Finalizando o projeto

Neste capítulo, vamos finalizar o nosso projeto. Para isso, iremos atribuir as funções de carregamento dos arquivos para podermos dialogar com as informações que passarmos para o nosso bot. É altamente recomendado que vocês busquem as funções das aulas anteriores para que não precisem reescrever do zero, além de usarem a base do código que fizemos no capítulo passado. O objetivo é permitir que o usuário escolha a fonte dos dados (site, PDF ou vídeo do YouTube) e que o conteúdo carregado seja utilizado para aprimorar as respostas do bot.

Lembre-se de instalar todas as bibliotecas necessárias antes de executar os códigos. Caso contrário, você poderá encontrar erros durante a execução.

```
!pip install langchain
!pip install langchain-groq
!pip install langchain-community
!pip install youtube_transcript_api==0.6.2
!pip install pypdf==5.0.0
```

Implementação dos Document Loaders

Na capítulo 19, as funções `carrega_site`, `carrega_pdf` e `carrega_youtube` estavam apenas declaradas sem funcionalidade. Agora, nós vamos implementá-las utilizando os **document loaders** do LangChain:

Carregamento de Sites

Vamos reaproveitar o que nós fizemos no capítulo 17 e adaptar aquela estrutura que dava acesso aos sites para o contexto da nossa aplicação:

```
from langchain_community.document_loaders import WebBaseLoader

def carrega_site():
    url_site = input('Digite a url do site: ')
    loader = WebBaseLoader(url_site)
    lista_documentos = loader.load()
    documento = ''
    for doc in lista_documentos:
        documento = documento + doc.page_content
    return documento
```

Esse código:

- Solicita uma URL ao usuário.
- Usa o **WebBaseLoader** para extrair o texto do site.

- Junta os textos extraídos em uma única string.(Com aquela estrutura que já conhecemos)

Carregamento de PDFs

Vamos reaproveitar o que nós fizemos no capítulo 18 e adaptar aquela estrutura que dava acesso ao PDF para o contexto da nossa aplicação:

```
from langchain_community.document_loaders import PyPDFLoader

def carrega_pdf():
    caminho = caminho = "/content/drive/MyDrive colab/RoteiroViagemEgito.pdf"
    loader = PyPDFLoader(caminho)
    lista_documentos = loader.load()
    documento = ''
    for doc in lista_documentos:
        documento = documento + doc.page_content
    return documento
```

Aqui:

- Um caminho fixo para um arquivo PDF é definido.
- O **PyPDFLoader** é usado para extrair o texto do documento.
- O texto extraído é concatenado em uma única string.

Carregamento de Vídeos do YouTube

Vamos reaproveitar o que nós fizemos no capítulo 18 e adaptar aquela estrutura que dava acesso ao Youtube para o contexto da nossa aplicação:

```
from langchain_community.document_loaders import YoutubeLoader

def carrega_youtube():
    url_youtube = input('Digite a url do vídeo: ')
    loader = YoutubeLoader.from_youtube_url(url_youtube, language=['pt'])
    lista_documentos = loader.load()
    documento = ''
    for doc in lista_documentos:
        documento = documento + doc.page_content
    return documento
```

Esse trecho:

- Solicita uma URL de vídeo do YouTube ao usuário.
- Utiliza o **YoutubeLoader** para extrair a transcrição do vídeo.
- Junta a transcrição em uma única string.

Mudanças na função resposta_bot

Na versão anterior, o chatbot utilizava apenas um prompt fixo. Agora, foi adicionado um espaço para inserir informações carregadas a partir de documentos externos:

Antes:

```
mensagens_modelo = [('system', 'Você é um assistente amigável chamado Asimo')]
```

Agora:

```
mensagem_system = '''Você é um assistente amigável chamado Asimo.  
Você utiliza as seguintes informações para formular as suas respostas: {informacoes}'''  
mensagens_modelo = [('system', mensagem_system)]
```

Passagem do Documento para o Chatbot

Antes, a função resposta_bot apenas recebia mensagens do usuário. Agora, também recebe o documento carregado:

Antes:

```
resposta = resposta_bot(mensagens)
```

Agora:

```
resposta = resposta_bot(mensagens, documento)
```

Exemplo do código final

```
#instalações necessárias  
!pip install langchain  
!pip install langchain-groq  
!pip install langchain-community  
!pip install youtube_transcript_api==0.6.2  
!pip install pypdf==5.0.0  
  
#Importações necessárias  
import os  
from langchain_groq import ChatGroq  
from langchain.prompts import ChatPromptTemplate  
from langchain.document_loaders import WebBaseLoader  
from langchain_community.document_loaders import WebBaseLoader  
from langchain_community.document_loaders import YoutubeLoader  
from langchain_community.document_loaders import PyPDFLoader  
  
#Configuração de API KEY e do modelo de linguagem  
api_key = 'Insira sua chave aqui'  
os.environ['GROQ_API_KEY'] = api_key  
chat = ChatGroq(model='llama-3.3-70b-versatile')
```

```
#Função de resposta do nosso bot
def resposta_bot(mensagens, documento):
    mensagem_system = '''Você é um assistente amigável chamado Asimo.
    Você utiliza as seguintes informações para formular as suas respostas: {informacoes}'''
    mensagens_modelo = [('system', mensagem_system)]
    mensagens_modelo += mensagens
    template = ChatPromptTemplate.from_messages(mensagens_modelo)
    chain = template | chat
    return chain.invoke({'informacoes': documento}).content

#Funções com os Loads de carregamento
def carrega_site():
    url_site = input('Digite a url do site: ')
    loader = WebBaseLoader(url_site)
    lista_documentos = loader.load()
    documento = ''
    for doc in lista_documentos:
        documento = documento + doc.page_content
    return documento

def carrega_pdf():
    caminho = '/content/drive/MyDrive/curso_ia_python/arquivos/RoteiroViagemEgito.pdf'
    loader = PyPDFLoader(caminho)
    lista_documentos = loader.load()
    documento = ''
    for doc in lista_documentos:
        documento = documento + doc.page_content
    return documento

def carrega_youtube():
    url_youtube = input('Digite a url do vídeo: ')
    loader = YoutubeLoader.from_youtube_url(url_youtube, language=['pt'])
    lista_documentos = loader.load()
    documento = ''
    for doc in lista_documentos:
        documento = documento + doc.page_content
    return documento

#Loop de interação com o usuario
print('Bem-vindo ao AsimoBot')

texto_selecao = '''Digite 1 se você quiser conversar com um site
Digite 2 se você quiser conversar com um pdf
Digite 3 se você quiser conversar com um vídeo de youtube '''

while True:
    selecao = input(texto_selecao)
    if selecao == '1':
        documento = carrega_site()
        break
    if selecao == '2':
        documento = carrega_pdf()
        break
    if selecao == '3':
```

```
    documento = carrega_youtube()
    break
print('Digite um valor entre 1 e 3')

mensagens = []
while True:
    pergunta = input('Usuario: ')
    if pergunta.lower() == 'x':
        break
    mensagens.append(('user', pergunta))
    resposta = resposta_bot(mensagens, documento)
    mensagens.append(('assistant', resposta))
    print(f'Bot: {resposta}')

print('Muito obrigado por usar o AsimoBot')
```

Conclusão

E assim, concluímos o nosso projeto. Note que algumas melhorias ficaram em aberto, como, por exemplo, a possibilidade de selecionar qualquer PDF, mas garanto que vocês conseguirão desenvolver isso sozinhos. O código está modular e simplificado, então vocês não terão dificuldades em fazer implementações extras, como essa e outras.

Agora é com vocês! Mãos à obra e bons experimentos!

21. O que fazer agora

Finalmente concluímos a nossa jornada em direção ao nosso primeiro Chatbot. Com essa conclusão, espera-se que os participantes tenham adquirido uma base sólida para explorar o potencial da linguagem Python e da IA. Tivemos como objetivo proporcionar um aprendizado prático e significativo, permitindo que os alunos compreendam como essas duas tecnologias, quando combinadas, podem se tornar uma ferramenta poderosa para a automação e a resolução de problemas.

O projeto desenvolvido ao longo do curso é um exemplo do que pode ser alcançado em um curto espaço de tempo. No entanto, as possibilidades de aplicação são vastas. Com mais tempo e dedicação, é possível expandir as funcionalidades e criar soluções ainda mais complexas e impactantes.

Direcionamentos para Aprofundamento

A partir deste ponto, os participantes podem explorar diversas áreas relacionadas à programação e à inteligência artificial. Abaixo estão algumas sugestões de tópicos e trilhas que podem ser úteis para quem deseja continuar aprendendo e desenvolvendo habilidades.

Aprofundamento em Aplicações de IA

Uma das áreas que podem ser exploradas é o aprofundamento em aplicações de IA. Isso inclui o estudo de diferentes tipos de modelos, como modelos que rodam localmente, modelos gratuitos e modelos pagos de provedores como OpenAI, Cohere e Google. Além disso, técnicas como **RAG (Retrieve Augmented Generation)** são essenciais para trabalhar com grandes volumes de dados, como livros ou documentos extensos.

Desenvolvimento de Agents Customizados

Outro tópico avançado é o desenvolvimento de **Agents Customizados**. Esses agents são capazes de tomar decisões e utilizar diversas ferramentas automaticamente, como extrair dados da internet ou de PDFs conforme a necessidade. Essa tecnologia permite a criação de soluções mais robustas e inteligentes, com um raciocínio próprio e um fluxo de pensamento mais estruturado.

Fundamentos de Data Science e Machine Learning

Para quem deseja entender a parte mais técnica por trás dos modelos de IA, é recomendável explorar os fundamentos de **Data Science e Machine Learning**. Esses conceitos são a base para a construção

de modelos de IA e incluem tópicos como estatística, algoritmos de machine learning e análise de dados.

Criação de Dashboards e Aplicações Web

Outra área de interesse é a criação de **Dashboards e Aplicações Web**. Isso permite transformar projetos desenvolvidos em ambientes como o Colab em ferramentas acessíveis para outras pessoas, como colegas de trabalho ou clientes. Aprender a construir interfaces web é essencial para quem deseja compartilhar suas soluções de forma mais ampla.

Análise e Tratamento de Dados

Por fim, a área de **Análise de Dados** é fundamental para quem deseja extrair o máximo valor de dados subutilizados. Isso inclui técnicas para limpeza, organização e análise de dados, permitindo a criação de insights e soluções mais eficientes.

Considerações Finais

Se você chegou até aqui, saiba que esse foi o primeiro passo em uma jornada de aprendizado contínuo. A programação e a inteligência artificial são áreas em constante evolução, e o conhecimento adquirido aqui pode ser expandido de diversas formas. Seja através de cursos avançados, projetos pessoais ou aplicações práticas, o importante é continuar explorando e desenvolvendo habilidades.

A programação tem o potencial de transformar a maneira como interagimos com a tecnologia e resolver problemas. Espera-se que este curso tenha despertado o interesse e a curiosidade para continuar aprendendo e aplicando esses conceitos em projetos cada vez mais complexos e impactantes.