



{JORNADA}  
PYTHON

AULA 4

# PYTHON DEV

```
def json_to_csv(json_file, csv_file):  
    with open(json_file) as f:  
        data = json.load(f)  
  
    # Create a CSV writer  
    csv_writer = csv.writer(csv_file)  
  
    # Write the header  
    csv_writer.writerow(data[0].keys())  
  
    # Write the data  
    for row in data:  
        csv_writer.writerow(row.values())
```



Parte 1

# Introdução

{JORNADA}  
**PYTHON**  
100% GRATUITO E ONLINE

# O que Vamos Aprender?

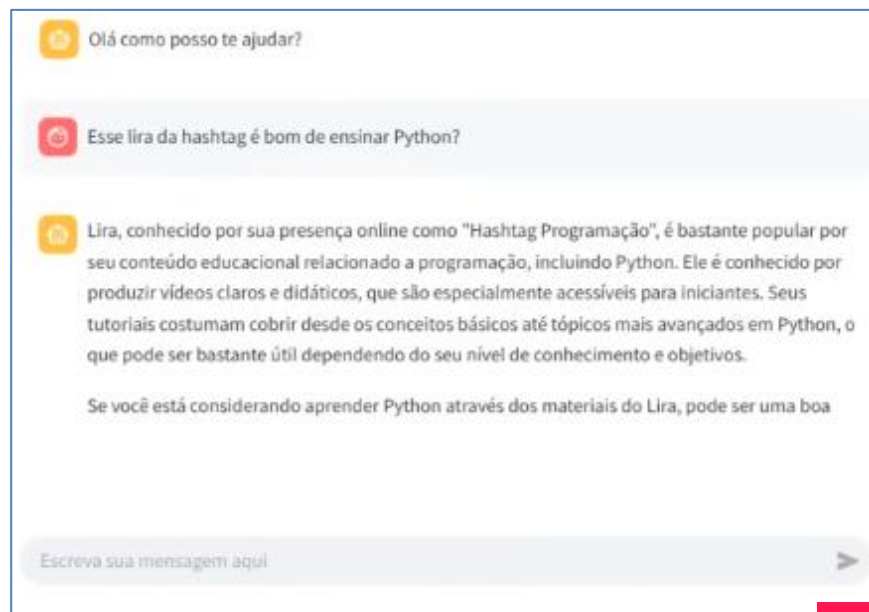
Na aula **Python Dev** nós vamos criar um Site com Inteligência Artificial. A proposta é desenvolver um ChatBot interativo, onde o usuário pode escrever mensagens e receber respostas automáticas, simulando um bate-papo com um assistente virtual. Essa funcionalidade é semelhante à que vemos em plataformas como o ChatGPT, mas agora será feita por você mesmo com Python.

A ideia é construir um chatbot completo, no qual a conversa acontece de forma dinâmica. O usuário envia uma pergunta, e o sistema responde utilizando tecnologias modernas de IA, como a OpenAI.

O mais interessante é que a aplicação será toda feita com a biblioteca Streamlit, que permite criar interfaces de sites diretamente com código Python de maneira simples e rápida. Sem precisar escrever códigos em HTML, CSS ou JavaScript.

Você vai entender como estruturar esse chat, como armazenar as mensagens, como criar a interface que simula um chat real, como conectar com a API da OpenAI (ChatGPT) e como controlar o comportamento da IA. Tudo será feito do zero, mesmo que você nunca tenha programado um site antes.

Dá uma olhada em como vai ficar o nosso site! Ao final da aula, teremos construído um verdadeiro chat com IA, com um layout limpo e moderno. O site exibe uma mensagem inicial de boas-vindas, e logo em seguida o usuário pode escrever sua pergunta no campo de texto. Assim que a mensagem é enviada, ela aparece no painel com um visual claro, enquanto a resposta da IA surge logo abaixo com outro visual.



Parte 2

# Bibliotecas Necessárias

# Instalação das Bibliotecas

Para esse projeto, vamos utilizar algumas bibliotecas diferentes das que usamos em aulas anteriores, já que agora o foco é criar um site com interface de chat e integração com inteligência artificial.

As bibliotecas que vamos instalar são:

- **Streamlit** – pip install streamlit
- **OpenAI** – pip install openai

A biblioteca **Streamlit** nos permitirá criar o site de forma simples usando apenas Python. Ela cuida automaticamente da parte visual (front-end) e do funcionamento da página (back-end).

Já a **OpenAI** é a biblioteca oficial da OpenAI. Através dela, conseguimos integrar a API do ChatGPT ao nosso programa, que nos permite fazer perguntas e receber respostas da inteligência artificial em tempo real.

Essas duas bibliotecas são fundamentais para que o projeto funcione. Caso você tente executar o código sem elas, vai encontrar mensagens de erro. Por isso, certifique-se de instalar tudo antes de começar a codar. Se quiser se aprofundar mais ou tiver dúvidas, aqui estão os links para a documentação oficial:

- **Streamlit** - <https://docs.streamlit.io/>
- **OpenAI** - <https://platform.openai.com/docs/overview>

Na documentação, você encontra exemplos de uso, formas de personalizar a interface, além de informações técnicas sobre parâmetros, funções e limitações. É sempre bom consultar essas fontes quando tiver dúvidas ou quiser fazer algo mais avançado.

Parte 3

# Conceitos Fundamentais do Python

{JORNADA}  
PYTHON

100% GRATUITO E ONLINE

# Conceitos Fundamentais do Python

## Listas

Antes de construir nosso chat com inteligência artificial, é importante entender como os dados da conversa serão organizados e utilizados dentro do código.

Para isso, vamos visualizar alguns conceitos e estruturas fundamentais do Python em um arquivo auxiliar: listas e dicionários.

As listas são conjuntos de informações organizadas em ordem. Por exemplo, podemos criar uma lista com nomes:


```
1 # lista
2 lista_nomes = ["lira", "jorge", "gi", "renan"]
```

Podemos adicionar novos nomes à essa lista com o método **.append()**:

```
1 # lista
2 lista_nomes = ["lira", "jorge", "gi", "renan"]
3 lista_nomes.append("julia") # adiciona informação em uma lista
4 print(lista_nomes)
```

E também podemos acessar os itens individualmente. Por exemplo, o primeiro item da lista (**posição 0**):

```
1 # lista
2 lista_nomes = ["lira", "jorge", "gi", "renan"]
3 lista_nomes.append("julia") # adiciona informação em uma lista
4 print(lista_nomes)
5
6 primeiro_item = lista_nomes[0]
7 print(primeiro_item)
```



```
● ['lira', 'jorge', 'gi', 'renan', 'julia']
lira
```



# Conceitos Fundamentais do Python

## Dicionários

Um dicionário em Python é uma estrutura onde temos chave e valor, desta forma: **{chave: valor, chave: valor}**

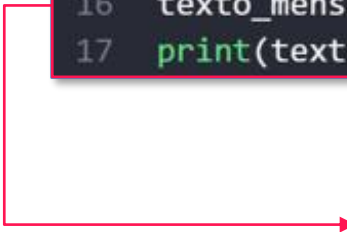
Dentro do nosso ChatBot, as mensagens serão definidas por quem enviou e pelo conteúdo da mensagem. Desse modo, essa estrutura será representada por um dicionário assim:

```
9  # dicionario python
10 # role = quem enviou a mensagem = "função"
11 # content = texto da mensagem = "conteudo"
12 # dicionario = {chave: valor, chave: valor}
13 mensagem = {"role": "user", "content": "Coe galera"}
```

Neste caso, "**role**" indica quem enviou a mensagem, e "**content**" é o conteúdo dela.

Podemos acessar as informações de um dicionário a partir da chave dele. Por exemplo, para acessar o conteúdo da mensagem:

```
9  # dicionario python
10 # role = quem enviou a mensagem = "função"
11 # content = texto da mensagem = "conteudo"
12 # dicionario = {chave: valor, chave: valor}
13 mensagem = {"role": "user", "content": "Coe galera"}
14
15 # 1 elemento -> dicionario[chave] -> valor
16 texto_mensagem = mensagem["content"]
17 print(texto_mensagem)
```



```
Python Dev/auxiliar.py"
Coe galera
```



# Conceitos Fundamentais do Python

## Juntando Lista + Dicionário

Também é possível combinar listas com dicionários, e é exatamente isso que iremos fazer para guardar o histórico completo de mensagens entre o usuário e o nosso ChatBot.

Cada mensagem será representada por um dicionário, e todas essas mensagens ficarão organizadas dentro de uma lista:

```
19 # lista + dicionario
20 lista_mensagens = [
21     {"role": "user", "content": "Coe galera"},
22     {"role": "assistant", "content": "Resposta da IA"},
23     {"role": "user", "content": "Tamo junto"}
24 ]
```

E da mesma forma que fizemos antes, podemos adicionar uma nova mensagem à lista:

```
26 lista_mensagens.append(
27     {"role": "assistant", "content": "Eu desisto de você"}
28 )
```

Para visualizar todas as mensagens de uma vez, podemos simplesmente imprimir a lista (**print(lista\_mensagens)**).

Ou, se quisermos exibir uma por uma, usamos um **for** para percorrer a lista:

```
30 print(lista_mensagens)
31
32 # exibir todos os itens de uma lista
33 for mensagem in lista_mensagens:
34     print(mensagem)
```

```
{'role': 'user', 'content': 'Coe galera'}
{'role': 'assistant', 'content': 'Resposta da IA'}
{'role': 'user', 'content': 'Tamo junto'}
{'role': 'assistant', 'content': 'Eu desisto de você'}
```

Essas estruturas são exatamente as mesmas que vamos usar no projeto real com IA.

Agora que você entendeu como funciona o armazenamento das mensagens, vai ser muito mais fácil acompanhar o restante do projeto.

Parte 4

# Criação de Sites e Sistemas com Python

{JORNADA}  
**PYTHON**  
100% GRATUITO E ONLINE

# Importando as Bibliotecas e Criando o Modelo

Agora que já instalamos as bibliotecas e entendemos como as mensagens serão armazenadas, chegou a hora de começar a construir nosso chat com inteligência artificial.

Todo o código será feito dentro do arquivo **main.py**. Nele, vamos criar a interface do site com **Streamlit**, exibir as mensagens e integrar com a IA da **OpenAI** para responder automaticamente o que o usuário digitar.

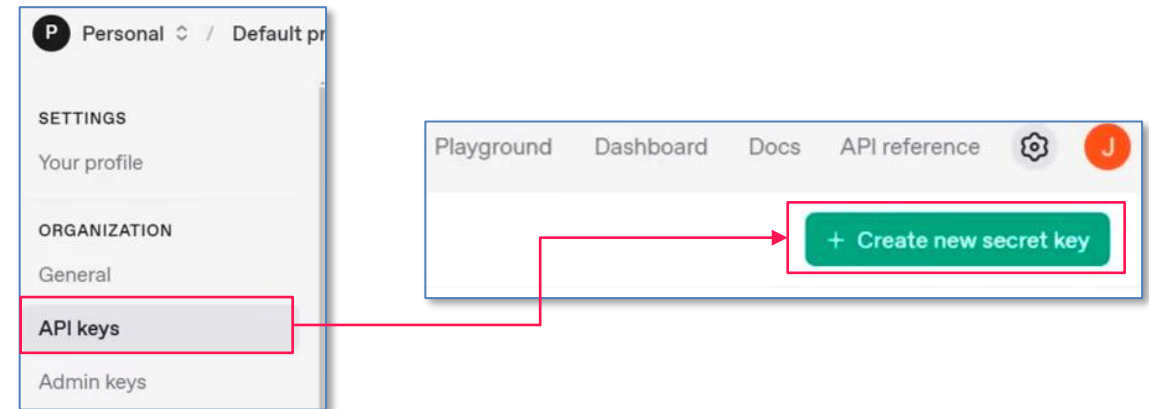
A primeira etapa é **importar as bibliotecas** que instalamos:

```
1 import streamlit as st
2 from openai import OpenAI
3
```

Em seguida, precisaremos **criar o modelo de IA**, porém, antes disso, você precisa **gerar a sua chave de API** da OpenAI, que é o que autoriza seu código a fazer perguntas e receber respostas da IA.

Siga esses passos:

- Acesse o site da OpenAI: <https://platform.openai.com>
- Faça login ou crie uma conta gratuita
- No menu lateral, clique em API Keys
- Clique em Create new secret key
- Copie a chave gerada e guarde com cuidado, pois ela não aparece novamente.



# Criando o Modelo e a Memória do ChatBot

Vamos usar essa chave para criar nosso modelo, substituindo o **"SUA\_CHAVE"** pela chave que você gerou:

```
1 import streamlit as st
2 from openai import OpenAI
3
4 modelo = OpenAI(api_key="SUA_CHAVE")
5
```

**Atenção:** Nunca compartilhe sua chave com outras pessoas. Essa chave está vinculada à sua conta e será usada para contabilizar os créditos utilizados pela API.

Com o modelo criado, agora vamos começar a construir o site. Primeiro, criamos um título com **st.write()**:

```
1 import streamlit as st
2 from openai import OpenAI
3
4 modelo = OpenAI(api_key="SUA_CHAVE")
5
6 st.write("### ChatBot com IA") # markdown
```

Agora, vamos criar a **memória do chat**. Como o Streamlit reinicia a página a cada nova ação do usuário, precisamos guardar as mensagens em uma estrutura que persiste entre as interações. Para isso, usaremos o **st.session\_state**:

```
1 import streamlit as st
2 from openai import OpenAI
3
4 modelo = OpenAI(api_key="SUA_CHAVE")
5
6 st.write("### ChatBot com IA") # markdown
7
8 # session_state = memoria do streamlit
9 if not "lista_mensagens" in st.session_state:
10     st.session_state["lista_mensagens"] = []
11
```

Essa lista será onde armazenamos todas as mensagens trocadas: do usuário e da IA.

# Exibindo o Histórico de Mensagens

Antes de capturar a nova mensagem, vamos exibir tudo que já foi enviado na conversa até agora:

```
1 import streamlit as st
2 from openai import OpenAI
3
4 modelo = OpenAI(api_key="SUA_CHAVE")
5
6 st.write("### ChatBot com IA") # markdown
7
8 # session_state = memoria do streamlit
9 if not "lista_mensagens" in st.session_state:
10     st.session_state["lista_mensagens"] = []
11
12 # exibir o histórico de mensagens
13 for mensagem in st.session_state["lista_mensagens"]:
14     role = mensagem["role"]
15     content = mensagem["content"]
16     st.chat_message(role).write(content)
```

Esse **for** percorre toda a lista e mostra cada mensagem no estilo visual de balão de chat (**st.chat\_message**), diferenciando quem enviou (**user** ou **assistant**).

# Exibindo as Mensagens do Usuário

Agora criamos o campo onde o usuário irá escrever sua mensagem (**chat\_input**) e, se o usuário digitar algo, salvamos a mensagem no histórico e a exibimos na tela:

```
1  import streamlit as st
2  from openai import OpenAI
3
4  modelo = OpenAI(api_key="SUA_CHAVE")
5
6  st.write("### ChatBot com IA") # markdown
7
8  # session_state = memoria do streamlit
9  if not "lista_mensagens" in st.session_state:
10     st.session_state["lista_mensagens"] = []
11
12  # exibir o histórico de mensagens
13  for mensagem in st.session_state["lista_mensagens"]:
14     role = mensagem["role"]
15     content = mensagem["content"]
16     st.chat_message(role).write(content)
17
18  mensagem_usuario = st.chat_input("Escreva sua mensagem aqui")
19
20  if mensagem_usuario:
21     # user -> ser humano
22     # assistant -> inteligencia artificial
23     st.chat_message("user").write(mensagem_usuario)
24     mensagem = {"role": "user", "content": mensagem_usuario}
25     st.session_state["lista_mensagens"].append(mensagem)
```



# Criação de Sites e Sistemas com Python

## Gerando a Resposta da IA

Depois que o usuário envia a pergunta, mandamos todo o histórico de mensagens para a IA. A IA usa esse contexto para gerar a resposta:

```
1 import streamlit as st
2 from openai import OpenAI
3
4 modelo = OpenAI(api_key="SUA_CHAVE")
5
6 st.write("### ChatBot com IA") # markdown
7
8 # session_state = memoria do streamlit
9 if not "lista_mensagens" in st.session_state:
10     st.session_state["lista_mensagens"] = []
11
12 # exibir o histórico de mensagens
13 for mensagem in st.session_state["lista_mensagens"]:
14     role = mensagem["role"]
15     content = mensagem["content"]
16     st.chat_message(role).write(content)
17
```

```
17
18 mensagem_usuario = st.chat_input("Escreva sua mensagem aqui")
19
20 if mensagem_usuario:
21     # user -> ser humano
22     # assistant -> inteligencia artificial
23     st.chat_message("user").write(mensagem_usuario)
24     mensagem = {"role": "user", "content": mensagem_usuario}
25     st.session_state["lista_mensagens"].append(mensagem)
26
27     # resposta da IA
28     resposta_modelo = modelo.chat.completions.create(
29         messages=st.session_state["lista_mensagens"],
30         model="gpt-4o"
31     )
32
33     resposta_ia = resposta_modelo.choices[0].message.content
34
```



# Criação de Sites e Sistemas com Python

## Exibindo a resposta da IA

Por fim, a resposta da inteligência artificial é exibida na tela e adicionada ao histórico de mensagens:

```
1 import streamlit as st
2 from openai import OpenAI
3
4 modelo = OpenAI(api_key="SUA_CHAVE")
5
6 st.write("### ChatBot com IA") # markdown
7
8 # session_state = memória do streamlit
9 if not "lista_mensagens" in st.session_state:
10     st.session_state["lista_mensagens"] = []
11
12 # exibir o histórico de mensagens
13 for mensagem in st.session_state["lista_mensagens"]:
14     role = mensagem["role"]
15     content = mensagem["content"]
16     st.chat_message(role).write(content)
17
18 mensagem_usuario = st.chat_input("Escreva sua mensagem aqui")
19
```

```
20 if mensagem_usuario:
21     # user -> ser humano
22     # assistant -> inteligencia artificial
23     st.chat_message("user").write(mensagem_usuario)
24     mensagem = {"role": "user", "content": mensagem_usuario}
25     st.session_state["lista_mensagens"].append(mensagem)
26
27     # resposta da IA
28     resposta_modelo = modelo.chat.completions.create(
29         messages=st.session_state["lista_mensagens"],
30         model="gpt-4o"
31     )
32
33     resposta_ia = resposta_modelo.choices[0].message.content
34
35     # exibir a resposta da IA na tela
36     st.chat_message("assistant").write(resposta_ia)
37     mensagem_ia = {"role": "assistant", "content": resposta_ia}
38     st.session_state["lista_mensagens"].append(mensagem_ia)
```

Com isso, o código para o nosso chat está completo! A cada nova mensagem, a IA responde com base na conversa toda, mantendo o histórico e considerando o contexto anterior.

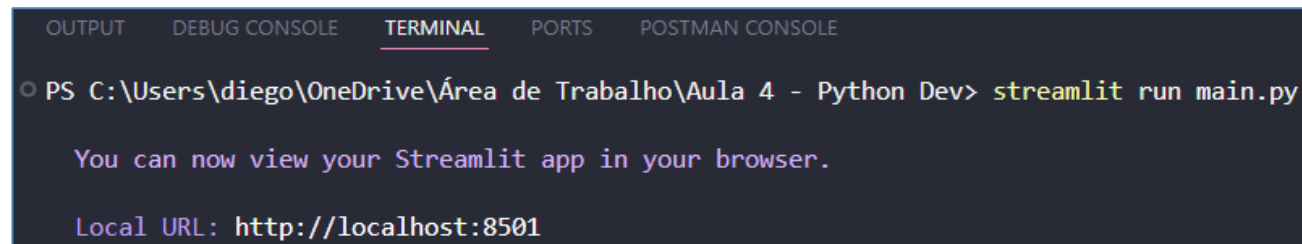
Parte 5

# Executando o Código

# Executando o Código do Chat

Diferente de outras aulas onde rodávamos o código com um clique no botão **Run**, neste projeto vamos executar o site usando um comando direto no terminal, já que estamos usando a biblioteca Streamlit.

Certifique-se de que está na mesma pasta onde está o arquivo **main.py**. Em seguida, digite o seguinte comando no terminal: **streamlit run main.py**

A screenshot of a terminal window with a dark background. At the top, there are tabs labeled 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL' (which is selected and underlined), 'PORTS', and 'POSTMAN CONSOLE'. The terminal shows a command prompt 'PS C:\Users\diego\OneDrive\Área de Trabalho\Aula 4 - Python Dev>' followed by the command 'streamlit run main.py'. Below the command, there is a message: 'You can now view your Streamlit app in your browser.' and then the 'Local URL: http://localhost:8501'.

Ao pressionar **Enter**, o Streamlit vai rodar o código e abrir o navegador com o site do seu ChatBot.

Se o navegador não abrir automaticamente, procure no terminal pela linha com o **Local URL**. É só clicar nesse link ou colá-lo no seu navegador para acessar o site.

# Visualizando a Página do ChatBot

Assim que abrir a página, você verá o seu site pronto, com o Chat esperando para interagir. Envie uma mensagem e, em poucos segundos, verá a resposta da inteligência artificial aparecer logo abaixo.

Toda a conversa é exibida de forma contínua, simulando um bate-papo real. Cada nova mensagem é registrada, e a IA responde com base no histórico anterior.

Esse projeto foi um pouco mais completo do que os anteriores, mas ainda assim você conseguiu criar algo impressionante com poucos comandos e usando apenas Python e duas bibliotecas principais.

Um ponto importante que sempre vale reforçar: use a documentação. Ela vai te ajudar tanto na instalação das bibliotecas, quanto no entendimento de como cada função funciona.

O próprio site da Streamlit, por exemplo, tem vários exemplos práticos de interface e outras funções que você pode usar.

Agora que você entendeu a lógica do projeto, tente refazer tudo do zero, sem olhar o gabarito. Isso vai te ajudar a fixar melhor o conteúdo, identificar o que realmente entendeu e o que ainda pode melhorar.



# Parte 6

# Conclusão

## Conclusão

# Conclusão

Nesse projeto, enfrentamos alguns desafios diferentes dos anteriores. Usamos bibliotecas novas, aprendemos a criar sites com Python usando Streamlit, e também entendemos como integrar um modelo de inteligência artificial no nosso código.

Foi preciso pensar em como organizar as mensagens da conversa, como manter o histórico de forma inteligente, como exibir tudo isso na tela e, o mais importante, como se comunicar com a IA da OpenAI usando Python.

Mesmo que você nunca tivesse programado um site antes, conseguiu colocar tudo isso em prática. Só com a aula, a reprodução do passo a passo e uma consulta à documentação, já foi possível montar um projeto funcional, completo e moderno.

A prática em programação é um dos pontos mais importantes no seu aprendizado. É praticando que você descobre formas diferentes de fazer a mesma coisa, aprende a resolver problemas, melhora sua lógica e começa a criar projetos com mais autonomia.

A programação, principalmente em Python, te abre muitas portas, pois você pode atuar em diversas áreas construindo diferentes tipos de projetos. Aqui na **Jornada Python** nós fizemos 4 projetos que você pode aplicar normalmente no dia a dia de uma empresa.

Para automatizar processos, para fazer análises de dados mais eficientes, para conseguir diminuir gastos, aumentar os lucros, para criar soluções, entre outras diversas possibilidades.

Não precisa começar pelo mais complexo, começando pelo simples você vai notar que as coisas vão funcionando e depois você vai juntando vários “simples” até chegar em um projeto mais robusto e complexo.

Espero que tenham gostado da **Jornada Python** e espero que continuem praticando e estudando Python!

# {JORNADA} PYTHON

100% GRATUITO E ONLINE

Ainda não segue a gente no Instagram e nem é inscrito no nosso canal do Youtube? Então corre lá!



@hashtagprogramacao



[youtube.com/hashtag-programacao](https://youtube.com/hashtag-programacao)

