MAKE SCHOOL

Packages & Blueprints

BEW 1.2

Agenda

MAKE SCHOOL

- Learning Outcomes
- Python Modules & Packages
- Activity: Describe Package Structure
- **BREAK**
- How to Start a New Project
- Activity: Start your Final Project
- Wrap-Up

Learning Outcomes

By the end of today, you should be able to...

1. **Explain** how modules & packages are used to organize Python code.
2. **Identify** and prevent circular dependencies in Python modules.
3. **Explain** why blueprints are useful for separating concerns in a project.
4. **Use** packages & blueprints to start a new project.

Warm-Up (5 minutes)

In a group of 3, answer:

- What's one thing causing you stress as we approach finals week?
- What is one thing you're looking forward to over the break?

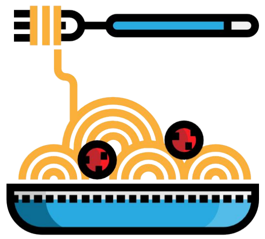
Python Modules & Packages

What is Modular Programming?

Modular programming is a design technique to break your code into separate parts, with minimal dependencies on each other.

You will see modular programming used in lots of different contexts. It minimizes bugs and makes it easier to find the source of errors.

On the opposite extreme of modular programming is "spaghetti code", where many parts all reference each other and it becomes more difficult to locate errors.



Modules vs. Packages

A module is a single file (or files) that is imported under a single import statement.

```
from my_module import my_function
```

A package is a collection of modules in directories that give a file hierarchy.

```
from my_package.foo.bar import my_function
```

We use packages in our projects to make it easier to organize the code.

__init__.py

`__init__.py` is a special file that declares a Python package. It used to be required to declare a package, but as of Python 3.3 it is no longer required.

However, it is still useful! For example, the **`unittest`** library will not search inside your folders unless they include `__init__.py`.

We can also declare variables & run setup code inside of `__init__.py`.

__init__.py

For example, we can use `__init__.py` to declare the `app` & `db` variables

(source):

```
# __init__.py
# (imports omitted for brevity)
app = Flask(__name__)
app.config.from_object(Config)

db = SQLAlchemy(app)

from books_app.routes import main

app.register_blueprint(main)

with app.app_context():
   db.create_all()
```

Circular Imports

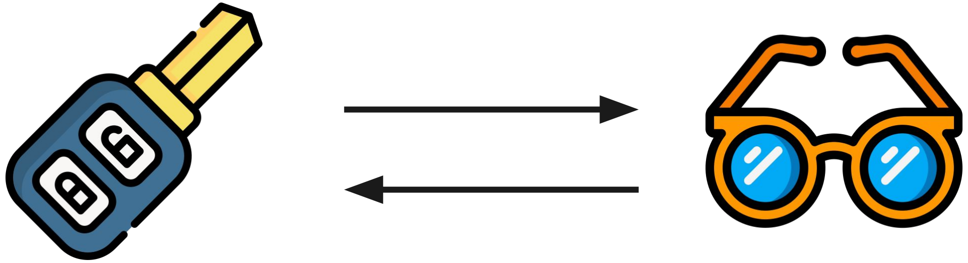
Ok, so... Why can't we put `from books_app.routes import main` at the top of the file?

That's because it would cause a **circular import**.

Circular Imports

A **circular import** occurs when two files are both importing each other, and both depend on each other to be initialized.

Think of it like a catch-22: *"I can't find my car keys without my glasses, but I left my glasses in the car so I can't get them without my car keys."*



Circular Imports

```python
# books_app/__init__.py
from flask import Flask
from flask_sqlalchemy import SQLAlchemy
from books_app.config import Config
import os
from books_app.routes import main

app = Flask(__name__)
app.config.from_object(Config)

db = SQLAlchemy(app)


app.register_blueprint(main)

with app.app_context():
   db.create_all()
```

```python
# books_app/routes.py
from flask import Blueprint
from books_app.models import Book,
Author, Genre


from books_app import app, db


main = Blueprint("main", __name__)
# ... routes ...
```

❌ app, db not yet initialized

Circular Imports

```python
# books_app/__init__.py
from flask import Flask
from flask_sqlalchemy import SQLAlchemy
from books_app.config import Config
import os

app = Flask(__name__)
app.config.from_object(Config)

db = SQLAlchemy(app)

from books_app.routes import main

app.register_blueprint(main)

with app.app_context():
   db.create_all()
```

```python
# books_app/routes.py
from flask import Blueprint
from books_app.models import Book,
Author, Genre


from books_app import app, db


main = Blueprint("main", __name__)
# ... routes ...
```

✅ app, db are initialized

Blueprints

Flask Blueprints

Flask Blueprints are another way to use modular programming to organize our code. They allow us to separate out Flask routes into multiple files.

So far, our applications have been pretty small. But for a larger website, you could have hundreds of routes. It makes sense to organize them into categories based on their purpose.

For example, we may have `main` & `auth` blueprints to separate out authentication routes. If your site has an API, you may want to have an `api` blueprint.

Flask Blueprints

We can **declare** a blueprint in the routes file:

```
main = Blueprint("main", __name__)

@main.route('/')
def homepage():
    # ...
```

Then we need to register the blueprint in the `__init__.py` file:

```
app.register_blueprint(main)
```

Flask Blueprints

Two blueprints can each have a route with the same name, and there won't be a naming conflict:

```
main = Blueprint("main", __name__)

@main.route('/')
def index():
    # ...
```

```
api = Blueprint("api", __name__)

@api.route('/api')
def index():
    # ...
```

```
url_for('main.index')
   -> '/'
```

```
url_for('api.index')
   -> '/api'
```

Activity (15 minutes)

With a partner, figure out who has most recently traveled to another state. That person will be the **interviewer**.

The **interviewee** should choose a homework assignment, share their screen, and explain the structure of the code. Make sure to mention the blueprint(s), setup code, & how to avoid a circular dependency.

After 5 minutes, switch partners.

MAKE SCHOOL

Break - 10 min

MAKE SCHOOL

How to Create a Project from Scratch

Creating a Project

A Flask project will typically have a file structure similar to the following (although it may look different):

```
project_folder/
    app_folder/
        static/
        templates/
        section_folder/
            forms.py
            routes.py
            tests.py
            __init__.py
        models.py
        config.py
        __init__.py
    app.py
    .env
    requirements.txt
    README.md
```

Create a Project

Check out the project starter code if you'd like help with starting your project.

Keep in mind that it's also OK to copy setup code from an existing project! Just provide attribution if you copy someone else's work.

MAKE SCHOOL

Wrap-Up