launch_code

LC101 2.6

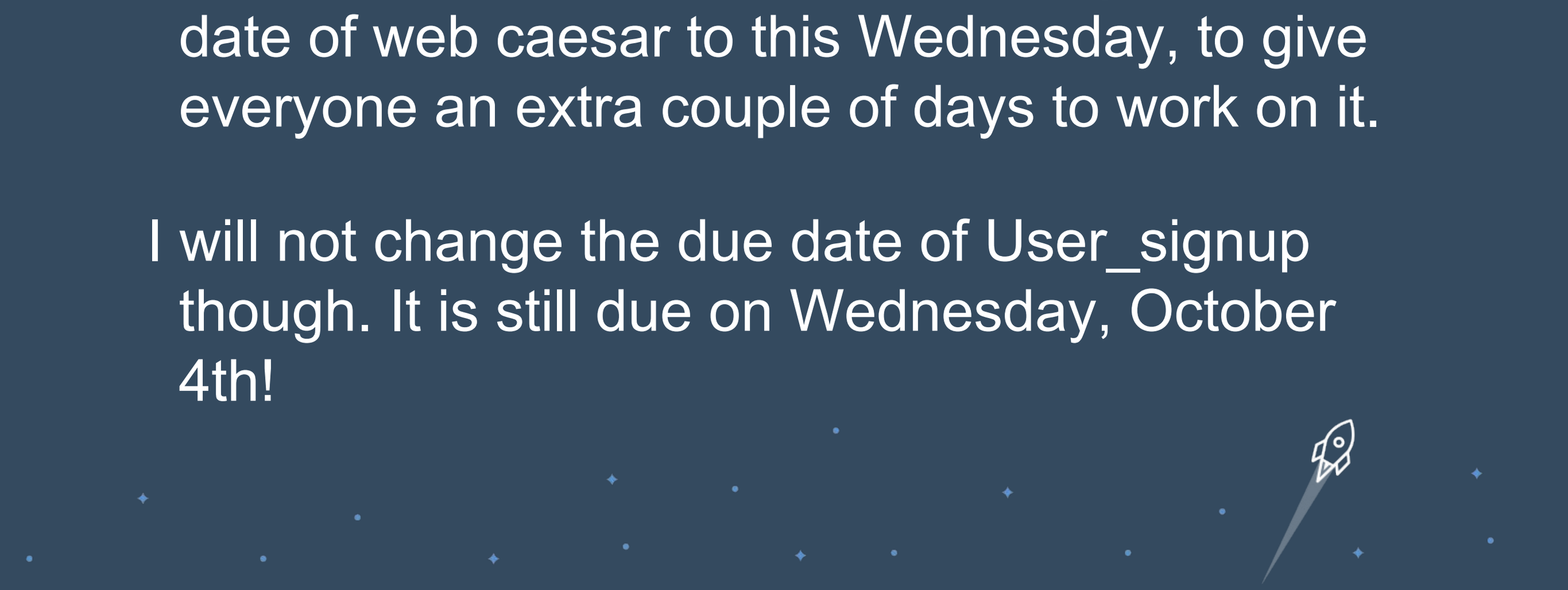
Class Agenda

1. Announcements
2. Last Class Review
3. New Material
4. Studio (Flicklist 4)

Announcement

Since I have been unavailable most of last week, and this past weekend, I am pushing the due date of web caesar to this Wednesday, to give everyone an extra couple of days to work on it.

I will not change the due date of User_signup though. It is still due on Wednesday, October 4th!



Review

Validation - Verifying user input

Client Side Validation - validating input on the front end, like with HTML Escaping

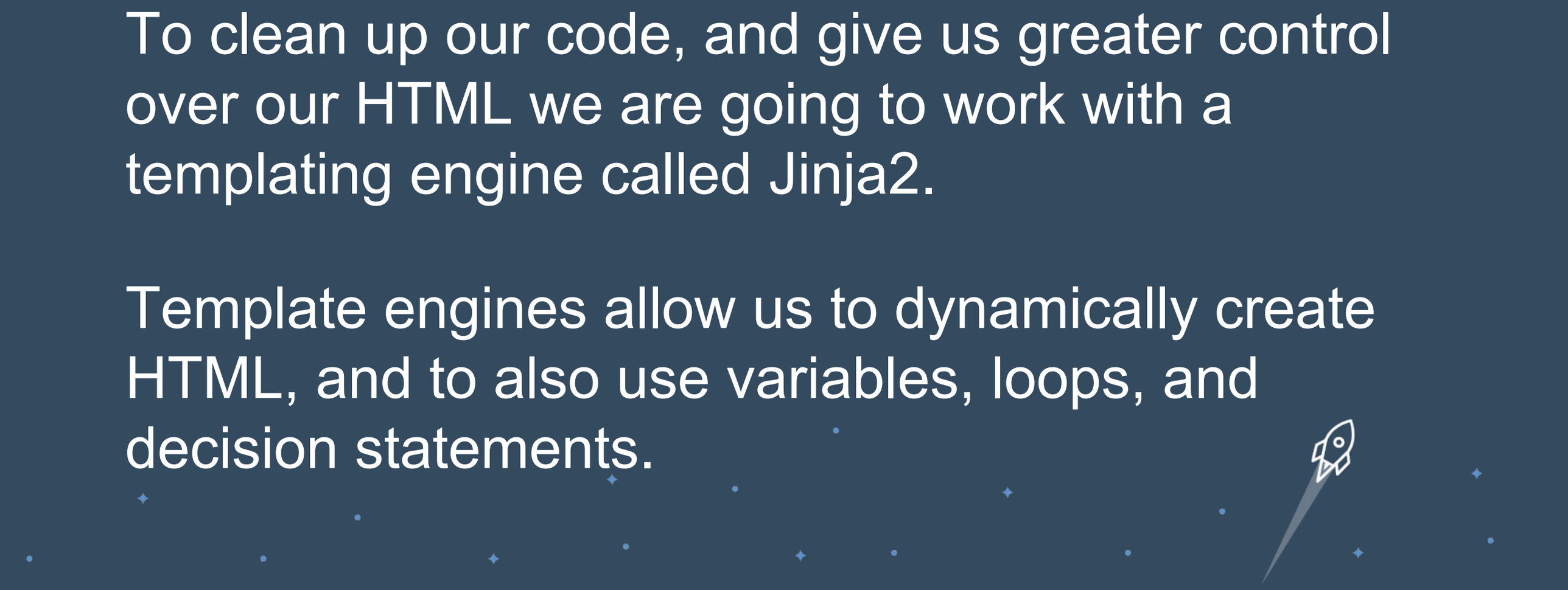
Server Side Validation - validating input on the back end, like with Python - Flask, and Redirect

Templating

So far we have been hard coding HTML into our Flask files.

To clean up our code, and give us greater control over our HTML we are going to work with a templating engine called Jinja2.

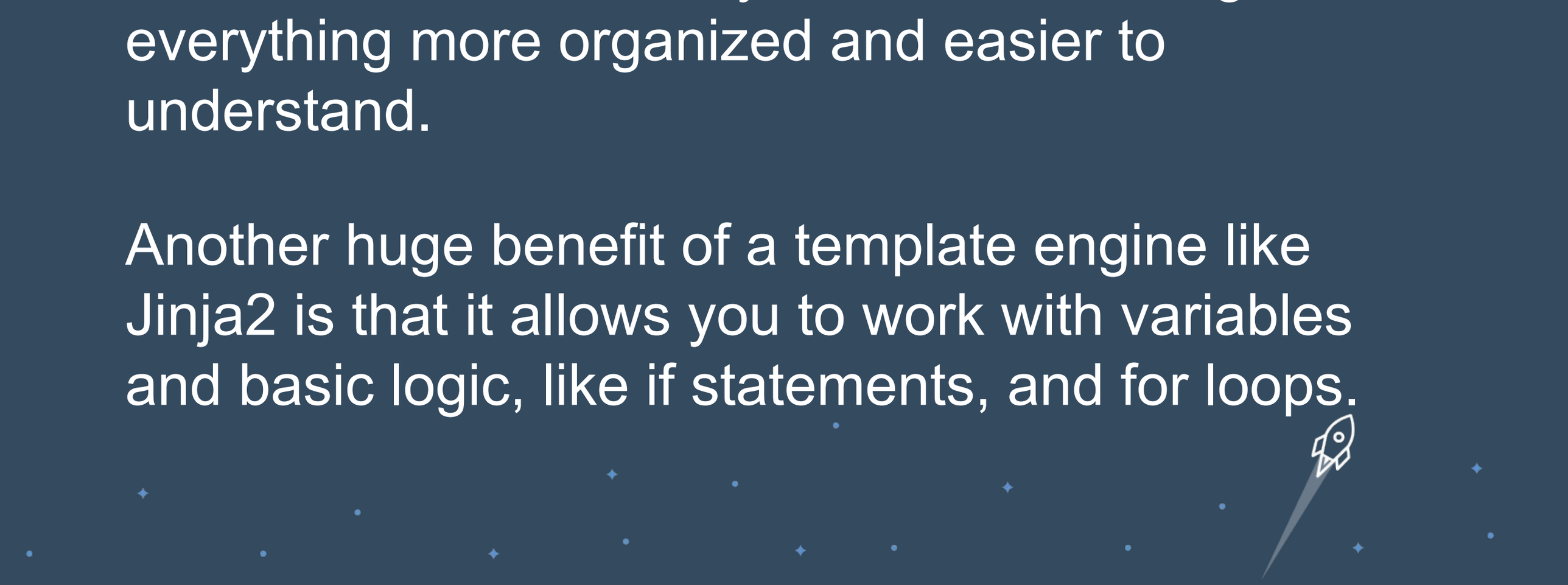
Template engines allow us to dynamically create HTML, and to also use variables, loops, and decision statements.



Jinja2

A template engine allows you to separate your HTML and the code for your server. Making everything more organized and easier to understand.

Another huge benefit of a template engine like Jinja2 is that it allows you to work with variables and basic logic, like if statements, and for loops.



Jinja2 Variables

In Jinja2 we can reference a variable by using double curly brackets {{*variable_name*}}

This gives us the ability to create dynamic web pages.

It also is much easier than using string.format() - as we need to manually set all of the variables, and hard code them to empty strings, the first time a form loads. Which is lots of extra work for us, and creates code that is harder to read.

Templates allow us to bypass that issue.

Let’s see an example: variable_example

Jinja2 continued

In order to use Jinja2 we had to do a few things to our main.py file.

We needed to import os, and import jinja2

We needed to create a template_dir

We needed to create a jinja_env, using the template_dir

We needed to create a template using jinja_env

Finally we needed to call .render() on the template we use

Jinja2 and Validation

One of the very cool things we learned about Jinja2 from our prep work is that it gives us the ability to HTML escape all templates with a global variable in our jinja_env.

Jinja2 Decision Statement

Jinja2 allows us to make if statements!

```
{% if something == True %}

<p>Something was True</p>

{% elif something == False %}

<p>Something was False</p>

{% endif %}
```

Jinja2 Loops

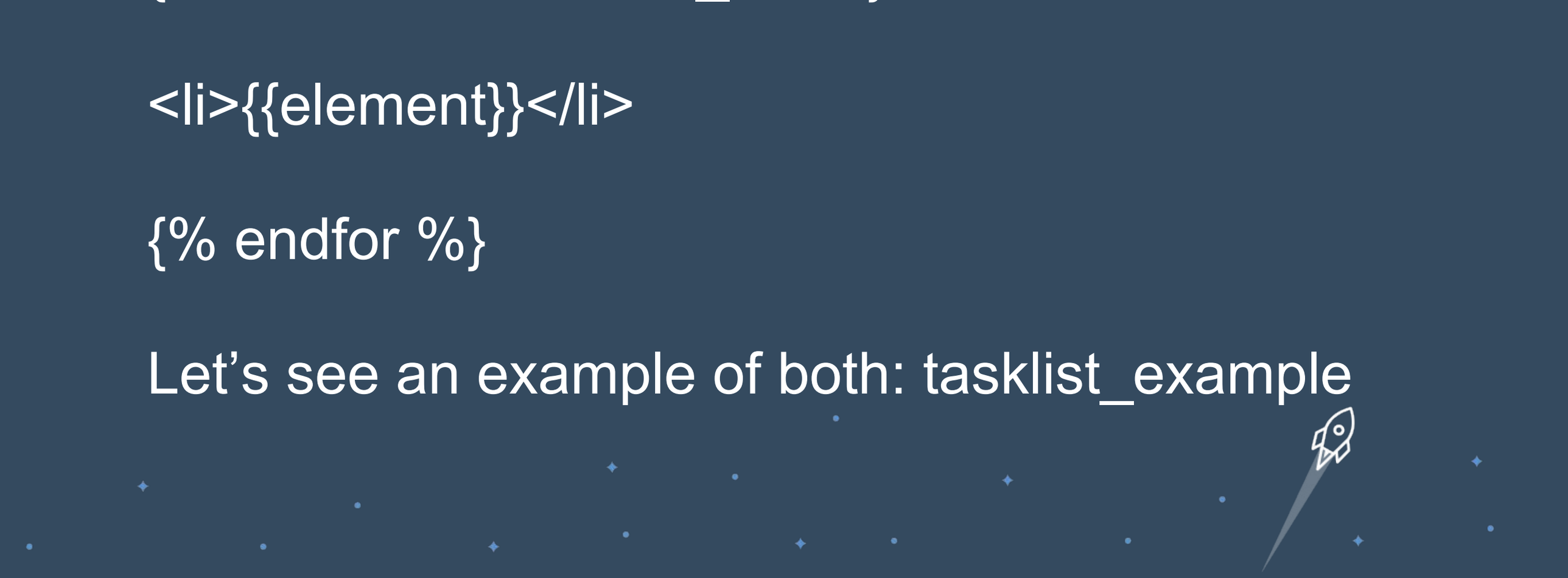
Jinja2 allows us to use looping statements!

{% for element in test_list %}

{{element}}

{% endfor %}

Let’s see an example of both: tasklist_example



Template Extensions

We can create HTML templates, and call them into other HTML files. We can create blocks, and then plug the blocks together.

{% block content %}

{% endblock %}

This creates a location we can put some code into later!

Example: block_example

Jinja2 Documentation

You can find the Jinja2 Documentation here:

http://jinja.pocoo.org/docs/2.9/

Render_template()

Our prep work had us manually create a jinja environment, create templates from that environment and then render them.

Flask gives us the ability to do this all in one step, with the render_template() method.

Example: render_example

Studio

Flicklist 4

Working with render_template, template extensions, and variables, ifs, and for loops with Jinja2.