PROJECT

Adopt a Pet
Imagine you are looking to add a new furry friend to your family! On the pet adoption website, you browse through the categories of animals and select the one you're interested in, which brings you to another page that contains a list of available pets. Then, you continue your search by further clicking on an individual pet to view its profile page.

Every time you navigate to a different webpage, your browser is making a request to the web server. Thanks to routing, the server knows exactly which endpoint should handle the request and can return the correct HTML page to display.

In this project, you'll use Python's Flask framework to create a simple pet adoption site that contains multiple routes.

Let's get started!

Tasks

17/17 Complete
Mark the tasks as complete by checking them off

Set up the Flask app

1.
At the top of **app.py**, import the `Flask` class from the `flask` module.

If you run your code now, you will see a `NoAppException` error, but we will fix that in the next step when you create your app!
Hint
The syntax for importing is:

```
from module_name import ClassName
```

2.
Create an instance of the `Flask` class, passing in `__name__`, and save the object to a variable called `app`.

If you run your code now, you will see a URL not found error, but we will fix that in the next step when you create your first route!
Hint
The syntax for instantiating a class is:

```
object_name = ClassName(...)
```

Create the index route

3.
To create the `index` route, first define a function called `index()` that returns an HTML `<h1>` element with the text `Adopt a Pet!`. Remember that HTML can be returned as a string.
Hint
The syntax for defining a function is:

```
def function_name():
    # code goes here
```

4.
Use the `route()` decorator to bind the URL path `'/'` to the `index()` function.

Run your code now and you should see the heading displayed on the page!
Hint
Add the `route()` decorator right before the function definition:

```
@app_object.route(url_path)
```

5.
Let’s add some more elements to the page. Right after the `<h1>` element, add a `<p>` element that contains the text `Browse through the links below to find your new furry friend:`.
Hint
If you want to create a multiline string, remember to use triple quotes.
6.
Now after the `<p>` element, create an unordered list using `<ul>`. The bulleted list should contain three items: `Dogs`, `Cats`, and `Rabbits`. Remember to use `<li>` to create each item.
Hint
The syntax for an unordered list is:

```
<ul>
    <li>...</li>
    <li>...</li>
    <li>...</li>
</ul>
```

Create the animals route

7.

The site is looking good so far! The next step is to create individual pages for each animal type and link them in the bulleted list. To do that, we'll add a new `animals` route.

First, define a function called `animals()`. In the function body, create a string containing an `<h1>` element with the text `List of pets`, and assign it to the variable `html`. Return `html` from the function.
Hint
The syntax for defining a function is:

```
def function_name():
    # code goes here
```

8.
Use the `route()` decorator to associate the `animals()` function with the URL pattern `'/animals/X'`, where `X` is a variable section of the URL. Name the variable part `pet_type`.
Hint
Add the `route()` decorator right before the function definition:

```
@app_object.route(url_path)
```

The syntax for specifying a variable section of the URL path is `<variable_name>`.
9.
Next, update the `animals()` function to take a parameter called `pet_type`. In the function body, modify the `<h1>` heading to read `List of X`, where `X` is `pet_type`.
Hint
You can use Python f-strings to help format the HTML string.
10.
We're ready to create links on the index page that links to each individual animal page! Inside the `index()` function, turn each bulleted list item into a link by adding an `<a>` element within each `<li>` element:

- `Dogs` should link to `'/animals/dogs'`
- `Cats` should link to `'/animals/cats'`
- `Rabbits` should link to `'/animals/rabbits'`

Now run your code and try clicking on the links!
Hint
The linked list items should look like this:

```
<li><a href="url_path">link_text</a></li>
```

Populate page content

11.
Using the file navigator near the top left corner of the code editor, open up **helper.py**. This file contains a dictionary named `pets` that contains some data that we can use to populate the webpages.

The `pets` dictionary contains three elements, one for each animal type. The key is the animal type and the value is a list of dictionaries, each of which contains info about an individual pet.

Start by importing the `pets` dictionary at the top of **app.py**.
Hint
The syntax for importing is:

```
from file_name import object_name
```

12.
Inside the `animals()` function, you'll be modifying `html` to display the names of all available pets that are of `pet_type`.

Right before the `return` statement, create a for loop that iterates over each element in the list of pets. You can access the appropriate list of pets in the `pets` dictionary by the key, `pet_type`. Inside the loop, create a `<li>` element for each pet's name and concatenate the string to `html`.

Then, make sure to concatenate the opening `<ul>` tag to `html` before the loop and the closing `</ul>` tag after the loop, such that the `<li>` elements would be nested inside the `<ul>` element.

If you run your code and navigate to each animal page, you can see a bulleted list of available pets!
Hint
The syntax for accessing dictionary elements by key is:

```
dictionary_name[key]
```

You can use the `+` operator (or shorthand `+=`) to concatenate strings.

Create the pet route

13.
The next step is to create and link to individual profile pages for each pet. To do that, we'll add a new `pet` route.

Define a function called `pet()` that is associated with the URL pattern `'/animals/X/#'`, where `X` and `#` are variable sections of the URL. The section indicated by `X` should be called `pet_type` and the section indicated by `#` should be called `pet_id`. Use a converter to specify that `pet_id` must be a positive integer.

Then, pass `pet_type` and `pet_id` to the `pet()` function.
Hint
The syntax for defining a function is:

```
def function_name():
    # code goes here
```
Add the `route()` decorator right before the function definition:

```
@app_object.route(url_path)
```
The syntax for specifying a variable section of the URL with a converter is `<converter:variable_name>`.
14.
In the function body, create a variable called `pet` that stores the profile information of the pet who is of `pet_type` and has index position `pet_id` in its list of pets.

In other words, first access the appropriate list of pets in the `pets` dictionary by the key, `pet_type`. Then, access the appropriate dictionary in the list of pets by the index position, `pet_id`.

Your resulting `pet` dictionary will look like this:

```
{
  'name': ...,
  'age': ...,
  'breed': ...,
  'description': ...,
  'url': ...
}
```
Hint
The syntax for accessing dictionary elements by key is:

```
dictionary_name[key]
```
The syntax for accessing list elements by index position is:

```
list_name[index]
```
15.
Return an HTML `<h1>` element containing the pet's name from the `pet()` function. You can access the pet's name from the `pet` dictionary you created in the previous step.
Hint

The syntax for accessing dictionary elements by key is:

```
dictionary_name[key]
```

16.

Now, we’re ready to create links on the animal page that links to each individual pet profile page! Inside the `animals()` function, turn each bulleted list item into a link by adding an `<a>` element within each `<li>` element.

The URL we want to link to should follow the pattern `'/animals/X/#'`, where `X` is `pet_type` and `#` is the index position. In order to get the latter, we must modify the for loop by using `enumerate()` to simultaneously loop over indices.

Once you’re done, run your code and try navigating to an individual pet’s profile page.

Hint

The linked list items should look like this:

```
<li><a href="url_path">link_text</a></li>
```

The syntax for using `enumerate()` to simultaneously loop over item and index is:

```
for idx, item in enumerate(iterable):
    # code goes here
```

17.

Finally, let’s add some more content to the profile page! Inside the `pet()` function, right after the `<h1>` element, add the following elements to display the profile info stored in the `pet` dictionary:

- `<img>` to display the photo at the given URL
- `<p>` that contains the pet’s description
- `<ul>` with two `<li>` for the pet’s breed and age

Hint

The syntax for an image element is:

```
<img src="url_path" />
```

If you want to create a multiline string, remember to use triple quotes.