

---

## **Python para iniciantes: do zero ao primeiro projeto**

Asimov Academy

# ASIMOV

## Conteúdo

<b>01. Boas-vindas ao Curso</b>	<b>4</b>
O que você vai aprender? . . . . .	4
Por que aprender Python? . . . . .	4
Como aproveitar ao máximo este curso? . . . . .	5
<b>02. Estrutura do Curso</b>	<b>6</b>
Como o curso está estruturado? . . . . .	6
Comparação entre os Métodos de Ensino . . . . .	6
O Projeto Prático . . . . .	6
<b>03. Quem Deveria Fazer Este Curso?</b>	<b>8</b>
Python para Diferentes Aplicações . . . . .	8
Para Quem Este Curso Não É Indicado? . . . . .	8
<b>04. O que é uma linguagem de programação</b>	<b>9</b>
Python: uma linguagem acessível e versátil . . . . .	9
Características do Python: . . . . .	9
<b>05. Instalando o Mu</b>	<b>10</b>
Por que utilizar o Mu? . . . . .	10
Instalando o Mu . . . . .	10
Passo 1: Baixar o Mu . . . . .	10
Passo 2: Instalar o Mu . . . . .	10
Passo 3: Primeira execução . . . . .	10
<b>06. Hello World, números e REPL</b>	<b>12</b>
Escrevendo o primeiro programa . . . . .	12
Arquivos Python e execução no Mu . . . . .	12
Trabalhando com números em Python . . . . .	12
Explorando o REPL (Read-Eval-Print Loop) . . . . .	12
<b>07. Strings e Variáveis</b>	<b>14</b>
Variáveis . . . . .	14
Regras para nomes de variáveis: . . . . .	14
<b>08. Formatação em Strings, Listas e Slicing</b>	<b>15</b>
Formatação de Strings . . . . .	15
Exemplo de f-string: . . . . .	15

Listas em Python . . . . .	15
Criando uma lista: . . . . .	15
Acessando elementos da lista: . . . . .	16
Acessando elementos de trás para frente: . . . . .	16
Obtendo o tamanho de uma lista: . . . . .	16
Slicing (Fatiamento) em Listas . . . . .	16
Exemplo de slicing: . . . . .	17
Outros exemplos de slicing: . . . . .	17
Slicing em Strings . . . . .	17
<b>09. Booleanos e Operadores de Comparação</b>	<b>18</b>
Tipo de Dado Booleano . . . . .	18
Operadores de Comparação . . . . .	18
Comentários no Código . . . . .	18
Operadores Lógicos: AND e OR . . . . .	19
AND (E lógico) . . . . .	19
OR (OU lógico) . . . . .	19
<b>10. If, Elif e Else</b>	<b>20</b>
Introdução ao if . . . . .	20
Identação no Python . . . . .	20
Usando múltiplas condições com and . . . . .	21
else: O que acontece se a condição for falsa? . . . . .	21
elif: Criando múltiplas condições . . . . .	21
Conclusão . . . . .	22
<b>11. For loop</b>	<b>23</b>
Criando uma lista e utilizando o for . . . . .	23
Importância da indentação . . . . .	23
Somando valores dentro do for . . . . .	24
Iterando sobre strings . . . . .	24
Usando o range no for . . . . .	25
<b>12. O que é Streamlit?</b>	<b>27</b>
Vantagens do Streamlit . . . . .	27
Instalação do Streamlit . . . . .	27
Criando um primeiro aplicativo com Streamlit . . . . .	27
Estrutura básica de um aplicativo Streamlit . . . . .	29

<b>13. Executando o Primeiro Aplicativo Streamlit</b>	<b>33</b>
Configuração do Ambiente . . . . .	33
Executando um Script Python . . . . .	33
Instalando o Streamlit . . . . .	33
Criando um Aplicativo Streamlit . . . . .	34
Configurando o Mu para Rodar Streamlit . . . . .	34
Criando um Executor para o Streamlit . . . . .	34
Conclusão . . . . .	35
<b>14. Carregando Dados no Pandas</b>	<b>36</b>
Bibliotecas Utilizadas . . . . .	36
Obtendo os Dados . . . . .	36
Importando e Carregando os Dados . . . . .	36
Exibindo os Dados no Streamlit . . . . .	37
Ajustando a Exibição . . . . .	39
Explorando os Dados no REPL . . . . .	39
<b>15. Criando o Primeiro Componente</b>	<b>40</b>
Trabalhando com o Pandas . . . . .	40
Acessando Dados . . . . .	40
Criando Filtros . . . . .	41
Criando um Slider no Streamlit . . . . .	41
<b>16. Construindo os Gráficos</b>	<b>43</b>
Importando o Plotly . . . . .	43
Criando um Gráfico de Barras . . . . .	43
Criando um Histograma de Preços . . . . .	45
Organizando os Gráficos em Colunas . . . . .	47
Conclusão . . . . .	48
<b>17. Finalizando o Projeto</b>	<b>49</b>
Criando uma Nova Página no Streamlit . . . . .	49
Carregando os Dados . . . . .	49
Criando o Select Box para Seleção de Livros . . . . .	50
Filtrando os Dados . . . . .	50
Extraindo Informações do Livro . . . . .	51
Exibindo as Informações do Livro . . . . .	52
Exibindo as Avaliações dos Livros . . . . .	53
Conclusão . . . . .	55

## 01. Boas-vindas ao Curso

Seja bem-vindo ao curso “**Python para Iniciantes: Do Zero ao Primeiro Projeto**”! Este curso foi criado para quem deseja aprender Python do jeito certo, partindo dos conceitos fundamentais até a construção de um projeto prático e aplicável ao mundo real.

### O que você vai aprender?

Ao longo deste curso, você terá uma jornada estruturada para dominar os fundamentos da programação e do Python. O curso é voltado para iniciantes, portanto, não é necessário conhecimento prévio em programação. Vamos explorar conceitos essenciais como:

- Sintaxe e estrutura básica do Python
- Variáveis e tipos de dados
- Controle de fluxo (condicionais e loops)
- Manipulação de dados com listas e dicionários
- Automação de tarefas
- Desenvolvimento de um **projeto prático**: um aplicativo web de análise de dados

### Por que aprender Python?

Python é uma das linguagens mais populares e versáteis da atualidade. Com ela, é possível atuar em diversas áreas, como:

- **Análise de Dados** – Trabalhar com grandes volumes de informação e criar relatórios interativos.
- **Automação** – Criar scripts para automatizar tarefas repetitivas.
- **Desenvolvimento Web** – Construir aplicações e APIs.
- **Ciência de Dados e Machine Learning** – Aplicar modelos inteligentes para resolver problemas complexos.

Python é utilizado por empresas como Google, Netflix e Instagram, tornando-se uma excelente opção para quem deseja ingressar na área de tecnologia.

### Como aproveitar ao máximo este curso?

Para obter os melhores resultados, siga estas dicas:

- **Pratique:** A programação se aprende escrevendo código! Teste cada conceito apresentado.
- **Não tenha medo de errar:** Os erros fazem parte do aprendizado. Leia as mensagens de erro e tente solucioná-los.
- **Interaja e pergunte:** Compartilhe suas dúvidas e participe de fóruns e comunidades.
- **Acompanhe os materiais extras:** Durante o curso, indicaremos links e referências para aprofundamento.

## 02. Estrutura do Curso

Este curso foi desenhado para oferecer uma abordagem **prática e intuitiva** ao aprendizado de Python. Em vez de estudar teoria de forma isolada, aplicaremos os conceitos diretamente em um projeto real, tornando o aprendizado mais dinâmico.

### Como o curso está estruturado?

O curso está dividido em **módulos progressivos**, para facilitar a assimilação dos conteúdos. Cada módulo contém:

- **Explicação teórica** – Conceitos fundamentais apresentados de forma clara.
- **Exemplos práticos** – Aplicação direta dos conceitos no código.
- **Exercícios e desafios** – Para reforçar o aprendizado com prática.
- **Projeto principal** – Um aplicativo web de análise de dados, desenvolvido ao longo do curso.

### Comparação entre os Métodos de Ensino

Método Tradicional	Nossa Abordagem
Aprender conceitos de forma isolada	Aprender conceitos aplicando-os no projeto
Exercícios teóricos sem contexto	Prática direcionada para resolver problemas reais
Pode ser cansativo e desmotivador	Envolvente, pois foca na construção de algo útil
Requer esforço extra para conectar conceitos	O aprendizado é contextualizado desde o início

### O Projeto Prático

O curso culmina na construção de um **aplicativo web interativo** que analisará os 100 livros mais vendidos da Amazon. Durante esse projeto, aprenderemos a:

- Coletar e processar dados

- Criar gráficos e visualizações interativas
- Construir uma interface simples para análise dos dados

Ao final do curso, você terá uma base sólida para seguir explorando o mundo da programação e da análise de dados com Python!



### 03. Quem Deveria Fazer Este Curso?

Este curso é ideal para qualquer pessoa que deseja aprender Python do zero e aplicá-lo em sua rotina profissional ou em novos projetos. Se você se encaixa em uma dessas categorias, este curso é para você:

**Profissionais de diversas áreas** – Querem automatizar tarefas repetitivas e otimizar processos.

**Analistas de Dados** – Buscam uma ferramenta poderosa para manipular e visualizar informações.

**Pesquisadores e Cientistas** – Precisam processar grandes volumes de dados.

**Estudantes e iniciantes em programação** – Desejam entrar no mundo da tecnologia.

**Empreendedores e gestores** – Querem usar a tecnologia para melhorar seus negócios.

#### Python para Diferentes Aplicações

Python pode ser utilizado em diversos cenários, como:

- **Substituir o Excel** – Manipulação de dados mais eficiente.
- **Automação de Tarefas** – Eliminação de processos manuais demorados.
- **Criação de Relatórios e Dashboards** – Análise de dados rápida e interativa.
- **Desenvolvimento Web e APIs** – Construção de sistemas e aplicações interativas.

#### Para Quem Este Curso Não É Indicado?

Este curso **não** é focado em **desenvolvimento web front-end**. Se o seu objetivo é criar sites com interfaces visuais interativas, o ideal seria estudar JavaScript, HTML e CSS. No entanto, Python é amplamente utilizado no **back-end**, sendo uma ótima escolha para quem deseja trabalhar com frameworks como **Django** e **Flask**.

## 04. O que é uma linguagem de programação

Linguagens de programação são como idiomas intermediários que permitem a comunicação com computadores. O código binário, a linguagem nativa dos computadores, é uma sequência de zeros e uns que controla os circuitos eletrônicos. No entanto, programar diretamente em código binário seria extremamente trabalhoso e ineficiente. Para facilitar essa comunicação, foram criadas linguagens de programação, que permitem que os programadores escrevam comandos em um formato mais compreensível para humanos.

Essas linguagens atuam como tradutores, convertendo instruções escritas pelo programador em comandos que a máquina pode processar. Existem diversas linguagens de programação, cada uma com suas particularidades e propósitos específicos. Algumas são mais voltadas para desenvolvimento web, outras para análise de dados, inteligência artificial, sistemas embarcados, entre outras áreas.

### Python: uma linguagem acessível e versátil

Python é uma das linguagens de programação mais populares atualmente. Ela se destaca por sua sintaxe simples e legível, o que a torna ideal para iniciantes e para quem deseja desenvolver projetos rapidamente. Além disso, Python possui uma vasta biblioteca de ferramentas que facilitam desde a automação de tarefas simples até o desenvolvimento de sistemas complexos.

#### Características do Python:

- **Sintaxe simplificada:** código mais legível e fácil de aprender.
- **Multiplataforma:** pode ser executado em diferentes sistemas operacionais sem modificações.
- **Bibliotecas e frameworks poderosos:** suporta automação, ciência de dados, desenvolvimento web, inteligência artificial, entre outras aplicações.
- **Comunidade ativa:** ampla documentação e suporte de desenvolvedores ao redor do mundo.

Por essas razões, Python tem sido amplamente adotado tanto por iniciantes quanto por empresas e profissionais experientes. Com ele, é possível criar desde scripts simples para automatizar tarefas do dia a dia até aplicações complexas de grande escala.

Nos próximos capítulos, exploraremos os principais conceitos da linguagem e como utilizá-la para resolver problemas reais.

## 05. Instalando o Mu

Para programar em Python, utilizaremos um ambiente de desenvolvimento integrado (IDE). O código Python é escrito em arquivos de texto e organizado seguindo regras específicas da linguagem. O uso de uma IDE facilita o processo de desenvolvimento, oferecendo funcionalidades como realce de sintaxe, execução de código e ferramentas úteis para depuração.

### Por que utilizar o Mu?

O Mu é uma IDE simplificada, ideal para iniciantes, pois oferece um ambiente de programação intuitivo e fácil de usar. Ele permite que você se concentre nos conceitos básicos de Python sem se preocupar com configurações complexas. Posteriormente, será possível utilizar IDEs mais avançadas, como o Visual Studio Code, para desenvolver projetos mais complexos.

### Instalando o Mu

#### Passo 1: Baixar o Mu

O Mu pode ser baixado gratuitamente para diferentes sistemas operacionais. Acesse o site oficial: [Clique Aqui](#)

Escolha a versão correspondente ao seu sistema operacional (Windows, macOS ou Linux) e faça o download do instalador.

#### Passo 2: Instalar o Mu

Após o download, siga as instruções de instalação:

- **Windows:** Execute o arquivo .exe, siga as instruções na tela e conclua a instalação.
- **macOS:** Abra o arquivo .dmg e arraste o Mu para a pasta de Aplicações.
- **Linux:** Siga as instruções específicas para sua distribuição, utilizando os comandos fornecidos no site oficial.

#### Passo 3: Primeira execução

Após instalar, abra o Mu e selecione o modo **“Python 3”** para começar a programar. Essa configuração garante que você esteja usando a versão correta da linguagem.

Agora você está pronto para escrever e executar seus primeiros programas em Python utilizando o Mu!

## 06. Hello World, números e REPL

### Escrevendo o primeiro programa

O primeiro passo em programação é exibir uma mensagem na tela. Em Python, usamos o comando `print()` para imprimir textos:

```
print("Olá, Mundo!")
```

Esse comando faz com que a frase “Olá, Mundo!” apareça na tela. A função `print()` recebe um argumento, que pode ser um texto (string) ou outros tipos de dados.

### Arquivos Python e execução no Mu

O código Python pode ser escrito em arquivos de texto com a extensão `.py` e executado em editores simples ou IDEs como o Mu. O Mu facilita a execução do código ao organizar os arquivos em uma pasta dedicada.

Para rodar um arquivo Python no Mu: 1. Escreva o código no editor. 2. Salve o arquivo com a extensão `.py`. 3. Clique no botão “Run” para executar o código.

### Trabalhando com números em Python

Python suporta diferentes tipos de dados numéricos, incluindo:

- **Inteiros (`int`):** números sem casas decimais (ex: 5, 100, -42).
- **Ponto flutuante (`float`):** números com casas decimais (ex: 3.14, 2.718, -0.5).

Podemos verificar o tipo de um valor usando a função `type()`:

```
print(type(10)) # int
print(type(3.14)) # float
```

### Explorando o REPL (Read-Eval-Print Loop)

O Mu possui um interpretador interativo, chamado REPL, que permite executar comandos Python em tempo real, sem a necessidade de salvar e rodar um arquivo. Isso facilita o aprendizado e a experimentação com a linguagem.

Para usar o REPL: 1. Abra o Mu. 2. Clique no botão “REPL”. 3. Digite comandos Python e pressione Enter para ver os resultados imediatamente.

O REPL é útil para testar pequenos trechos de código e entender melhor o comportamento da linguagem.

Nos próximos capítulos, exploraremos mais recursos essenciais do Python!

## 07. Strings e Variáveis

Após aprender sobre números, `print()` e ações no Python, vamos explorar as **strings**, que são textos. Para declará-las, utilize aspas duplas ou simples:

```
texto = "Olá, Rodrigo"
outro_texto = 'Olá, Mundo'
```

É possível concatenar strings, ou seja, uni-las, usando o operador `+`:

```
mensagem = "Olá" + " Rodrigo"
print(mensagem) # Saída: Olá Rodrigo
```

No entanto, não podemos concatenar strings com números diretamente:

```
idade = 25
mensagem = "Idade: " + str(idade) # Necessário converter o número para string
print(mensagem) # Saída: Idade: 25
```

### Variáveis

Variáveis são como caixas para armazenar valores. Para criar uma, utilize um nome e o operador de atribuição `=`:

```
saudacao = "Olá"
nome = "Rodrigo"
mensagem = saudacao + ", " + nome
print(mensagem) # Saída: Olá, Rodrigo
```

As variáveis podem ser usadas para guardar informações e realizar operações.

### Regras para nomes de variáveis:

- Utilize apenas **letras, números e underlines**.
- Não inicie com números.
- Letras **maiúsculas e minúsculas diferenciam** as variáveis.

Com este conhecimento sobre **strings** e **variáveis**, podemos avançar para a construção do dashboard.

## 08. Formatação em Strings, Listas e Slicing

### Formatação de Strings

No Python, podemos combinar variáveis dentro de strings utilizando **f-strings**. Isso facilita a formatação de textos dinâmicos de maneira legível e eficiente.

#### Exemplo de f-string:

```
A = "Olá"
B = 3
C = "Rodrigo"
D = f"{A}, {C}! O número é {B}."
print(D)
```

#### Saída:

Olá, Rodrigo! O número é 3.

- O **f** antes das aspas indica que a string será formatada.
- O que estiver entre **{ }** será substituído pelo valor da variável correspondente.
- Podemos utilizar quantas variáveis forem necessárias dentro da string.

### Listas em Python

As listas são estruturas que armazenam vários elementos em uma única variável. Podemos armazenar diferentes tipos de dados dentro da mesma lista.

#### Criando uma lista:

```
minha_lista = ["Olá", "Rodrigo", 2, 2.5, "Teste"]
print(minha_lista)
```

#### Saída:

['Olá', 'Rodrigo', 2, 2.5, 'Teste']



### Acessando elementos da lista:

Podemos acessar elementos individuais da lista utilizando **índices**.

```
print(minha_lista[0]) # Primeiro elemento  
print(minha_lista[1]) # Segundo elemento
```

### Saída:

```
Olá  
Rodrigo
```

No Python, os índices começam em **0**. Assim, o primeiro elemento está na posição 0, o segundo na 1, e assim por diante.

### Acessando elementos de trás para frente:

Podemos utilizar índices negativos para acessar elementos de trás para frente:

```
print(minha_lista[-1]) # Último elemento  
print(minha_lista[-2]) # Penúltimo elemento
```

### Saída:

```
Teste  
2.5
```

### Obtendo o tamanho de uma lista:

Para saber quantos elementos existem em uma lista, utilizamos `len()`:

```
print(len(minha_lista))
```

### Saída:

```
5
```

## Slicing (Fatiamento) em Listas

Podemos extrair subconjuntos de elementos de uma lista utilizando **slicing**.

### Exemplo de slicing:

```
sub_lista = minha_lista[1:4] # Do índice 1 até 3 (o 4 não é incluído)
print(sub_lista)
```

### Saída:

```
['Rodrigo', 2, 2.5]
```

- O primeiro número indica o início (inclusivo).
- O segundo número indica onde deve parar (exclusivo).

### Outros exemplos de slicing:

- `lista[:3]` -> Do início até o índice 2.
- `lista[2:]` -> Do índice 2 até o final.
- `lista[-3:]` -> Três últimos elementos da lista.

### Slicing em Strings

Strings são sequências de caracteres e também suportam slicing.

```
texto = "Rodrigo"
print(texto[:4]) # Pegando os primeiros 4 caracteres
```

### Saída:

```
Rodr
```

Assim como nas listas, podemos utilizar slicing para manipular strings facilmente.

## 09. Booleanos e Operadores de Comparação

Até o momento você já passou por números, variáveis, métodos, listas e um pouquinho de *slicing*. O próximo passo agora é aprender sobre **booleanos**.

Booleanos são como o *sim* e o *não* de uma linguagem de programação. Eles representam **verdadeiro** e **falso** e são dois valores reservados: `True` e `False`.

### Tipo de Dado Booleano

No Python, `True` e `False` são palavras reservadas e precisam ser escritas com a primeira letra maiúscula. Podemos verificar seu tipo com a função `type()`:

```
print(type(True)) # Saída: <class 'bool'>
print(type(False)) # Saída: <class 'bool'>
```

### Operadores de Comparação

Os booleanos por si só não têm muito sentido, mas são extremamente úteis quando combinados com **operadores de comparação**. Esses operadores permitem comparar valores e retornar `True` ou `False`. Os principais operadores são:

Operador	Descrição	Exemplo	Resultado
<code>&gt;</code>	Maior que	<code>3 &gt; 2</code>	<code>True</code>
<code>&lt;</code>	Menor que	<code>3 &lt; 2</code>	<code>False</code>
<code>&gt;=</code>	Maior ou igual	<code>2 &gt;= 2</code>	<code>True</code>
<code>&lt;=</code>	Menor ou igual	<code>2 &lt;= 2</code>	<code>True</code>
<code>!=</code>	Diferente de	<code>3 != 2</code>	<code>True</code>
<code>==</code>	Igual a	<code>3 == 2</code>	<code>False</code>

Esses operadores são fundamentais para **tomada de decisões** dentro de um programa.

### Comentários no Código

Podemos adicionar **comentários** para organizar melhor o código. Comentários são ignorados pelo Python e servem apenas para anotações.

```
# Isso é um comentário  
print(3 > 2) # Comparação retorna True
```

## Operadores Lógicos: AND e OR

Além dos operadores de comparação, temos os operadores **lógicos** AND e OR.

### AND (E lógico)

Retorna True se **ambas** as condições forem verdadeiras:

```
print(True and True)    # Saída: True  
print(True and False)   # Saída: False  
print(3 > 2 and 2 > 1)   # Saída: True  
print(3 > 2 and 2 < 1)   # Saída: False
```

### OR (OU lógico)

Retorna True se **pelo menos uma** das condições for verdadeira:

```
print(True or False)    # Saída: True  
print(False or False)   # Saída: False  
print(3 > 2 or 2 < 1)    # Saída: True  
print(1 > 3 or 2 < 1)    # Saída: False
```

## 10. If, Elif e Else

A penúltima aula desta parte introdutória de Python apresenta o conceito de **estrutura de controle de fluxo**, ou seja, a capacidade de permitir que o código siga por um caminho ou outro, dependendo de uma regra previamente estabelecida.

### Introdução ao `if`

O `if`, na tradução literal, significa “se”. Ele permite perguntar algo ao computador e dizer: **se** alguma condição for verdadeira, então execute um determinado código.

```
idade = 18

if idade > 18:
    print("Você pode dirigir")

print("Fim do programa")
```

Ao executar esse código, ele imprime apenas:

Fim do programa

Isso acontece porque `18 > 18` é falso. Para corrigir isso, podemos usar `>=`:

```
if idade >= 18:
    print("Você pode dirigir")
```

Agora a saída será:

Você pode dirigir  
Fim do programa

### Identação no Python

No Python, a indentação define blocos de código. O que estiver **indentado** após um `if` faz parte dele. Podemos usar **quatro espaços** ou **um `tab`** para isso.

```
if idade >= 18:
    print("Você pode dirigir")
    print("Parabéns!")
```

Isso garante que ambas as linhas sejam executadas se a condição for verdadeira.

## Usando múltiplas condições com and

Podemos combinar condições com and:

```
idade = 18
possui_carteira = False

if idade >= 18 and possui_carteira:
    print("Você pode dirigir")
```

Esse código **não imprime nada**, porque possui\_carteira é False.

Se trocarmos and por or, teremos um comportamento mais permissivo:

```
if idade >= 18 or possui_carteira:
    print("Você pode dirigir")
```

Agora, qualquer uma das condições sendo verdadeira, o código será executado.

## else: O que acontece se a condição for falsa?

O else define o que será executado caso a condição do if **não** seja verdadeira:

```
if idade >= 18 and possui_carteira:
    print("Você pode dirigir")
else:
    print("Você não pode dirigir")
```

Saída:

Você não pode dirigir

## elif: Criando múltiplas condições

Podemos ter um terceiro caso intermediário usando elif:

```
if idade >= 18 and possui_carteira:
    print("Você pode dirigir")
elif possui_carteira:
    print("Você apenas possui a carteira, espere a idade")
else:
    print("Você não pode dirigir")
```

Agora, diferentes saídas podem ser geradas dependendo das variáveis.

## Conclusão

A estrutura de controle de fluxo nos permite decidir **quais trechos do código** serão executados. Isso é essencial para a construção de lógicas mais complexas.

Na próxima aula, vamos aprender sobre **o for**, que nos permite repetir códigos automaticamente até que alguma condição seja atendida!

## 11. For loop

O último elemento estudado em Python, antes do início do desenvolvimento do projeto, é o **for**. O **for** é uma **estrutura de repetição**, também chamada de **laço**, que permite repetir um trecho de código um determinado número de vezes.

O **for** é uma palavra reservada do Python e aparece em verde, assim como outras palavras já apresentadas. Sua tradução literal seria algo como “**para cada**”, o que faz sentido quando combinada com determinados elementos.

### Criando uma lista e utilizando o for

Vamos começar criando uma lista:

```
a = [1, 2, 3, 4, 5]
```

É uma lista de tamanho **5**, onde o elemento **0** é 1, o elemento **4** é 5, e todos os elementos são números inteiros.

No **for**, é preciso combinar um **iterável** (como essa lista, por exemplo). O **for** vai decompor esse iterável e, ao mesmo tempo que faz isso, vai executar um trecho de código.

Isso pode parecer confuso no início, mas vamos ver na prática:

```
for x in a:  
    print(x)
```

O que está acontecendo aqui?

- O **for** pega cada valor da lista a, um de cada vez.
- Esse valor é atribuído à variável x.
- O **print(x)** exibe esse valor na tela.
- O processo se repete até todos os valores da lista terem sido percorridos.

### Importância da indentação

Aqui, a mesma regra do **if** se aplica: a **indentação** do código indica que o que está abaixo pertence ao **for**. Esse código será executado a cada iteração. Se quisermos adicionar mais comandos dentro do laço, basta manter a indentação.

Quando rodamos o código acima, o resultado será:



1  
2  
3  
4  
5

Se tivéssemos uma lista de **100.000 elementos**, o **for** faria esse processo de maneira igualmente rápida.

### Somando valores dentro do for

Podemos fazer operações dentro do laço, como somar todos os elementos da lista:

```
b = 0
for x in a:
    b = b + x
print(b)
```

O que acontece aqui:

1. Criamos `b = 0`.
2. Para cada `x` na lista `a`, somamos `x` ao valor de `b`.
3. No final, `b` conterá a soma de todos os valores da lista.

A execução do código nos dá:

15

O processo funciona assim:

- `b = 0 + 1 -> b = 1`
- `b = 1 + 2 -> b = 3`
- `b = 3 + 3 -> b = 6`
- `b = 6 + 4 -> b = 10`
- `b = 10 + 5 -> b = 15`

### Iterando sobre strings

O **for** também pode ser usado com strings:

```
for letra in "Python":
    print(letra)
```

Saída:

P  
y  
t  
h  
o  
n

### Usando o range no for

O Python tem uma forma mais fácil de gerar sequências de números com o **range**:

```
for x in range(0, 10, 1):  
    print(x)
```

Aqui:

- O **primeiro valor** (0) é o ponto de partida.
- O **segundo valor** (10) é onde ele para (mas não inclui).
- O **terceiro valor** (1) é o passo (de quantos em quantos ele anda).

Resultado:

0  
1  
2  
3  
4  
5  
6  
7  
8  
9

Se quisermos contar de **2 em 2**, basta mudar o passo:

```
for x in range(0, 10, 2):  
    print(x)
```

Saída:

0  
2  
4  
6  
8

Se não informarmos o **valor inicial**, ele assume **zero**. Se não informarmos o **passo**, ele assume **1**:

```
for x in range(10):  
    print(x)
```

Isso equivale a `range(0, 10, 1)`.

## 12. O que é Streamlit?

O **Streamlit** é uma biblioteca Python voltada para a criação de aplicações web interativas, especialmente útil para análise de dados e aprendizado de máquina. Ele permite desenvolver interfaces gráficas com poucas linhas de código, sem a necessidade de experiência com desenvolvimento web.

Com o Streamlit, é possível transformar scripts Python em dashboards e ferramentas interativas acessíveis via navegador.

### Vantagens do Streamlit

- **Fácil de usar:** Sintaxe simples e intuitiva.
- **Rápido desenvolvimento:** Permite a criação de aplicativos sem precisar configurar servidores ou aprender frameworks web complexos.
- **Integração com bibliotecas populares:** Funciona bem com Pandas, Matplotlib, Plotly, entre outras.
- **Execução em tempo real:** Atualiza a interface automaticamente quando o código é modificado.
- **Compatibilidade com Jupyter Notebooks:** Possui funcionalidades semelhantes, facilitando a migração de scripts.

### Instalação do Streamlit

Para instalar o Streamlit, utilize o seguinte comando no terminal:

```
pip install streamlit
```

Após a instalação, verifique se o Streamlit foi instalado corretamente executando:

```
streamlit hello
```

Esse comando executará um aplicativo de demonstração do Streamlit no navegador.

### Criando um primeiro aplicativo com Streamlit

Para iniciar um projeto, crie um arquivo Python e adicione o seguinte código:

```
import streamlit as st

st.title("Meu Primeiro Aplicativo com Streamlit")
st.write("Este é um exemplo simples de um aplicativo web usando Python e Streamlit.")
```

Para executar o aplicativo, use o comando:

```
streamlit run nome_do_arquivo.py
```

O Streamlit abrirá automaticamente uma aba no navegador exibindo a aplicação.



**Figure 1:** 1- Primeiro Aplicativo com Streamlit

## Estrutura básica de um aplicativo Streamlit

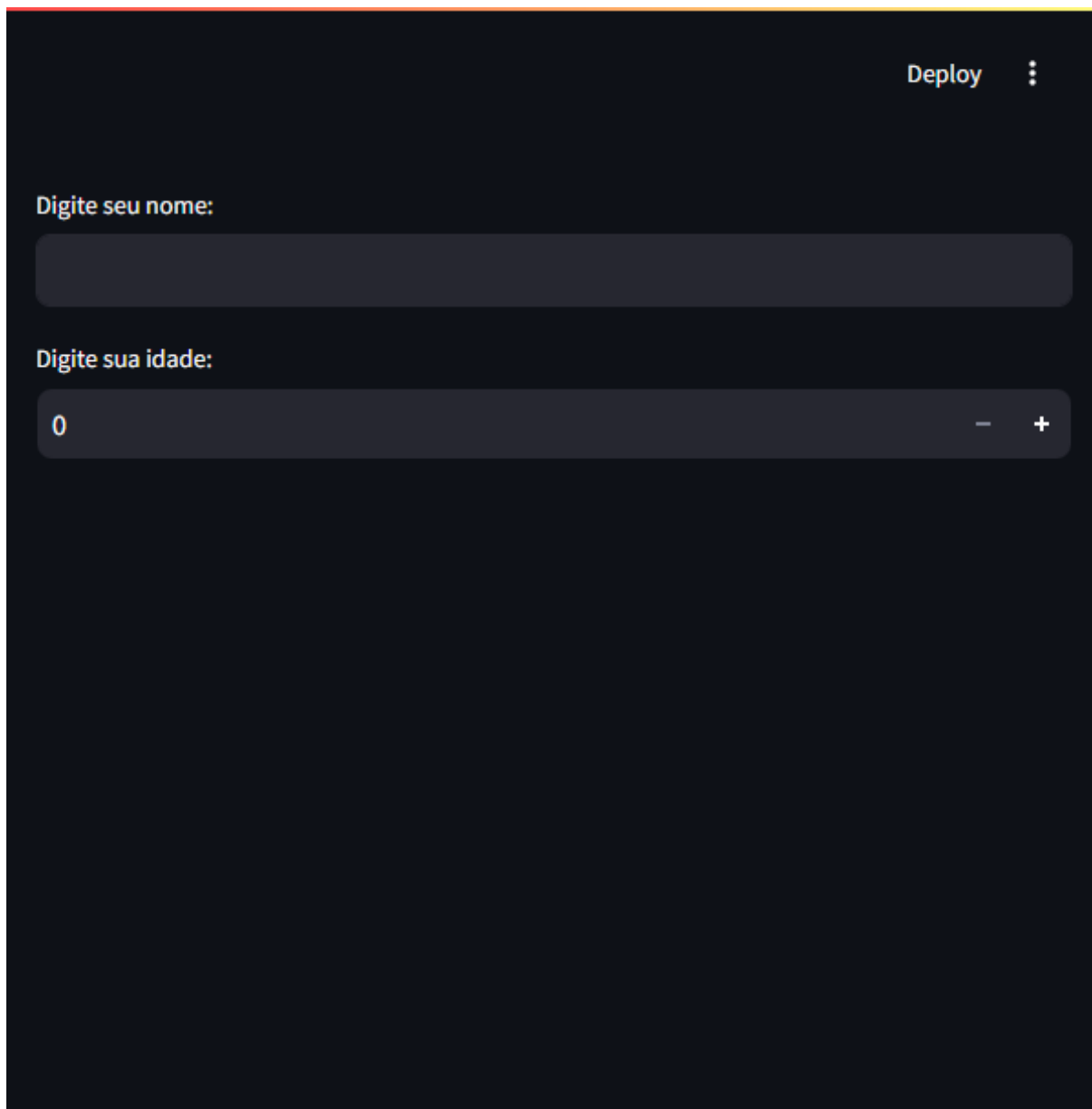
O Streamlit fornece diversos componentes para criar interfaces interativas:

- **Títulos e textos:**

```
st.title("Título Principal")
st.header("Cabeçalho")
st.subheader("Subcabeçalho")
st.text("Texto simples")
```

- **Entrada de dados:**

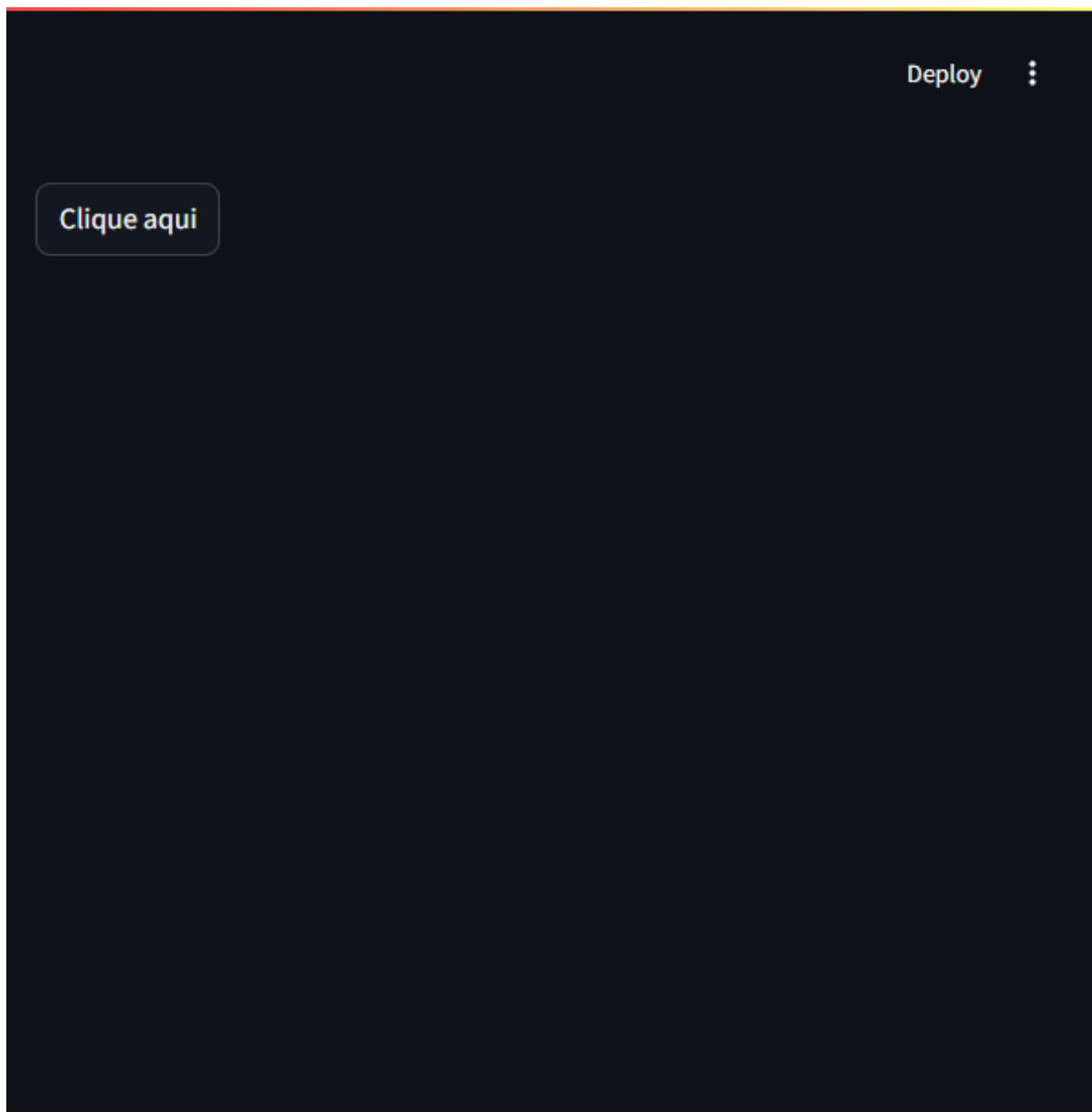
```
nome = st.text_input("Digite seu nome:")
idade = st.number_input("Digite sua idade:", min_value=0, max_value=120)
```



**Figure 2:** 2- Text Input e Number Input Streamlit

- **Botões e interatividade:**

```
if st.button("Clique aqui"):
    st.write("Botão pressionado!")
```



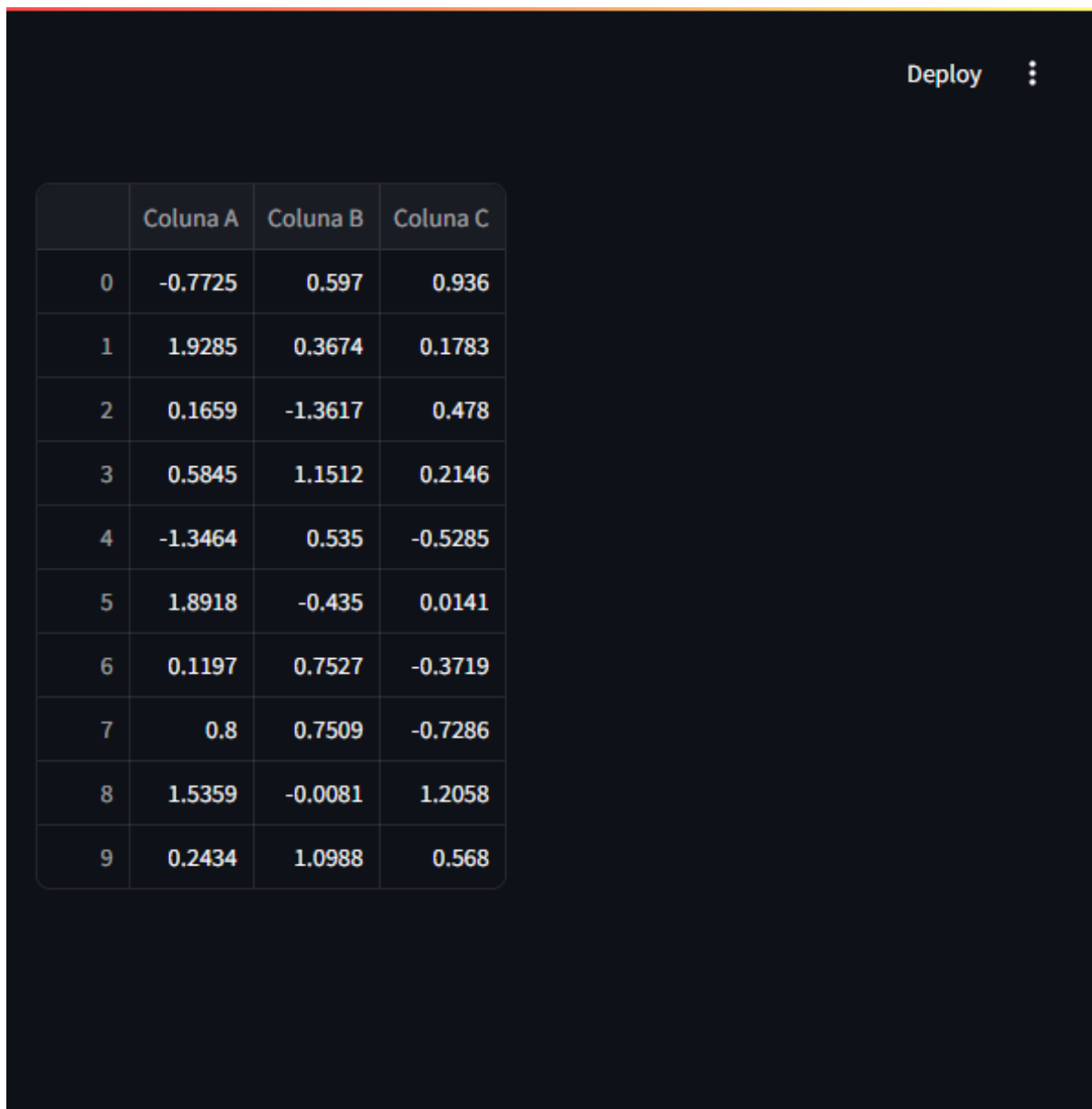
**Figure 3:** 3- Botão Streamlit

- **Gráficos e tabelas:**

```
import pandas as pd
import numpy as np

df = pd.DataFrame(
    np.random.randn(10, 3),
    columns=['Coluna A', 'Coluna B', 'Coluna C']
)
st.dataframe(df)
```





	Coluna A	Coluna B	Coluna C
0	-0.7725	0.597	0.936
1	1.9285	0.3674	0.1783
2	0.1659	-1.3617	0.478
3	0.5845	1.1512	0.2146
4	-1.3464	0.535	-0.5285
5	1.8918	-0.435	0.0141
6	0.1197	0.7527	-0.3719
7	0.8	0.7509	-0.7286
8	1.5359	-0.0081	1.2058
9	0.2434	1.0988	0.568

**Figure 4:** 4- Dataframe Streamlit

Com esses elementos, é possível construir aplicativos interativos para visualização e análise de dados.

## 13. Executando o Primeiro Aplicativo Streamlit

Antes de começar a utilizar o Streamlit, é necessário entender alguns conceitos sobre Python e como ele funciona. O Streamlit exige algumas configurações adicionais para rodar corretamente no ambiente de desenvolvimento Mu.

### Configuração do Ambiente

O Mu foi escolhido como ambiente de desenvolvimento porque ele simplifica o processo de instalação do Python, que pode ser mais complicado em alguns sistemas operacionais, especialmente no Windows. Para quem usa Linux ou MacOS, o Python já vem instalado por padrão, enquanto no Windows é necessário instalar manualmente.

Quando utilizamos o Mu, ele cria um ambiente isolado de Python, garantindo que todos os pacotes necessários possam ser instalados e utilizados sem problemas. Isso evita possíveis erros de configuração que podem ocorrer ao instalar o Python manualmente.

### Executando um Script Python

Para executar um script Python no terminal, seguimos os seguintes passos:

1. Criamos um arquivo chamado `olamundo.py` com o seguinte conteúdo:

```
print("Hello World")
```

2. No terminal, acessamos a pasta onde o arquivo está localizado:

```
cd caminho/para/a/pasta
```

3. Rodamos o script utilizando o comando:

```
python3 olamundo.py
```

Isso imprimirá “Hello World” no terminal.

Por trás dos panos, o Mu executa exatamente esse comando quando rodamos um script diretamente nele.

### Instalando o Streamlit

O Python não vem com pacotes de terceiros instalados por padrão. Para instalar o Streamlit, utilizamos o gerenciador de pacotes `pip`:

```
python3 -m pip install streamlit
```

Esse comando baixa e instala o Streamlit no ambiente Python utilizado. Caso o Python não esteja instalado corretamente no sistema, esse comando pode falhar, especialmente no Windows.

### Criando um Aplicativo Streamlit

Após instalar o Streamlit, podemos criar um arquivo Python para rodar nossa primeira aplicação. Criamos um arquivo chamado `app.py` e adicionamos o seguinte código:

```
import streamlit as st

st.write("Olá, Mundo!")
```

Para executar esse script fora do Mu, utilizamos o seguinte comando no terminal:

```
streamlit run app.py
```

Isso abrirá uma página no navegador com a mensagem “Olá, Mundo!”.

### Configurando o Mu para Rodar Streamlit

O Mu tem um ambiente isolado de Python, então os pacotes precisam ser instalados diretamente dentro dele. Para isso:

1. Clicamos na engrenagem de configurações do Mu.
2. Acessamos a aba “Pacotes de Terceiros”.
3. Adicionamos os pacotes necessários (exemplo: `streamlit`, `pandas`, `plotly`).
4. Clicamos em “OK” e aguardamos a instalação.

### Criando um Executor para o Streamlit

O Mu não suporta diretamente a execução do `streamlit run`. Para contornar essa limitação, criamos um arquivo chamado `executar_streamlit.py` com o seguinte código:

```
import sys
from streamlit.web import cli as stcli

sys.argv = ["streamlit", "run", "app.py"]
stcli.main()
```

Sempre que quisermos rodar nossa aplicação, ao invés de usar `streamlit run app.py`, executamos `executar_streamlit.py` diretamente pelo Mu.

## **Conclusão**

Com essa configuração, conseguimos rodar aplicações Streamlit dentro do Mu sem precisar sair do ambiente. Essa abordagem facilita o desenvolvimento e permite criar aplicações web interativas de maneira simples e prática.

## 14. Carregando Dados no Pandas

### Bibliotecas Utilizadas

Ao configurar os pacotes de terceiros, além do **Streamlit**, incluímos também **Pandas** e **Plotly**:

- **Pandas**: Biblioteca para manipulação de dados.
- **Plotly**: Biblioteca para criação de gráficos interativos.

Sempre que necessário, podemos consultar a documentação oficial dessas bibliotecas para mais informações.

- Documentação **Pandas**: [Clique Aqui](#).
- Documentação **Plotly**: [Clique Aqui](#).
- Documentação **Streamlit**: [Clique Aqui](#).

### Obtendo os Dados

Os dados utilizados estão disponíveis no site [Kaggle](#), uma plataforma voltada para análise de dados e Machine Learning. Para baixar os arquivos:

1. Acesse o Kaggle e faça o download do conjunto de dados **Top 100 Bestselling Book Reviews on Amazon**: [Clique aqui](#).
2. Extraia os arquivos e adicione-os à pasta `datasets` dentro do seu projeto.
3. Os arquivos que utilizaremos são:
  - `customer_reviews.csv`
  - `Top-100 Trending Books.csv`

Cada arquivo contém:

- **Top-100 Trending Books.csv**: Ranking, nome do livro, preço, avaliação, autor e ano de publicação.
- **customer reviews.csv**: Comentários de usuários, avaliação do review, data do comentário e identificador do livro.

### Importando e Carregando os Dados

No script Python, comece importando a biblioteca Pandas e atribua um apelido (pd):

```
import pandas as pd
```

Agora, crie duas variáveis para armazenar os DataFrames:

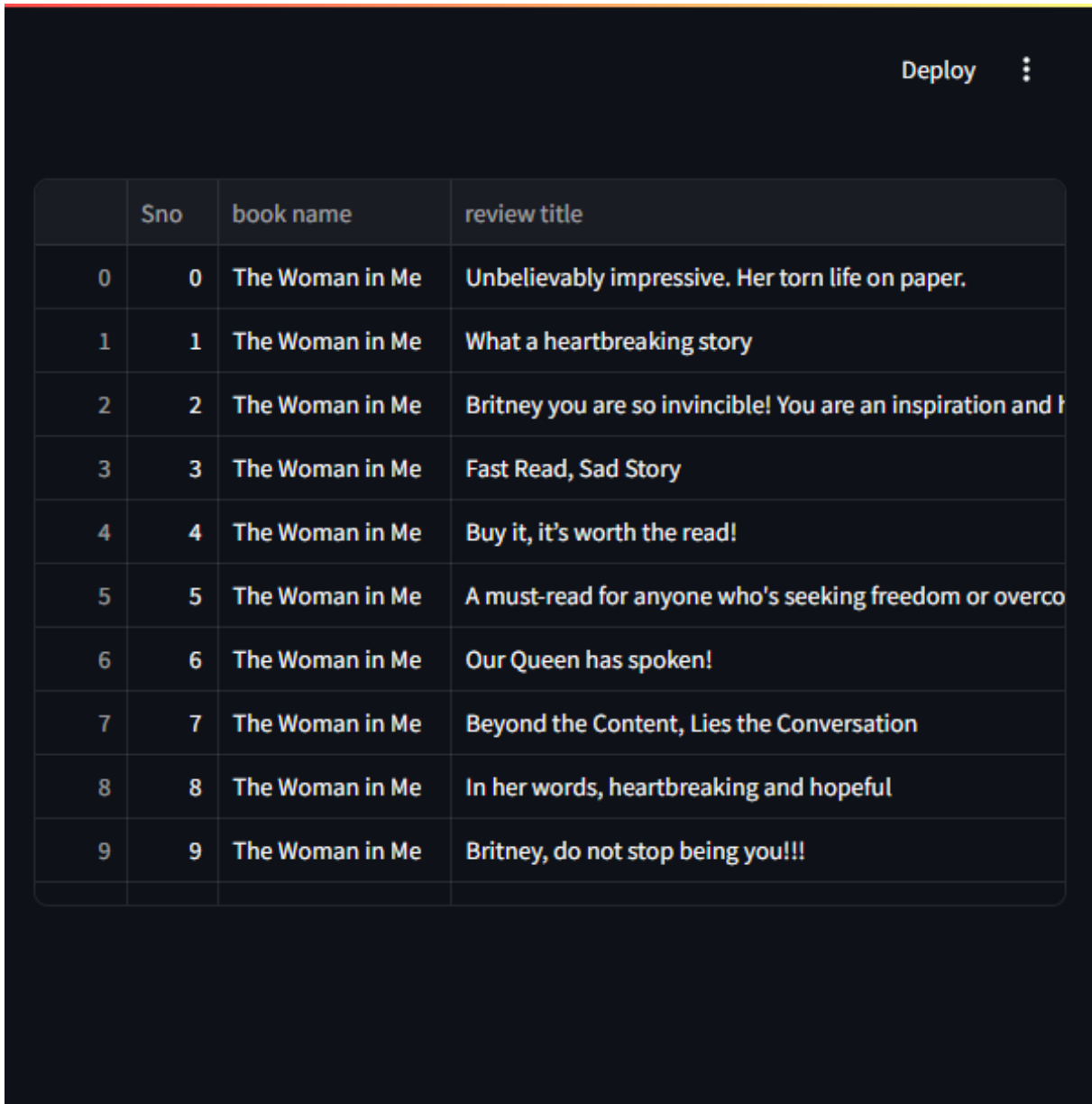
```
df_reviews = pd.read_csv("datasets/customer_reviews.csv")  
df_top100_books = pd.read_csv("datasets/Top-100 Trending Books.csv")
```

O método `read_csv()` do Pandas é responsável por ler arquivos no formato CSV e transformá-los em um **DataFrame**, estrutura utilizada para manipulação de dados tabulares.

### Exibindo os Dados no Streamlit

Podemos visualizar a tabela diretamente no Streamlit ao adicionar o seguinte código:

```
import streamlit as st  
  
st.write(df_reviews)
```



The screenshot shows a web application interface with a dark theme. In the top right corner, there is a 'Deploy' button and a vertical ellipsis menu icon. Below this, a table displays book reviews. The table has four columns: an index column, 'Sno', 'book name', and 'review title'. There are 10 rows of data, all for the book 'The Woman in Me'. The reviews are as follows:

	Sno	book name	review title
0	0	The Woman in Me	Unbelievably impressive. Her torn life on paper.
1	1	The Woman in Me	What a heartbreaking story
2	2	The Woman in Me	Britney you are so invincible! You are an inspiration and I
3	3	The Woman in Me	Fast Read, Sad Story
4	4	The Woman in Me	Buy it, it's worth the read!
5	5	The Woman in Me	A must-read for anyone who's seeking freedom or overco
6	6	The Woman in Me	Our Queen has spoken!
7	7	The Woman in Me	Beyond the Content, Lies the Conversation
8	8	The Woman in Me	In her words, heartbreaking and hopeful
9	9	The Woman in Me	Britney, do not stop being you!!!

**Figure 5:** 1- Datasets Kaggle Streamlit

Isso exibe o conteúdo do DataFrame de maneira interativa, permitindo rolar, buscar e filtrar os dados.

## Ajustando a Exibição

O **Streamlit** oferece algumas configurações visuais que podem ser ajustadas. Para expandir a largura da tela e melhorar a visualização:

```
st.set_page_config(layout="wide")
```

Além disso, é possível alternar entre o **modo claro** e o **modo escuro** nas configurações do Streamlit.

## Explorando os Dados no REPL

Para entender melhor o funcionamento do Pandas, é recomendável abrir um **REPL** (Read-Eval-Print Loop) no terminal e testar comandos de manipulação de dados. Dessa forma, podemos experimentar diferentes métodos do Pandas sem precisar atualizar o Streamlit constantemente.



## 15. Criando o Primeiro Componente

Dando continuidade, agora há uma série de elementos novos. No lado esquerdo, o script em Streamlit está sendo executado, garantindo que o servidor continue funcionando. Um REPL foi aberto para testar algumas funcionalidades do Pandas e entender melhor os dados disponíveis. O objetivo é manipular essas informações e inseri-las na aplicação.

### Trabalhando com o Pandas

O Pandas é uma biblioteca que funciona como uma versão do Excel dentro do Python, permitindo o trabalho com tabelas. Para utilizá-lo corretamente, é necessário importar e carregar os dados dentro da instância do Python.

```
import pandas as pd
```

Para exemplificação, considere os DataFrames `df_top100_books` e o `df_reviews`, que já foram carregados:

```
df_reviews = pd.read_csv("datasets/customer_reviews.csv")
df_top100_books = pd.read_csv("datasets/Top-100 Trending Books.csv")
```

A estrutura de um DataFrame pode ser verificada utilizando:

```
print(type(df_reviews)) # Retorna <class 'pandas.core.frame.DataFrame'>
```

Os DataFrames possuem índices e colunas que permitem a manipulação eficiente dos dados. Algumas consultas básicas incluem:

```
print(df_reviews.index) # Exibe o índice do DataFrame
print(df_reviews.columns) # Exibe as colunas disponíveis
```

### Acessando Dados

Para acessar uma coluna específica:

```
print(df_reviews["book name"]) # Exibe os valores da coluna "book name"
```

Caso ocorra um erro ao acessar uma coluna, verifique se o nome está correto, incluindo espaços e maiúsculas/minúsculas.

Para filtrar os dados com base em um critério específico, pode-se utilizar expressões condicionais:

```
dfFiltrado = df_reviews[df_reviews["reviewer rating"] > 4]
```

Esse comando retorna apenas as linhas onde a avaliação (`reviewer rating`) seja maior que 4.

## Criando Filtros

Filtros podem ser utilizados para segmentar dados, por exemplo, livros abaixo de um determinado preço:

```
dfTopBooks = df_top100_books[df_top100_books["book price"] < 50]
```

Caso seja necessário encontrar o valor mínimo e máximo dentro de uma coluna específica:

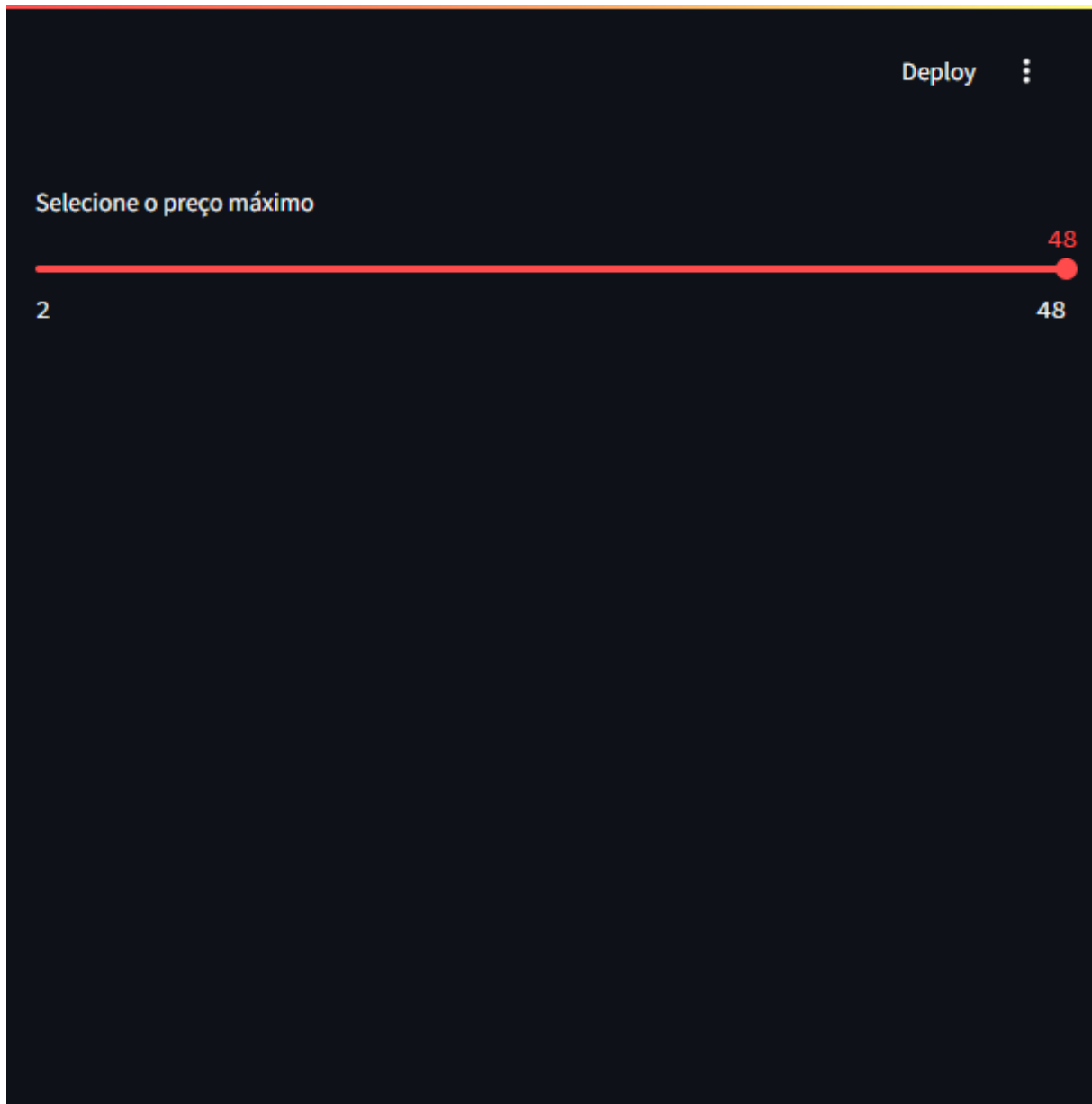
```
priceMax = dfTopBooks["book price"].max()
priceMin = dfTopBooks["book price"].min()
```

## Criando um Slider no Streamlit

Para permitir a interação com os dados dentro da aplicação, pode-se utilizar um slider do Streamlit. A documentação oficial do Streamlit contém informações detalhadas sobre cada componente disponível. Um exemplo de slider:

```
import streamlit as st

price_slider = st.slider(
    "Selecione o preço máximo", # Texto exibido acima do slider
    min_value=int(priceMin),    # Valor mínimo
    max_value=int(priceMax),    # Valor máximo
    value=int(priceMax)         # Valor inicial
)
```



**Figure 6:** 2- Slider Streamlit

Com isso, o slider é renderizado na interface, permitindo que o usuário defina um preço máximo para os livros exibidos na aplicação.

Essa introdução ao Pandas e ao Streamlit fornece os primeiros passos para manipular e visualizar dados interativamente. Nos próximos capítulos, serão exploradas mais funcionalidades para aprimorar a aplicação.

## 16. Construindo os Gráficos

Para dar continuidade ao projeto, é importante notar como, com poucas linhas de código, já conseguimos estruturar um projeto interessante. Neste tópico, vamos aprender a criar gráficos no Python utilizando a biblioteca **Plotly**.

### Importando o Plotly

O **Plotly** possui diversas funcionalidades, sendo uma delas a biblioteca expressa (*express*), que permite a criação de gráficos de maneira simplificada. Para utilizá-la, importe o módulo **plotly.express** e atribua um apelido para facilitar seu uso:

```
import plotly.express as px
```

### Criando um Gráfico de Barras

Para visualizar a distribuição dos anos de publicação dos livros dentro da faixa de preço selecionada, é possível criar um histograma utilizando o **Plotly**. Antes disso, é necessário contar a quantidade de ocorrências de cada ano na coluna *year of publication* da base de dados, utilizando o método **value\_counts()** do **Pandas**:

```
df_top100_books['year of publication'].value_counts()
```

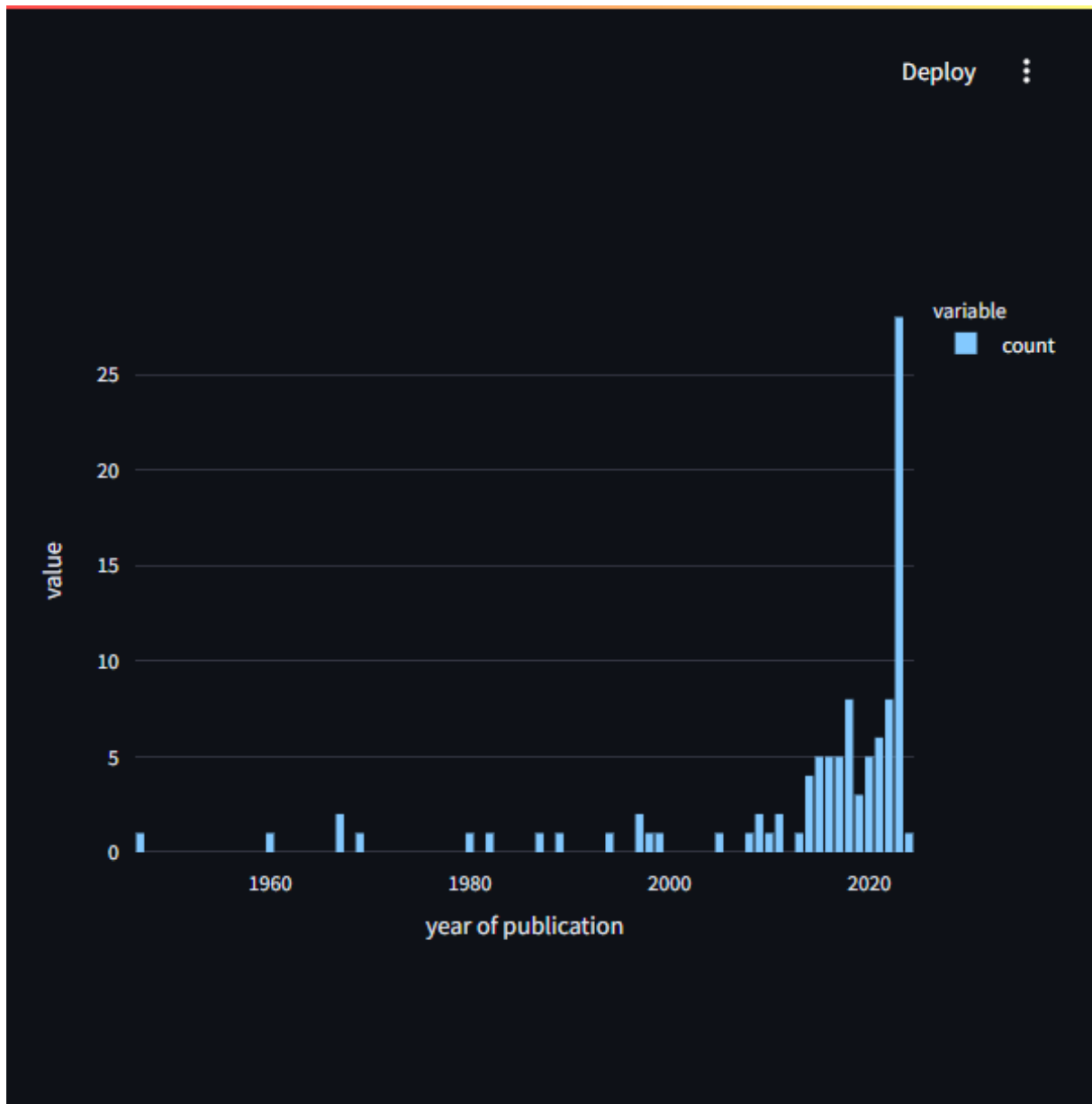
Esse método retorna a contagem de ocorrências de cada ano na base de dados.

Agora, utilizando o **Plotly Express**, podemos criar um gráfico de barras:

```
fig = px.bar(df_top100_books['year of publication'].value_counts())
```

No **Streamlit**, para exibir esse gráfico, utilize o comando:

```
st.plotly_chart(fig)
```



**Figure 7:** 3- Plottly Chart Streamlit

Ao executar esse código, o gráfico será exibido na interface do **Streamlit**, mostrando a distribuição dos livros por ano de publicação.

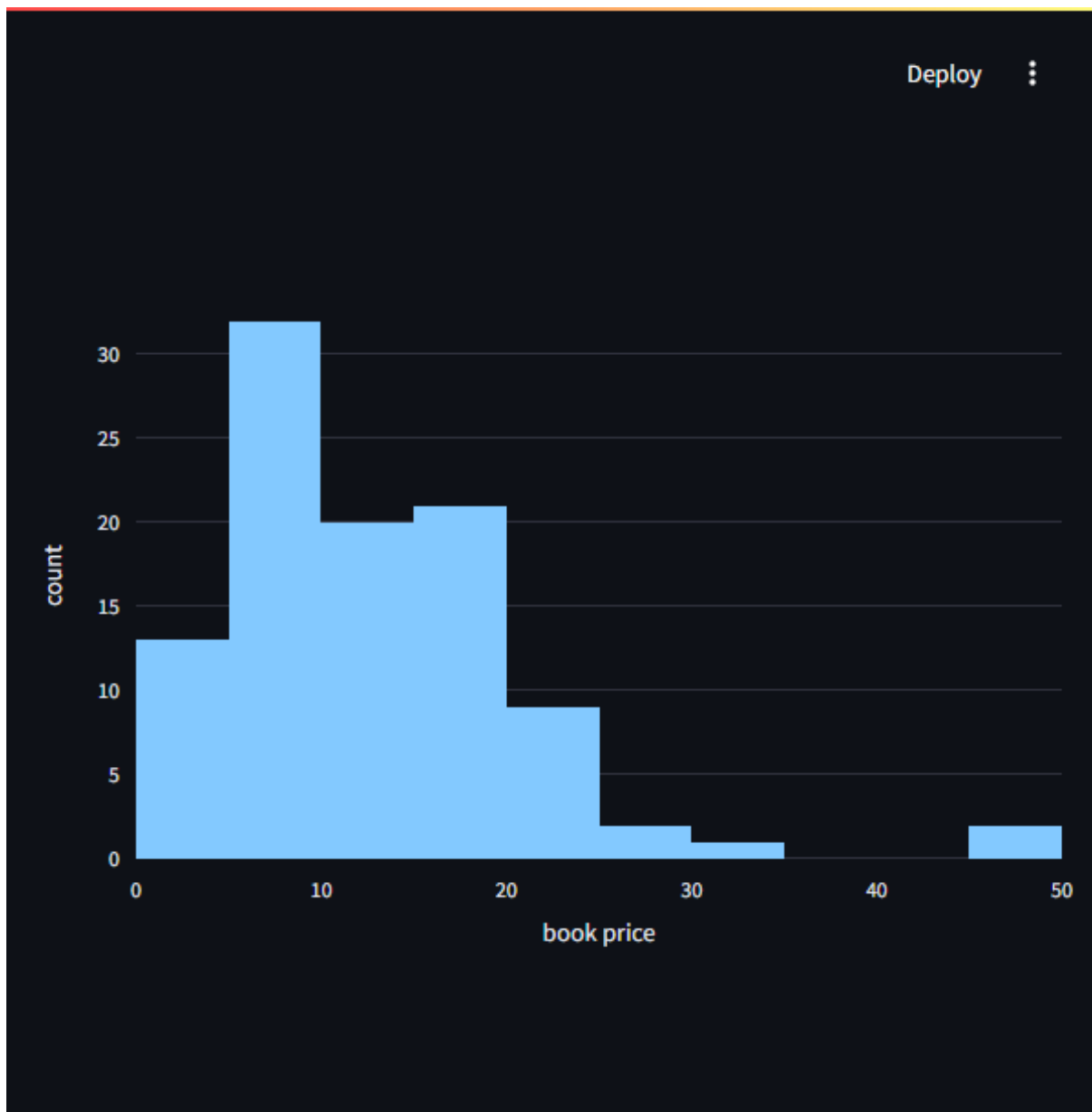
## Criando um Histograma de Preços

Para visualizar a distribuição dos preços dos livros, utilizamos um histograma, que pode ser criado com o método **histogram** do **Plotly Express**:

```
fig2 = px.histogram(df_top100_books, x='book price')
```

Para exibir o gráfico no **Streamlit**:

```
st.plotly_chart(fig2)
```



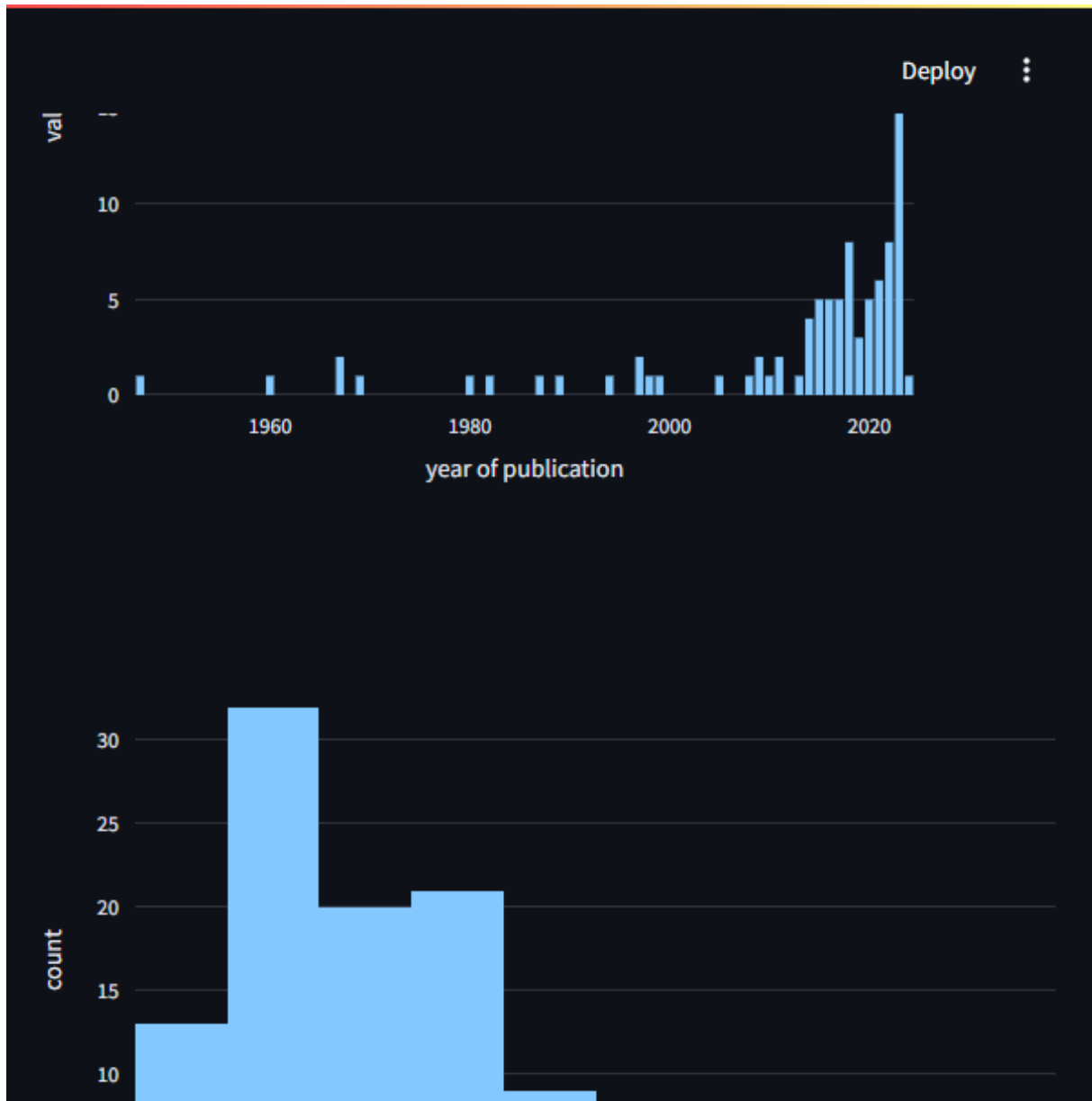
**Figure 8:** 4- Plottly Chart Streamlit

Agora temos dois gráficos exibidos, um representando a distribuição dos anos de publicação e outro representando a distribuição dos preços.

## Organizando os Gráficos em Colunas

No **Streamlit**, os gráficos podem ser organizados lado a lado utilizando o método **st.columns()**. Para dividir a tela em duas colunas e exibir um gráfico em cada uma delas, utilize:

```
col1, col2 = st.columns(2)
col1.plotly_chart(fig)
col2.plotly_chart(fig2)
```



**Figure 9:** 5- Plotly Chart Column Streamlit



Isso permite uma melhor disposição visual dos gráficos na interface.

## Conclusão

Com poucas linhas de código, conseguimos criar e exibir gráficos interativos para análise de dados. Além disso, organizamos a visualização utilizando colunas no **Streamlit**. Com essa base, é possível expandir a análise incluindo mais filtros e novos gráficos para explorar diferentes aspectos dos dados. No próximo tópico, finalizaremos o projeto e o curso.

## 17. Finalizando o Projeto

Nesta etapa, será finalizada a segunda página do projeto, chamada **Books Reviews**. Essa página terá um elemento de seleção de livros, que difere do slider da página anterior. Será utilizada uma caixa de seleção (*select box*), que permitirá listar e selecionar os livros individualmente.

Ao escolher um livro, serão exibidas informações como: - Nome do livro - Gênero - Preço - Avaliação (*rating*) - Ano de publicação

Além disso, será feita a exibição das descrições associadas a cada livro.

### Criando uma Nova Página no Streamlit

Para adicionar novas páginas no **Streamlit**, é necessário criar uma pasta específica dentro do projeto.

1. No diretório do projeto, criar uma pasta chamada `pages`.
2. Criar um novo arquivo dentro dessa pasta, por exemplo: `bookreviews.py`.
3. Importar a biblioteca **Streamlit** e testar a nova página:

```
import streamlit as st

st.write("Teste")
```

4. Salvar e atualizar o projeto. A nova página já estará disponível no menu lateral do Streamlit.

### Carregando os Dados

Para exibir as informações dos livros, é necessário carregar os dados corretamente. O primeiro passo é importar a biblioteca **Pandas** e ler os arquivos de dados:

```
import pandas as pd
import streamlit as st

df_reviews = pd.read_csv("datasets/customer_reviews.csv")
df_top100_books = pd.read_csv("datasets/Top-100 Trending Books.csv")
```

Em seguida, criar uma lista com os nomes únicos dos livros disponíveis:

```
books = df_top100_books["book title"].unique()
```

Essa lista será utilizada na caixa de seleção.

## Criando o Select Box para Seleção de Livros

O **Streamlit** possui um componente chamado `selectbox`, que permite criar uma caixa de seleção.

```
book = st.sidebar.selectbox("Escolha um livro", books)
```

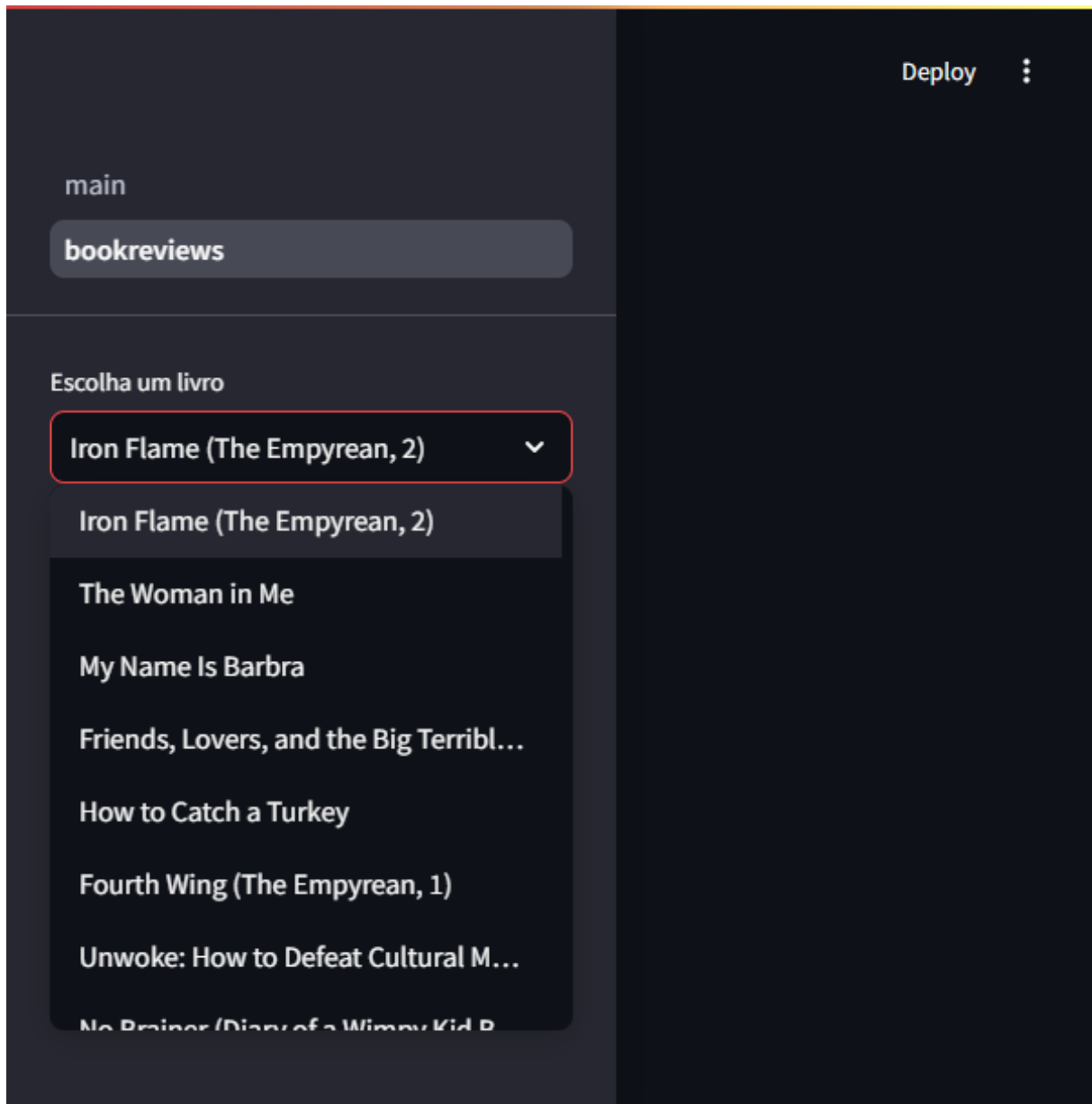
Com isso, o usuário poderá selecionar um livro na barra lateral do aplicativo.

## Filtrando os Dados

Após selecionar um livro, é necessário filtrar os dados do **DataFrame** para exibir apenas as informações do livro escolhido:

```
df_book = df_top100_books[df_top100_books["book title"] == book]
df_reviews_filtered = df_reviews[df_reviews["book name"] == book]
```

No código acima: - `df_book` contém apenas os dados do livro selecionado. - `df_reviews_filtered` contém apenas as avaliações associadas ao livro escolhido.



**Figure 10:** 1- Bookreviews Page

### Extraindo Informações do Livro

Como o **Pandas** retorna um subconjunto dos dados ao fazer um filtro, é necessário extrair os valores individuais corretamente:

```
book_title = df_book["book title"].iloc[0]
book_genre = df_book["genre"].iloc[0]
book_price = df_book["book price"].iloc[0]
```

```
book_rating = df_book["rating"].iloc[0]
book_year = df_book["year of publication"].iloc[0]
```

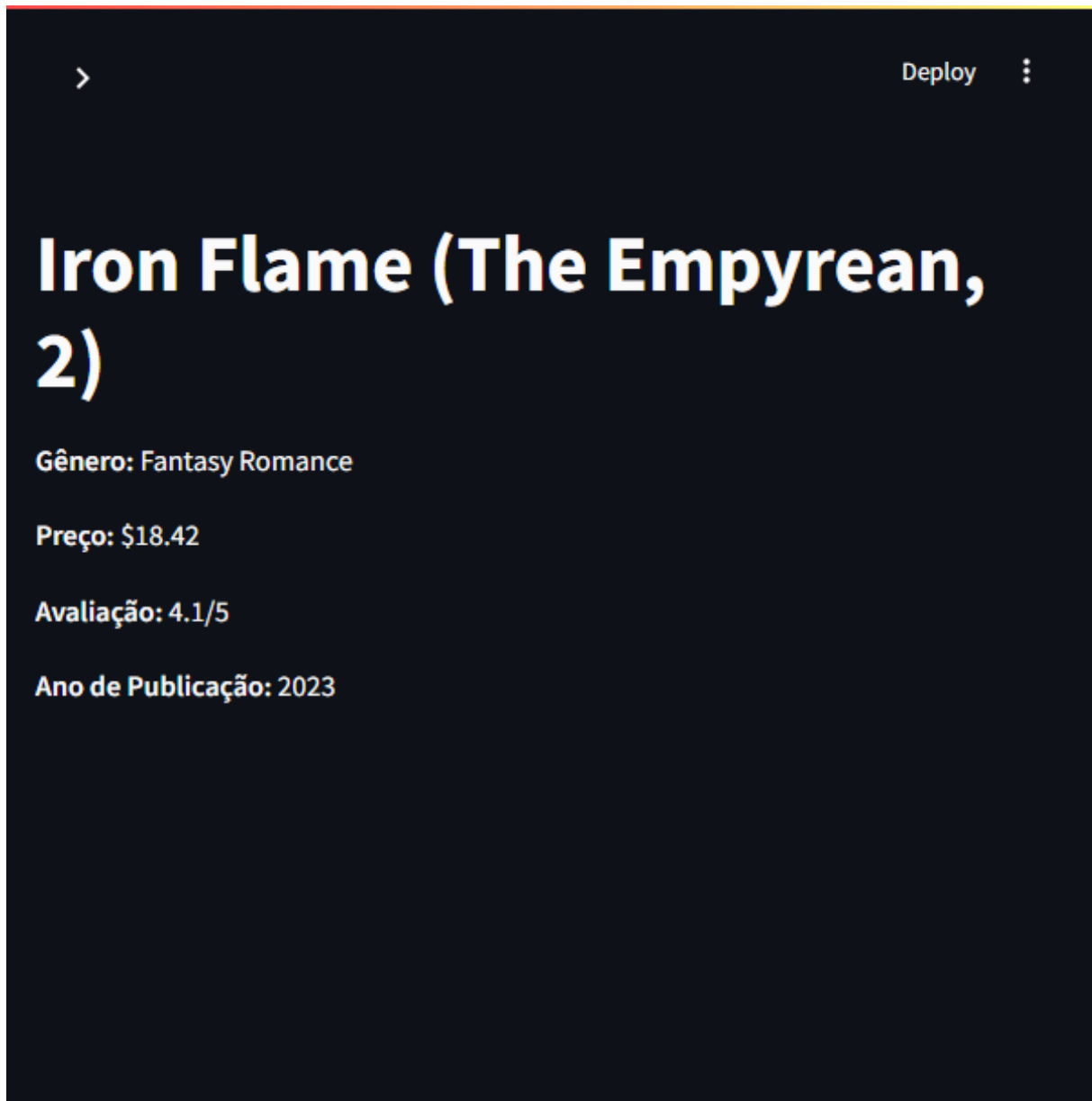
A função `.iloc[0]` garante que apenas um valor seja retornado, evitando problemas ao exibir os dados no Streamlit.

### Exibindo as Informações do Livro

Agora, os dados podem ser exibidos na tela utilizando os elementos de texto do **Streamlit**:

```
st.title(book_title)
st.write(f"***Gênero:** {book_genre}")
st.write(f"***Preço:** ${book_price:.2f}")
st.write(f"***Avaliação:** {book_rating}/5")
st.write(f"***Ano de Publicação:** {book_year}")
```

O uso de `:.2f` no preço garante que o valor seja exibido com duas casas decimais.



**Figure 11:** 2- Bookreviews Page

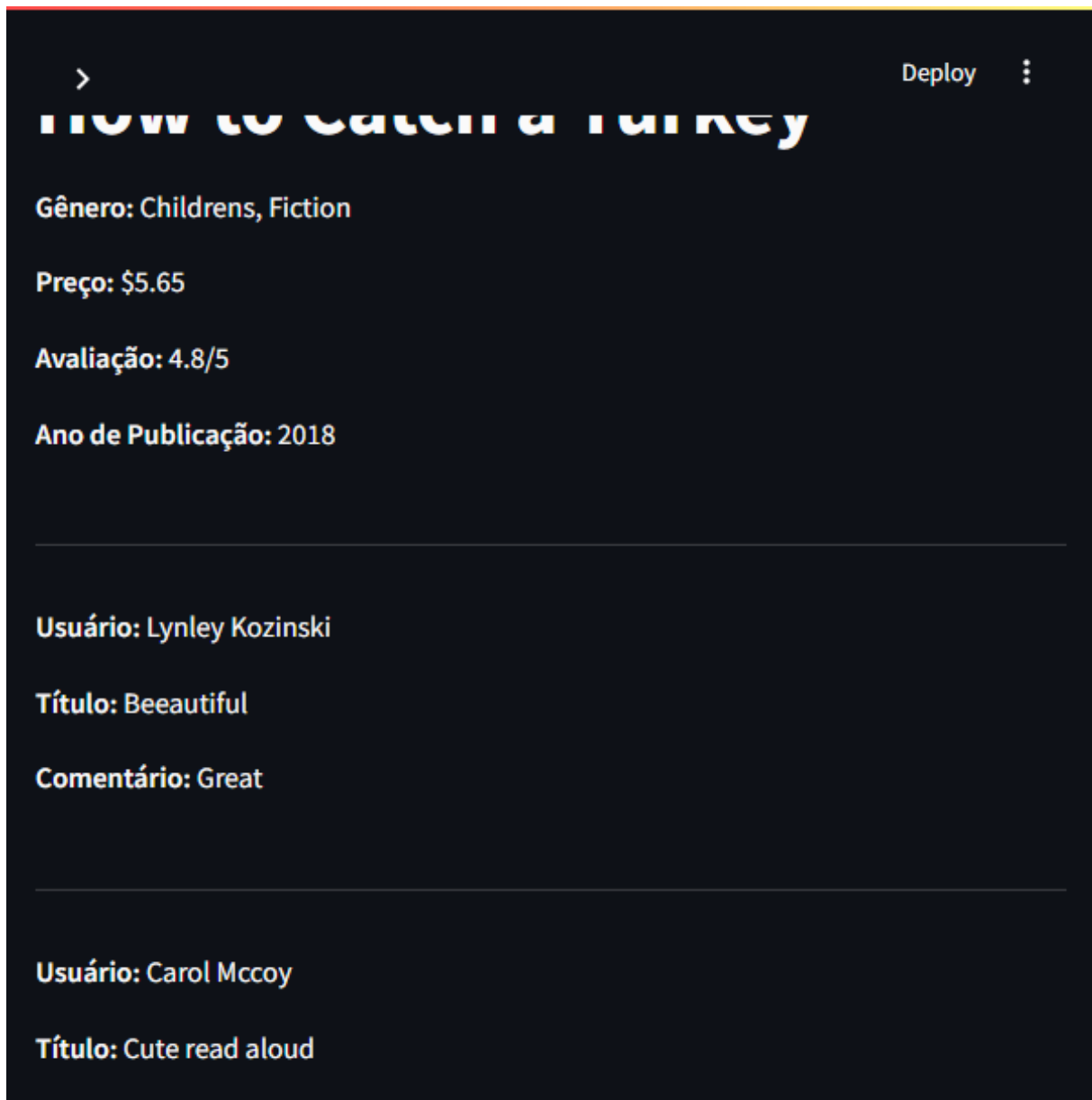
### Exibindo as Avaliações dos Livros

Para exibir as avaliações, é possível percorrer os dados filtrados e exibi-los com uma linha de separação entre cada avaliação:

```
st.write("----")
for _, review in df_reviews_filtered.iterrows():
    st.write(f"**Usuário:** {review['reviewer']}")
```

```
st.write(f"**Título:** {review['review title']}")
st.write(f"**Comentário:** {review['review description']}")
st.write("---")
```

O método `.iterrows()` percorre cada linha do **DataFrame**, permitindo exibir os comentários um a um.



**Figure 12:** 3- Bookreviews Page

## Conclusão

Com isso, a página **Books Reviews** está finalizada. Agora, o usuário pode selecionar um livro e visualizar suas informações, além de ler as avaliações associadas. Essa abordagem modular permite adicionar novas funcionalidades de forma simples, conforme necessário.

Código completo do projeto:

### app.py

```
import streamlit as st

import pandas as pd
import plotly.express as px

st.set_page_config(layout="wide")

df_reviews = pd.read_csv("datasets/customer reviews.csv")
df_top100_books = pd.read_csv("datasets/Top-100 Trending Books.csv")

st.write(df_reviews)

dfFiltrado = df_reviews[df_reviews["reviewer rating"] > 4]
dfTopBooks = df_top100_books[df_top100_books["book price"] < 50]

priceMax = dfTopBooks["book price"].max()
priceMin = dfTopBooks["book price"].min()

price_slider = st.slider(
    "Selecione o preço máximo", # Texto exibido acima do slider
    min_value=int(priceMin),    # Valor mínimo
    max_value=int(priceMax),    # Valor máximo
    value=int(priceMax)         # Valor inicial
)

fig = px.bar(df_top100_books['year of publication'].value_counts())
fig2 = px.histogram(df_top100_books, x='book price')

col1, col2 = st.columns(2)
col1.plotly_chart(fig)
col2.plotly_chart(fig2)
```

### bookreviews.py

```
import streamlit as st

import pandas as pd
import plotly.express as px

df_reviews = pd.read_csv("datasets/customer reviews.csv")
df_top100_books = pd.read_csv("datasets/Top-100 Trending Books.csv")

books = df_top100_books["book title"].unique()
```



```
book = st.sidebar.selectbox("Escolha um livro", books)

df_book = df_top100_books[df_top100_books["book title"] == book]
df_reviews_filtered = df_reviews[df_reviews["book name"] == book]

book_title = df_book["book title"].iloc[0]
book_genre = df_book["genre"].iloc[0]
book_price = df_book["book price"].iloc[0]
book_rating = df_book["rating"].iloc[0]
book_year = df_book["year of publication"].iloc[0]

st.title(book_title)
st.write(f"**Gênero:** {book_genre}")
st.write(f"**Preço:** ${book_price:.2f}")
st.write(f"**Avaliação:** {book_rating}/5")
st.write(f"**Ano de Publicação:** {book_year}")

st.write("---")
for _, review in df_reviews_filtered.iterrows():
    st.write(f"**Usuário:** {review['reviewer']}")
    st.write(f"**Título:** {review['review title']}")
    st.write(f"**Comentário:** {review['review description']}")
    st.write("---")
```