Authoring EVITA Training Modules

Pedagogical Design and Technical Implementation

Ashwin Mohanan (RISE), EVITA Project

10 December 2025

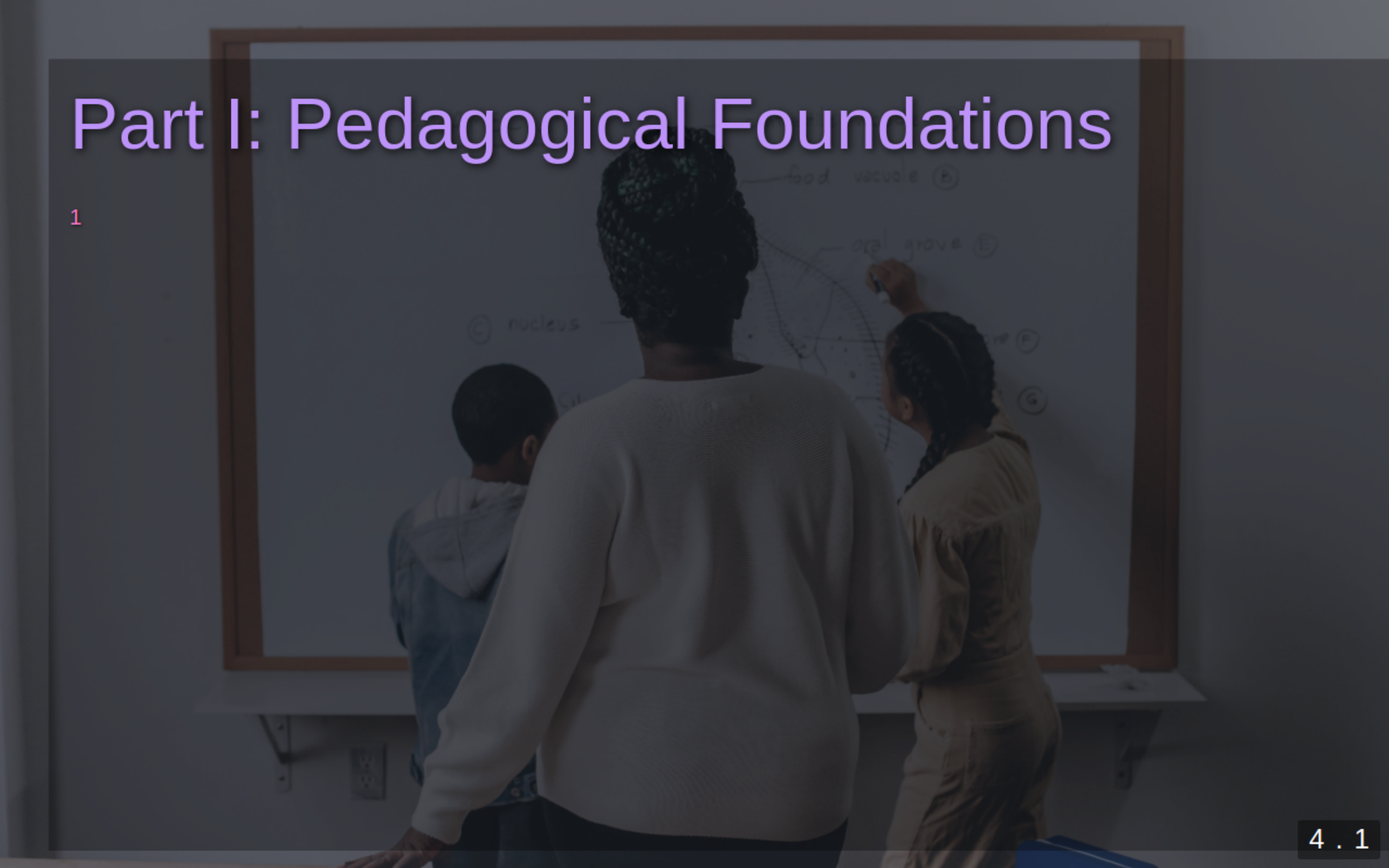
Overview: Creating Effective Training Modules for EVITA

This primer covers:

- *Pedagogical foundations for effective learning*
  - Backwards design from skills trees and objectives
  - Formative assessment techniques
  - Managing cognitive load
- *Technical toolchain overview*
  - Practical implementation with MyST and sphinx-evita

Part I: Pedagogical Foundations

1



Why Pedagogy Matters

> *” Remember that no lesson survives first contact with learners…*
>
> *…that every lesson is too short for the teacher and too long for the learner…*
>
> *…and that nobody will be more excited about the lesson than you are.” — Teaching Tech Together*

Key insights:

Backwards design from the skills and learning outcome leads to focussed content

Design for learners’ *actual mental models*, not assumed knowledge

Combination of *formative assessment and summative assessements*

Cognitive load management prevents overwhelm, and leads to better learning outcomes

Learning science principles guide effective content creation

Evidence-based approach:

2

Backwards Design

3

Start with the End in Mind

The Three Stages

Identify desired results — What should learners be able to *do*?

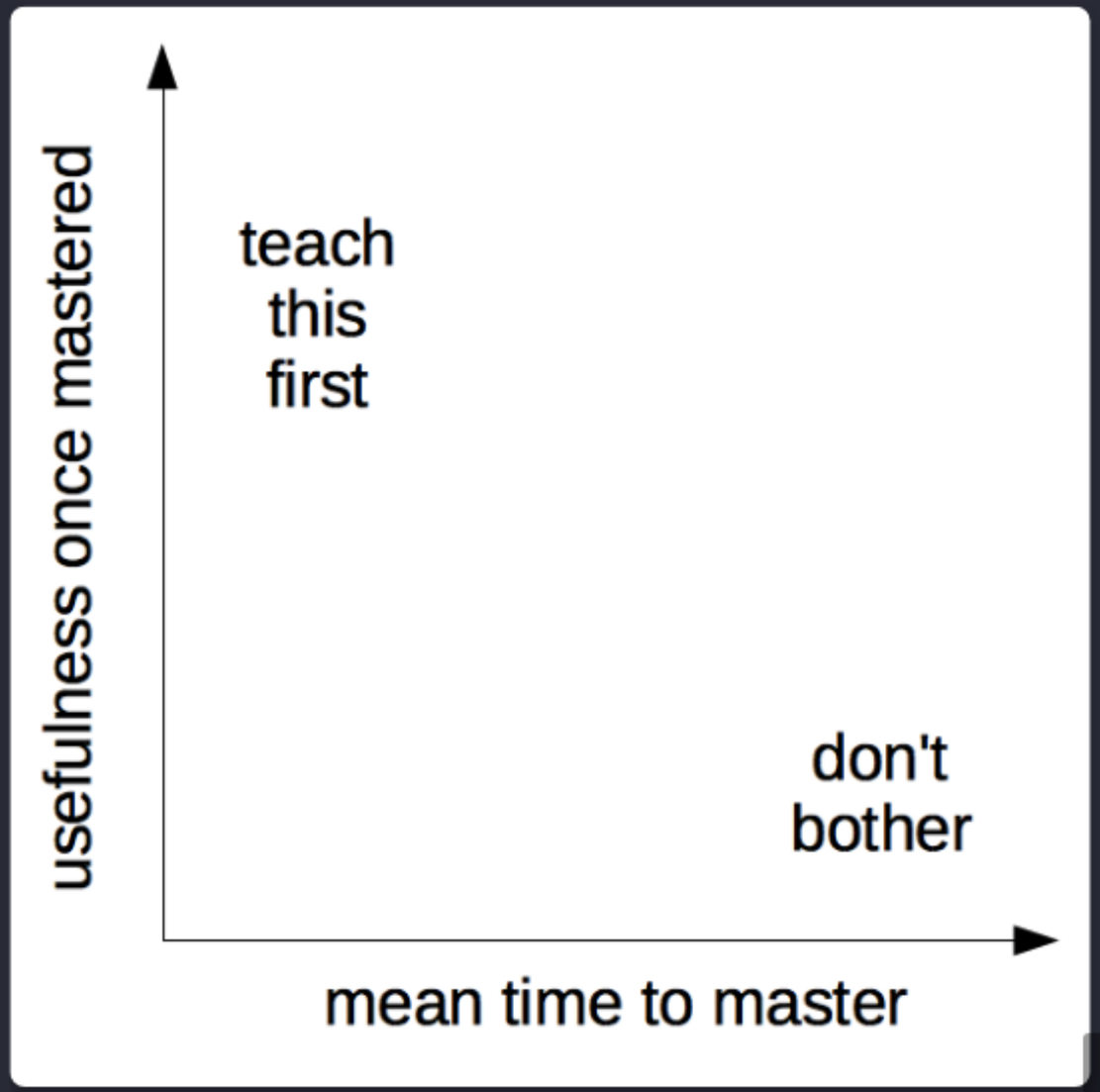
Determine acceptable evidence — How will you *know* they learned?

Plan learning experiences — What activities will help them get there?

Never start with content! Start with skills and outcomes.

Deciding what to teach: visual guide

Source:



EVITA Skills Tree Approach

Skills tree = Visual representation of all skills

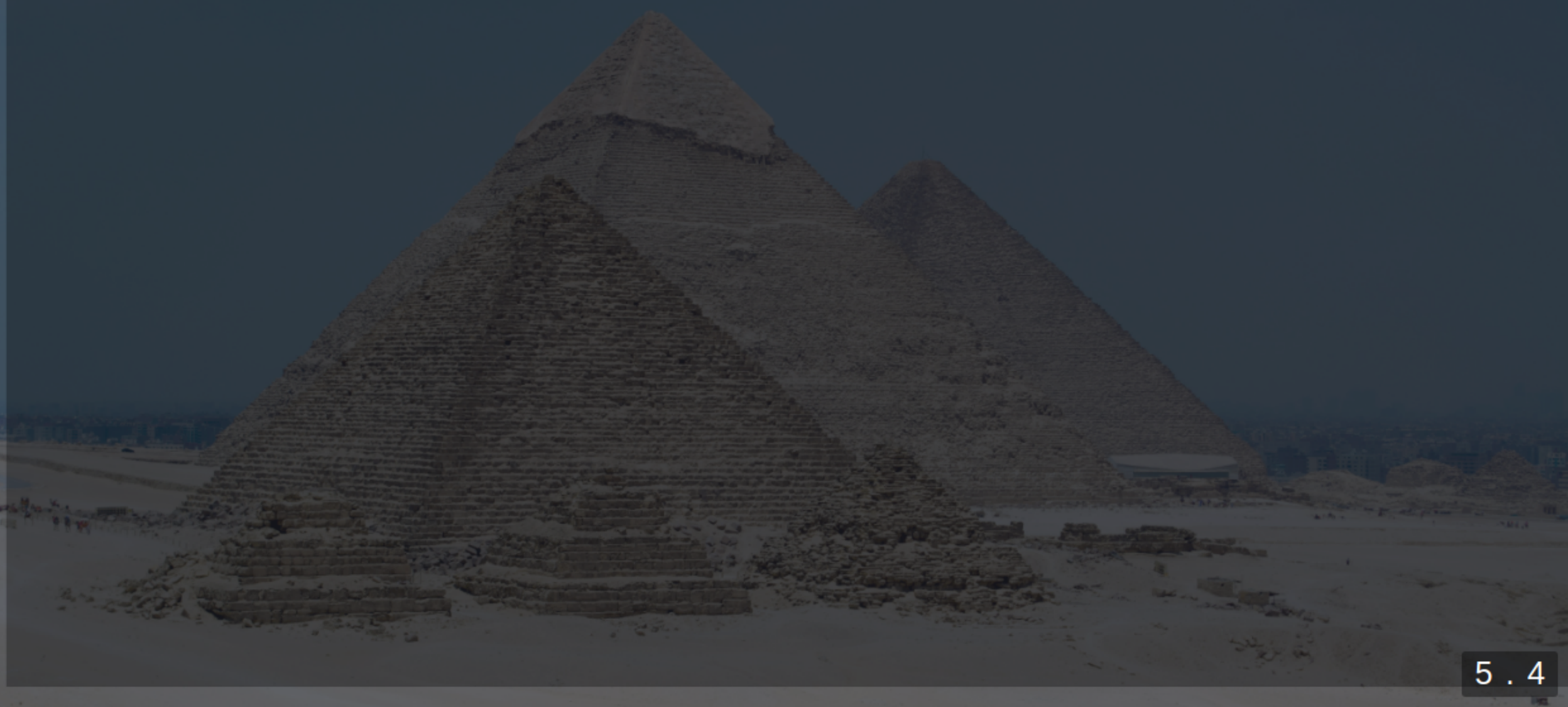
```
Foundation Taxonomy (HPC Certification Forum)
    ├── Node Skills (coarse grained)
    │   ├── Leaf Skill A
    │   └── Leaf Skill B
    └── Node Skills (coarse grained)
        └── Leaf Skill C
```

Benefits:

- Learning pathways:
- Helps learners navigate their path
- Prevents cognitive overload by chunking
- Links to HPC Certification Forum skill tree

Using Bloom’s Taxonomy

4



Bloom’s Taxonomy: A Framework for Learning

Hierarchical levels of cognitive skills:

Remember — Recall facts and basic concepts

Understand — Explain ideas or concepts

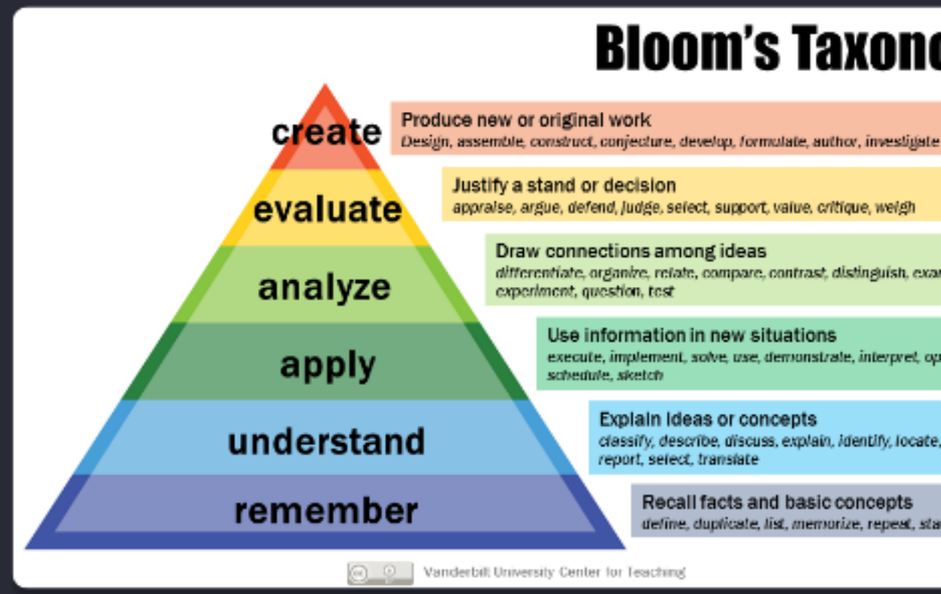
Apply — Use information in new situations

Analyze — Draw connections among ideas

Evaluate — Justify a decision or course of action

Create — Produce new or original work

Key insight: Higher levels require mastery of lower levels



Writing Learning Objectives with Bloom's Taxonomy

Episode-level

Example from an episode in demo module

```
:::{objectives}
- Understand limitations of Python's standard library for
  large data processing
- Understand the logic behind NumPy ndarrays and learn to use
  some NumPy numerical computing tools
- Learn to use data structures and analysis tools from Pandas
:::
```

Use action verbs aligned with learning levels:

1. **Remember**: define, list, recall, recognize, state
2. **Understand**: describe, explain, paraphrase, summarize
3. **Apply**: execute, implement, solve, use, demonstrate
4. **Analyze**: compare, contrast, differentiate, examine
5. **Evaluate**: assess, critique, judge, justify, test
6. **Create**: design, construct, develop, formulate, author

Principle: Match verbs to intended learning depth

Module Learning Outcomes

Module-level

In the landing page (index.md), we start with

“By the end of this module, learners should:

- Have a good overview of available tools and libraries for improving performance in Python
- Know libraries for efficiently storing, reading and writing large data
- Be comfortable working with NumPy arrays and Pandas dataframes for data analysis using Python”

Note: Always **link outcomes** to specific skills in the **skill tree**!

Formative Assessment

Assessment *For* Learning

Formative assessment = Frequent, low-stakes checks during learning

Summative assessment = or exams / quizes. Takes place at the end of the lesson.

> *Learners don't pass or fail formative assessment; instead, it gives both the teacher and the learner feedback on how well they are doing and what they should focus on next.*

> *Summative assessment is like a driver's test: it tells the learner whether they have mastered the topic and the teacher whether their lesson was successful*

Rule of thumb: Formative assessment every 10-15 minutes

6

Purpose:

Reveals misconceptions *immediately*

Guides instruction in real-time

Builds confidence through success

Types of Formative Assessment

Multiple Choice with Diagnostic Distractors

Each wrong answer reveals a specific misconception

Other Techniques

- **Parsons problems**: Rearrange scrambled code
- **Fill-in-the-blank**: Complete partial solutions
- **Predict the output**: Before running code
- **Type-along exercises**: Guided practice
- **Exercises with solutions**: Hands-on practice

Example Exercise from Demo Module

````
:::::{exercise}

Start with ...

:::{code-block} python
:emphasize-lines: 2

def foo():
    ...
:::

::::{solution}

You can ...

:::{code-block} python

class Bar:
    pass

def foo():
    return Bar()
:::
::::
:::::
````

Always nest solutions inside exercises!

Managing Cognitive Load and memory management

7

Working Memory is Limited

The bottleneck: ~4-7 items in working memory

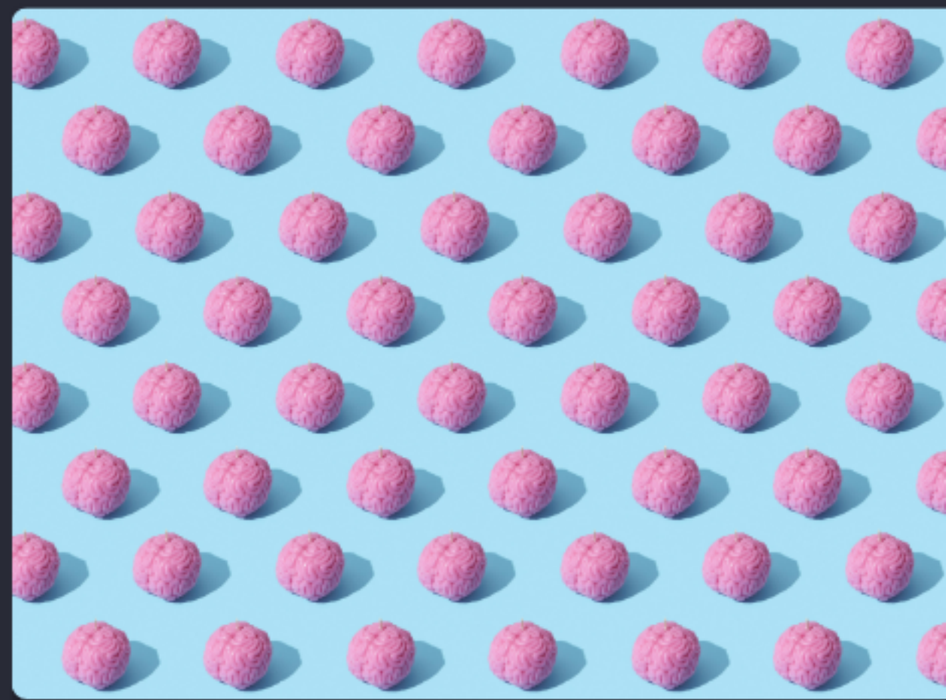
Three Types of Cognitive Load

Intrinsic: is what people have to keep in mind in order to absorb new material,

Germane: is the (desirable) mental effort required to link new information to old,

Extraneous: unnecessary distractions

> *Our goal as teachers is to maximize the memory available to **handle intrinsic load**, which means **reducing the germane load** at each step and **eliminating the extraneous load**.*



8

Strategies to reduce Cognitive Load

Stick to the script

Conflicting information splits the attention and leads to unnecessary cognitive load.

Copy-pasting vs manual typing□

Manually typing out code into a programming environment takes significantly longer time than copy-pasting it from lesson material, but learners usually strongly prefer it.

The importance of going slow

Understand

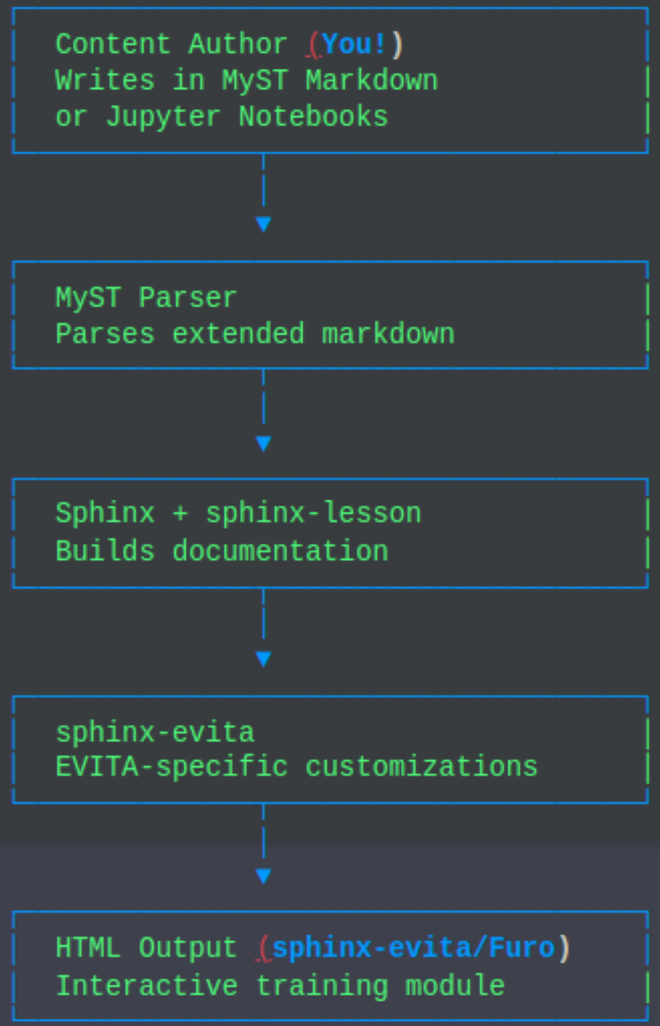
- Expert awareness gap
- Taking advantage of errors
- Meta-talk

Part II: Technical Implementation

The EVITA Toolchain

10

How the Tools Fit Together



Tool Roles

MyST Parser: Markdown → Sphinx AST

Extended CommonMark syntax

Directives and roles

Technical authoring features

MyST-NB: Jupyter Notebook → Sphinx AST

Support for rendering and executing Jupyter notebooks

Also support executing lightweight markdown files generated using Jupytext.

Sphinx: Documentation generator

Cross-references, TOC management

Multiple output formats

sphinx-lesson: Educational directives

Exercises, objectives, prerequisites

From CodeRefinery project

sphinx-evita: EVITA customizations

Theme integration (Furo)

Project-specific features

sphinx-autobuild /

JupyterLab + jupyterlab-myst: Live preview

Real-time rendering

Interactive development

Setting Up Your Environment

11

Local Development Setup

Step 1: Create Virtual Environment

```
python3 -m venv evita-env
source evita-env/bin/activate
# Linux/Mac
```

Step 2: Install Dependencies

```
pip install --group notebook .
```

This installs (after reading `pyproject.toml`):

Sphinx and sphinx-evita

And all other tools such as MyST parser, sphinx-autobuild as dependencies

JupyterLab with MyST extension

Record your lesson-specific sependencies in the notebook dependency group!

12

Live Development Server

Step 3: Launch Live Preview

```
make livehtml
```

What this does:

Builds your module using `sphinx-autobuild`

Starts local web server

Watches for file changes

Auto-rebuilds on save

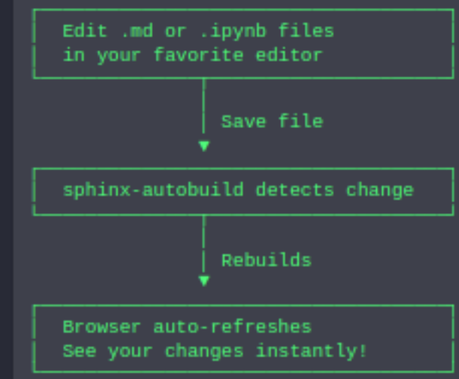
Benefits:

Immediate feedback

No CI/CD dependency

Fast iteration

Works offline



Access via local HTTP server: Usually `http://127.0.0.1:8000`

No manual rebuild needed!

MyST Markdown Basics

13

MyST = Extended Markdown Syntax

Markedly Structured Text

Standard Markdown Works

```
# Heading 1
## Heading 2

**bold** and *italic*

- Bullet lists
1. Numbered lists

`inline code`

[link text](https://example.com)
```

Plus Extensions

Directives (block-level elements)

Roles (inline semantic markup)

Math equations

Cross-references

Footnotes

Directives: Block-Level Elements

Syntax:

```
:::{directive-name}
:option1: value
:option2: value

Content goes here
:::
```

Example:

```
:::{note}
This is an important note!
:::
```

Common directives: `code-block`, `figure`, `table`, `admonition`

Roles: Inline Semantic Markup

Syntax: `{role-name}`content``

Examples:

- `{doc}`other-page`` — Link to another document
- `{ref}`section-label`` — Link to labeled section
- `{cite}`author2024`` — Citation reference
- `{kbd}`Ctrl+C`` — Keyboard shortcut
- `{math}`E=mc^2`` — Inline math

Code Blocks with Features

Basic Code Block

````
```python
def hello():
    print("Hello, EVITA!")
```
````

With Line Emphasis

````
```{code-block} python
:emphasize-lines: 2,3
:linenos:

def calculate():
    # These lines are highlighted
    result = sum(range(10))
    return result
```
````

Math Equations

Inline Math

```
Einstein showed that {math}`E=mc^2`.
```

Math Block

```
$$
\int_0^\infty e^{-x^2} dx = \frac{\sqrt{\pi}}{2}
$$
```

You can also use `{math}` as a directive

Images and Figures

Simple Image

```
![Alt text](./images/array-indexing-1D.png)
```

With Size and Alignment

```
![](./images/pandas_dataframe.svg){w=30em align=center}
```

Tip: Store images in `episodes/images/` directory

sphinx-evita Directives

Module Structure Directives

Prerequisites (in index.md)

```
:::{prereq}

- Basic experience with Python
- Basic experience in working in a Linux-like terminal
- Some prior experience in working with large or small datasets
:::
```

Place on your landing page to set expectations!

Episode Structure

Learning Objectives

```
:::{objectives}
- Understand limitations of Python's standard library
  for large data processing
- Understand the logic behind NumPy ndarrays and learn
  to use some NumPy numerical computing tools
- Learn to use data structures and analysis tools from Pandas
:::
```

Place at the start of each episode!

Instructor Note

```
:::{instructor-note}
- 25 min teaching/type-along
- 25 min exercising
:::
```

Use for:

- Timing guidance
- Common misconceptions
- Teaching tips
- Setup requirements

Meant as inline guidance for instructors, but visible to all!

Admonition Types: Information

Passive Information

```
:::{note}
Important emphasis within content
:::

:::{seealso}
Related resources and external links
:::

:::{demo}
Code for demonstration (non-participatory)
:::

:::{discussion}
Discussion prompts for learners
:::
```

Admonition Types: Active Learning

Encouraging Participation

```
:::{type-along}
Live coding sessions at slower pace
:::

:::{exercise}
Tasks for learners to complete
:::

:::{solution}
Exercise answers (nest inside exercise)
:::

:::{tip}
Helpful suggestions
:::
```

Admonition Types: Warnings

Different Severity Levels

```
:::{caution}
You should copy the code above to a separate code block,
or change its cell type from Markdown to Code.
:::

:::{warning}
You might get different results from the above code example.
:::

:::{attention}
If you get an error like xxx, you should import numpy
before the first line `import numpy as np`.
:::
```

Use appropriately: Don’t overuse warnings!

See Also directive: Further Resources

External References

```
:::{seealso}

- [Introduction to running R, Python, Julia, and Matlab in HPC](https://uppmax.github.io/R-python-julia-matlab-HPC/)
- [Practical Intro to GPU Programming using Python](https://github.com/ENCCS/webinar_documents/tree/main/2024-oct-24-python)
- [Using Python in an HPC environment](https://uppmax.github.io/HPC-python/)
- [Python for Scientific Computing](https://aaltoscicomp.github.io/python-for-scicomp/)

:::
```

Advanced Features

Collapsible Content

````
::::{solution}
:class: dropdown

Click to reveal the solution...
::::
````

Tabbed Content (System-Specific)

````
::::{tabs}
:::{group-tab} Linux
```bash
sudo apt install docker
```
:::

:::{group-tab} macOS
```bash
brew install docker
```
:::
::::
````

PDF Slides Embedding

Embedding Presentations

```
:::{pdfembed} slides/efficient-array-computing.pdf
:::
```

Requirements:

- Store PDF in `_static/slides/` directory
- Provide both PDF and source (PPTX/ODP)
- Referenced from episode markdown

Module Structure

Required Files and Directories

```
content/
├── conf.py               # Sphinx configuration
├── index.md              # Landing page
├── episodes/             # Teaching content
│   ├── ModuleName-0-Setup.ipynb
│   ├── ModuleName-1-Topic.md
│   ├── ModuleName-2-Topic.ipynb
│   ├── images/           # Episode images
│   │   └── diagram.svg
│   ├── code/             # Code examples
│   │   └── example.py
│   └── quiz/             # Summative assessment
│       ├── quiz.md
│       └── solution.md
├── instructor-guide.md   # Teaching recommendations
├── reference-for-learners.md  # Glossary, resources
└── _static/
    └── slides/           # Presentation files
        ├── topic.pdf
        └── topic.pptx
```

The index.md File

Landing Page Structure

Must include:

1. **Title and description** (1-3 paragraphs)
2. **Prerequisites** (using `{prereq}` directive)
3. **Table of contents** (using `{toctree}`)
   - Software setup section
   - Episodes section
   - Reference section
4. **Learning outcomes** with skill tree links
5. **Credit** section
6. **License** information

Table of Contents with toctree

```
:::{toctree}
:caption: Software setup
:maxdepth: 1

episodes/Python-HPDA-0-SoftwareSetup
:::

:::{toctree}
:caption: Episodes
:maxdepth: 1

episodes/Python-HPDA-1-Motivation
episodes/Python-HPDA-2-EfficientArrayComputing
:::

:::{toctree}
:caption: Reference
:maxdepth: 1

instructor-guide
reference-for-learners
:::
```

Episode Naming Convention

Recommended Pattern

```
ModuleName-N-EpisodeName.{md,ipynb}
```

Examples:

- `Python-HPDA-0-SoftwareSetup.ipynb`
- `Python-HPDA-1-Motivation.ipynb`
- `Python-HPDA-2-EfficientArrayComputing.md`

Benefits:

- Clear ordering
- Module identification
- Descriptive names

Episode Content Structure

Each Episode Should Have

1. **Title** (H1 heading)
2. **Introduction** (brief context)
3. `{objectives}` directive
4. `{instructor-note}` directive
5. **Content sections** with:
   - `{demo}`, `{type-along}`, or `{discussion}`
   - `{exercise}` with nested `{solution}`
   - Images, code blocks, examples
6. `{keypoints}` summary
7. `{seealso}` references

Jupyter Notebook vs Markdown

When to Use Each

Jupyter Notebooks (.ipynb):

- Interactive code cells
- Executable examples
- Output embedded in document
- Great for data analysis, scientific computing

Markdown (.md):

- Simpler editing
- Better version control
- Faster rendering
- Great for conceptual content

Both are supported! Choose what fits your content.

The conf.py File

Sphinx Configuration

Key elements:

```
# Project information
project = 'High Performance Data Analytics in Python'
author = 'Your Name'

# Extensions
extensions = [
    'myst_nb',           # MyST Notebook support
    'sphinx_lesson',     # Educational directives
    'sphinx_evita',      # EVITA customizations
    # ...
]
```

Summary

You're Ready to Create!

The pedagogical foundation:

Start backwards from skills tree and learning outcomes

Assess frequently with formative assessments every 10-15 minutes

Manage cognitive load with scope of the module

Structure each episode with objectives, content, and key points

The technical toolkit:

MyST markdown provides rich, readable content authoring

sphinx-evita and other extensions give you educational scaffolding

`make livehtml` offers instant feedback during development

Start Writing Today

You have (nearly) everything you need:

Start at (sphinx-evita docs)

Pedagogical best practices (Teaching Tech Together, ENCCS instructor training)

Demo module will be released soon.

Your learners need your expertise!

The EVITA community benefits from each module y create. Start small — even a single well-designed episode makes a difference.

Questions? Challenges?

Reach out to the EVITA team. We're building this together.

Resources & References

Pedagogical Foundations

Teaching Tech Together (Greg Wilson) https://teachtogether.tech/en/index.html
Comprehensive guide to evidence-based teaching practices, including backwards design, formative assessment, and cognitive load theory

ENCCS Instructor Training https://enccs.github.io/instructor-training/ *Training program fo technical instructors, covers lesson design, interactive teaching, and best practices for HP education*

Resources & References (cont.)

Technical Documentation

sphinx-evita Documentation https://sphinx-evita.readthedocs.io/en/latest/ *Main documentation for the EVITA Sphinx extension, includes installation, configuration, and overview*

sphinx-evita Module Author Reference https://sphinx-evita.readthedocs.io/en/latest/ref-module-authors.html *Complete reference for all available directives, admonitions, and features for creating EVITA modules*

MyST Parser Documentation https://myst-parser.readthedocs.io/en/latest/ *Extended Markdown syntax guide, configuration options, and advanced features*

Resources & References (cont.)

Related Projects

Sphinx (documentation generator) https://www.sphinx-doc.org/ *Core documentation build system*

sphinx-lesson (CodeRefinery) https://coderefinery.github.io/sphinx-lesson/ *Educational directives and conventions for lesson materials*

Furo (Sphinx theme) https://pradyunsg.me/furo/ *Modern, responsive theme used by EVIT*

HPC Certification Forum https://www.hpc-certification.org/wiki/skill-tree/b *Comprehensiv HPC skill tree for mapping learning outcomes*

1. Background photo by Katerina Holmes on Pexels↩
2. Photo by Katerina Holmes on Pexels↩
3. Background photo by Ivan S on Pexels↩
4. Background photo by Mouad Mabrouk on Pexels↩
5. Background photo by Andy Barbour on Pexels↩
6. Photo by Andy Barbour on Pexels↩
7. Background photo by DS stories on Pexels↩
8. Photo by DS stories on Pexels↩
9. Background photo by ThisIsEngineering on Pexels↩
10. Background photo by suntorn somtong on Pexels↩
11. Background photo by Lukas on Pexels↩
12. Photo by Lukas on Pexels↩
13. Background photo by Lukas on Pexels↩
14. Background photo by cottonbro studio on Pexels↩