Web Application Development using Python

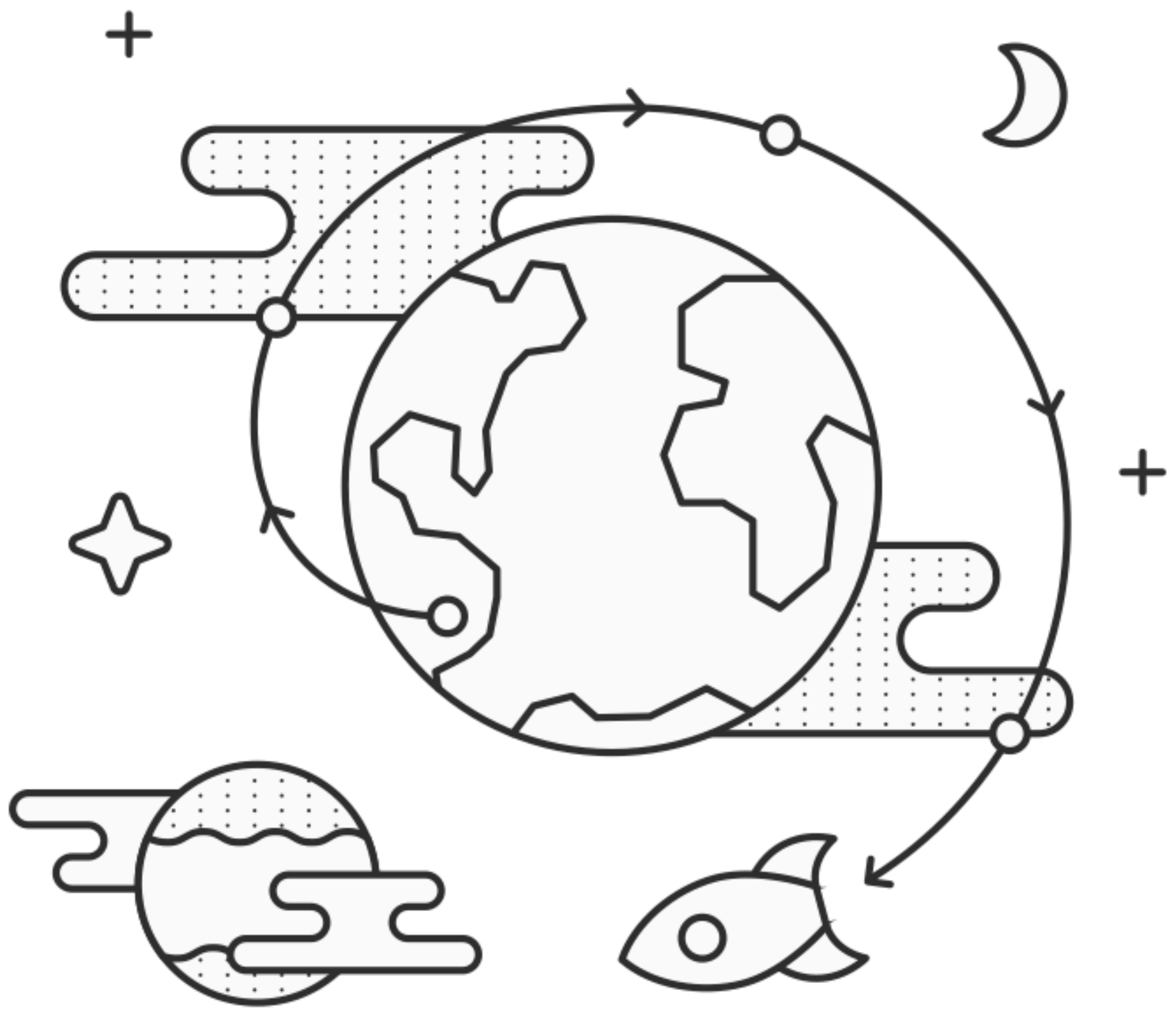
Flask - Part 1

Prepared by George Khoury

Outline

- **Routing**
  - Variable Rules
  - Unique URLs / Redirection Behavior
- **Templates**
  - Template Engine
  - Rending Templates
  - How to organize templates?
  - Jinja2

Routing



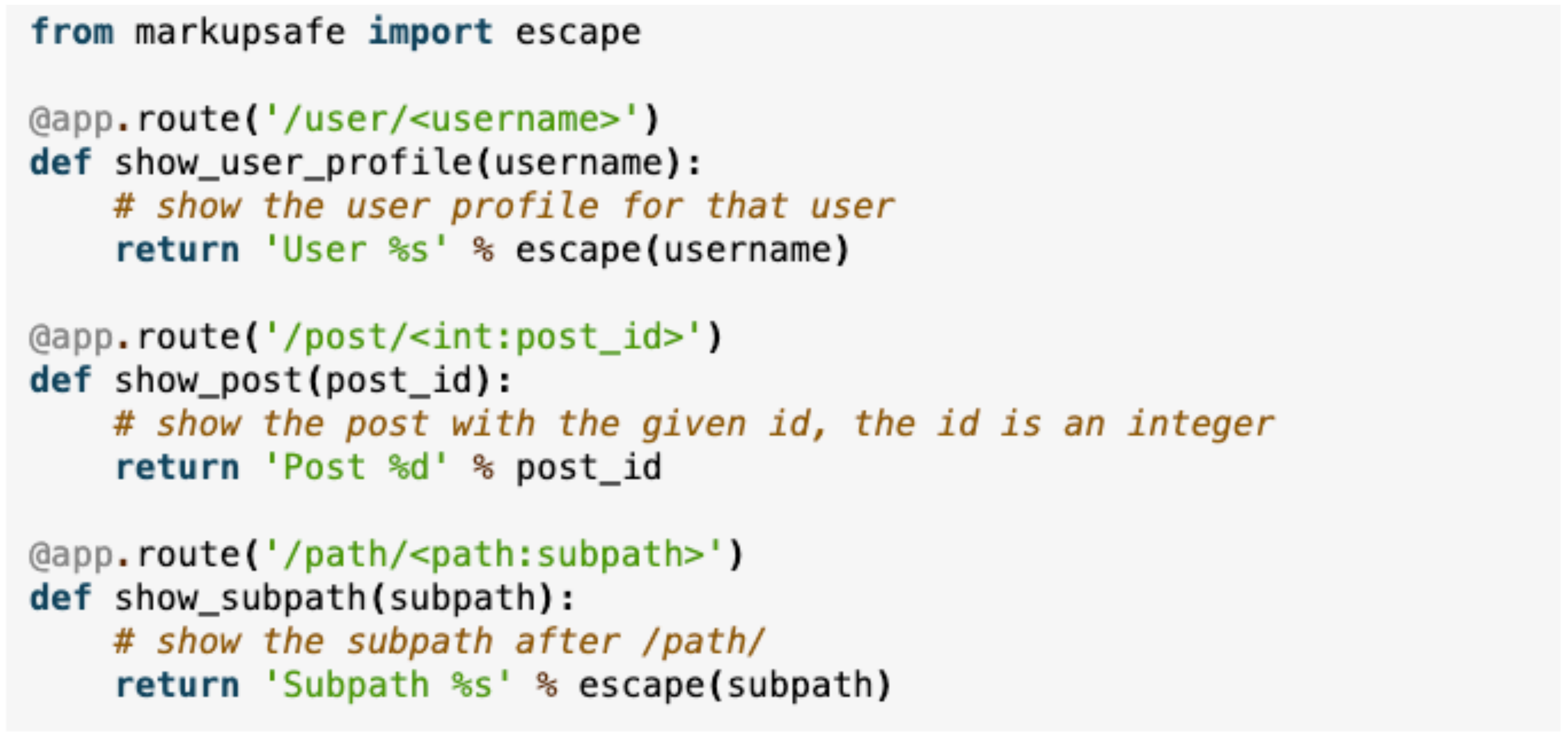
Routing
Basics

- Modern web applications use meaningful URLs to help users.
- Users are more likely to like a page and come back if the page uses a meaningful URL they can remember and use to directly visit a page.
- Use the `route()` decorator to bind a function to a URL.
- You can do more! You can make parts of the URL dynamic and attach multiple rules to a function.

Routing

Variable Rules

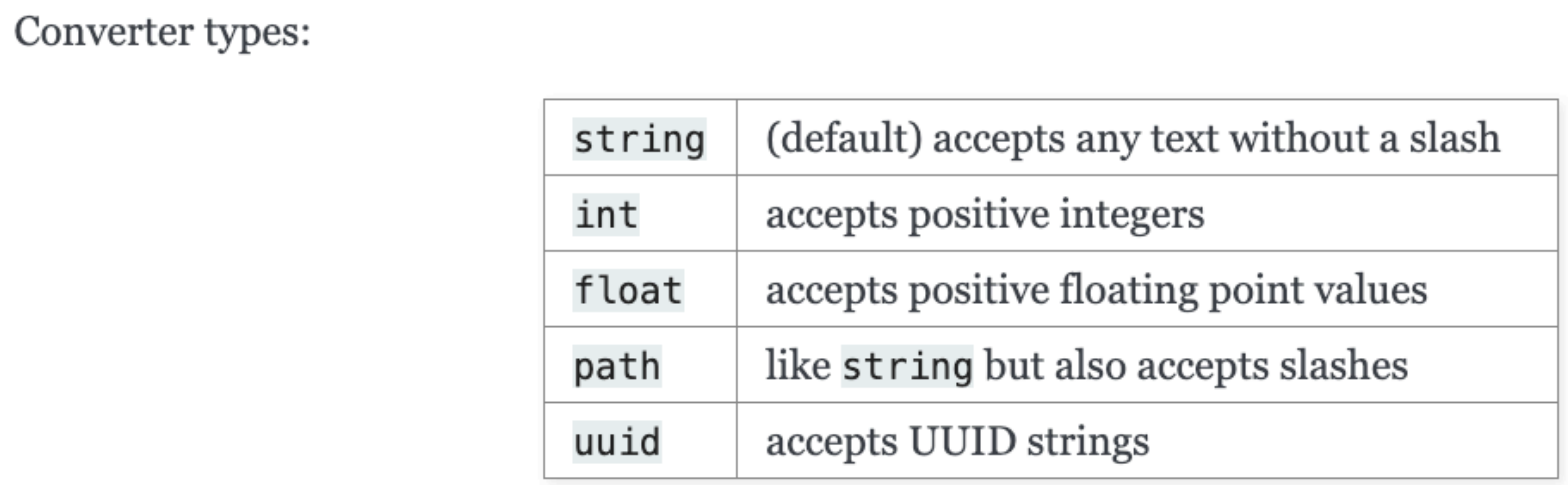
- You can add variable sections to a URL by marking sections with `<variable_name>`.
- Your function then receives the `<variable_name>` as a keyword argument.
- Optionally, you can use a converter to specify the type of the argument like `<converter:variable_name>`.



```python
from markupsafe import escape

@app.route('/user/<username>')
def show_user_profile(username):
    # show the user profile for that user
    return 'User %s' % escape(username)

@app.route('/post/<int:post_id>')
def show_post(post_id):
    # show the post with the given id, the id is an integer
    return 'Post %d' % post_id

@app.route('/path/<path:subpath>')
def show_subpath(subpath):
    # show the subpath after /path/
    return 'Subpath %s' % escape(subpath)
```

Routing

Converter Types



Converter types:

`string`	(default) accepts any text without a slash
`int`	accepts positive integers
`float`	accepts positive floating point values
`path`	like `string` but also accepts slashes
`uuid`	accepts UUID strings

Routing

Unique URLs / Redirection Behavior

- The canonical URL for the **projects** endpoint **has a trailing slash**.
  - It’s similar to a folder in a file system. If you access the URL without a trailing slash, Flask redirects you to the canonical URL with the trailing slash.
- The canonical URL for the **about** endpoint **does not have a trailing slash**.
  - It’s similar to the pathname of a file.
  - Accessing the URL with a trailing slash produces a 404 “Not Found” error.
  - This helps keep URLs unique for these resources, which helps search engines avoid indexing the same page twice.

```python
@app.route('/projects/')
def projects():
    return 'The project page'

@app.route('/about')
def about():
    return 'The about page'
```

Routing

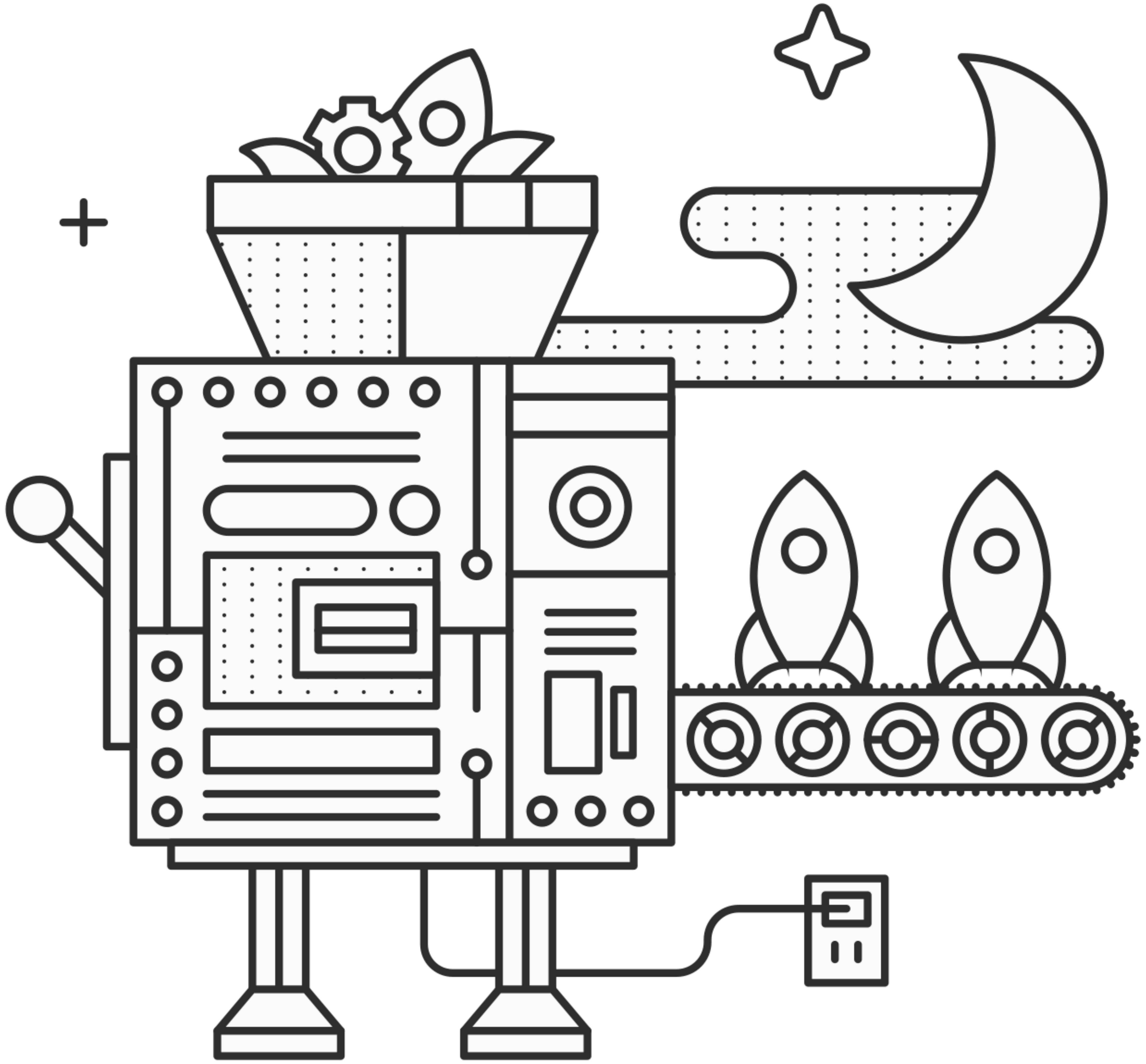
URL Building

- To build a URL to a specific function, use the `url_for()` function.
- It accepts the name of the **function** as its first argument and any number of keyword arguments, each corresponding to a variable part of the URL rule.
- Unknown variable parts are appended to the URL as query parameters.

```python
from flask import Flask, url_for
from markupsafe import escape

app = Flask(__name__)

@app.route('/')
def index():
    return 'index'

@app.route('/login')
def login():
    return 'login'

@app.route('/user/<username>')
def profile(username):
    return '{}\'s profile'.format(escape(username))

with app.test_request_context():
    print(url_for('index'))
    print(url_for('login'))
    print(url_for('login', next='/'))
    print(url_for('profile', username='John Doe'))
```

```
/
/login
/login?next=/
/user/John%20Doe
```

Templates



Templates

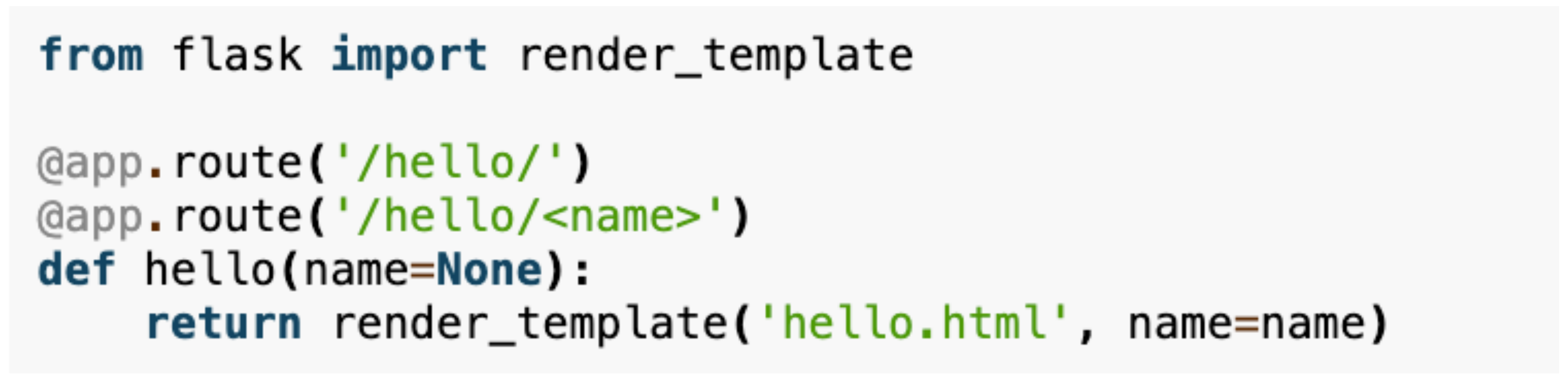
Template Engine

- Generating HTML from within Python is not fun, and actually pretty cumbersome because you have to do the HTML escaping on your own to keep the application secure.
- Because of that Flask configures the Jinja2 template engine for you automatically.
- While Flask doesn’t force us to use any particular templating language, it assumes that we’re going to use Jinja.
  - **Most of the developers in the Flask community use Jinja, and I recommend that you do the same.**

Templates

Rendering Templates

- To render a template you can use the `render_template()` method.
- All you have to do is provide the name of the template and the variables you want to pass to the template engine as keyword arguments.
- Here's a simple example of how to render a template.

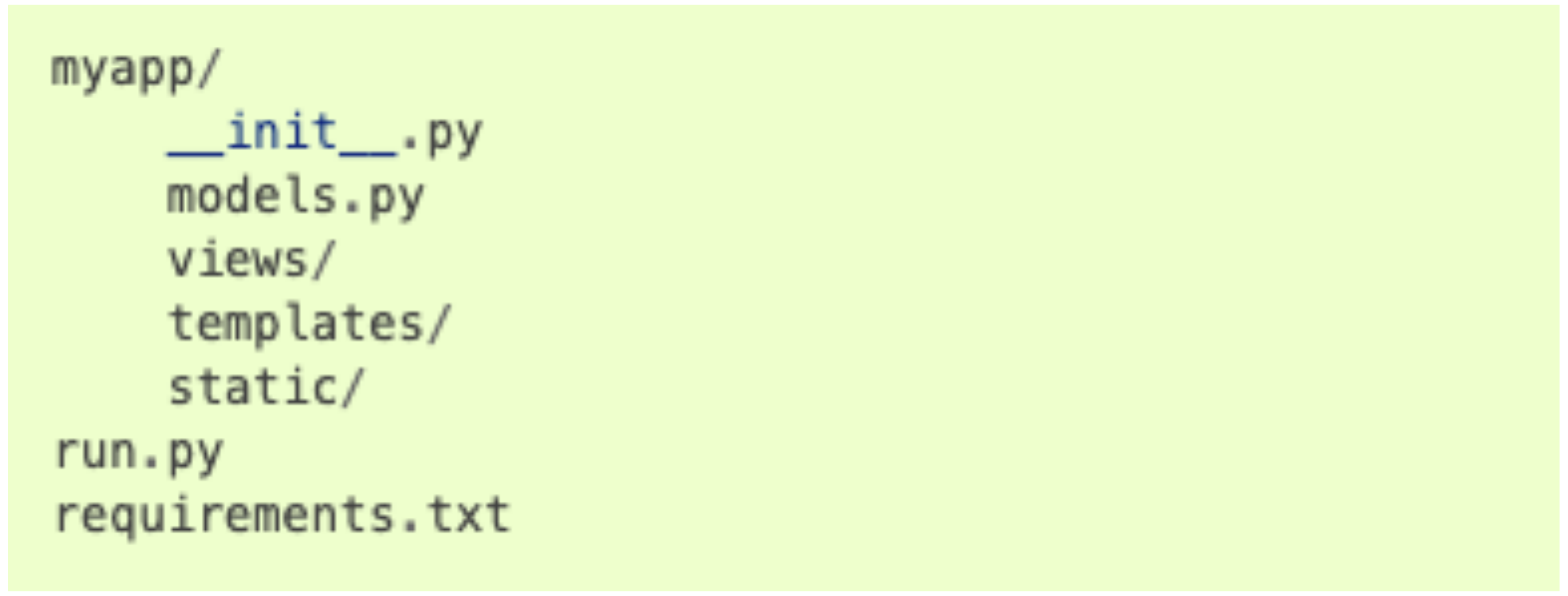


```
from flask import render_template

@app.route('/hello/')
@app.route('/hello/<name>')
def hello(name=None):
    return render_template('hello.html', name=name)
```

Templates

How to organize templates?

- Flask will look for `templates` in the templates folder.
- The structure of the `templates` directory parallels the structure of our routes.



```
myapp/
    __init__.py
    models.py
    views/
    templates/
    static/
run.py
requirements.txt
```

Templates

Jinja2

- The Jinja documentation does a great job of explaining the syntax and features of the language.
- I won’t reiterate it all here, always refer to the Jinja Template Designer Documentation.

A Quick Note

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

Learning Resources

- https://flask.palletsprojects.com/en/1.1.x/quickstart/#routing
- https://flask.palletsprojects.com/en/1.1.x/quickstart/#variable-rules
- https://flask.palletsprojects.com/en/1.1.x/quickstart/#unique-urls-redirection-behavior
- https://flask.palletsprojects.com/en/1.1.x/quickstart/#url-building
- https://flask.palletsprojects.com/en/1.1.x/quickstart/#rendering-templates
- https://jinja.palletsprojects.com/en/2.11.x/templates/#synopsis