Este é o CS50

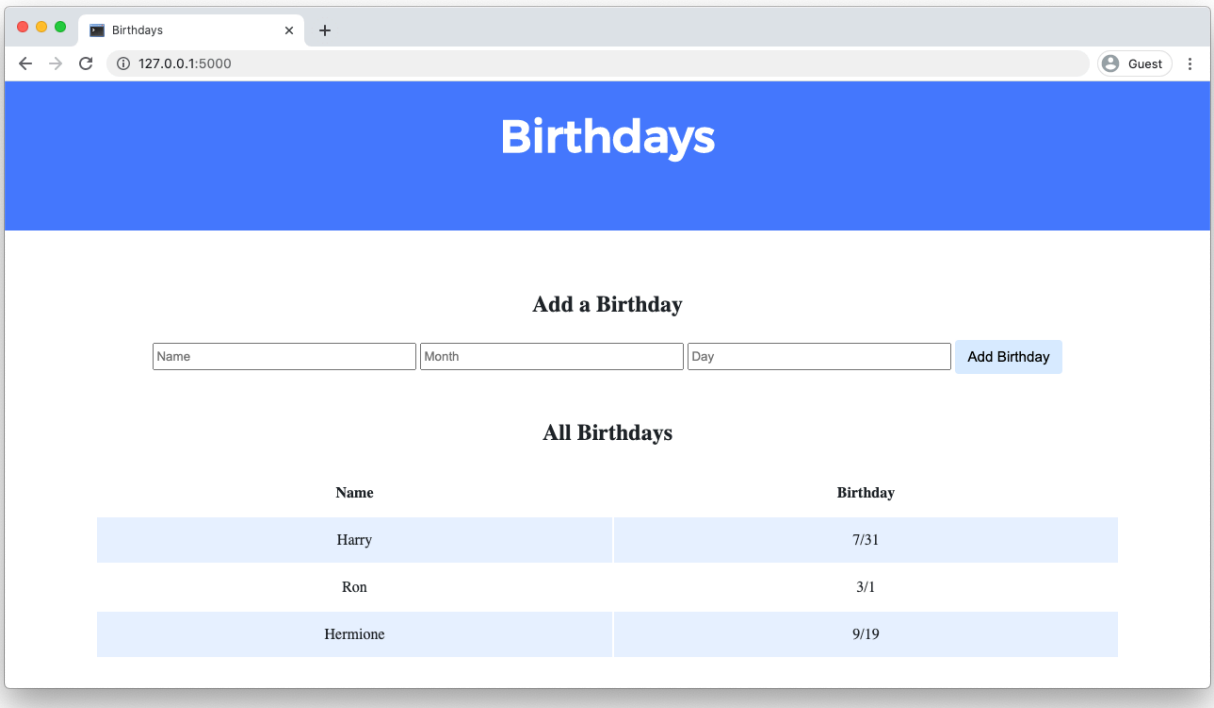
Introdução à Ciência da Computação (CS50)

OpenCourseWare

Doar (https://cs50.harvard.edu/donate)

David J. Malan (https://cs.harvard.edu/malan/)
malan@harvard.edu
(https://www.facebook.com/dmalan) (https://github.com/dmalan)
(https://www.instagram.com/davidjmalan/) (https://www.linkedin.com/in/malan/)
(https://www.reddit.com/user/davidjmalan)
(https://www.threads.net/@davidjmalan) (https://twitter.com/davidjmalan)

Aniversários



Problema a resolver

Crie um aplicativo web para registrar os aniversários dos amigos.

Começando

▼ **Baixe o código de distribuição.**

Abra cs50.dev (https://cs50.dev) .

Comece clicando dentro da janela do terminal e, em seguida, execute `cd` o comando. Você verá que o "prompt" será semelhante ao abaixo.

```
$
```

Clique dentro da janela do terminal e execute o seguinte comando.

```
wget https://cdn.cs50.net/2024/fall/psets/9/birthdays.zip
```

Em seguida, pressione Enter para baixar um arquivo ZIP chamado [nome do arquivo] `birthdays.zip` em seu espaço de código. Preste atenção ao espaço entre [nome do arquivo] `wget` e o URL a seguir, ou qualquer outro caractere!

Agora execute

```
unzip birthdays.zip
```

para criar uma pasta chamada `birthdays` . Você não precisa mais do arquivo ZIP, então pode executar

```
rm birthdays.zip
```

e responda com “y” seguido de Enter quando solicitado para remover o arquivo ZIP que você baixou.

Agora digite

```
cd birthdays
```

Em seguida, pressione Enter para entrar (ou seja, abrir) esse diretório. Seu prompt agora deve ser semelhante ao abaixo.

```
birthdays/ $
```

Se tudo correr bem, você deverá executar

```
ls
```

e você deverá ver os seguintes arquivos e pastas:

```
app.py  birthdays.db  static/  templates/
```

Se você encontrar algum problema, siga esses mesmos passos novamente e veja se consegue determinar onde errou!

Entendimento

Em `app.py`, você encontrará o início de uma aplicação web Flask. A aplicação possui uma rota (`/`) que aceita tanto `POST` requisições (após o `if`) quanto `GET` requisições (após o `else`). Atualmente, quando a `/` rota é requisitada via `GET`, o `index.html` template é renderizado. Quando a `/` rota é requisitada via `POST`, o usuário é redirecionado de volta para `/` via `GET`.

`birthdays.db` é um banco de dados SQLite com uma tabela, `birthdays`, que possui quatro colunas: `id`, `name`, `month`, e `day`. Já existem algumas linhas nessa tabela, embora, futuramente, sua aplicação web suporte a inserção de novas linhas!

No `static` diretório, você encontrará um `styles.css` arquivo contendo o código CSS para esta aplicação web. Não é necessário editar este arquivo, mas fique à vontade para fazê-lo se desejar!

No `templates` diretório existe um `index.html` arquivo que será renderizado quando o usuário visualizar sua aplicação web.

Detalhes da implementação

Implemente e desenvolva um aplicativo web que permita aos usuários armazenar e acompanhar suas datas de aniversário.

- Quando a `/` rota for solicitada via `GET`, seu aplicativo web deverá exibir, em uma tabela, todas as pessoas em seu banco de dados juntamente com suas datas de nascimento.
  - Primeiro, em `app.py`, adicione lógica ao seu `GET` tratamento de requisições para consultar o `birthdays.db` banco de dados e obter todos os aniversários. Passe todos esses dados para o seu `index.html` template.
  - Em seguida `index.html`, adicione a lógica para renderizar cada aniversário como uma linha na tabela. Cada linha deve ter duas colunas: uma coluna para o nome da pessoa e outra coluna para a data de nascimento.
- Quando a `/` rota for solicitada via `POST`, seu aplicativo web deverá adicionar um novo aniversário ao seu banco de dados e, em seguida, renderizar novamente a página inicial.
  - Primeiro, em `index.html`, adicione um formulário HTML. O formulário deve permitir que os usuários digitem um nome, o mês de aniversário e o dia de aniversário. Certifique-se de que o formulário seja enviado para `/` (sua "ação") com um método de `post`.

- Em seguida `app.py`, adicione a lógica de `POST` tratamento de requisições para `INSERT` uma nova linha na `birthdays` tabela, com base nos dados fornecidos pelo usuário.

Opcionalmente, você também pode:

- Adicionar a possibilidade de excluir e/ou editar registros de aniversário.
- Adicione quaisquer funcionalidades adicionais de sua escolha!

Dicas

Clique nos botões abaixo para ler algumas dicas!

▼ **Crie um formulário através do qual os usuários possam enviar suas datas de nascimento.**

Em `index.html`, observe o seguinte TODO:

```
<!-- TODO: Create a form for users to submit a name, a month, and a day -->
```

Lembre-se de que, para criar um formulário, você pode usar o `form` elemento HTML. Você pode criar um `form` elemento HTML com as seguintes tags de abertura e fechamento:

```
<form>
</form>
```

É claro que um formulário ainda precisa de campos de entrada (e um botão para que o usuário possa enviar o formulário!). Lembre-se de que `input` os elementos HTML criam, entre outras coisas, caixas de entrada dentro de um formulário. Você pode especificar o `type` atributo `text` para permitir que eles aceitem campos de texto `text` ou `number` texto. Também atribua `input` um atributo `id` aos elementos `name` para que você possa diferenciá-los.

```
<form>
    <input name="name" type="text">
    <input name="month" type="number">
    <input name="day" type="number">
</form>
```

Seu formulário pode se beneficiar de um botão que o usuário possa clicar para enviar seus dados. Adicione um `input` elemento do tipo `<button>` `submit`, que permitirá ao usuário fazer exatamente isso. Se você quiser que o próprio botão tenha um texto explicativo, tente definir o `value` atributo `text`.

```
<form>
    <input name="name" type="text">
    <input name="month" type="number">
    <input name="day" type="number">
```

```
    <input type="submit" value="Add Birthday">
</form>
```

Para onde os dados do usuário serão enviados? Atualmente, para lugar nenhum! Lembre-se de que você pode especificar um `action` atributo do formulário para definir qual rota deve ser solicitada após o envio do formulário. Os dados do formulário serão enviados juntamente com a solicitação resultante. O `method` atributo especifica qual método de solicitação HTTP usar ao enviar o formulário.

```
<form action="/" method="post">
    <input name="name" type="text">
    <input name="month" type="number">
    <input name="day" type="number">
    <input type="submit" value="Add Birthday">
</form>
```

Com isso, seu formulário deve estar perfeitamente funcional, embora ainda possa ser melhorado! Considere adicionar `placeholder` valores para dar um toque especial:

```
<form action="/" method="post">
    <input name="name" placeholder="Name" type="text">
    <input name="month" placeholder="Month" type="number">
    <input name="day" placeholder="Day" type="number">
    <input type="submit" value="Add Birthday">
</form>
```

Considere também adicionar alguma *validação no lado do cliente* para garantir que o usuário coopere com a intenção do seu formulário. Por exemplo, um `input` campo do tipo `< input type` `number` ="text" />` também pode ter um atributo `min` `min` e `max` um atributo `max` especificados, que determinam o valor mínimo e máximo que um usuário pode inserir.

```
<form action="/" method="post">
    <input name="name" placeholder="Name" type="text">
    <input name="month" placeholder="Month" type="number" min="1" max="12">
    <input name="day" placeholder="Day" type="number" min="1" max="31">
    <input type="submit" value="Add Birthday">
</form>
```

▼ **Adicionar o formulário enviado por um usuário ao banco de dados**

Em `app.py`, observe o seguinte TODO:

```
# TODO: Add the user's entry into the database
```

Lembre-se de que o Flask possui alguns métodos úteis para acessar dados de formulários enviados via `POST` ! Em particular:

```
# Access form data
request.form.get(NAME)
```

onde `NAME` se refere ao `name` atributo do `input` elemento específico com os dados enviados. Se seus `input` elementos fossem nomeados `name`, `month`, e `day`, você poderia acessar (e armazenar!) seus valores respectivamente da seguinte forma:

```
# Access form data
name = request.form.get("name")
month = request.form.get("month")
day = request.form.get("day")
```

Agora, os valores enviados pelo usuário nos elementos de entrada `name`, `month`, e `day` estão disponíveis para você como variáveis Python.

O próximo passo é adicionar esses valores ao seu banco de dados! Graças a esta linha em particular.

```
db = SQL("sqlite:///birthdays.db")
```

`app.py` Já foi estabelecida uma conexão com o servidor `birthdays.db` sob o nome <nome_do_servidor> `db`. Agora você pode executar consultas SQL chamando o comando `db.execute` com uma consulta SQL válida. Se você quisesse adicionar o aniversário de Carter em 1º de janeiro, poderia executar a seguinte instrução SQL:

```
INSERT INTO birthdays (name, month, day) VALUES('Carter', 1, 1);
```

Configure `app.py` para executar a mesma consulta, mas com marcadores de posição para os valores a serem inseridos, da seguinte forma:

```
# Access form data
name = request.form.get("name")
month = request.form.get("month")
day = request.form.get("day")

# Insert data into database
db.execute("INSERT INTO birthdays (name, month, day) VALUES(?, ?, ?)", name, month,
```

E pronto! Tente enviar o formulário, abrir o arquivo `birthdays.db` e usar uma `SELECT` consulta para visualizar o conteúdo da `birthdays` tabela. Você deverá ver os dados do formulário enviado disponíveis.

À medida que você cria aplicações mais avançadas, também precisará adicionar *validação no servidor* : ou seja, uma maneira de verificar se os dados do usuário são válidos *antes* de qualquer outra ação! Uma das primeiras validações que você pode fazer é verificar se o usuário enviou algum dado! Se você tentar recuperar dados de um formulário `request.form.get` onde o usuário não enviou nenhum, `request.form.get` o resultado será uma string vazia. Você pode verificar esse valor em Python da seguinte forma:

```
# Access form data
name = request.form.get("name")
if not name:
    return redirect("/")

month = request.form.get("month")
if not month:
    return redirect("/")

day = request.form.get("day")
if not day:
    return redirect("/")

# Insert data into database
db.execute("INSERT INTO birthdays (name, month, day) VALUES(?, ?, ?)", name, month,
```

Agora, você não irá inserir uma linha até ter certeza de que o usuário forneceu todos os dados necessários.

Algumas outras coisas ainda podem dar errado! E se o usuário, de fato, não fornecer um valor numérico para ` `month` id` ou `value` `day`? Uma maneira de verificar é `try` converter o valor para um inteiro com `getInt()` `int` e, se a conversão falhar, redirecionar o usuário de volta para a página inicial.

```
# Access form data
name = request.form.get("name")
if not name:
    return redirect("/")

month = request.form.get("month")
if not month:
    return redirect("/")
try:
    month = int(month)
except ValueError:
    return redirect("/")

day = request.form.get("day")
if not day:
    return redirect("/")
try:
    day = int(day)
except ValueError:
    return redirect("/")

# Insert data into database
db.execute("INSERT INTO birthdays (name, month, day) VALUES(?, ?, ?)", name, month,
```

E mesmo que o usuário tenha inserido um número, é melhor verificar se ele está dentro do intervalo correto!

```
# Access form data
name = request.form.get("name")
```

```
if not name:
    return redirect("/")

month = request.form.get("month")
if not month:
    return redirect("/")
try:
    month = int(month)
except ValueError:
    return redirect("/")
if month < 1 or month > 12:
    return redirect("/")

day = request.form.get("day")
if not day:
    return redirect("/")
try:
    day = int(day)
except ValueError:
    return redirect("/")
if day < 1 or day > 31:
    return redirect("/")

# Insert data into database
db.execute("INSERT INTO birthdays (name, month, day) VALUES(?, ?, ?)", name, month,
```

▼ **Renderizar aniversários em** `birthdays.db`

Depois que um usuário puder enviar datas de aniversário e armazená-las em `birthdays.db`, sua próxima tarefa é garantir que essas datas de aniversário sejam exibidas em `index.html`.

Primeiro, você precisará recuperar todos os aniversários de `birthdays.db`. Você pode fazer isso com a seguinte consulta SQL:

```
SELECT * FROM birthdays;
```

Veja o seguinte TODO em `app.py`:

```
# TODO: Display the entries in the database on index.html
```

Considere configurar `app.py` para executar esta consulta SQL sempre que a página for carregada com uma `GET` requisição:

```
# Query for all birthdays
birthdays = db.execute("SELECT * FROM birthdays")
```

Agora, todos os aniversários na `birthdays` tabela `birthdays.db` estão disponíveis para você em uma variável Python chamada `birthdays`. Em particular, os resultados da consulta SQL são armazenados como uma lista de dicionários. Cada dicionário representa uma linha retornada pela consulta e cada chave no dicionário corresponde a um nome de coluna da `birthdays` tabela (ou seja, “nome”, “mês” e “dia”).

Para renderizar esses aniversários em `index.html`, você pode usar `render_template` a função do Flask. Você pode especificar que o aniversário `index.html` deve ser renderizado com a `birthdays` variável especificando um argumento de palavra-chave, também chamado `birthdays`, e atribuindo a ele o valor da `birthdays` variável que você acabou de criar.

```
# Query for all birthdays
birthdays = db.execute("SELECT * FROM birthdays")

# Render birthdays page
return render_template("index.html", birthdays=birthdays)
```

Para ficar claro, o nome no lado esquerdo da vírgula `=` (, `birthdays`) é o nome sob o qual você pode acessar os dados de aniversários dentro `index.html` dela.

Agora que `index.html` você tem acesso aos dados de aniversário, pode usar o Jinja para renderizar os dados corretamente. O Jinja, assim como o Python, pode percorrer os elementos de uma lista. E o Jinja, assim como o Python, pode acessar elementos de um dicionário por suas chaves. Nesse caso, a sintaxe do Jinja para fazer isso é o nome do dicionário, seguido por um `&` `.`, e então o nome da chave a ser acessada.

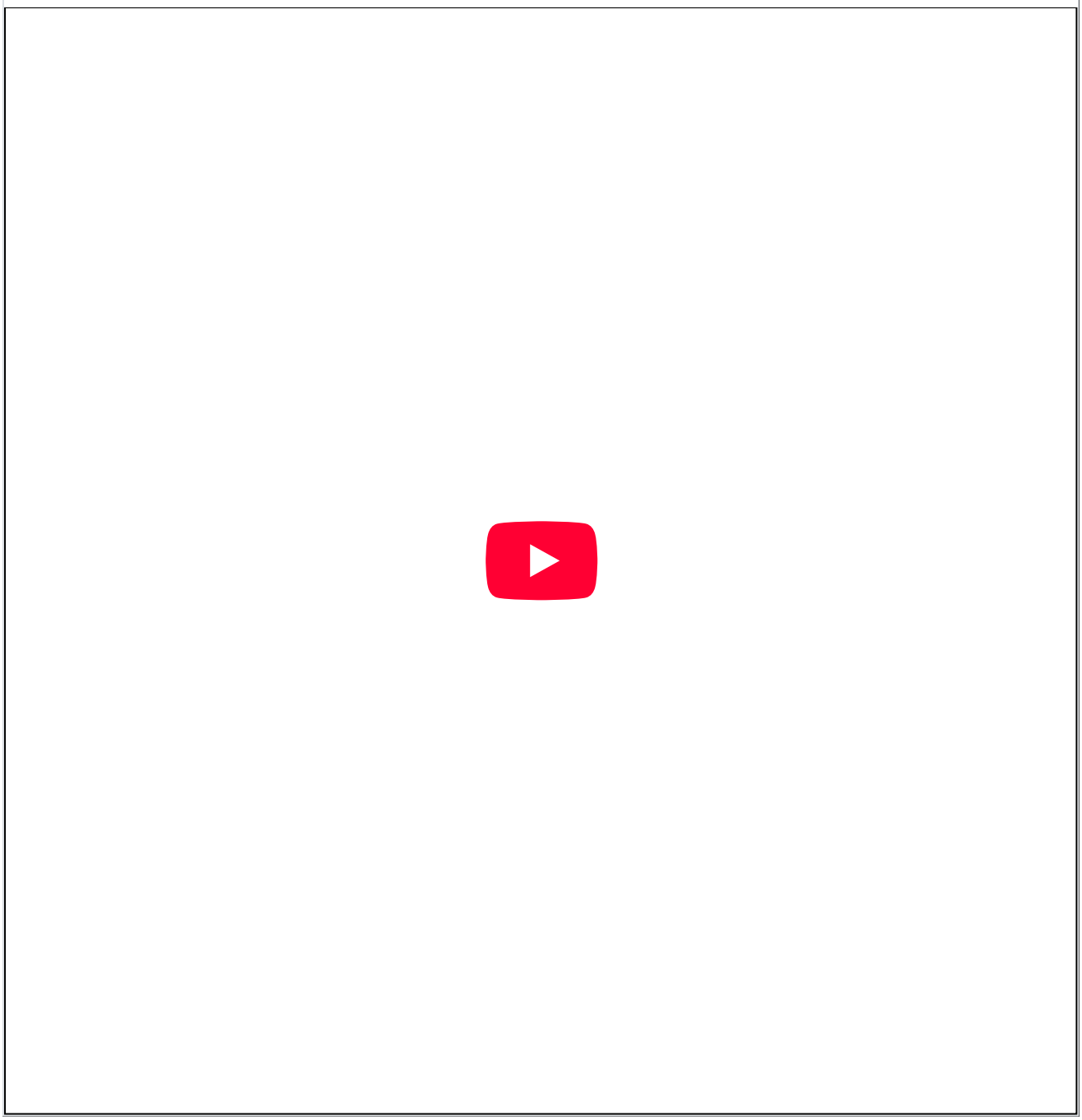
```
{% for birthday in birthdays %}
    <tr>
        <td></td>
        <td>/</td>
    </tr>
{% endfor %}
```

E pronto! Tente recarregar a página para ver as datas de aniversário exibidas.

Passo a passo

> Este vídeo foi gravado quando o curso ainda utilizava o ambiente de desenvolvimento integrado (IDE) CS50 para escrever código. Embora a interface possa parecer diferente do seu ambiente de código, o comportamento dos dois ambientes deve ser bastante similar!

CS50



▶ **Não sabe como resolver?**

Testando

Não `check50` para este conjunto de problemas! Mas certifique-se de testar sua aplicação web adicionando algumas datas de aniversário e verificando se os dados aparecem na tabela conforme o esperado.

Execute o seguinte comando `flask run` no terminal, estando no `birthdays` diretório correto, para iniciar um servidor web que execute a sua aplicação Flask.

Como enviar

```
submit50 cs50/problems/2025/x/birthdays
```