Web Application Development using Python

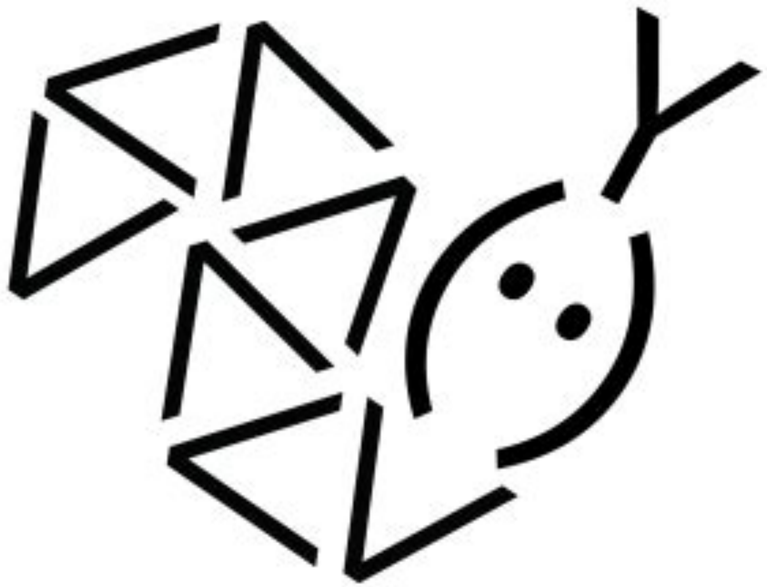
Introduction to Flask - Part 1

Prepared by George Khoury

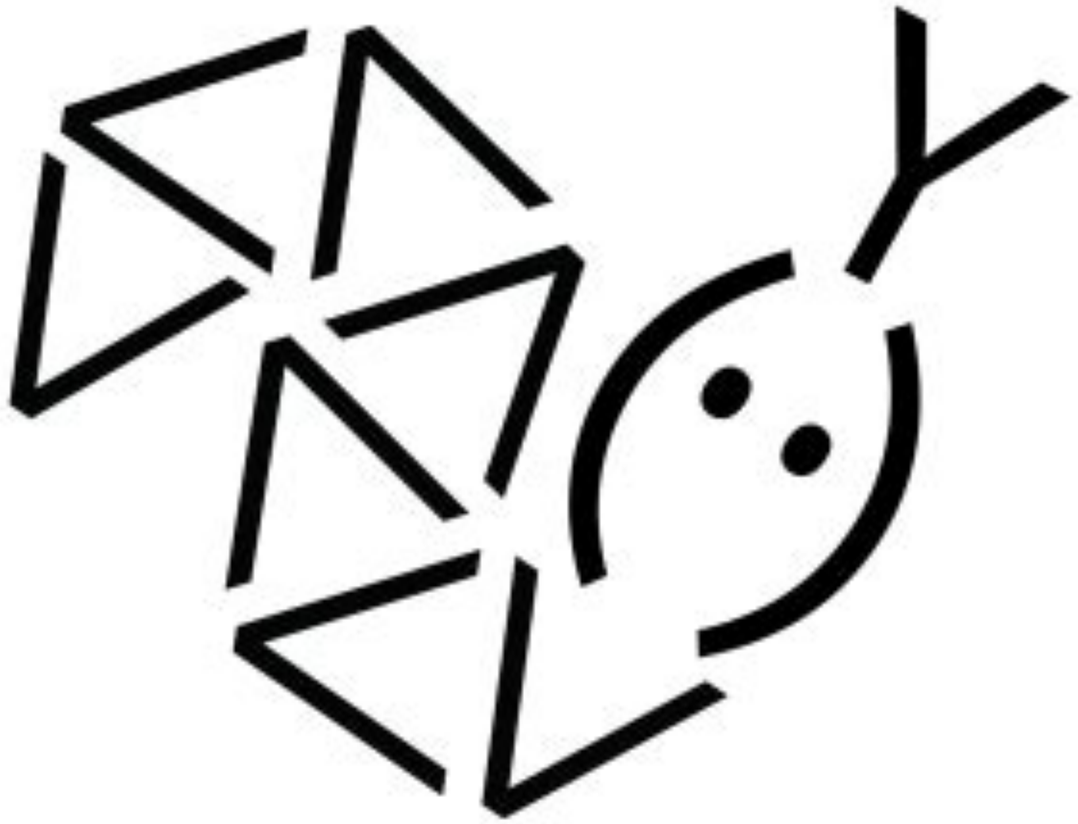
Outline

- Python virtual environments
- What are web applications?
- Project structure
- What is Flask?
  - Create a Flask app
  - Basic routing
  - `render_template()`

Python Virtual Environments

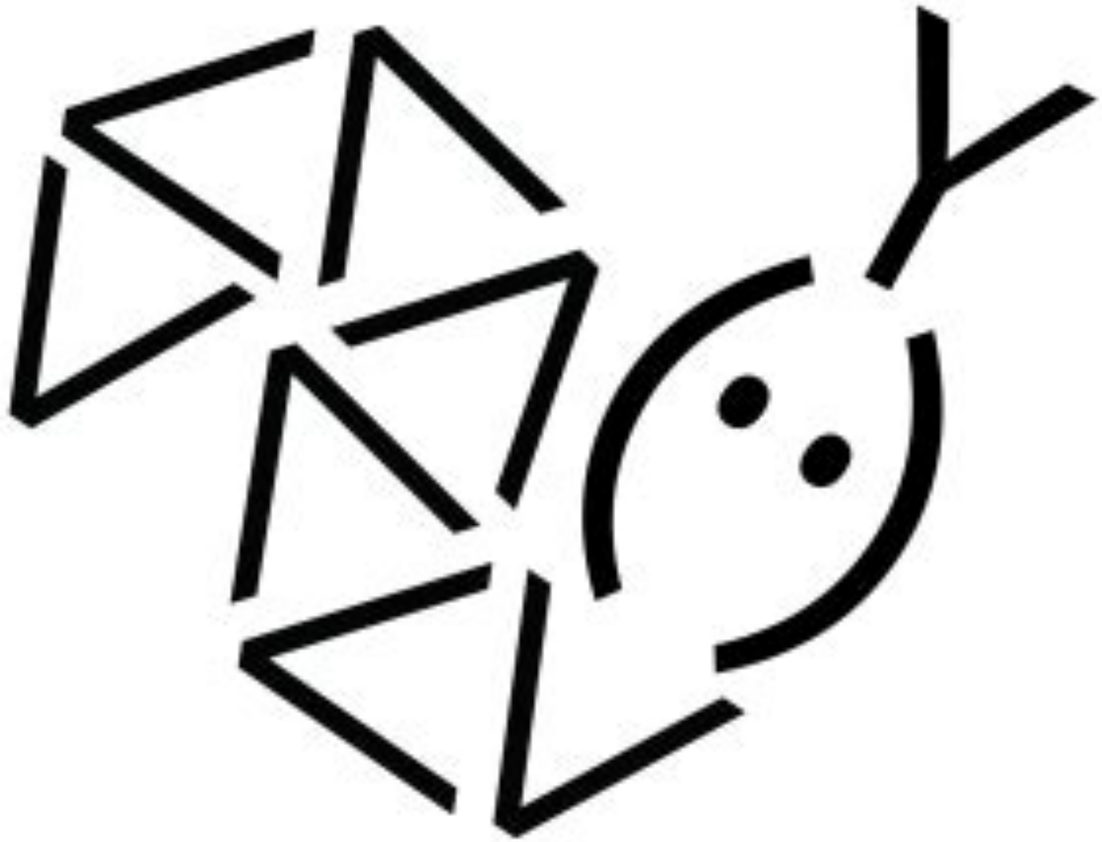


What is a virtual environment?



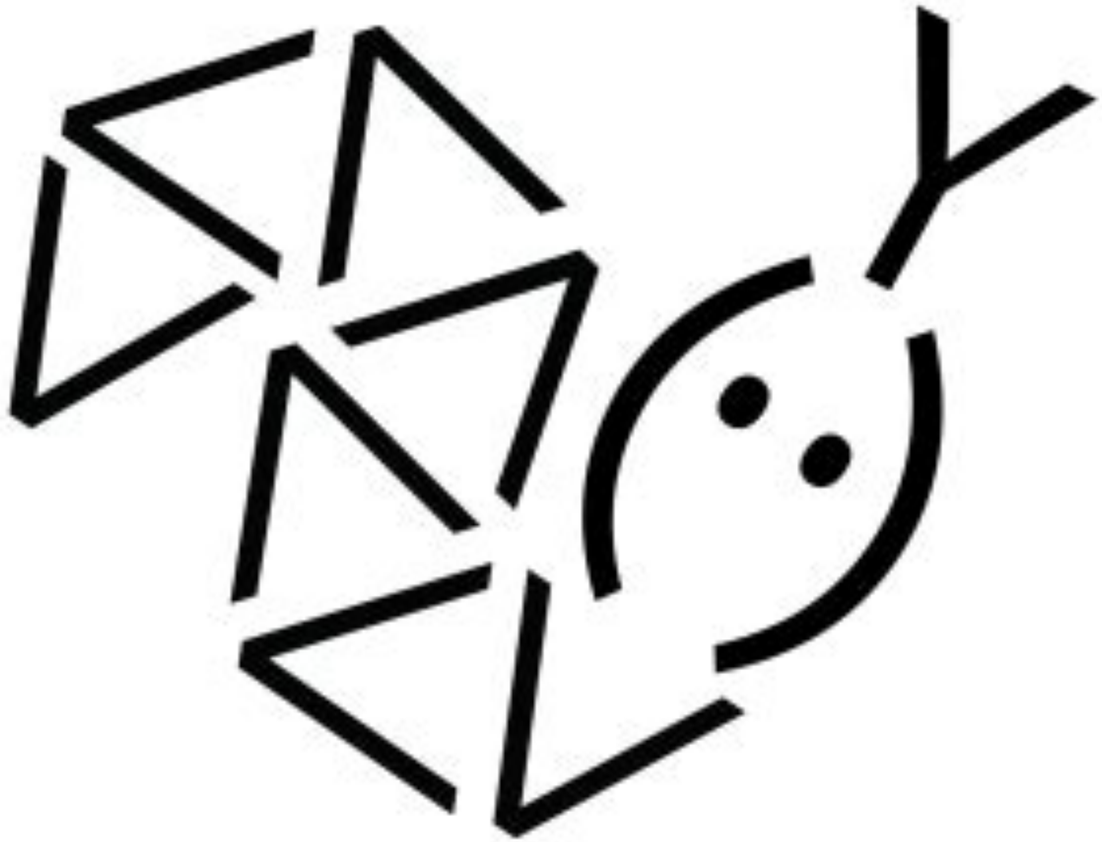
What problem does a virtual environment solve?

- Virtual environments are used to manage the dependencies for your project, both in **development** and in **production**
- Different versions of Python and the standard libraries may be required for different projects
- Virtual environments allow us to easily switch between to different environments without breaking compatibility for other projects



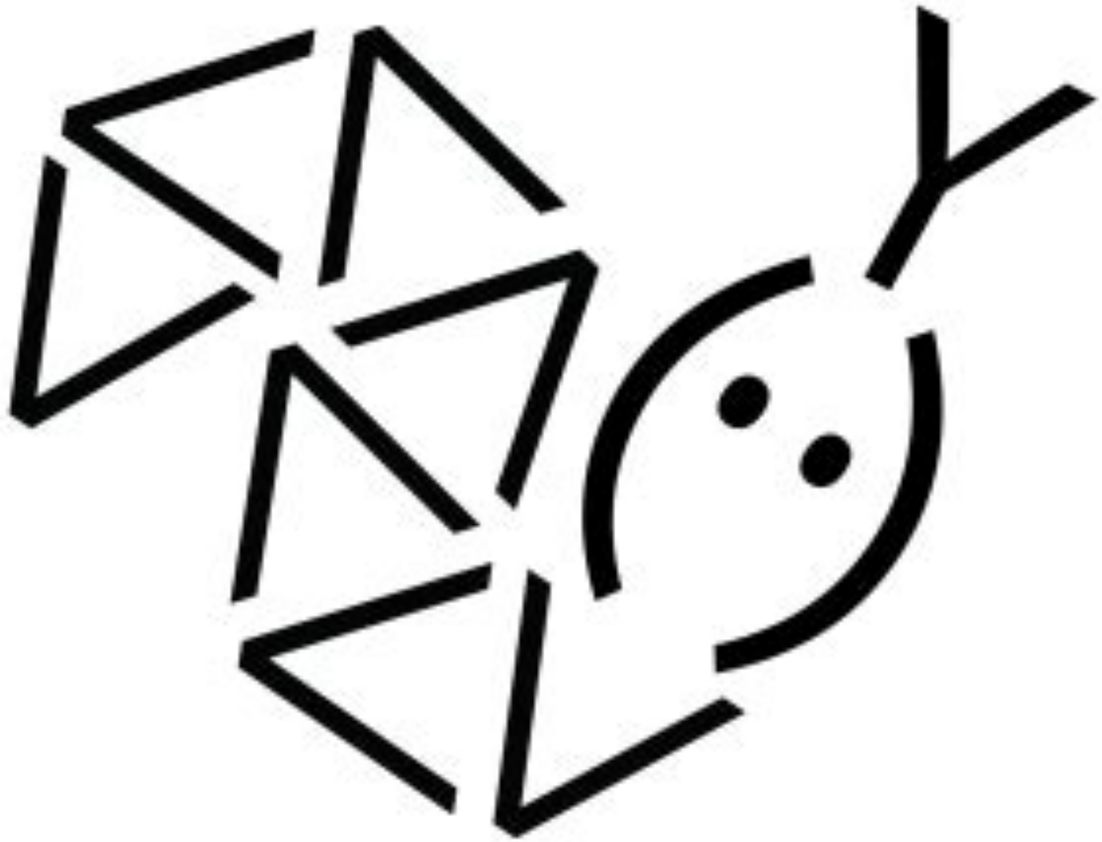
How to create a virtual environment?

- **Python 3 is bundled with the `venv` module**
  - `python -m venv myenv`
- A common directory location for a virtual environment is `.venv`
  - This name keeps the directory typically hidden in your shell
  - The name that explains why the directory exists

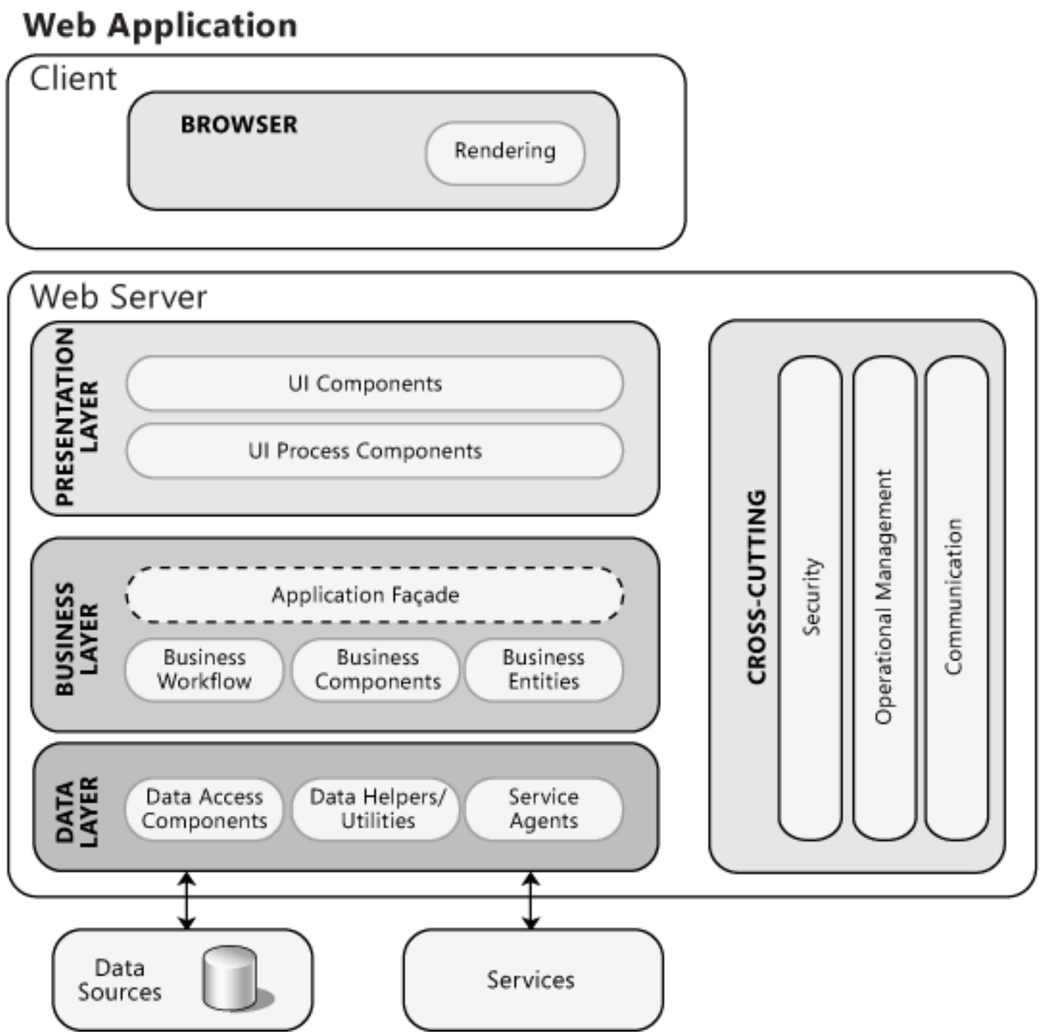


Using your environment

- We can activate our newly created environment by running:
  - `source .venv/bin/activate`
- Now that our environment is active, we can use pip to install packages.
  - `pip install flask`
- To deactivate the environment you can use:
  - `deactivate`

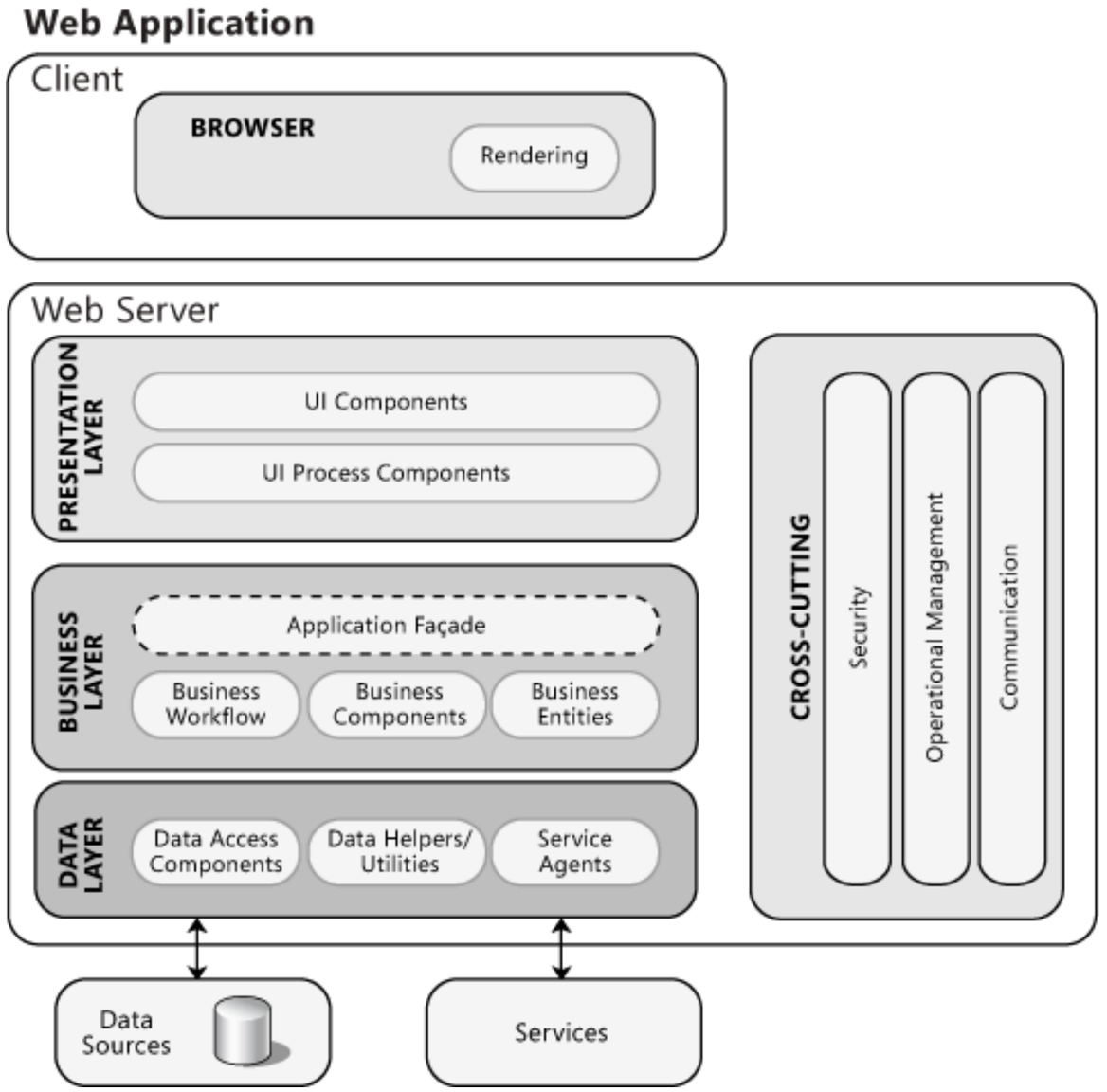


What are web applications?

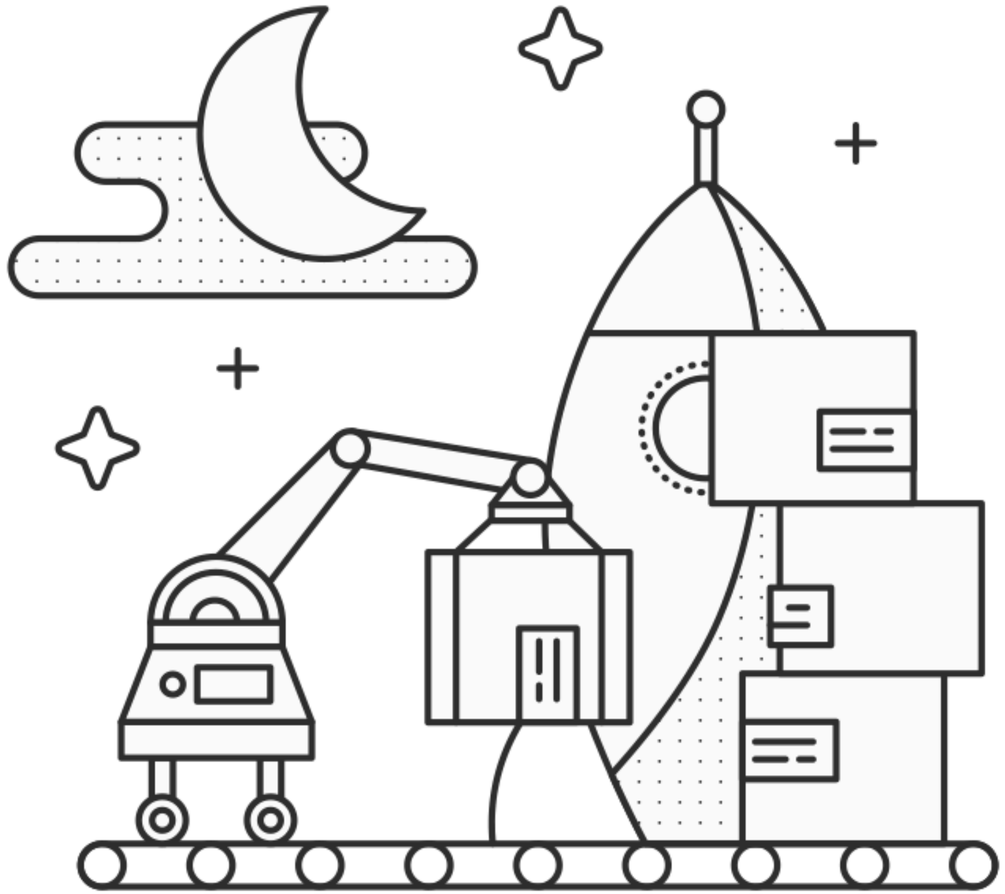


What are web applications?

- An application that runs completely on the user's browser is called a web application.
- An interface similar to client-server application is provided to the user in a web application and the user interacts with in same manner as the client-server application.
- A web application can provide the same functionality similar to client-server application.
- As these applications run on the browser so they can run on any platform or operating system having a web browser.
- For example, a word processor, like Google Docs can also be a web application that may allow the users to download data into their hard disk drives.
- GMail, Outlook, Office 365 are examples.

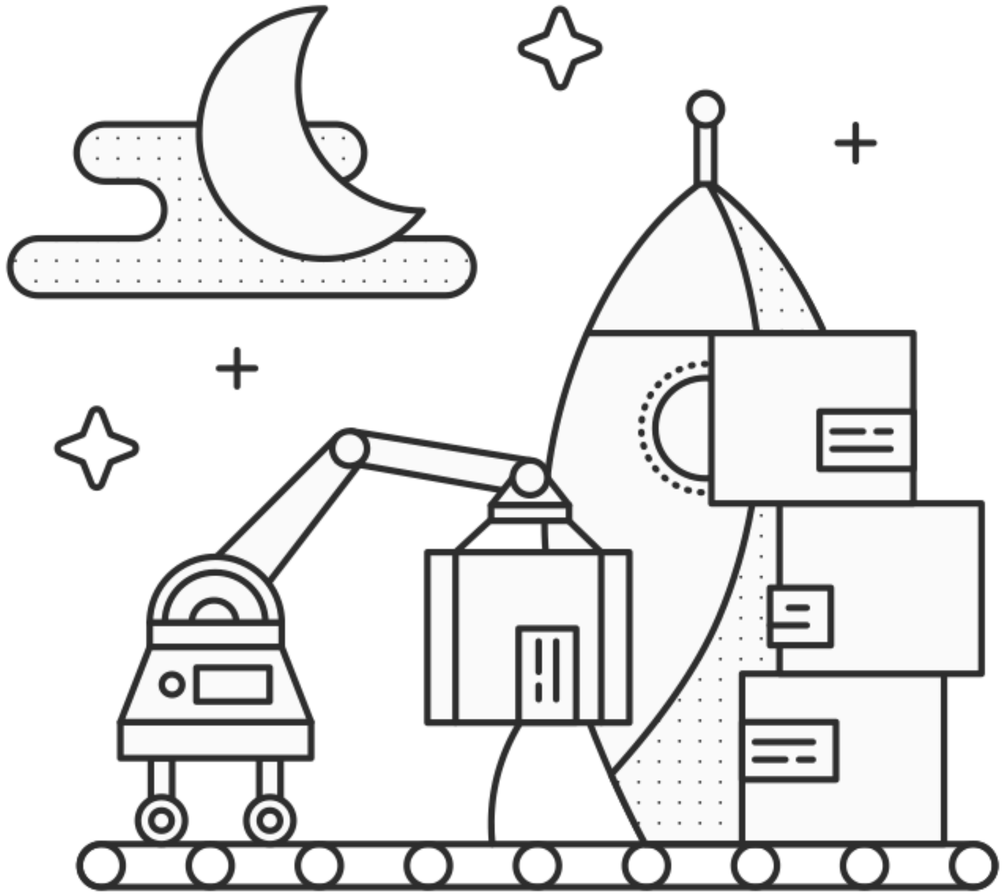


Project Structure



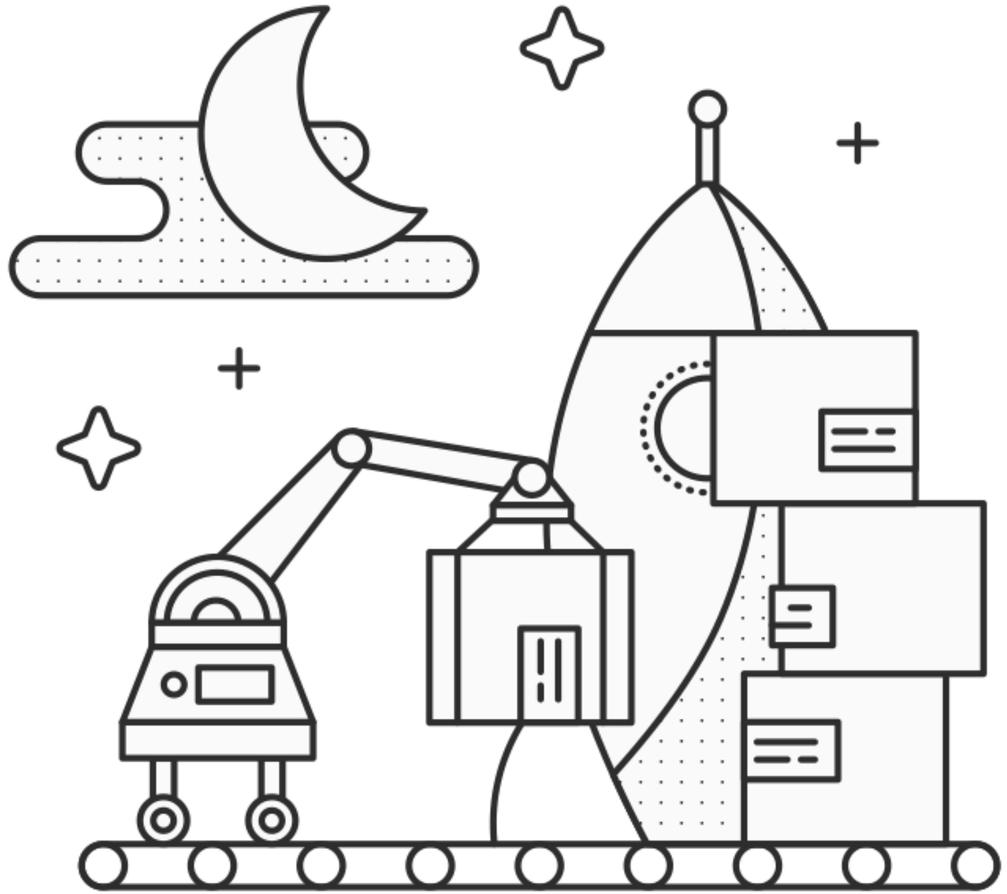
Organization patterns

- **Single module**
  - When you have a few routes
  - Few hundred lines of code
- **Package**
  - For more complex projects, we can factor out the different components of our app into a group of inter-connected modules — a package



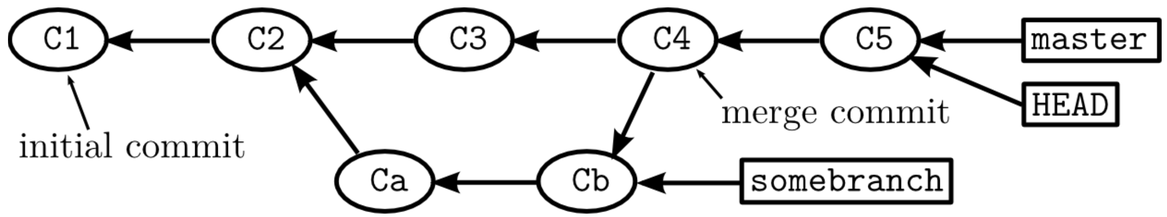
First project

- We will build a simple web application with some basic routes
- We will organise this application as a single module application
- **hello_flask/**
  - **`app.py` -** contains our flask app application
  - **`.venv`** - contains our virtual environment



Remember Git

- Remember that you can always use Git to keep track of your project.
- Keep your working directory and working tree clean.
- Write informative commit messages.
- Commit often. Remember to push to remote.



Flask Basics

Flask application

- After install Flask, you can use the flask class to create an instance of your application.
  - `from flask import Flask # import flask`
  - `app = Flask(__name__)   # create a new flask application`
- You can then start defining routes for your web application. Flask makes this easy by providing the decorator `@route('')`
- The Flask instance we created has a lot of methods we can use, we will explore this later on in this course.

Running your application

- Now we have an application to test out, we can use the `run()` method from Flask to run the development server with our application deployed.
- Or from the project root, we can run:
  - `export FLASK_APP=app.py`
  - `flask run`
- The development server binds to 127.0.0.1 on port 5000 by default.

Debug mode

- Enabling debug mode will reload the server on code changes.
- It also provides you with a debugger to further debug your code.
- To enable debug mode you can pass the **`debug=True`** parameter to the **`run()`** method.
  - `app.run(debug=True)`
- Or using the environment variable
  - `export FLASK_ENV=development`
  - `export FLASK_DEBUG=1`

Routing basics

- Use the `route()` decorator to bind a function to a URL.
- Use variable rules to receive variables from the URL
  - @app.route(‘/user/<username>')
  - @app.route(‘/post/<int:post_id>')
- You can define more than one route for the same function.

Resources

- **Virtual Environments**
  - https://docs.python.org/3/tutorial/venv.html
  - https://flask.palletsprojects.com/en/1.1.x/installation/#virtual-environments
  - https://exploreflask.com/en/latest/environment.html
- **Flask**
  - https://flask.palletsprojects.com/en/1.1.x/tutorial/#tutorial
  - https://flask.palletsprojects.com/en/1.1.x/quickstart/#routing
  - https://flask.palletsprojects.com/en/1.1.x/quickstart/#variable-rules
  - https://exploreflask.com/en/latest/conventions.html
  - https://exploreflask.com/en/latest/organizing.html

Resources

- **Jinja2**
  - https://jinja.palletsprojects.com/en/2.11.x/
  - https://jinja.palletsprojects.com/en/2.11.x/templates/#variables
  - https://jinja.palletsprojects.com/en/2.11.x/templates/#filters
  - https://jinja.palletsprojects.com/en/2.11.x/templates/#list-of-control-structures
  - https://jinja.palletsprojects.com/en/2.11.x/templates/#template-inheritance