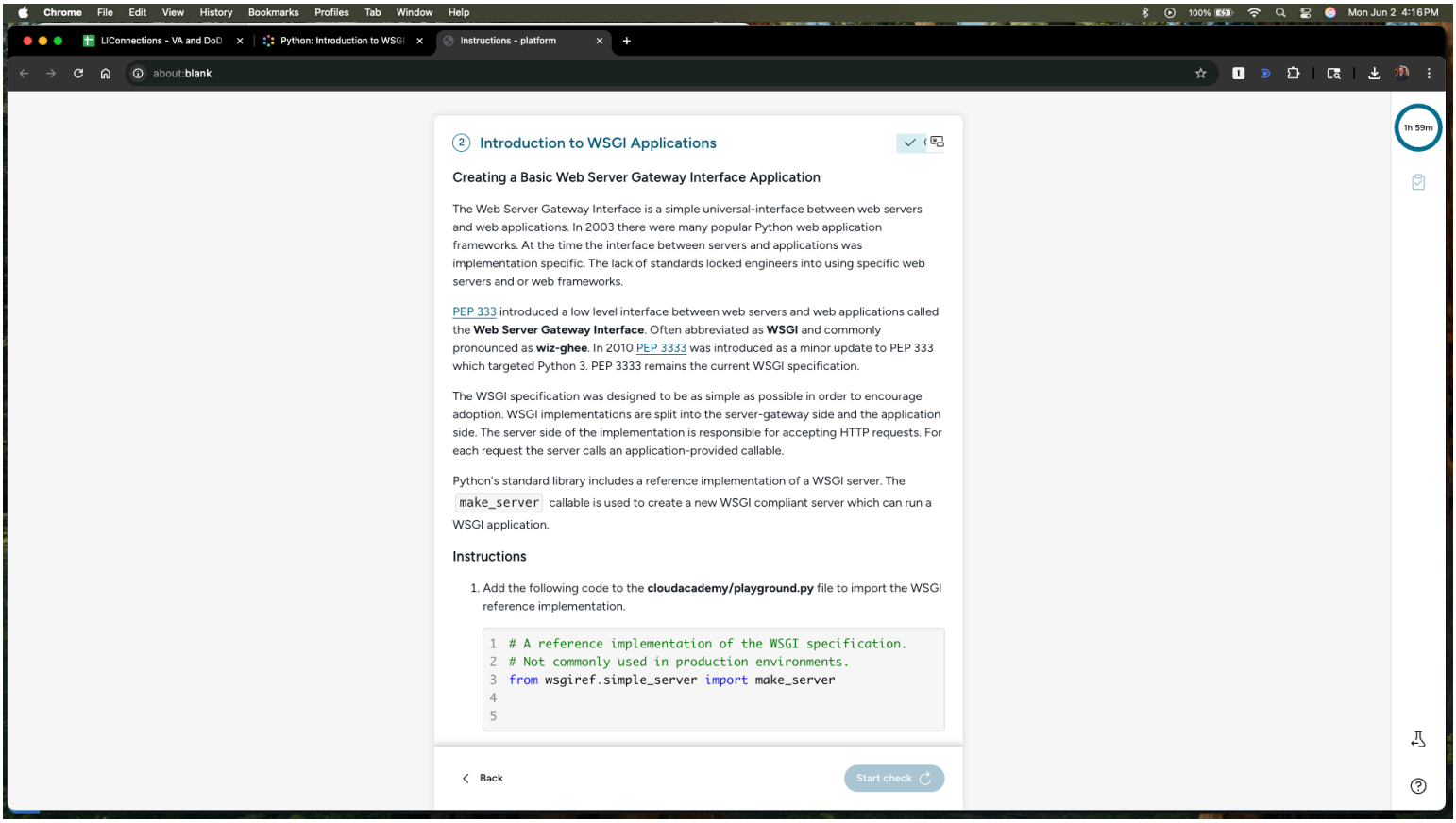
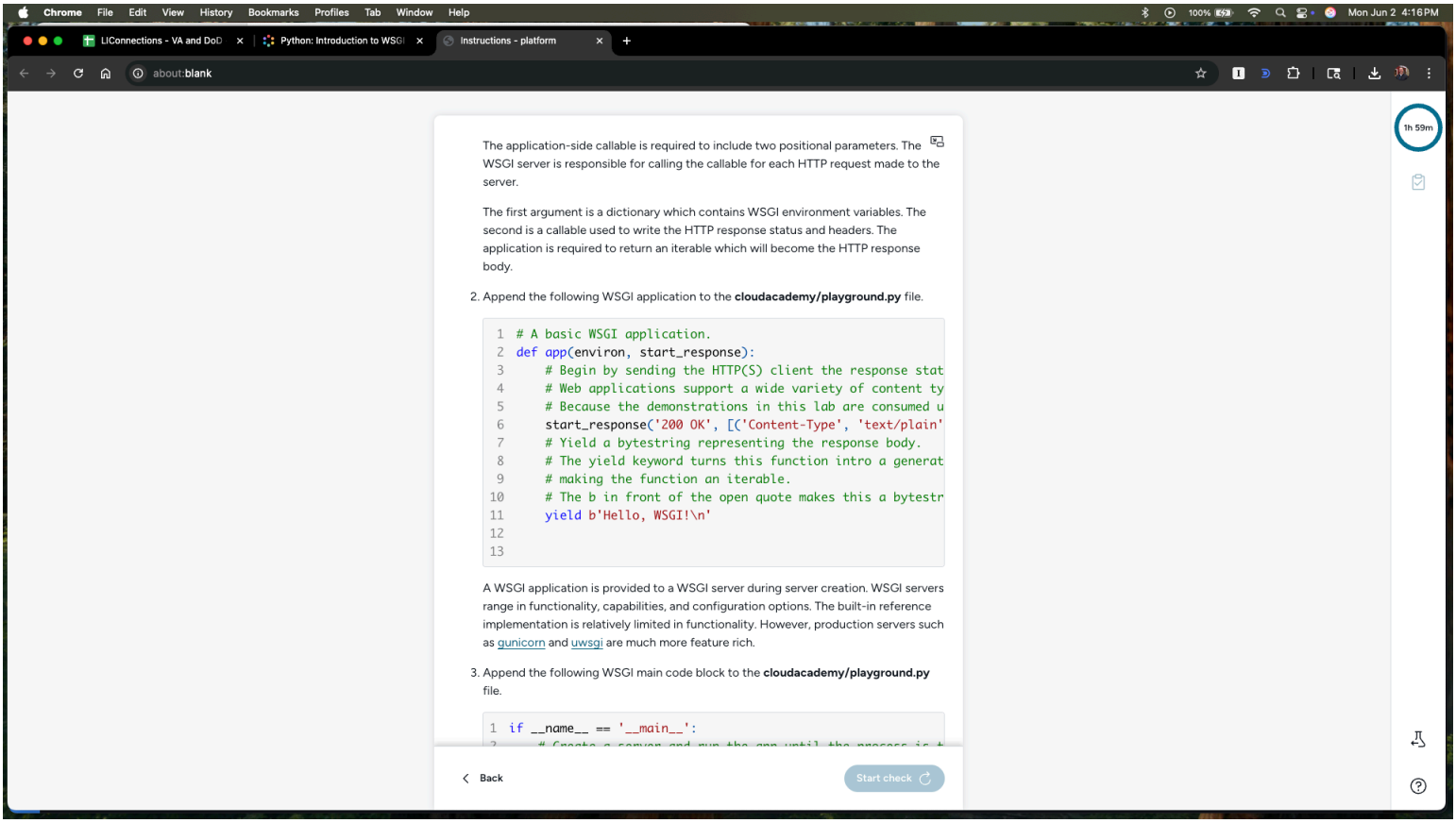
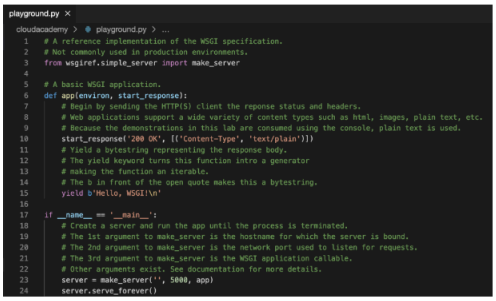

Chrome File Edit View History Bookmarks Profiles Tab Window Help
Mon Jun 2 4:16 PM
LIConnections - VA and DoD
Python: Introduction to WSGI
Instructions - platform
about:blank
1h 59m
2 Introduction to WSGI Applications
Creating a Basic Web Server Gateway Interface Application
The Web Server Gateway Interface is a simple universal-interface between web servers and web applications. In 2003 there were many popular Python web application frameworks. At the time the interface between servers and applications was implementation specific. The lack of standards locked engineers into using specific web servers and or web frameworks.
PEP 333 introduced a low level interface between web servers and web applications called the Web Server Gateway Interface. Often abbreviated as WSGI and commonly pronounced as wiz-ghee. In 2010 PEP 3333 was introduced as a minor update to PEP 333 which targeted Python 3. PEP 3333 remains the current WSGI specification.
The WSGI specification was designed to be as simple as possible in order to encourage adoption. WSGI implementations are split into the server-gateway side and the application side. The server side of the implementation is responsible for accepting HTTP requests. For each request the server calls an application-provided callable.
Python's standard library includes a reference implementation of a WSGI server. The make_server callable is used to create a new WSGI compliant server which can run a WSGI application.
Instructions
1. Add the following code to the cloudacademy/playground.py file to import the WSGI reference implementation.
1 # A reference implementation of the WSGI specification.
2 # Not commonly used in production environments.
3 from wsgiref.simple_server import make_server
4
5
Back
Start check


Chrome File Edit View History Bookmarks Profiles Tab Window Help
Mon Jun 2 4:16 PM
LIConnections - VA and DoD
Python: Introduction to WSGI
Instructions - platform
about:blank
The application-side callable is required to include two positional parameters. The WSGI server is responsible for calling the callable for each HTTP request made to the server.
The first argument is a dictionary which contains WSGI environment variables. The second is a callable used to write the HTTP response status and headers. The application is required to return an iterable which will become the HTTP response body.
2. Append the following WSGI application to the cloudacademy/playground.py file.
# A basic WSGI application.
def app(environ, start_response):
# Begin by sending the HTTP(S) client the response stat
# Web applications support a wide variety of content ty
# Because the demonstrations in this lab are consumed u
start_response('200 OK', [('Content-Type', 'text/plain'
Yield a bytestring representing the response body.
The yield keyword turns this function intro a generat
making the function an iterable.
The b in front of the open quote makes this a bytestr
yield b'Hello, WSGI!\n'
A WSGI application is provided to a WSGI server during server creation. WSGI servers range in functionality, capabilities, and configuration options. The built-in reference implementation is relatively limited in functionality. However, production servers such as gunicorn and uwsgi are much more feature rich.
3. Append the following WSGI main code block to the cloudacademy/playground.py file.
if __name__ == '__main__':
1h 59m
Back
Start check

3. Append the following WSGI main code block to the **cloudacademy/playground.py** file.

```
if __name__ == '__main__':
    # Create a server and run the app until the process is t
    # The 1st argument to make_server is the hostname for wh
    # The 2nd argument to make_server is the network port us
    # The 3rd argument to make_server is the WSGI applicatio
    # Other arguments exist. See documentation for more deta
    server = make_server('', 5000, app)
    server.serve_forever()

```



4. Start the WSGI server by running the **cloudacademy/playground.py** file in the **IDEs terminal** pane: **(Terminal > New Terminal)**

```
python3 cloudacademy/playground.py
```

Back

Start check

Chrome File Edit View History Bookmarks Profiles Tab Window Help 100% Mon Jun 2 4:17 PM

LIConnections - VA and DoD | Python: Introduction to WSGI | Instructions - platform

about:blank

1h 59m

4. Start the WSGI server by running the **cloudacademy/playground.py** file in the **IDE terminal** pane: **(Terminal > New Terminal)**

```
python3 cloudacademy/playground.py
```

The WSGI server will block until it receives an interrupt signal. Testing the application requires another terminal pane.

5. Split the terminal pane by clicking the split button on the right hand side of the terminal.

Going forward the terminal running the WSGI server will be referred to as the **WSGI terminal**.

The newly created terminal will be used to send HTTP requests and receive responses using curl. Curl is a command line tool used for interacting with URLs.

Going forward the terminal running curl will be referred to as the **curl terminal**.

A WSGI server is ready to accept HTTP requests once started.

When the server is started it binds to a hostname and port. The call to

Back

Start check

A WSGI server is ready to accept HTTP requests once started.

When the server is started it binds to a hostname and port. The call to `make_server('', 5000, app)` listens for HTTP requests to localhost on port 5000. The hostname argument specifies an empty string which binds to the local host.

6. View the response in the **curl terminal**.

```
curl http://localhost:5000/
```

At their core WSGI applications are just Python callables with two positional parameters. WSGI applications are run by a WSGI server that calls a WSGI application for each HTTP request.

⭐ Proceed to the next step ⭐

Validations

On Track - Getting Started With WSGI Applications

Ensure a WSGI application callable named `app` exists in the **cloudacademy/playground.py** file.

Report an issue

Back

Start check