Co-funded by the European Union

Project Nr. 2022-1-FR01-KA220-HED-000086863

lightcode

Strengthening the Digital Transformation of Higher Education Through Low-Code

8. Let's explore the LightCode platform

KarmicSoft

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

Erasmus+ Project
lightcode

Co-funded by the European Union

PROJECT'S COORDINATOR

Paris Dauphine University
France

PROJECT'S PARTNERS

University of Macedonia Greece

University of Niš
Serbia

Karmic Software Research
France

REACH Innovation
Austria

University of Zagreb
Faculty of Economics & Business

University of Zagreb
Croatia

Symplexis
Greece

symplexis.eu

TABLE OF CONTENTS

OVERVIEW

Introduction

In a world where technology shapes every field, the ability to develop digital solutions is no longer reserved for expert programmers. The LightCode platform introduces a fresh and accessible way to explore software development without requiring years of technical training. This module offers you the opportunity to take your first steps into the world of low-code application development by exploring a platform specifically designed to simplify complex technologies and foster creativity.

Through this introductory exploration, you will become familiar with the platform's environment, try out basic functionalities, and experience first-hand how ideas can be turned into working prototypes—without writing extensive code. There is no expectation of mastering the tool at this stage. Instead, you are invited to "play" with the platform, discover its potential, and ignite your curiosity for what's possible. Whether you dream of building applications or simply wish to better understand how digital solutions are created, this exploration will open new doors to your future career opportunities.

Why is this Module Important?

Imagine standing before a locked door that leads to a world of innovation, creativity, and new career possibilities. Traditionally, only those fluent in complex programming languages held the keys. Today, however, low-code platforms like LightCode provide a simpler, more inclusive key—one that opens this door to everyone, regardless of their technical background.

Consider the story of **Emma**, a psychology student with a passion for judo. While helping to organize a local judo tournament, Emma noticed that managing beverage orders for the participants and attendees was a chaotic, manual process. Instead of struggling with spreadsheets or waiting for a

technical expert, she turned to the LightCode platform. In just a few hours, she built a simple application that allowed users to place and track beverage orders directly from their phones.

This small but practical solution made the event run smoothly and saved countless hours of confusion. More importantly, it showed Emma that she didn't need to be a programmer to solve real-world problems. This experience sparked her curiosity about technology and how she could apply these skills in her future career.

This module is your first step toward that same empowerment. You will explore the LightCode platform in a safe, guided environment, understanding how digital tools can turn your ideas into reality. While you are not expected to become an expert, this first encounter might just be the spark that changes the way you see your professional future.

Learning Outcomes

By the end of this module, you will be able to:

- Navigate the LightCode platform and explore its main features.
- Customize simple apps using visual building blocks.
- Format content using basic **Markdown** for clearer, more attractive presentations.
- Understand and read simple **YAML** structures used to define app configurations.
- Discover basic **Python** logic to control simple object behaviors and automate transitions.

Prerequisites

This module is designed for **absolute beginners**—no programming experience is required. However, to help you feel confident as you explore, it's helpful to have:

- Basic familiarity with using a web browser and common digital tools (like email, Google Docs, or Microsoft Word).
- A general understanding of how information is organized, such as items in a shopping list or rows in a spreadsheet.
- Most importantly, an open and curious mindset, ready to explore and have fun without the pressure of being perfect!

💡 *Think about it this way: if you know how to order a pizza online, you already have the skills you need to use LightCode! By the end of this module, you'll even be able to understand, configure and even build a simple app that calls its own functions—just like clicking a button to order your favorite pizza or beverage.*

🪪 **So what, no driver's license needed to use LightCode?** Exactly! We've designed this platform to make your journey as smooth and engaging as possible. However, we also believe it's important to promote responsible resource use, acknowledge the effort behind content creation, and ensure fair access.

That's why **advanced features—including Creative Mode—are reserved for selected learners**. You might need to ask your instructor or project mentor to unlock these capabilities for you. But don't worry—even without access to Creative Mode, you can explore, learn, and fully enjoy everything this introductory experience offers.

And the best part? **Access to this level of discovery remains completely free**. When you're ready for more, we'll be right here to guide you to the next stage of your adventure!

STRUCTURE OF THE MODULE

📚 **Structure of the Module – Your Learning Journey**

- **Start with the Big Picture**

Explore how technology is no longer reserved for expert coders. Discover how the LightCode platform empowers everyone to turn ideas into interactive solutions.

- **Meet Emma and Unlock the First Door**

Follow the inspiring story of Emma, a psychology student, and see how low-code tools can solve real-life problems—even managing a busy judo tournament!

- **Get Comfortable on the LightCode Main Page**

Take your first tour of the platform's home base. Learn how to navigate, access tutorials, and explore projects—all from one central hub.

- **Jump into the Tutorial Playground!**

This is your creative sandbox. Click, try things out, make mistakes, and have fun! Remember, there's no wrong way to explore here.

- **Play with Building Blocks—Your New Digital LEGO®**

Experiment with Text, Image, Video, Map, Data, and Object blocks. Combine them to create something unique and exciting.

- **Take On Your First Interactive Challenge**

Complete small missions that reinforce what you've learned. Can you help Lucky the dog find a park on the map?

- **Solve the Riddle of Lucky & Wanda**

Put your critical thinking to the test. What happens if Lucky's sleeping and his bestie barks? Dive into objects and interactions to find out!

- **Pause and Reflect**

Take a moment to step back. What did you discover? What surprised you? Share your thoughts with classmates or your instructor.

- **Discover the Hidden Creative Mode**

With the right access, step beyond exploration and start shaping your own digital environments. Change layouts, customize blocks, and build the app you envision.

- **Go Behind the Scenes with the Configuration Interface**

Learn how to configure blocks visually—no coding needed. Modify icons, titles, and layouts with just a few clicks.

- **Make Your Content Shine with Markdown**

Add bold, colorful, and structured content using Markdown formatting. It's an easy way to impress and a powerful skill for your future career.

- **Peek Behind the Curtain with YAML**

Curious how apps are really structured? Explore YAML and see how it powers everything you've created behind the scenes.

- **Master the Art of Objects, States, and Behavior at the Judo Café**

See how digital objects interact, change states, and bring your app to life—just like running a real coffee shop!

- **Understand How Apps Think with Business Logic**

Explore how actions like placing orders and managing payments work through methods and state transitions.

- **Design Like a Pro in the Class Designer**

Discover how classes, attributes, and methods are built to define real-world logic inside your app.

- **Take the Next Step with Python Logic (Advanced)**

Ready for more power? Learn how embedded Python code makes your apps smarter and more dynamic.

- **Don’t Forget to Test—Quality First!**

Before you celebrate, run tests to make sure everything works perfectly. Like Emma, always taste the coffee before you serve it!

- **Wrap It All Up with a Final Quiz!**

Test your knowledge, reflect on your journey, and unlock your LightCode Explorer Badge. You’re ready for the next adventure!

LIGHT-CODE, STEP BY STEP

1. Welcome to the LightCode Main Page

💡 In this section, you will **step directly into the LightCode interface** and experience what it feels like to interact with applications not only as a passive user, but as a creator. Don't worry about making mistakes—this is your sandbox to explore, click, and have fun!

💡 When you first open LightCode, you'll land on the **Main Page**. Think of this as your personal starting point—just like the homepage of an online store, but instead of buying things, you're about to build and explore!

Here's what you'll see:

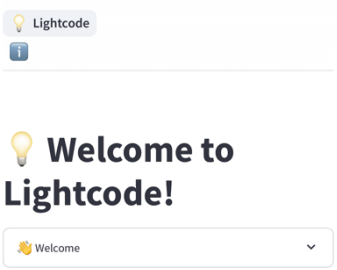
- 👩‍🎓 **Learn Section**: A quick introduction reminding you that LightCode is here to help you build apps with little or no code.
- **Discover Your Options**:
    - ✓ Discover basic building blocks.
    - ✓ Read tutorials to guide you through examples.
    - ✓ Create and customize modules and blocks.
    - ✓ Build your own apps when you're ready.
    - ✓ Share your creations with others!

At the top, you'll find helpful icons and menus:

- 👤 **User Profile**: Personal settings and preferences.

- 📋 **Custom Menu**: Quick access to tutorials, your projects, and the platform's core features.
- **ℹ Information Section**: Copyright and helpful documentation links.

➡️ **Tip:** If you're ever lost, just return to this main page by clicking the LightCode logo on the top left corner. If you're on your mobile device, it's similar.



2. Ready? Let's Enter the Tutorial Playground!

From the main page, head to the **Tutorial 101** module. This is your playground to explore and experiment with LightCode's building blocks. Don't worry about making mistakes—this space is designed for discovery!

💡 In this section, you will **step directly into the LightCode interface** and experience what it feels like to interact with applications not as a passive user, but as a creator. Don't worry about making mistakes—this is your sandbox to explore, click, and have fun!

3. Meet the Basic Building Blocks

Everything you see and interact with in LightCode is called a **block**. Just like building with LEGO®, you can combine different blocks to create something new. These are the fundamental types of blocks you will encounter:

- 📖 **Text Block**: Displays formatted text. Try finding hidden hints and colourful messages!
- 🐕 **Image Block**: Adds images to your app—like our friendly mascot Lucky the dog!
- 📽️ **Video Block**: Play embedded videos directly within your app interface.
- 🗺️ **Map Block**: Try interacting with the map. Zoom in and out to discover hidden spots!
- 📈 **Data Blocks**: Explore editable Grids and Charts—change the data and watch what happens!
- 📦 **Object Blocks**: These are more advanced but fun to explore. They can hold data, behaviors, and even custom actions.

💡 ***Did you notice?*** *Every block has a small icon and a title. These help you quickly identify what each block does. Hover over or click them to reveal helpful tips!*

4. Try It Yourself: Interactive Discovery

Here's your first mission:

- 👩‍💻 Open the Tutorial 101 module.
- 🎯 Find the **Text Block** that says *"Man's best friend 🐶"*.
- 🔎 Hover over it to reveal the hidden hint.

- 🗺️ Go to the Map Block and try zooming in on Paris—can you find a park for Lucky the dog?
- 📈 In the Data Grid, edit Lucky's energy level. What changes do you observe?

Remember, **just like ordering a pizza online**, every action you perform (clicking, selecting, entering data) sends a simple instruction. In the background, the platform is calling functions to make things happen—without you writing any code!

5. Explore the Riddle

🧩 Ready for a bit of fun? Try solving the riddle about Lucky and Wanda:

"What happens if Lucky's sleeping and his bestie barks?"

Try interacting with the objects and forms to figure it out. No worries if you don't find the answer immediately—this is all part of the discovery!

6. Reflect & Share Your Experience

At the end of your exploration, take a moment to reflect:

- Which blocks were easy and intuitive to use?
- Did you discover any hidden hints or Easter eggs?
- Which part of the interface are you curious to explore further in the next modules?

💬 Feel free to share your feedback directly and discuss your experience with your classmates.

⚠️ ***Warning.*** *For the following, you'll need an account and be logged-in. If you don't have one, ask your instructor. Otherwise, continue the exploration as an end-user. There are always more gems and easter-eggs and come back later to continue the interactive learning.*

7. Discover the Configuration / Creative Mode

Have you ever wished you could rearrange a website or app to better suit your needs? In LightCode, that's not just possible—it's encouraged! With the proper access, you can enter the **Creative Mode**, where you're no longer just exploring; you're actively shaping your environment.

🛠️ How to Enter Creative Mode

- Look for the ⚙️ **Gear Icon** next to any block title or at the top of the page.
- Click it to open the **Configuration Panel**, where you can:
    - Change the block's title, icon, or description.
    - Add or remove help hints for other users.
    - Adjust visual styles such as borders, colours, and media attachments.

- Configure data sources if the block is interactive (for example, linking a data grid to a specific dataset).

👷 **Example:** Click the ⚙️ gear on the **Text Block** titled *"Man's best friend 🐶"*. Try changing the title to *"Lucky's Playground"* and update the image if you have one!

🧩 Drag & Drop to Build Your Own Page Layout

In LightCode, a page is called a **Module**, and you can fully customize its layout:

1. **Activate Layout Mode** by clicking the *layout* button.
2. **Drag & Drop Blocks** to rearrange their order directly on the screen.
3. **Add New Blocks**:
   - Click the ➕ button to add new Text, Image, Video, Map, or Data blocks.
4. **Group Blocks**: Create logical sections by visually grouping related blocks together.
5. **Preview Your Layout**: Instantly see how it looks and works for end-users.

🎨 Creative Mode in Action: Try This Mini-Challenge!

- Enter the Creative Mode on the Tutorial 101 module.

- Rearrange the page so that:
    - The **Map Block** appears directly after the **Man's Best Friend** 🐶 image.
    - Add a new **Text Block** titled *"Fun Fact About Dogs"* and include a quick fact.
    - Try removing one block and see how it affects the layout.

📣 Don't worry—nothing is permanent! You can always reset the layout if you refresh your page. Be aware, you'll also loose your modifications. More later about how to save them and where there are really stored.

💡 **Pro Tip:** Think of Creative Mode like arranging your living space. You decide what goes where, what's highlighted, and how everything flows. And just like home décor, it's fun to try new arrangements until it feels right!

8. Behind the Scenes: Exploring the Visual Configuration Interface

In LightCode, every block can be configured directly through a clean and intuitive visual editor. No need to worry about code—if you can fill in a form, you can customise a block!

📖 Example: Editing a Simple Text Block

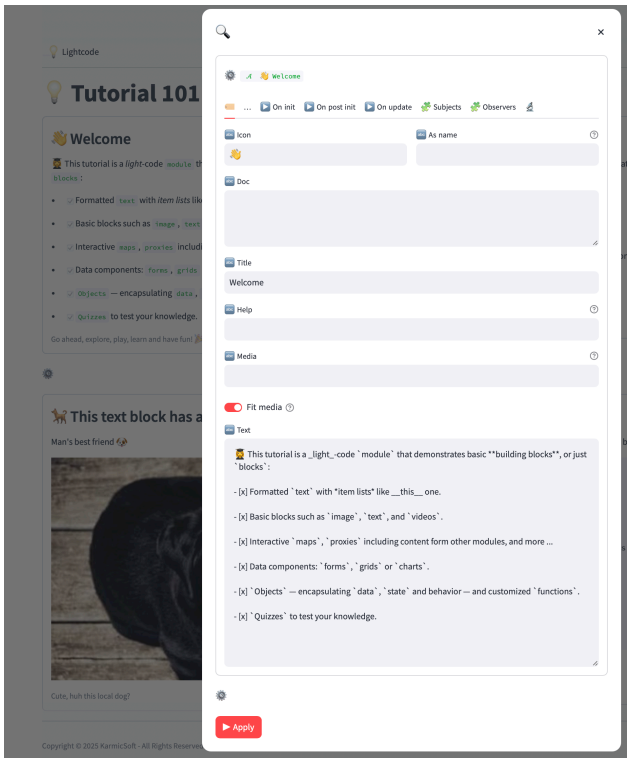
Here's what you see when you click the ⚙️ **gear icon** on a block:

- 📝 **Icon**: Pick an emoji or symbol that visually represents your block. Example: 👋 for a welcome message.
- 📚 **Doc**: Add detailed documentation or notes for collaborators (optional).
- 🏷️ **Title**: This appears at the top of the block. Keep it short and descriptive, like "Welcome" or "Fun Fact."
- 💡 **Help**: Provide quick tips or hints that appear when users hover over the block.
- 🖼️ **Media**: Upload or link images to make the block more engaging (toggle the "Fit media" switch to adjust image sizing).
- 📝 **Text**: This is the main content area, where you enter your formatted text. You can add bold, italic, coloured text, and even interactive checklists.

➡️ Try This Now!

- Click the ⚙️ gear icon under the "Welcome" block.
- Change the icon to 🎉, update the title to "Let's Celebrate!", and modify the text to include your favorite motivational quote.
- Click **Apply** and watch the block update instantly!

💡 ***Pro Tip:*** *You can always come back and fine-tune your content later. Start simple and iterate—just like arranging blocks in a LEGO® set!*



📄 Bonus: Add Some Style to Your Text with Markdown

When editing the content of a block, you're not limited to plain text. LightCode uses a simple and powerful formatting language called **Markdown**. It's a universal language for creating beautifully formatted text without needing complicated tools.

💼 **And here's a secret:** mastering basic Markdown is a great skill to add to your resume—it's widely used in tech documentation, GitHub projects, emails, and professional reports!

✍️ Markdown Basics at a Glance

Action	Markdown Syntax	Example Result
Bold Text	**bold** or __bold__	**bold**
Italic Text	*italic* or _italic_	*italic*
Code Highlight	`code`	code
Headings	# Heading 1 to ###### Heading 6	Large → Small Headings
Lists	- item or * item	• item
Checklists	- [x] Done / - [ ] To do	✓ / ☐
Line Break	End line with 2 spaces	(Creates new line)
Links	[Text](URL)	Visit LightCode
Images	![Alt text](image_url)	
Quotes	> Quoted text	> Quoted text
Horizontal Rule	--- or ***	————————————

💡 Try This Now!

In the Text Block editor:

- Add **Your Name** to make your name bold.
- Create a quick list of your skills using - for bullets.
- Add a checkbox to track your goals:

- [x] Completed this tutorial

- [ ] Start building my first app

📣 ***Pro Tip:*** *You can even add Markdown formatting to your LightCode modules for clear and professional presentations. It's a small skill that can make a big difference in how your ideas are communicated! Moreover, the usage*

9. Behind the Curtain: Welcome to YAML

Just like Markdown lets you format text beautifully, **YAML** (Yet Another Markup Language) allows you to describe and structure entire apps, including how blocks behave and interact.

💼 **And guess what? YAML is a highly valued skill in modern IT!** It's widely used for configuring cloud services, automating deployments, and describing complex data structures in a clean, human-readable way. Add "Basic YAML Knowledge" to your CV, and you're already ahead of the game!

📖 A Quick Story: Why YAML Exists

In the early 2000s, developers were tired of writing long, messy configuration files in formats like XML. They wanted something simpler, more readable, and closer to how humans naturally structure information. That's how YAML was born—a clean, indentation-based language that feels almost like writing plain text, but with the power to control complex systems.

🛠️ **Example:** How Our "Welcome" Block Looks in YAML

```yaml
- !Text
  title: "Welcome 1z"
  icon: "👋"
  border: True
  text: |
    🧑‍🚀🎓 This tutorial is a _light_-code `module` that demonstrates basic **building blocks**, or just `blocks`:

    - [x] Formatted `text` with *item lists* like __this__ one.

    - [x] Basic blocks such as `image`, `text`, and `videos`.

    - [x] Interactive `maps`, `proxies` including content form other modules, and more ...

    - [x] Data components: `forms`, `grids` or `charts`.

    - [x] `Objects` – encapsulating `data`, `state` and behavior – and customized `functions`.

    - [x] `Quizzes` to test your knowledge.

  footer: Go ahead, explore, play, learn and have fun! 🎉
```

🧩 Let's Decode This Together

- !Text → Custom tag defining block of type **Text**.
- title → The block title shown on screen.
- icon → The emoji icon for quick visual recognition.
- border: True → Visually frames the block with a border.
- text: | → Starts a multi-line formatted text block using Markdown.
- - → the minus sign prefixes an element of a list …
- footer → Adds a closing message under the main content.

➡️ Notice how the indentation makes the structure easy to read? That's the magic of YAML: understandable by both humans and machines 👀

💡 Try This Now!

- Imagine creating a personal greeting block. Write a YAML block with your own title, a fun icon, and a motivational message.
- Example challenge:

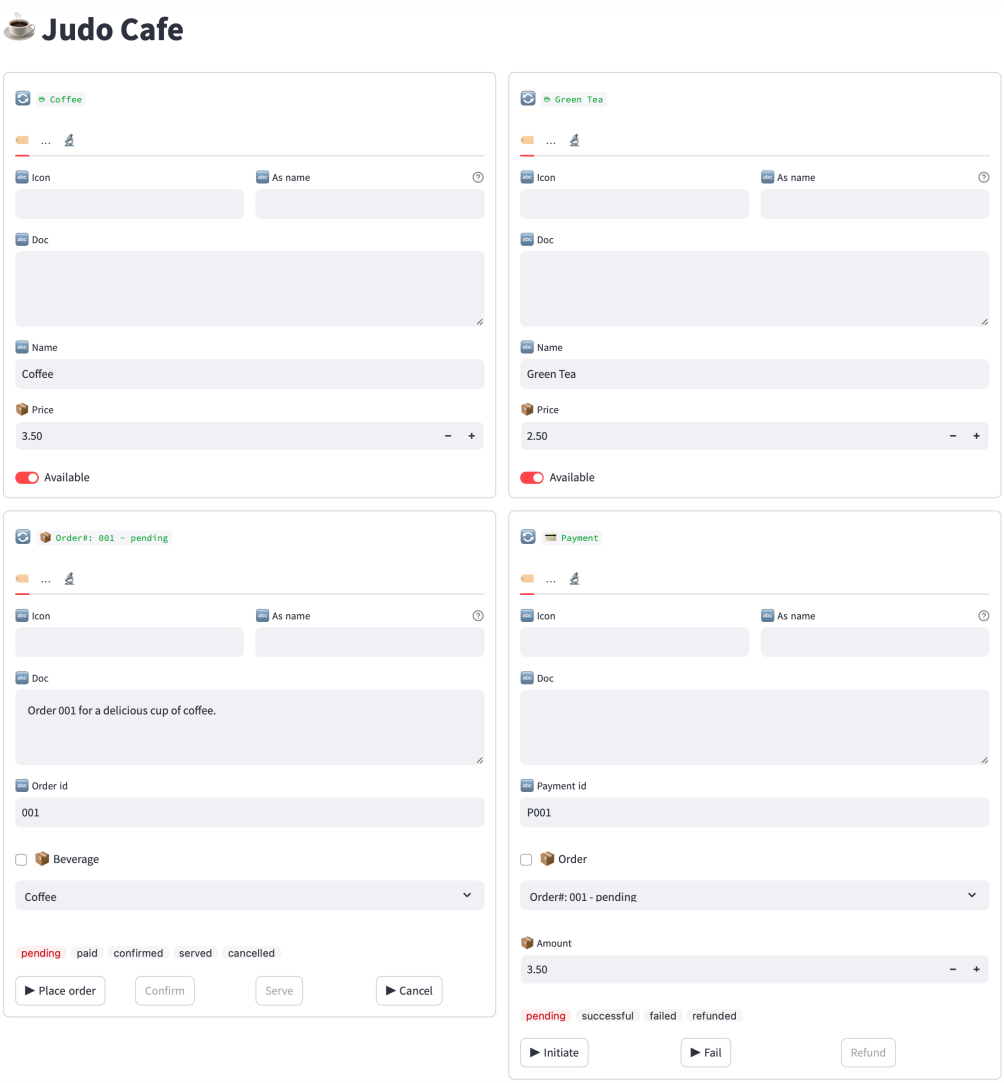
```
- !Text
  title: "My Goals"
  icon: "🚀"
  border: True
  text: |
    - [x] Complete the LightCode tutorial.
    - [ ] Build my first app.
    - [ ] Share it with the world!
  footer: "One small step today, one giant leap tomorrow! 🌕"
```

📣 ***Pro Tip:*** *YAML is everywhere—from configuring AI models to setting up your own website. Learning this now isn't just for LightCode; it's a superpower you'll use again and again! Add it to your resume now. And by the way, why don't you write your resume in YAML?* 👀

Let's walk through this as if we're teaching Emma—step by step, keeping it engaging and concrete.

10. Discovering Objects, States, and Behavior: Meet the Judo Café!

Remember Emma, our psychology student and judo enthusiast? She's back! This time, she's organizing a local judo tournament and wants to manage the beverage orders smoothly. Instead of juggling sticky notes and cash boxes, she built a simple app with LightCode to handle it all. Welcome to the **Judo Café** module!



📚 Step 1: What Are Objects?

In the digital world, **objects** are like real-life things—but smarter! They don't just exist; they also remember stuff (data) and know how to do things (behavior).

- ☕ **Beverage Object**: Represents a drink like Coffee, Tea, or Matcha.
    - Data: Name, Price, Availability.
    - Behavior: Simply exists for now—its role will become clearer soon!
- 📦 **Order Object**: Represents a customer's drink order.
    - Data: Order ID, Selected Beverage, Status.
    - Behavior: Can place, confirm, serve, or cancel an order.
- 💳 **Payment Object**: Manages payments for orders.
    - Data: Payment ID, Amount, Linked Order.
    - Behavior: Can initiate, fail, or refund a payment.

Just like in the real café, these digital objects interact with each other!

🎮 Step 2: How Do States Work?

Every object can be in a particular **state**—this describes what's happening right now.

- 📦 **Order States**:
    - pending → Just placed, waiting for payment.
    - paid → Payment completed.

- confirmed → Barista confirmed the order.
- served → Your beverage is served!
- cancelled → Order was cancelled.

- 💳 **Payment States**:
  - pending → Waiting for the customer to pay.
  - successful → Payment completed.
  - failed → Payment didn't go through.
  - refunded → Customer got their money back.

➡️ These states change automatically when Emma (or the app user) clicks buttons like **Place Order**, **Confirm**, or **Refund**. Behind the scenes, the object's state changes and new options become available or hidden.

💡 Think About It This Way:

Ordering a coffee online?

- You start in pending by selecting your drink.
- Click **Place Order**, and it moves to paid.
- The barista clicks **Confirm** and starts preparing it.
- Finally, it's marked as served.

If something goes wrong, you can cancel or refund before it's too late! Or at least, it should work this way 👀

🧩 Try This in the Judo Café App!

1. Configure your order by choosing a beverage.
2. Watch the state move from pending → paid → confirmed → served.
3. Try cancelling an order while it's still pending—notice that this option disappears after confirmation? That's business logic in action!
4. How about stock? Did you notice something?

📝 New Skill for Your CV: Object Modelling and State Management!

This is a core concept in modern app development. You're now familiar with:

- Creating and interacting with digital objects.
- Managing their lifecycle through states.
- Using visual interfaces to trigger behaviors.

Still there? Great! Let's move on to the next key concept: **Behavior**.

🧠 Adding Business Logic: How Objects Exhibit Behavior

Objects aren't just static pieces of information—they can also **act**. In programming, we say that objects have **behavior** defined by methods.

Let's revisit Emma's Judo Café app. She didn't want to just store information about drinks—she wanted her app to know how to:

- Process new orders,

- Confirm when drinks are prepared,
- Handle payments,
- And even cancel or refund orders when needed.

That's where **methods** come in!

📚 What Is a Method?

A **method** is like a skill or action the object knows how to perform. It often triggers a state change or performs a calculation.

Example Behaviors in the Judo Café:

- 📦 **Order Methods**:
  - place_order(): Moves the order from pending to paid.
  - confirm(): Marks the order as confirmed.
  - serve(): Marks the order as served.
  - cancel(): Cancels the order before it's prepared.
- 💳 **Payment Methods**:
  - initiate(): Starts the payment process and moves it to successful.
  - fail(): Marks the payment as failed.
  - refund(): Refunds the payment and changes the state accordingly.

💡 **Important Note:**

Each method defines:

- When it can be called (**preconditions**).
- What happens after it runs (**postconditions**).

➡️ Example: You can only run serve() on an order that's already confirmed. Trying to serve an order before it's confirmed? That's not allowed—the app protects you from that mistake automatically!

🎮 **Interactive Exercise: Explore Behaviors**

1. Do you see the Order object inspector?
2. Click through the available buttons to trigger methods like **Place Order**, **Confirm**, and **Serve**.
3. Watch how available options change as the order moves through states.

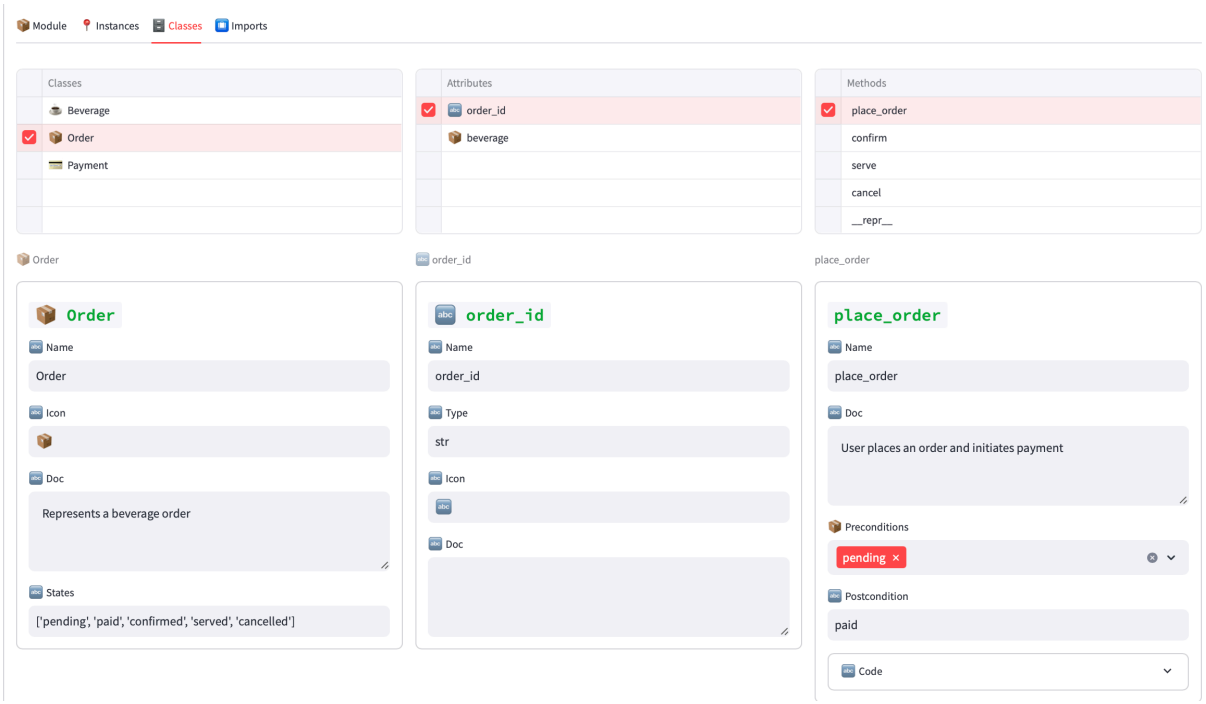
❌ Try clicking a button that shouldn't be available—see how it's disabled? That's built-in business logic at work!

📣 **Pro Tip for Your Resume:**

You now understand how to model object behaviors and manage their lifecycle—this is a critical skill used in designing real-world business applications!

11. Behind the Magic: Designing the Order Class

So far, you've interacted with objects like orders and payments. Now, let's go deeper and see how they're **designed** behind the scenes using the **Class Designer**. This is where the real structure and behavior of your app are defined.



📦 What is a Class?

A **Class** is a blueprint for creating objects. It defines:

- What data (attributes) each object will have.
- What actions (methods) it can perform.
- How it behaves when its state changes.

In our Judo Café, the **Order** class models the entire lifecycle of a beverage order.

👷 Exploring the Class Designer: The Order Class

From the screenshot you shared, we can see the Class Designer divided into three key areas:

Attributes (Data)

This section defines the data each Order object will store:

- 📌 **order_id**: A unique string that identifies each order.
- 📌 **beverage**: Links the order to a specific drink (like Coffee or Green Tea).

Methods (Behavior)

This section defines what the Order can *do*.

- 🏷️ **place_order**:
    - **Description**: "User places an order and initiates payment."
    - **Preconditions**: The order must be in the pending state before this action is possible.
    - **Postcondition**: After placing the order, its state automatically moves to paid.
    - ✅ This ensures business logic is respected—no skipping steps!

Other methods available but not selected in this view include:

- confirm(), serve(), and cancel()—each with its own conditions and effects.

States (Lifecycle Management)

This section controls how the object transitions through different states:

- pending, paid, confirmed, served, cancelled.

➡️ These states ensure that actions happen in the correct order. For example, you can't serve a drink before it's confirmed!

🎮 **Try This Challenge:**

1. Open the Class Designer for the **Order** class.
2. Review the attributes—could you add a new one like customer_name?
3. Explore the method place_order. Change its postcondition to confirmed and see how that alters the order flow (then don't forget to change it back!).

💡 **Pro Tip:**

Understanding classes, attributes, methods, and states puts you a step closer to designing full applications. Add "Basic Object-Oriented Design" to your resume—you've earned it!

12. The Ultimate Power: Adding Custom Logic with Python

In LightCode, not only can you visually design classes and behaviors, but you can also inject real Python code to control advanced logic. This is where apps become truly **smart**.

⚠️ Important Warning: Handle with Care!

What you're seeing here is the most powerful tool available in LightCode—**direct Python scripting embedded into your app's behavior**. With great power comes great responsibility!

Because this feature allows you to directly manipulate object states and control business logic, improper use could lead to unintended results or even break the app's workflow.

That's why **access to this feature may require you to first pass a few qualification tests or gain approval from your instructor or system administrator**. This ensures that you fully understand how your changes affect the system before making them live.

💡 **Good to Know:** Even without writing code, you can achieve a lot through the visual designer. But when you're ready to take the next step, this skill will open entirely new doors in your development journey!

Now let's break down what's happening inside the place_order method of the **Order** class:

Code

```
assert self.beverage, "⚠️Please make a choice first"
pass  # Transition happens automatically
if self.beverage._book():
  self.state = 'paid'
else:
  raise ValueError("Beverage not available")
```

📚 Line-by-Line Explanation

assert self.beverage, "⚠️Please make a choice first"

- ✅ **What it does:**

Before doing anything else, the app checks if the user has selected a beverage.

- 🚨 **If not:** It stops the process and shows the warning: *"Please make a choice first."*

pass # Transition happens automatically.

- This is just a placeholder. The framework can automatically handle some transitions based on defined preconditions and postconditions. The dash sign allows you to comment statements so you can remember your intentions.

if self.beverage._book():

self.state = 'paid'

- ✅ **What it does:**

It tries to **book the selected beverage** (e.g., check if it's still in stock).

- If successful, it updates the order's state to paid.

else:

raise ValueError("Beverage not available")

- 🚨 **What it does:**

If the booking fails (maybe the drink just sold out!), it raises an error and prevents the order from proceeding.

🎯 Summary of the Logic Flow:

1. Did the user choose a beverage?
2. Is it available?
3. If yes → mark as paid.
4. If no → stop and alert the user.

💼 Add This to Your Resume: Python Scripting for Business Logic!

You now know how to:

- Write simple Python validations.
- Handle errors gracefully with assert and raise.

- Control state transitions directly through code.

This is no longer just low-code—it's professional-level business logic design!

13. Ultimate Gate: Testing – Because Quality Matters!

Before any changes or new logic go live in your app, it's critical to **test** them. Just like Emma wouldn't serve a drink before tasting it, you shouldn't launch new behaviors without checking they work as expected.

In LightCode, every time you create or update classes, methods, or scripts, the system can run tests to make sure everything is functioning properly.

📋 What You See in the Test Report:

```
Loading drinks
System Under Test: 🍹 Beverages
## ☻ Beverage
ℹ️ A refreshing drink

## 📦 Order
ℹ️ Represents a beverage order
## 💳 Payment
ℹ️ Handles payment for an order
# ✅ Done with no errors.
```

- ✅ **Done with no errors** means your logic is solid—great job! As a bonus, you might see some balloons 🎈
- ❌ If errors show up, it means something went wrong (maybe a missing attribute or incorrect state transition).

🎯 Why Is This Important?

- Ensures your app is stable before end-users interact with it.
- Helps catch logical mistakes early.
- Builds trust that your business logic works exactly as designed.

💼 Pro Tip for Your Resume:

You now have practical experience with **automated testing and validation workflows**—a critical skill in professional software development!

📝 Reflection Exercise: Can You Describe How the App Works?

Look carefully at the diagram below. Without using any technical jargon, try to explain in plain English how the Judo Café app handles beverage orders and payments.

Here are some guiding questions to help you:

1. **How does the process start?**
    - What's the first thing a customer must do?
2. **What happens after a beverage is selected?**
    - Are there any checks before moving forward?
3. **How does the app know if the beverage is available?**
    - What do you think happens if it isn't?
4. **What steps follow after placing the order?**
    - Who confirms and serves the order?
5. **When does payment happen?**
    - What are the possible outcomes of the payment?
6. **Can an order be canceled? If yes, when?**

💡 **Pro Tip:** If you can explain it clearly in simple words, you're already thinking like a software designer!

📣 *Feel free to share your explanation with your instructor or group—discussing different interpretations can reveal new insights!*

🎓 14. Final Quiz – LightCode: Introduction to Low-Code

Part 1: Multiple Choice (Choose the Correct Answer)

1. **What is a block in LightCode?**

   a) A completed application

   b) A visual and logical unit of content or functionality

   c) A user profile setting

2. **Which language is used to format text in LightCode?**

   a) HTML

   b) YAML

   c) Markdown

3. **In object-oriented design, what is a class?**

   a) A single instance of data

   b) A blueprint for creating objects

   c) A payment method

4. **What happens if you try to place an order without selecting a beverage?**

   a) The app automatically picks one for you

   b) An error message appears: "⚠️ Please make a choice first"

c) The order goes through anyway

5. **Which of the following is NOT a state of an Order?**

   a) pending

   b) confirmed

   c) cooking

Part 2: True or False

1. You can always access Creative Mode without restrictions.
2. YAML is used in LightCode to structure app configuration.
3. An Order can be served before payment is confirmed.

Part 3: Reflection – Write Your Answer

In 3-4 sentences, describe how the beverage ordering process works in the Judo Café app. Try to use plain English without technical jargon.

Hint: Start from beverage selection, and think about what happens at each step up to payment.

Part 4: Bonus Challenge (Optional)

Use YAML to add a new beverage called "Iced Tea" priced at $3.00.

CONCLUSION

Congratulations on completing this introductory journey into the world of low-code development with LightCode! You've not only explored the platform's key features but also discovered how digital solutions can be created, configured, and tested without advanced programming skills.

As you continue your learning path, you'll dive deeper into collaborative work environments (see **Module 9: Collaborate with Your Team**), learn how to manage complex data structures efficiently (**Module 10: Data Management**), and automate processes to enhance productivity (**Module 11: Automating Processes**).

Your journey will culminate in mastering essential best practices for data validation and consistency (**Module 12: Data Validation & Consistency**) and learning how to secure your applications effectively (**Module 13: Securing Your App**).

The skills you've gained here—navigating the platform, working with objects and states, applying logic with Python, and using industry-standard tools like Markdown and YAML—are already valuable additions to your professional profile.

Remember: Every great application starts with exploration and curiosity. You now have the keys to continue building, collaborating, and creating impactful digital solutions for the future.

REFERENCES

- Paris Dauphine University. (2023). *WP3.2: Fundamental Elements of Low-Code*. Erasmus+ Project KA220-HED-E1CD9927.

  Low-code design principles, the importance of modularity, Model-Driven Engineering (MDE), and Agile integration in low-code environments.

- Beck, K., et al. (2001). *Manifesto for Agile Software Development*. Agile Alliance. Retrieved from https://agilemanifesto.org/

  Agile methodologies shaping low-code iterative development approaches.

- Humble, J., & Farley, D. (2010). *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*. Addison-Wesley Professional.

  Continuous testing and validation practices applied within the LightCode platform's automated testing framework.

- CommonMark Initiative. (2024). *CommonMark Specification (Version 0.30)*. Retrieved from https://commonmark.org/

  Markdown syntax standards, providing a language-agnostic, specification.

- YAML.org. (2024). *YAML Language Specification (Version 1.2)*. Retrieved from https://yaml.org/spec/1.2/spec.html

  Standard reference for YAML configuration language, used extensively in LightCode app modeling.

- Python Software Foundation. (2024). *The Python Language Reference (Version 3.12)*. Retrieved from https://docs.python.org/3/

This chapter is licensed under the Creative Commons Attribution–NonCommercial 4.0 International License (CC BY-NC 4.0). Free use, reuse, adaptation, and sharing are permitted for non-commercial purposes. Author: Michel Zam (KarmicSoft)