Building and Deploying Course Websites with Python, Quarto, and Mkdocs

Alish Chhetri and Gregory M. Kapfhammer

May 15, 2025

What Problem Are We Solving?

- Course sites as **static archives** rather than **living resources**
- Students view materials as **products to consume** rather than **projects to contribute to**
- Interactive examples require **separate tools** or external **platforms**
- Content becomes **quickly outdated** without sustainable update processes

> 💡 **Ultimate Goal**: Introduce how using packages like Quarto make it possible to create **interactive** and **maintainable** course websites

Building and Deploying with Quarto

- **Quarto**: Accessible markdown-based framework
- **Python & WebAssembly**: Easily run Python in the browser
- **GitHub**: Version control and collaborative workflows
- **GitHub Actions**: Automated testing and deployment

- 💡 **Ultimate Goal**: Build content-focused, interactive, collaborative course sites that instructors can maintain with minimal effort.
- **Next Steps**: Give examples of content you can immediately use! See Algorithmology.org for more details!

Interactive Fibonacci Sequence

Python Code ⟳ Start Over ▷ Run Code

```python
def fibonacci(n):
    """Calculate the Fibonacci sequence up to n terms."""
    fib_sequence = [0, 1]

    if n <= 0:
        return []
    if n == 1:
        return [0]

    # Generate Fibonacci sequence
```

```
First 10 Fibonacci numbers: [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
```

- 💡 **Task**: Try modifying the code to calculate more terms or implement a different algorithm!

Setting Up the Course Infrastructure

1. Create a GitHub repository with Quarto structure
2. Configure navigation, styling, and organization in `_quarto.yml`
3. Set up GitHub Actions for automated deployment
4. Create templates for student contributions
5. Publish to GitHub Pages or Netlify

Course Roles & Contributions

Instructor’s Role

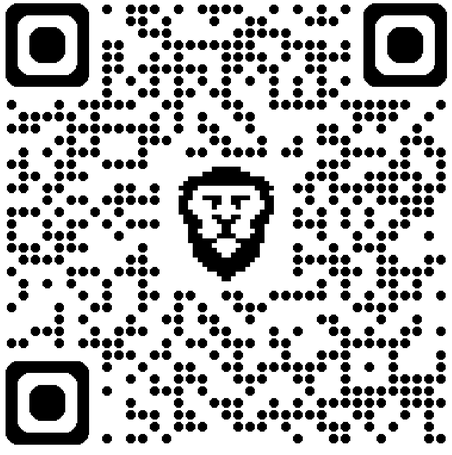
- Create course content structure
- Establish contribution guidelines
- Review and provide feedback
- Guide collaborative learning
- Maintain core infrastructure

Student’s Role

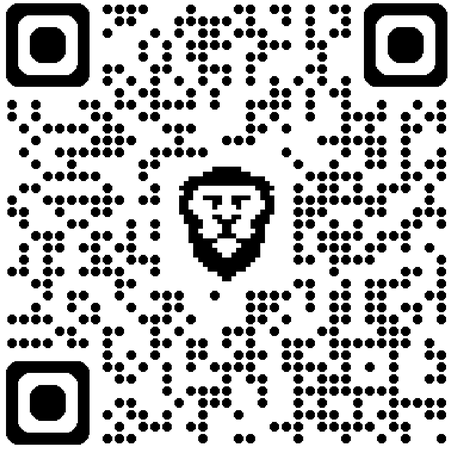
- Fork repository for contributions
- Create content in assigned areas
- Submit pull requests for review
- Engage in peer feedback
- Create technical documentation

Real World Examples

Course Website



GitHub Repository



Future Improvements

- **Browser-Based IDE Experience**: Complete development environment in the browser
- **Enhanced WebAssembly**: Better error messages and debugging for Python in browser
- **Collaborative Features**: Note taking and shared whiteboarding tools
- **Automated Feedback**: AI-powered assistance for student contributions
- **Enhanced Visualizations**: Interactive data and algorithm visualizations

Why You Should Consider This Approach

Educational Benefits

- **Student Ownership & Investment**: Students become stakeholders in course content
- **Living Educational Resources**: Materials continuously evolve and improve
- **Industry-Standard Practices**: Students gain experience with professional tools
- **Transparent Learning Process**: Public materials encourage higher quality work
- **Adaptable for All Class Sizes**: Effective from seminars to large lectures

Technical Benefits

- **Simplified Content Creation**: Markdown reduces technical barriers
- **Focus on Educational Value**: Infrastructure complexity is abstracted away
- **Browser-Based Interactivity**: Code execution without local installations
- **Comprehensive Searchability**: Find content across all course materials
- **Collaborative Development**: Modern version control and workflows