launch_code

LC101 2.11

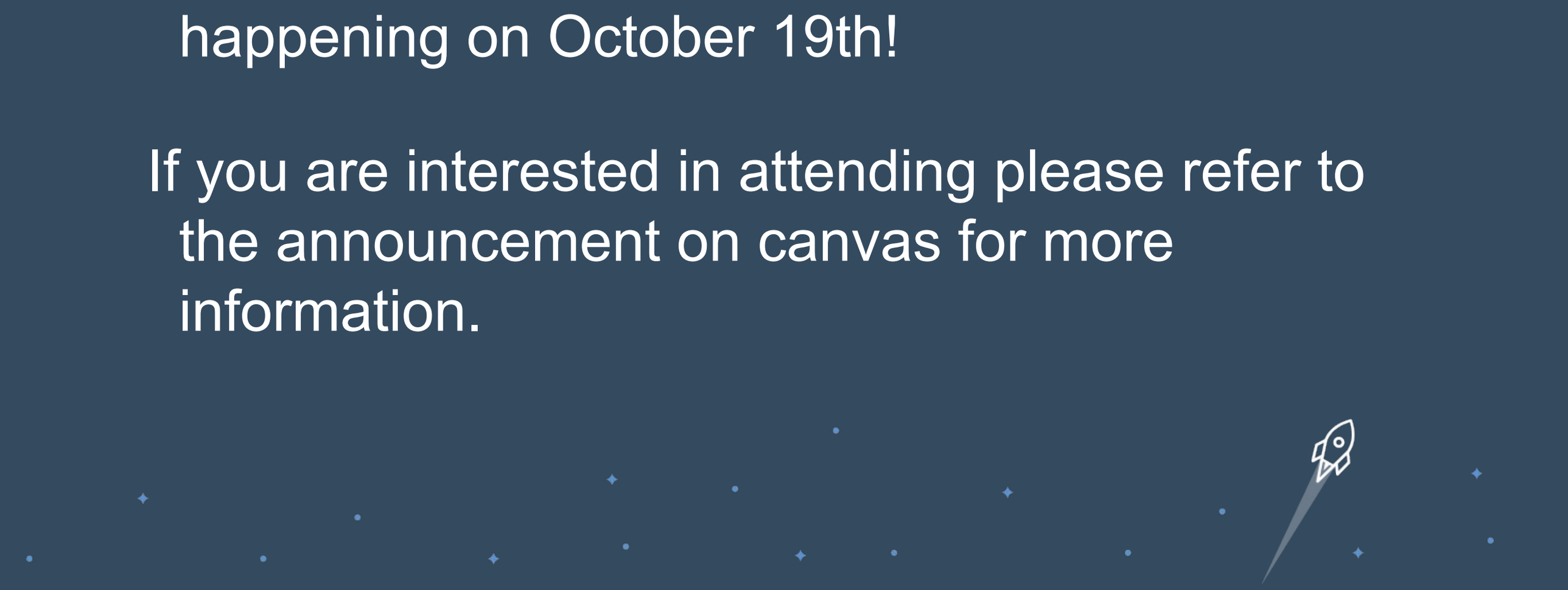
Class Agenda

1. Last Class Review
2. Announcement (Geek Gala)
3. Lecture
4. Work time for projects!

Announcement

GeekGala - LaunchCode's birthday party is happening on October 19th!

If you are interested in attending please refer to the announcement on canvas for more information.



Review

ORM - Object Relational Mapping

1. Creating a user, and DB from phpMyAdmin
2. Configuring our DB connection from a Flask Application
3. Creating a db object from SQLAlchemy
4. Creating classes and linking them to our DB using the db object
5. Using HTML forms to get input from the user then adding, and committing to our DB from our flask app
6. Querying our DB from our Flask app, and using the results to validate information, or display to the user

Session

Last class we talked about a session.

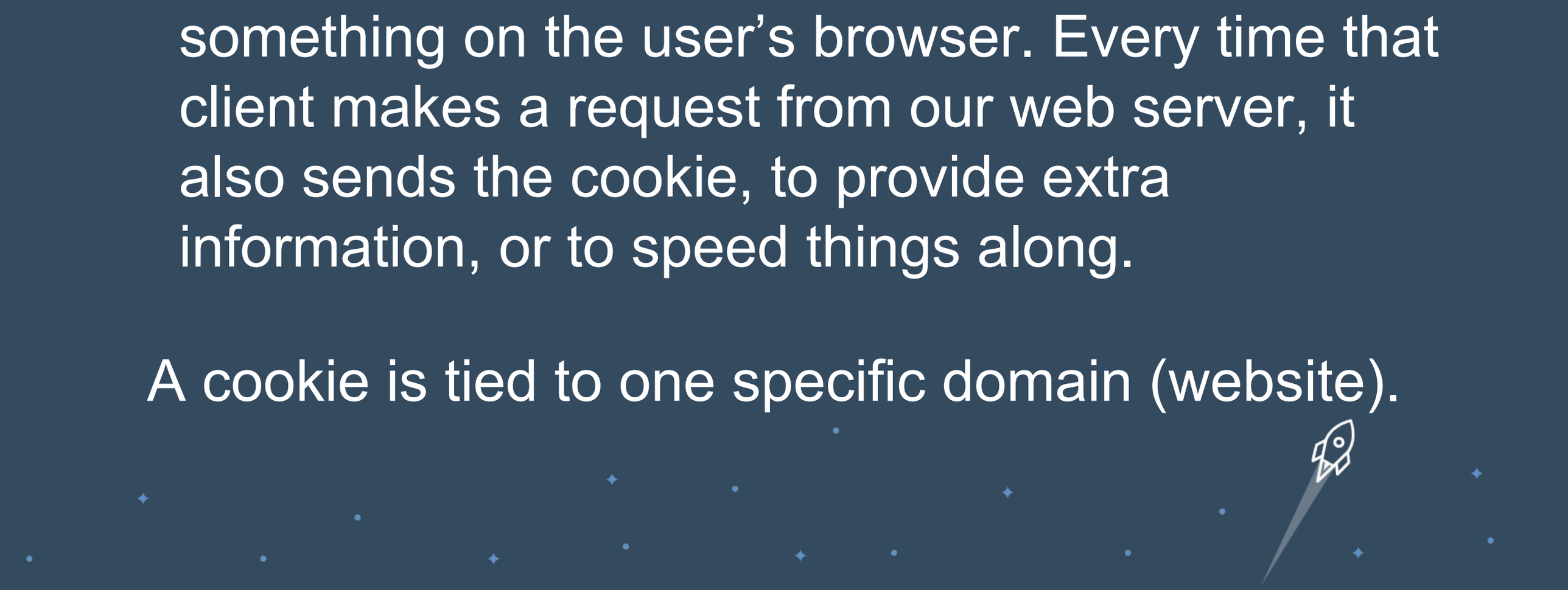
A session is server-side temporary storage. In our example last class we created a session that stored the user's email address, and saved it until the user logged out. This way our application "knew" the user was logged in.

Cookie

Today we expand on the idea of temporary storage, by talking about cookies.

A cookie is client-side storage. We save something on the user's browser. Every time that client makes a request from our web server, it also sends the cookie, to provide extra information, or to speed things along.

A cookie is tied to one specific domain (website).



Common Uses for Cookies

In our prep work, it outlined three things cookies are commonly used for:

1. Storing user login information.
2. Storing small pieces of information to keep the user from hitting the DB multiple times.
3. Tracking for ad purposes.

HTTP: Request & Response

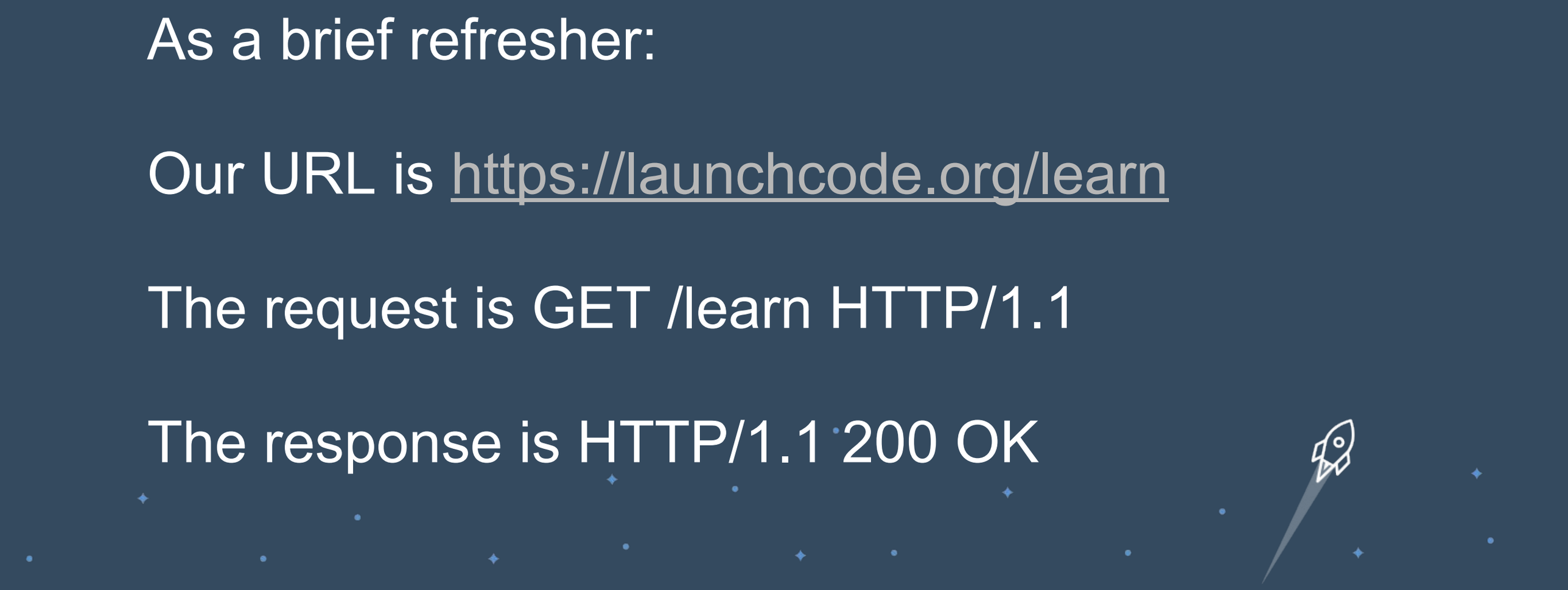
At this point in our class, we have seen HTTP requests, and responses many times.

As a brief refresher:

Our URL is https://launchcode.org/learn

The request is GET /learn HTTP/1.1

The response is HTTP/1.1 200 OK



Cookie works through HTTP Headers

Early in the class we talked about how an HTTP Request, and HTTP Response can contain headers that carry additional information. Cookies are among that additional information.

HTTP request POST /login HTTP/1.1

HTTP response Set-Cookie:user_id=12345; HTTP/1.1 200 OK

HTTP request Cookie:user_id=12345; GET /home HTTP/1.1

HTTP response HTTP/1.1 200 OK

Cookie Domains

The domain of your cookie is how your browser knows when to send your cookie.

It will only send the cookies stored in the browser when you connect to the specific domain stated in the cookie.

Browser Cookie Settings

Each browser has settings for cookies. You may have turned them off, or cleared them, or blocked a site from storing a cookie on your browser. Your website's users may be the same! You can’t assume cookies will work for every user.

You can find these settings in **firefox** by going to: “about:preferences#privacy” in your browser.

You can then select Use custom settings for history in the History drop down box, and it will give you options on how to handle cookies!

Cookie Expiration

Cookies expire when you close your browser, unless you specifically set their expiration date.

You do this when you are setting the cookie.

Set-Cookie:user_id=12345; Expires=Tue, 1, Jan 2025 00:00:0 GMT;

Flask - Cookies

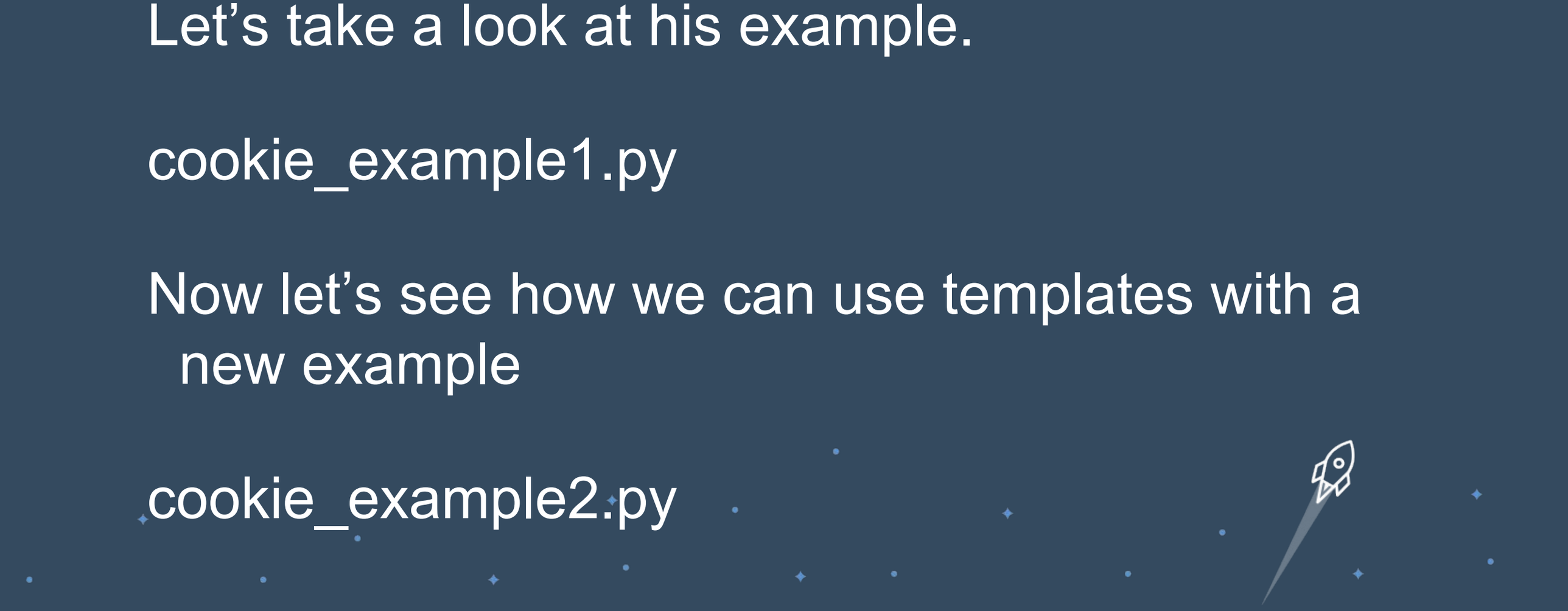
In our prepwork we saw Chris work with cookies, by importin make_response

Let’s take a look at his example.

cookie_example1.py

Now let’s see how we can use templates with a new example

cookie_example2.py



Cookie based sessions

Last class we worked with session[‘email’] = email

That’s how we set a user to being “logged in” to our application.

At the time we didn’t really know much about it other than it was a way to store information through states of our application.

What we’ve learned now is that a session is information stored on a server, and a cookie is stored in a browser. However, Flask sends the server session to the browser as a cookie, and automatically encrypts it for us!