



THE LONDON SCHOOL  
OF ECONOMICS AND  
POLITICAL SCIENCE ■

130 EST 1895  
years

# FROM TYPIST TO ORCHESTRATOR: Rethinking Core Skills in Programming Education in the Age of AI Agents

**Dr Marcos Barreto**

**Associate Professor (Education) in Data Science**

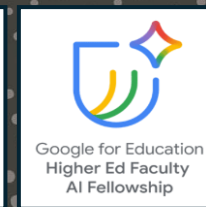
Department of Statistics

e: [m.e.barreto@lse.ac.uk](mailto:m.e.barreto@lse.ac.uk)

w: [marcosebarreto.github.io](https://marcosebarreto.github.io)

Research Showcase  
(26 May 2026)

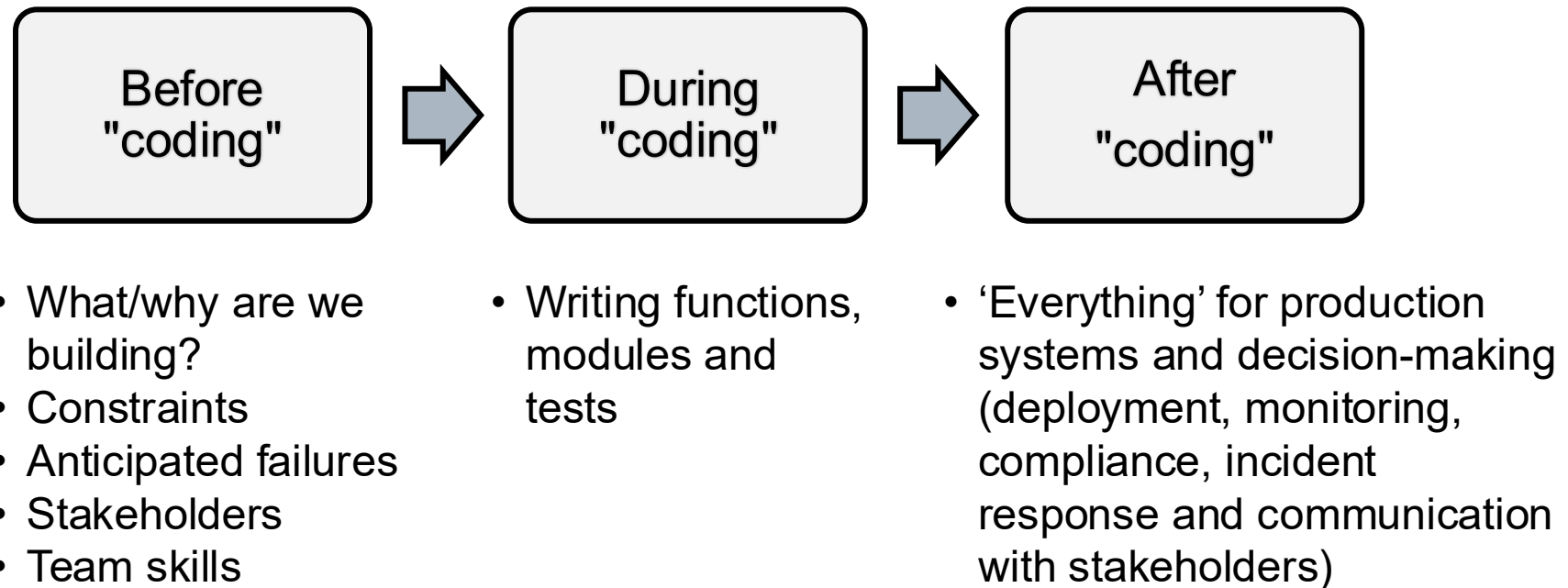
Capacity Building on Generative AI Tools  
for Programming Education  
(Talent Development Awards 2024-2025)



# Programming – concept and cycle

**Programming** is writing **computer code** (or **program**) to solve a **problem**.<sup>[1]</sup>

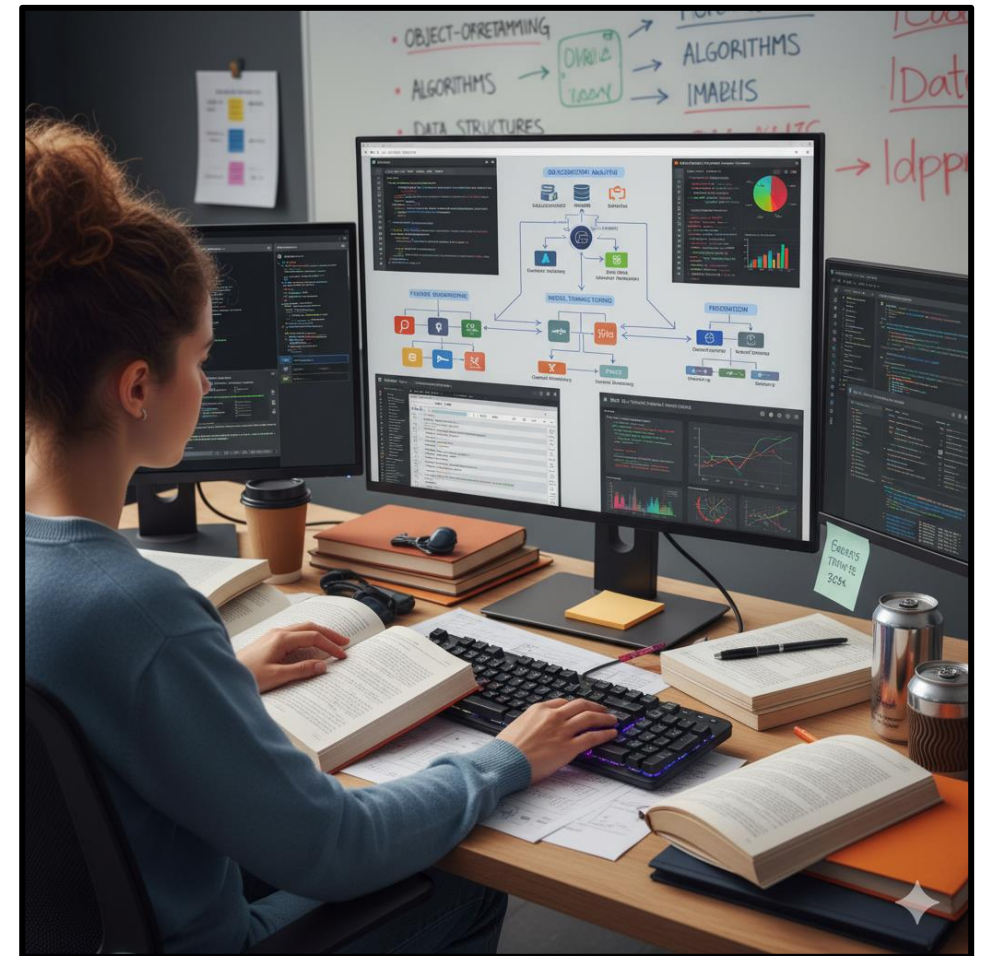
Usually, three stages:<sup>[2]</sup>



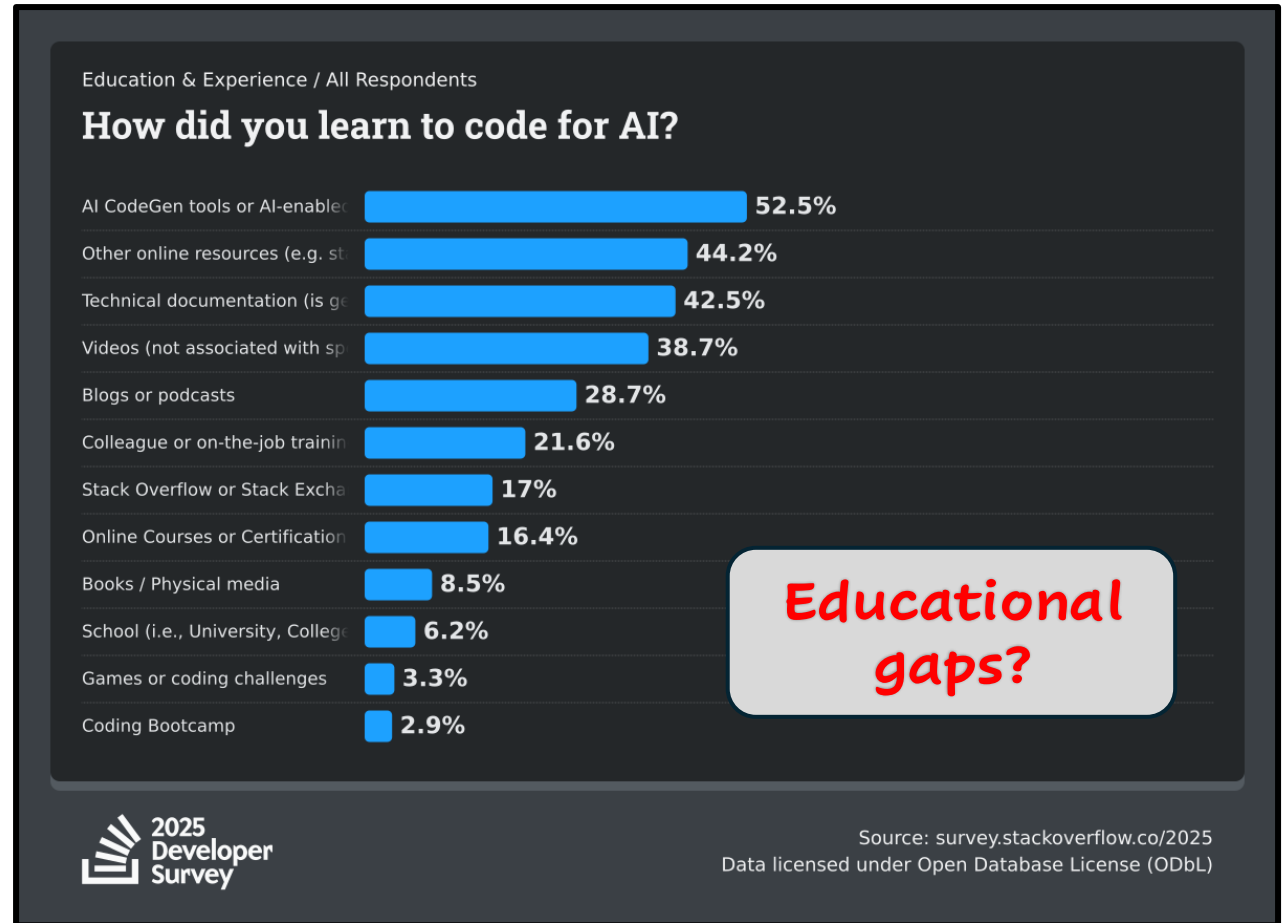
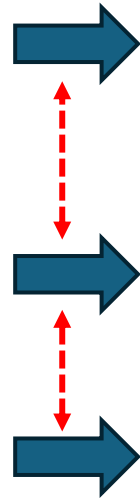
# Programming education – Higher Education

**Focus:** correctness in controlled settings

- Toy datasets
- Minimal real data / API usage
- Single-file solutions (in most cases)
- “Fill-in-the-blank” notebooks
- “Mechanical” scaffolded exercises  
(**simplified** before coding setting)
- Minimal debugging skills
- Minimal (or **inexistent**) after coding setting
- Presentation restricted to plotting/discussing results  
(technical interpretation)



# Programming education – elsewhere

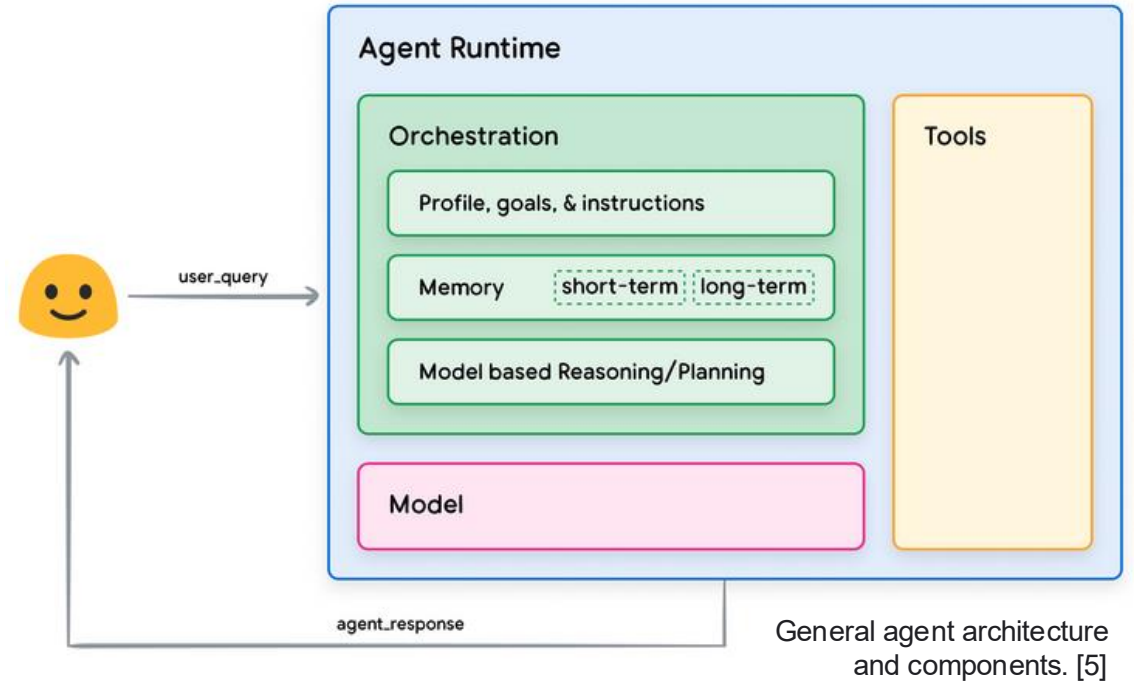
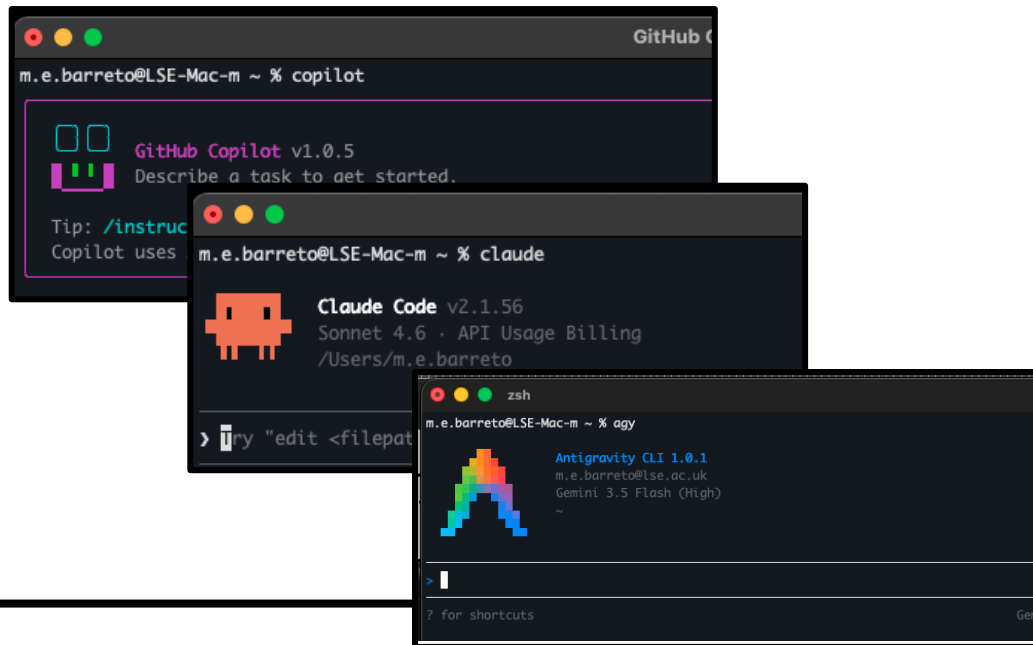


'Vibe coding' named word of the year by Collins Dictionary. [3]

Learning how to code for AI (n=28,172 – 57.5%). [4]

# Agentic frameworks

- **infrastructure**, abstractions, and tools for building agents
- emphasis on **autonomy**, planning, and goal-directed behaviour.



# Agentic frameworks – task examples

“Implement a REST API endpoint that returns the top 10 most frequently accessed URLs from a web server log file. Include unit tests and documentation.”



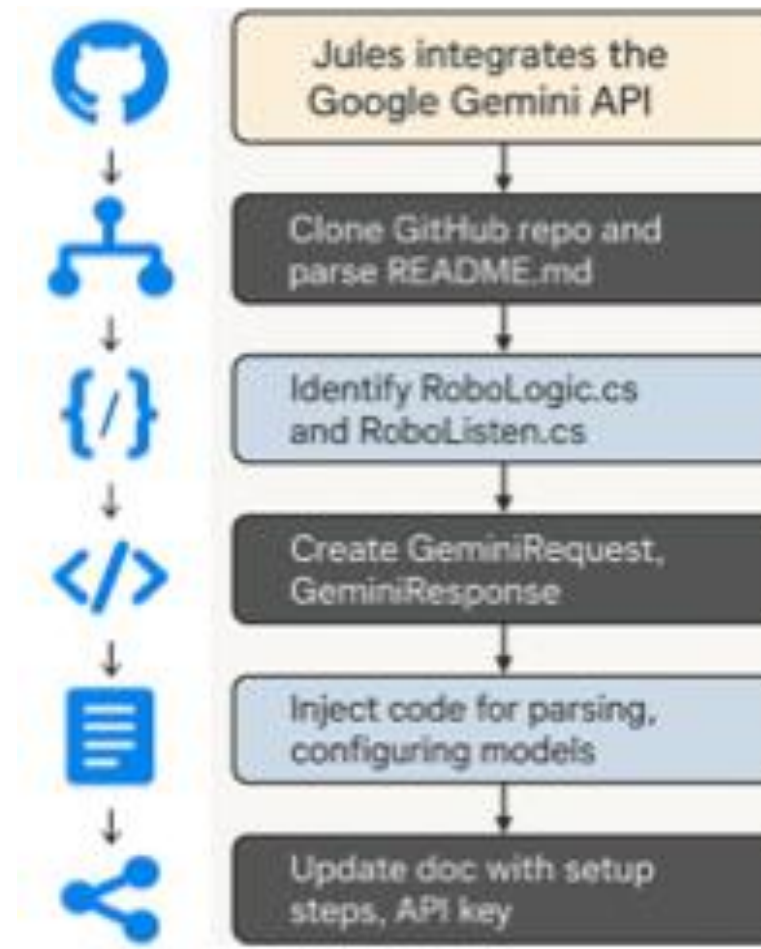
- Parse log files
- Count URLs
- Return computed frequencies
- Write unit tests
- Identify failing cases
- Refine code
- Generates documentation strings
- Produce human-readable API documentation

# Agentic frameworks – task examples

“Integrate the Google Gemini API into the R1 robot”.

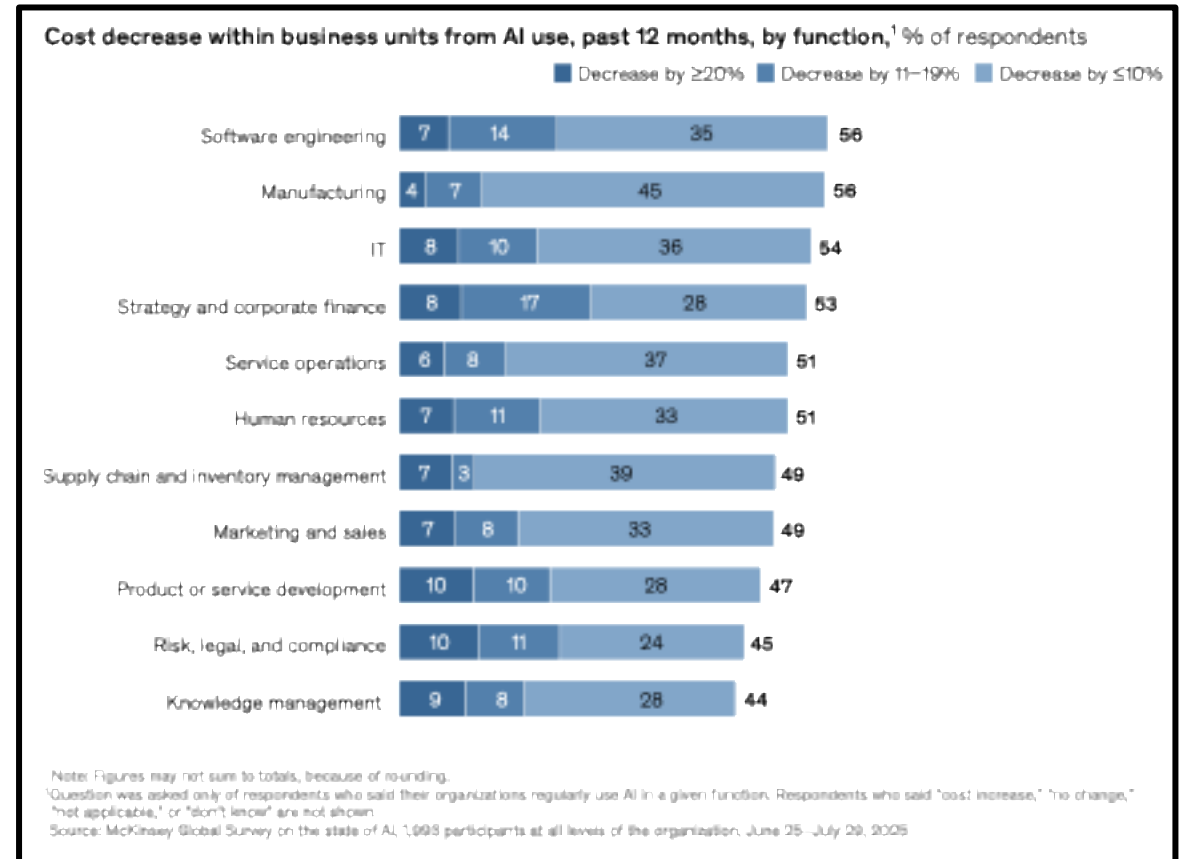
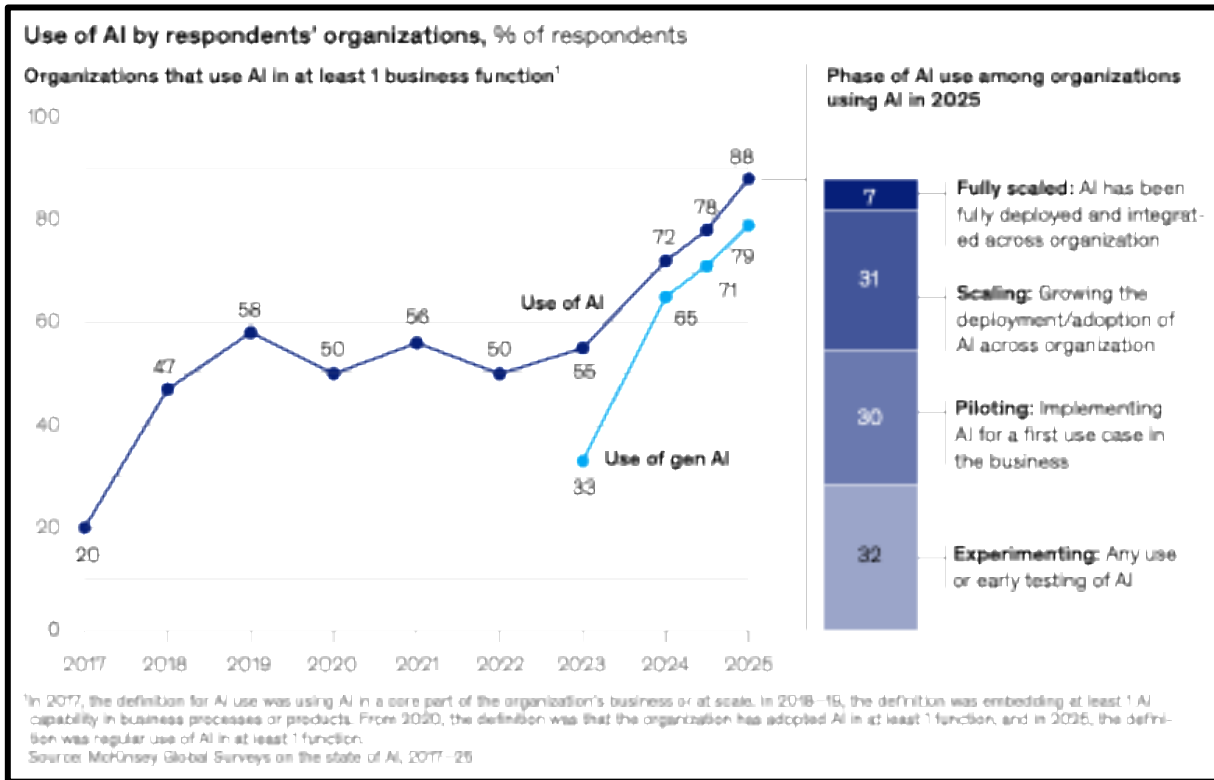
Fig. 4: Agentic Coding Example-API Integration with Jules. This flowchart illustrates how Jules autonomously integrates the Google Gemini API, executing tasks from repository analysis and code modification to documentation and Git operations demonstrating a complete, multi-step software development workflow typical of advanced agentic coding systems.

Agentic coding example. [7]



# Adoption of agentic frameworks

Still incipient but increasingly being piloted



Data captured from 25 June to 29 July 2025, 1,993 participants, 105 nations. [8]

# Adoption of agentic frameworks (cont.)

Viewed as promising approach

**AWS SUMMIT**  
EMEA ON-DEMAND HUB

[Session Details](#) [Take th](#)

## Enterprises are doubling down on Agents

<b>33%</b> of enterprise software apps will include agentic AI by 2028, up from less than 1% in 2024.	<b>15%</b> of day-to-day work decisions will be made autonomously through agentic AI by 2028.
--	--

Gartner, "Top strategic Technology Trends for 2025," October 2024

Gartner, "Top Strategic Technology Trends: agentic AI – The evc Experience", February 2025

Fullscreen

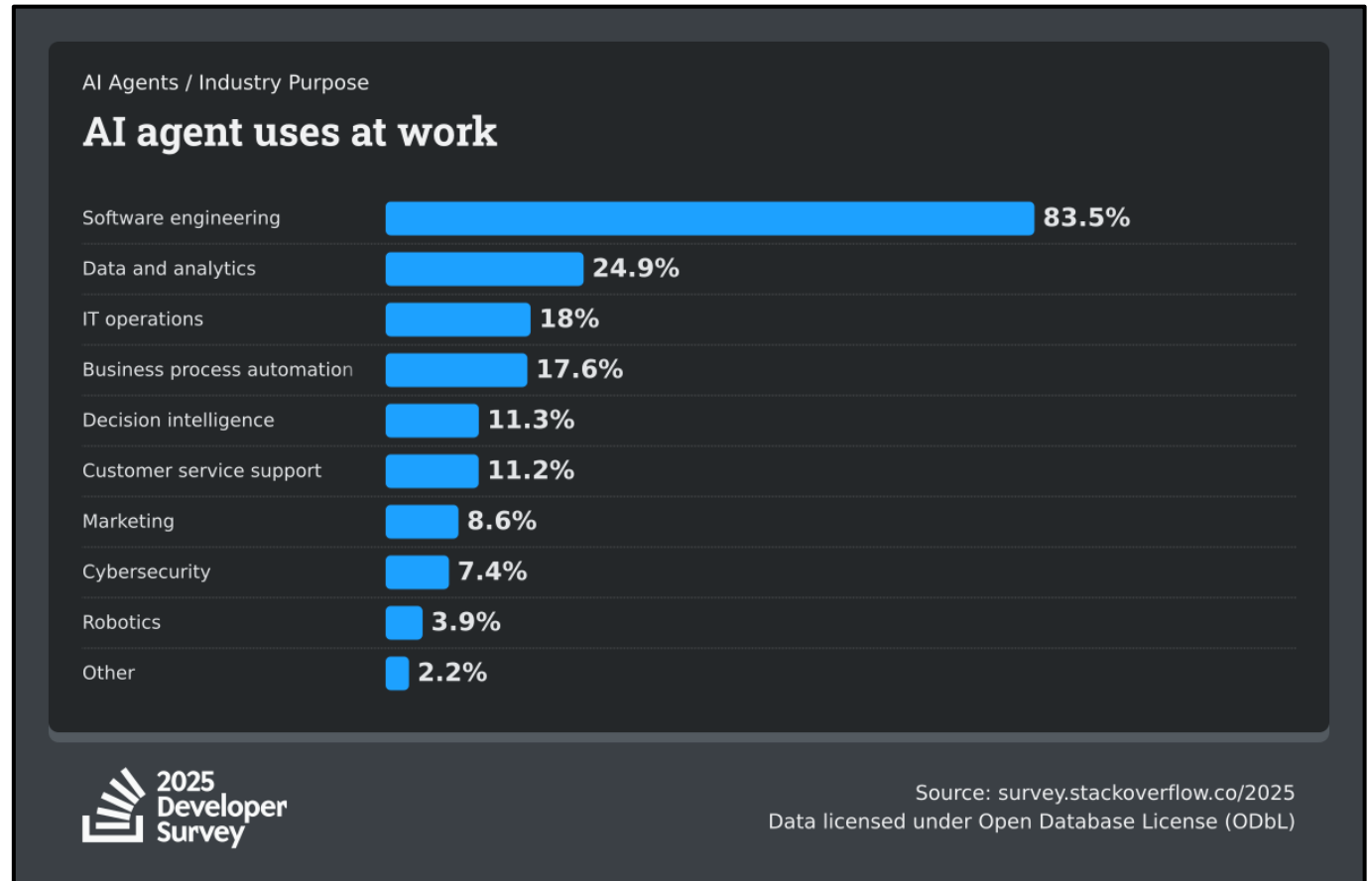
|| AWS LIVE © 2025 Amazon.com, Inc. or its affiliates. All rights reserved.

📺 🗂️ 🖥️ ⚙️ 🔄

AWS Summit London 2026  
<https://aws.amazon.com/events/summits/london/>

# Adoption of agentic frameworks (cont.)

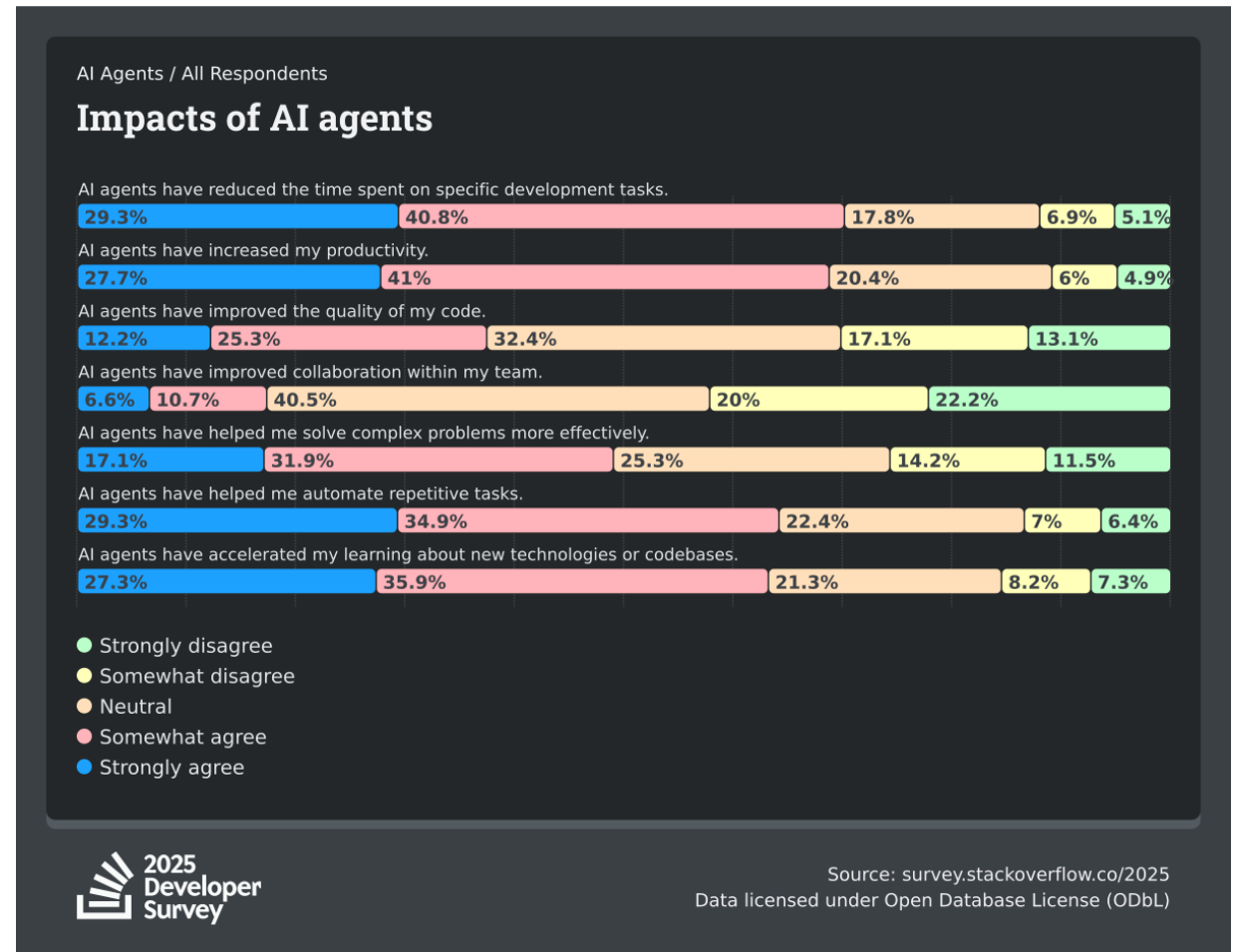
What **industry purposes** or **specific tasks** are you using AI agents in your development work?



Adoption of AI agents (n=12,301 – 25.1%). [4]

# Adoption of agentic frameworks (cont.)

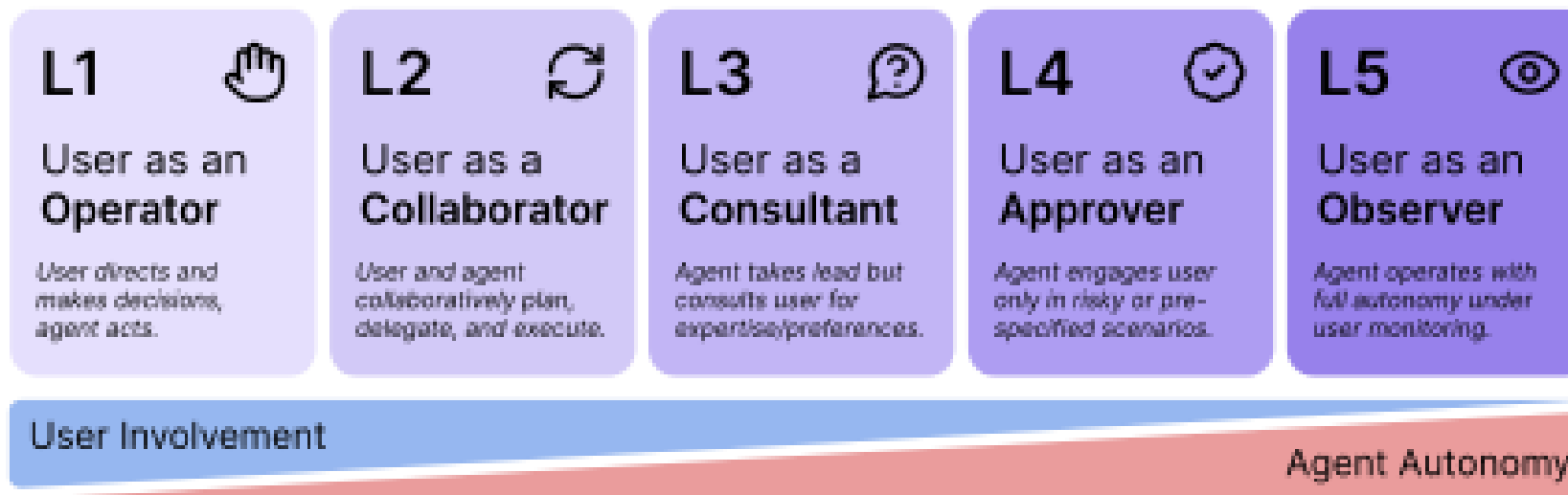
To what extent do you agree with the following statements regarding the **impact of AI agents** on your **work as a developer**?



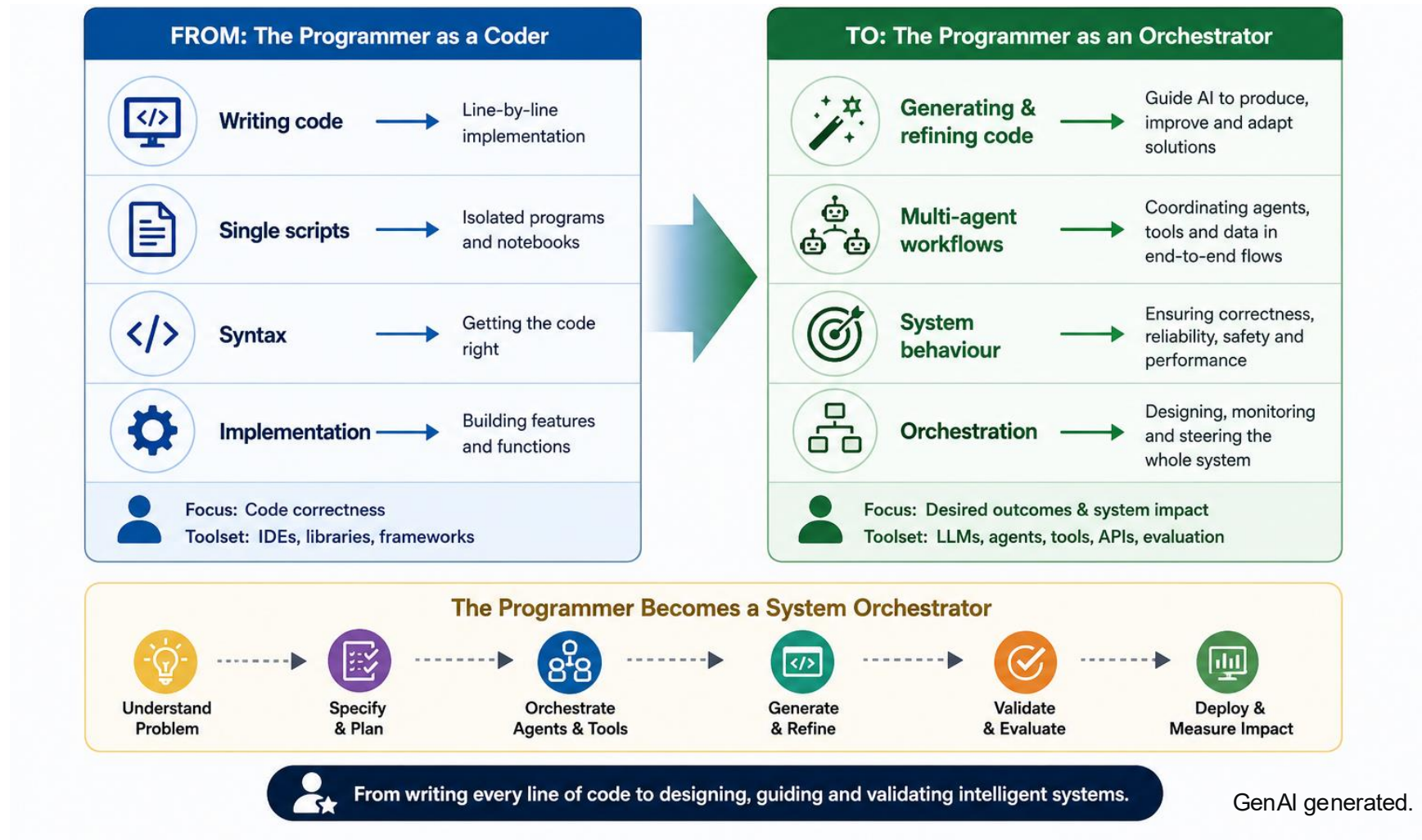
Adoption of AI agents (n=12,823 – 26.2%). [4]

# Agency levels

- **Agency**: the capacity to act intentionally
- **Autonomy**: the ability to operate without a human operator for a protracted period of time (robotics/AI contexts)



# New programming competence



# What's already happening in LSE classrooms



## GENIAL: When are **GEN**erative AI Tools a Catalyst for **L**earning?

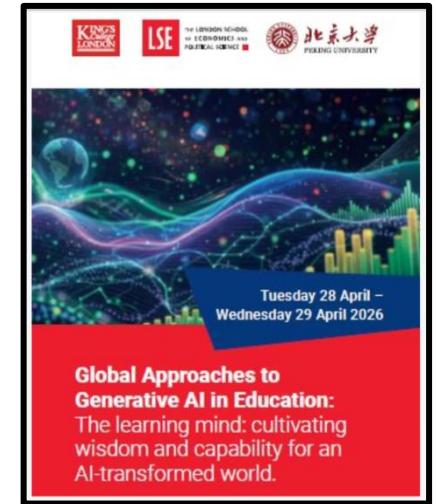
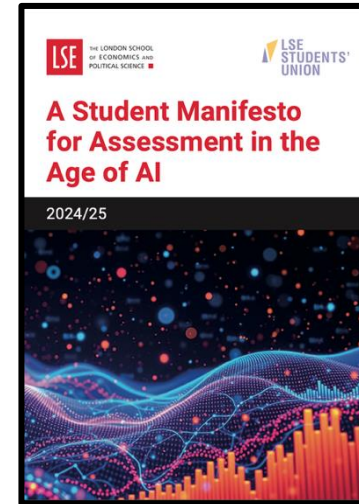
A Collaborative Focus Group

<https://lse-dsi.github.io/genial/>

- 1<sup>st</sup> LSE-funded project on GenAI in Education (2023-2024)
- Seven courses, 4 departments, 220 students
- AI-supported formative classroom exercises, including programming

## AI in education showcases

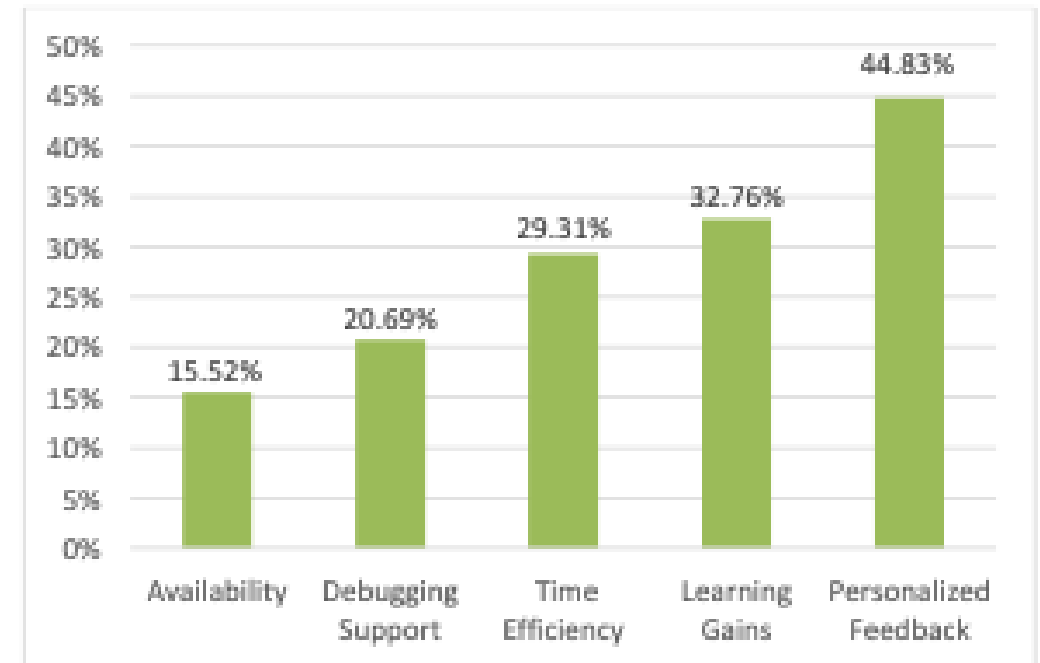
The Eden Centre hosted several sessions throughout the 2025/26 academic year. These 45-minute interactive meet-ups (hybrid attendance) provided an opportunity for colleagues to present their innovative use of AI in teaching, learning, assessment, or curriculum design and development in both qualitative and quantitative disciplines in the social sciences.



- AI-assisted coding and database modelling
- Agentic AI in behavioural intervention systems
- LLM as assignment
- AI-enhanced feedback

# Emerging pedagogical approaches

- **AI as tutor**
  - providing step-by-step feedback
  - acting as teaching assistants
- **AI as collaborator**
  - pair programming + debugging
  - iterative refinement
- **AI as object of reasoning**
  - students debugging AI-generated code
  - students revising tasks, implementation plans, and unit tests
- ... and many other reported use cases, surveys, and case-control studies



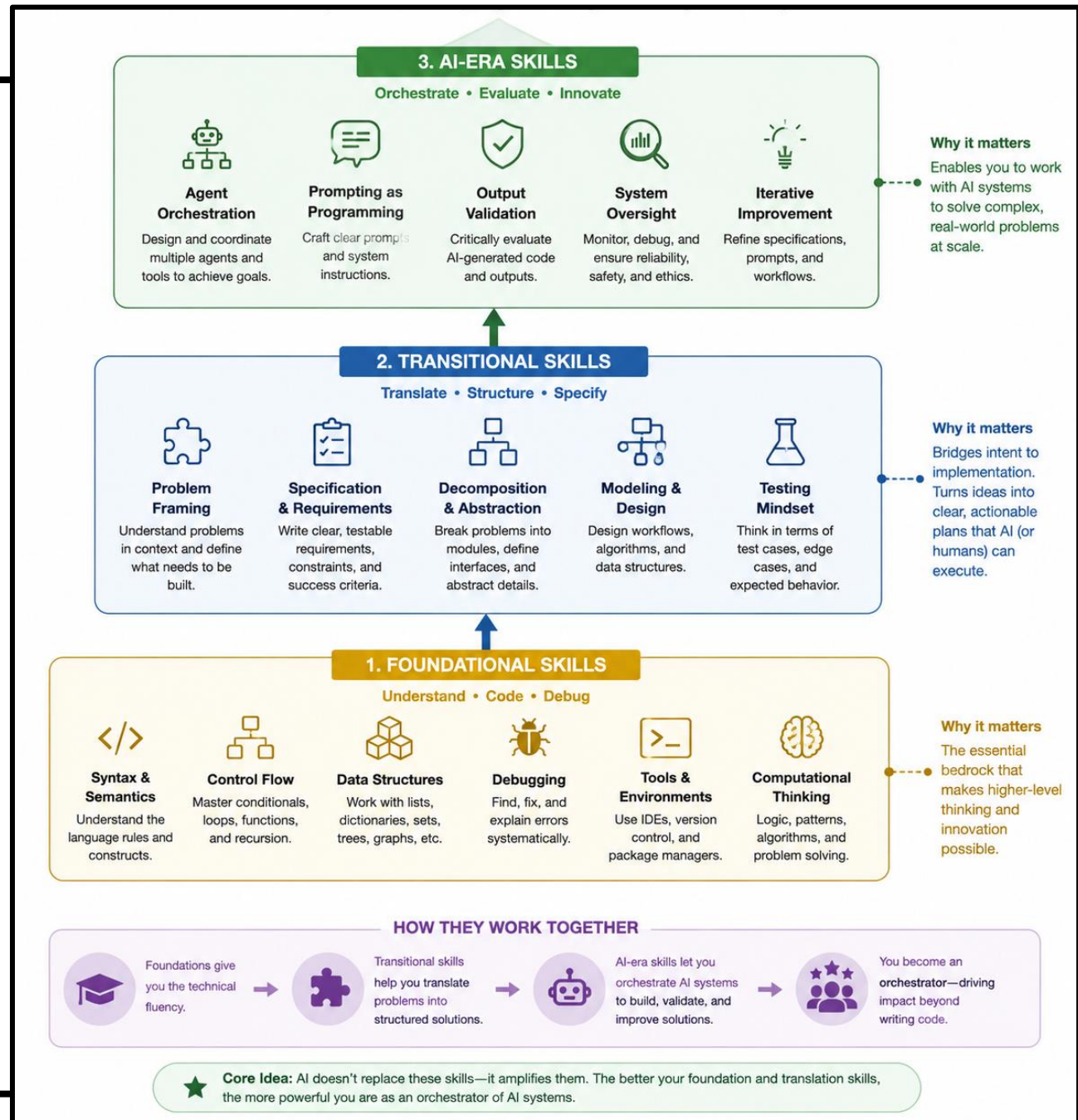
**Benefits of learning with AI agents.**

Analysis of 58 peer-reviewed studies published between 2022 and 2025 [10]

# New programming competence (expanded)

- Baseline **AI programming skills**
  - Foundational (non-negotiable)
  - Transitional
  - Agentic AI

GenAI generated.



# Specification-driven design (SSD)

Spec-driven development with ADK and Antigravity  
<https://codelabs.developers.google.com/sdd-adk-antigravity#2>

- **Spec-driven design** is about making technical decisions explicit, reviewable, and evolvable via a series of “**steering documents**” that guide agent behaviour and tasks.
- The workflow becomes:  
**Spec → Plan → Tasks → Code**
- Requirements can be specified via **EARS** (Easy Approach to Requirements Syntax):

**WHEN [condition/event]**  
**THE SYSTEM SHALL [expected behavior]**

```
sdd-adk-agents-agy/
├── .agents/
│   ├── workflows/ # SDD slash commands (/speckit.*) - manual trigger
│   │   ├── speckit.specify.md
│   │   ├── speckit.clarify.md
│   │   ├── speckit.plan.md
│   │   ├── speckit.tasks.md
│   │   ├── speckit.analyze.md
│   │   ├── speckit.implement.md
│   │   ├── speckit.checklist.md
│   │   └── speckit.constitution.md
│   ├── skills/ # Antigravity skills (loaded on demand, agent determined)
│   │   ├── mcp-toolbox-postgres/
│   │   │   └── SKILL.md # MCP Toolbox config skill
│   │   └── repo-research/
│   │       └── SKILL.md # Repo analysis skill
│   └── rules/ # Always-active context
├── .specify/ # spec-kit SDD templates and memory
│   ├── memory/constitution.md
│   └── templates/
├── scripts/
├── restaurant_concierge/ # ADK agent package
│   ├── __init__.py
│   ├── agent.py # LlmAgent + ToolContext tools + Toolbox integration
│   └── .env # Vertex AI configuration
├── server.py # FastAPI server wrapping the agent
├── tools.yaml # MCP Toolbox tool definitions
├── scripts/ # Setup scripts
└── pyproject.toml
```

# Bottlenecks

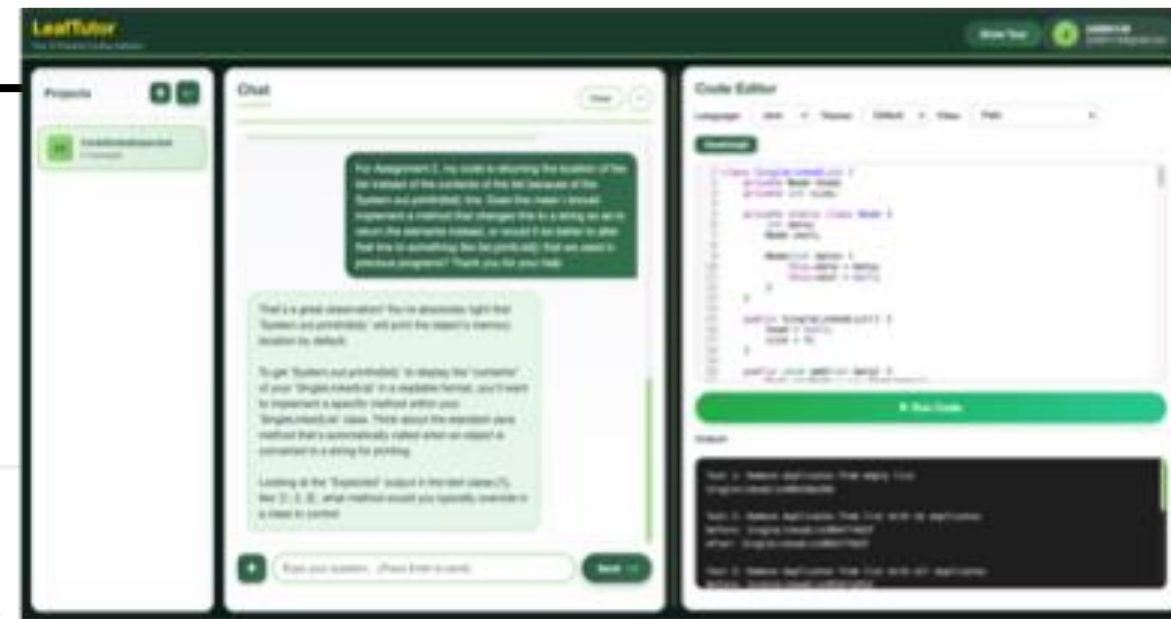
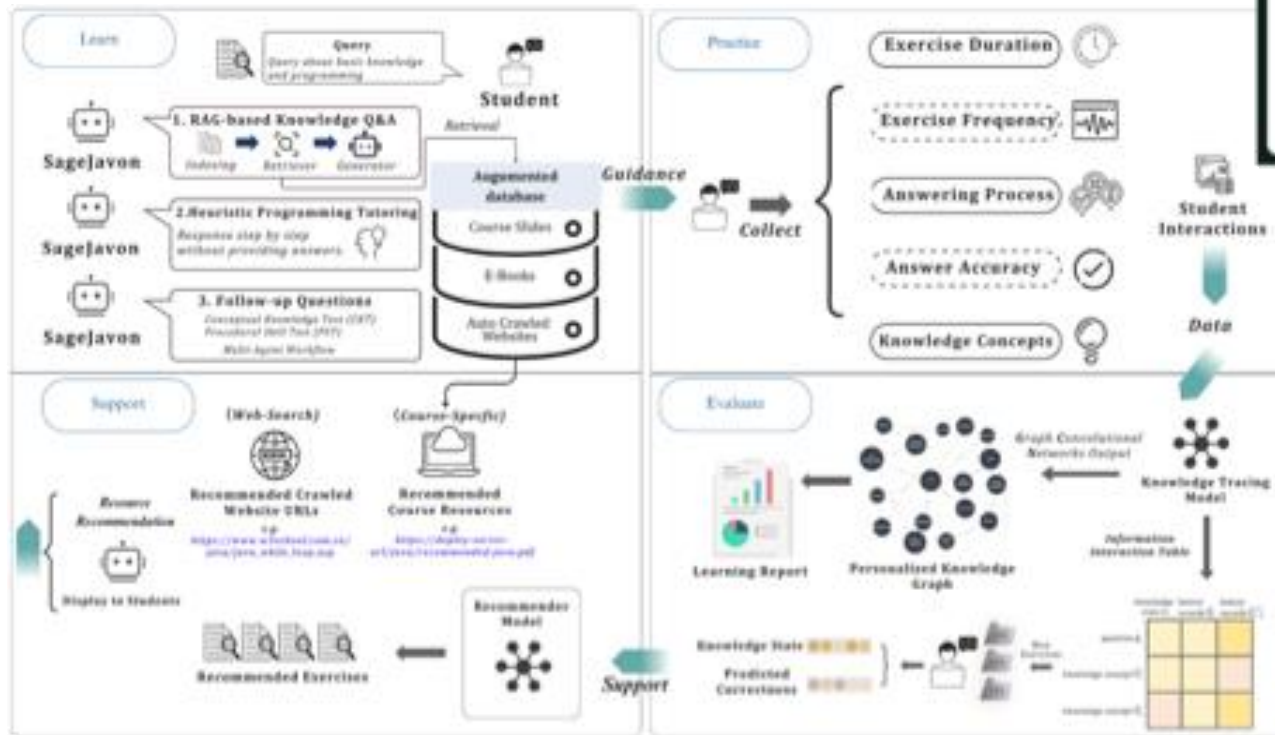
- Academic **“technical debt”**
    - **Messy** use cases (including data)
    - Problem specification under **“system thinking”**:
      - **logical**: what needs to be developed
      - **analytical**: high-level components, interaction
      - **computational**: modules, rules, data flow
      - **procedural**: optimal execution, best practices
    - Access to tooling + **agentic framework literacy**
    - **Validation**
-

# What have we heard today?

Speaker	Skills / Capabilities / Approaches
<b>Tatiana</b>	<b>AI Research Agents:</b> idea => tree-based experimentation => write up; well-defined specification; AIRSbench triplets (problem, dataset, metric) mapped to Agent (research, plan, ...); 3-tier paradigm (validator, discoverer, innovator)
<b>Dimitrios</b>	<b>GenAI in Finance:</b> ensemble of models approving card transactions; best case for agents is coding (known ROIs); agents mostly for internal processes, not customer-facing yet
<b>David</b>	<b>GenAI and Creativity:</b> brainstorming, ideation, writing; divergent thinking, problem solving, creative writing; novelty and usefulness; discussion structure, iterative refinement, and interaction design; creativity as engineering problem
<b>Yi Wei</b>	<b>Lesion segmentation:</b> guided model (mouse click), predictive/interpretable features/biomarkers; 2 years for full-automatic models; agentic systems for feature recommendation (CT scan and image segmentation)
<b>Niek</b>	<b>Multicalibration:</b> optimisation of predictive performance metrics; manual specification of segments
<b>Don</b>	<b>Repository automation:</b> code “magicians”; Continuous AI; “constraint-guided” programming; performance engineering; formal verification
<b>Chengchun</b>	<b>LLM reasoning:</b> code benchmarks; prompting approaches; reward model

# What we can do today

- **AI tutor integration**  
(especially for first-time programmers)



Code Not As Expected Error. Interaction demonstrating LeafTutor diagnosing a non-error output issue. [11]

Fig. 2. The architecture of the SageJavon Platform, comprising four main modules: [Learn], [Practice], [Evaluate], and [Support]. (i) Upon entering SageJavon, students can first use the [Learn] module to access the Heuristic Dialogue Learning Module for explanations of Java-related concepts. (ii) If students identify gaps in their knowledge, they can then use the [Practice] module, which recommends targeted exercises for practice. (iii) The [Evaluate] module assesses the student's code based on their exercise history and predicts their knowledge level. (iv) After determining the student's knowledge level, the [Support] module provides a comprehensive learning diagnosis and recommends tailored resources to facilitate further improvement.

Architecture of SageJavon. [11]

# What we can do today

- AI debugging labs

The screenshot displays the HYPOCOMPASS interface, which is divided into several sections:

- Problem: first\_num\_greater\_than (A):** A text box containing the problem description: "Write a Python function first\_num\_greater\_than(numbers\_list, key) that takes a list of integers (numbers\_list) and an integer key (key), and returns the first number in the list that is greater than the key. If there is no number greater than the key, then you should return None." Below this is a blue box with instructions: "Now you are chatting with a student. Please explain to them why their code is wrong by selecting the right explanation from the list. If you are right, the student will fix their code accordingly!"
- Test Suite Development (B):** A section for creating test cases. It includes an "Add Test Case" form with "Input" and "Expected output" fields, and an "Add Test Group" form with an "Enter test group name" field. A blue button labeled "Evaluate Test Suite" is positioned below these forms. Below the forms is a table titled "Your Test Cases" with columns for "Passed?" and "Actual Output".
- Student's Current Code (D):** A code editor showing the student's current Python code for the function.
- View Code Differences (F):** A section for comparing code, showing two versions of the function with differences highlighted in red and green.
- Office Hour Queue (C):** A vertical queue of student avatars and names: Bob: Start helping, Chelsea: Waiting, and Dave: Waiting. A circular arrow icon (E) indicates the queue's flow.
- Office Hour! (E):** A chat interface showing a conversation between the user and an LLM-agent. The chat includes messages from the user and the agent, such as "Increase the test case given that [1, 2, 3, 4] is supposed to output 2. However, the given behavior is unexpected in the program's logic here." and "Oh, I see my code got the test case wrong. Could you explain what's wrong with my code?". There are buttons for "Yes", "No", and "Explain more".

Figure 1: In HYPOCOMPASS, given a problem description (A), a student user (in the role of a Teaching Assistant) needs to compile a test suite (B) and assist multiple LLM-simulated agents (e.g., *Bob*, *Chelsea*, *Dave*) in an Office Hour Queue (C) through a chat interface (E). Each LLM-agent acts as a novice seeking help with a buggy solution (D) and provides feedback to the user (F).

# • Specification-first assignments (LSE ST207 – Databases)

Start new conversation in Google\_AGY

Let's work on a database modelling and programming exercise. The context is a company database with information on employees and their dependants, projects and departments. Employees work for a department, have a supervisor, who is also an employee, and may have one or more dependants. Each department controls a number of projects, where one or more employees work. Each department is managed by one employee. Refer to the attached ER model for more details, but use the CSV files to be uploaded to the project repository for the real table and column names, along with relationships. Your task is to help me creating a Python notebook with code to create a "company" database in sqlite, define the database schema for all tables and relationships, import the data from the csv files and write SQL queries to answer the following questions: 1) group projects per department and list all employees working in each project; 2) find the top 5 employees based on the number of projects they work for; and 3) generate a list of all supervisors and their supervisees. When we agree on the final code, we will generate unit tests for the database. We will open a browser to access Google Colab and import the code and the datasets.

## 1. Database scenario



Revise the implementation plan to ensure that: 1) Mgr\_ssn (FK) points to Ssn (so it's a self-reference for the table); 2) data insertion will respect all FK rules and load the data correctly; and 3) when showing SQL results, use appropriate tabular format so the results are clear. Regarding the unit test and verification plan: FK constraints are likely to be checked during data loading and join operations, so there's no need to check them again. How do you plan to check for query results correctness? Explain. Also, we will run this code on Google Colab, so the python3 command to run the unittest may not work. Revise as well.

## 4. Refinement

## 2. Planning and analysis

Running background command

```
~/.../Google_AGY $ find . -name "*.csv"
```

Checked command status

Thought for <1s

Checked command status

Thought for 2s

Edited Task

**Planning and Analysis**

Analyzed the CSV files and ER model. Developed an implementation plan for creating the SQLite database, importing data, and writing queries.

Files Edited

- Implementation Plan

Progress Updates

- Analyzing ER model and searching for CSV files
- Requesting user review of the implementation plan

I have analyzed the ER model and the provided CSV files. I've created an implementation plan for the database modelling exercise. Please review the plan, and once approved, I will proceed with creating the SQLite database, importing the data, and writing the SQL queries and unit tests.

Implementation Plan

Implementation plan for the database modeling and programming exercise.

## 3. Implementation plan

Implementation Plan - Database Modelling and Programming Exercise

This plan outlines the steps to create a SQLite database for a company, import data from CSV files, and implement SQL queries and unit tests.

**Proposed Changes**

**Database Setup**

- Create `company.db` using SQLite.
- Define schema based on CSV file structures:
  - `employee`: `Fname`, `Minit`, `Lname`, `Ssn` (PK), `Bdate`, `Address`, `Gender`, `Salary`, `Mgr_ssn` (FK), `Dno` (FK)
  - `department`: `Dname`, `Dnumber` (PK), `Super_ssn` (FK), `Super_startdate`
  - `dept_locations`: `Dnumber` (FK), `Dlocation` (PK part)
  - `project`: `Pname`, `Pnumber` (PK), `Plocation`, `Dnum` (FK)
  - `works_on`: `Essn` (FK), `Pno` (FK), `Hours`
  - `dependent`: `Essn` (FK), `Dependent_name` (PK part), `Sex`, `Bdate`, `Relationship`

**Data Import**

- Use `pandas` or built-in `csv` module to read CSV files.
- Insert data into SQLite tables.

**SQL Queries**

- Query 1:** Group projects by department and list employees in each.
  - Join `department`, `project`, `works_on`, and `employee`.
- Query 2:** Top 5 employees by number of projects.
  - Group `works_on` by `Essn`, count projects, join `employee` for names.
- Query 3:** Supervisors and supervisees.
  - Self-join `employee` on `Mgr_ssn = Ssn`.

- **Specification-first assignments**  
(LSE ST207 – Databases – *cont.*)

5. Database creation

**Database Schema Definition**  
Planning and analysis completed. Starting the execution phase by creating the SQLite database and defining the schema for all tables, including the self-referencing manager relationship.

Files Edited  
• setup\_db.py Task

Progress Updates Collapse all ▾  
1 Creating SQLite database and defining schema with self-referencing FKs

Edited Task View  
Edited • setup\_db.py +132 -0 △ 1

Run command?  
~/.../Google\_AGY \$ python3 /Users/m.e.barreto/Tools/Google\_AGY/setup\_db.py

Ask every time ^

Reject Run ^

1 Step Requires Input Expand <

6. Data loading

both plans seem fine. Proceed with the implementation.

**Database Schema Definition**  
Database schema defined. Encountered a timeout during initial data import attempt. Now re-running the import with direct sqlite3 calls to ensure completion and correct foreign key handling.

Files Edited  
• import\_data.py • setup\_db.py Task

Progress Updates Expand all <  
1 Creating SQLite database and defining schema with self-referencing FKs  
2 Importing data from CSV files and verifying integrity

**Database Schema Definition and Data Import**  
Initial database setup encountered issues with missing tables and timeouts. Re-running the initialization and import together to ensure `company.db` is correctly populated and foreign keys are verified.

Files Edited  
• setup\_and\_import.py

Progress Updates Collapse all ▾  
1 Re-initializing database and importing data using direct sqlite3 calls to ensure correct table creation and loading.

Edited Task  
Edited • setup\_and\_import.py +142 -0 △ 4

Ran background command Open ↗  
~/.../Google\_AGY \$ python3 /Users/m.e.barreto/Tools/Google\_AGY/setup\_and\_import.py

Imported data into employee  
Imported data into department

```
> setup_and_import.py
1 import sqlite3
2 import csv
3 import os
4
5 def create_schema(cursor):
6     # Enable foreign keys
7     cursor.execute("PRAGMA foreign_keys = ON;")
8
9     # Drop tables if they exist
10    tables = ['dependent', 'works_on', 'pr
11    'employee']
12    for table in tables:
13        cursor.execute(f"DROP TABLE IF EXI
14
15    # Create tables
16    cursor.execute("""
17    CREATE TABLE employee (
18        Fname TEXT NOT NULL,
19        Minit TEXT,
20        Lname TEXT NOT NULL,
21        Ssn TEXT PRIMARY KEY,
22        Bdate TEXT,
23        Address TEXT,
24        Gender TEXT,
25        Salary REAL,
26        Mgr_ssn TEXT,
27        Dno INTEGER,
28        FOREIGN KEY (Mgr_ssn) REFERENCES employee(Ssn),
29        FOREIGN KEY (Dno) REFERENCES department(Dnumber)
30    );
31    """)
32
33    cursor.execute("""
34    CREATE TABLE department (
35        Dname TEXT NOT NULL,
36        Dnumber INTEGER PRIMARY KEY,
37        Super_ssn TEXT NOT NULL,
38        Super_startdate TEXT,
39        FOREIGN KEY (Super_ssn) REFERENCES employee(Ssn)
40    );
41    """)
```

Shouldn't this be OFF instead of ON to allow for creating all tables with no FK errors?

Cancel Add Comment

Artifact for database creation and data loading

- **Specification-first assignments** (LSE ST207 – Databases – *cont.*)

7. Query implementation

```

> run_queries.py
1 import sqlite3
2 import pandas as pd
3
4 def run_queries(db_path='/Users/m.e.barreto/Tools/Google_AGY/company.db'):
5     conn = sqlite3.connect(db_path)
6
7     print("---- Query 1: Projects per Department and Employees ----")
8     query1 = """
9     SELECT
10        d.Dname AS Department,
11        p.Pname AS Project,
12        e.Fname || ' ' || e.Lname AS Employee
13    FROM department d
14    JOIN project p ON d.Dnumber = p.Dnum
15    JOIN works_on w ON p.Pnumber = w.Pno
16    JOIN employee e ON w.Essn = e.Ssn
17    ORDER BY d.Dname, p.Pname;
18    """
19    df1 = pd.read_sql_query(query1, conn)
20    print(df1.to_string(index=False))
21    print("\n")
22
23    print("---- Query 2: Top 5 Employees by Number of Projects ----")
24    query2 = """
25    SELECT
26        e.Fname || ' ' || e.Lname AS Employee,
27        COUNT(w.Pno) AS Project_Count
28    FROM employee e
29    JOIN works_on w ON e.Ssn = w.Essn
30    GROUP BY e.Ssn
31    ORDER BY Project_Count DESC, Employee ASC
32    LIMIT 5;
33    """
34    df2 = pd.read_sql_query(query2, conn)
35    print(df2.to_string(index=False))
36    print("\n")
37
  
```

Artifact for database query

8. Test of database queries

**Query Implementation**  
Database schema defined and data successfully imported from all CSV files. Foreign key integrity verified. Moving to query implementation phase.

Files Edited  
run\_queries.py Task

Progress Updates Collapse all

1 Writing and executing SQL queries for project grouping, top employees, and supervisor hierarchy.

Edited Task View

Edited run\_queries.py +54 -0 1

Run command?

```
~/.../Google_AGY $ python3 /Users/m.e.barreto/Tools/Google_AGY/run_queries.py
```

Ask every time ^ Reject Run

1 Step Requires Input Expand

Waiting...

**Query Implementation**  
Database schema defined and data successfully imported from all CSV files. Foreign key integrity verified. Moving to query implementation phase.

Files Edited  
run\_queries.py Task

Progress Updates Collapse all

1 Writing and executing SQL queries for project grouping, top employees, and supervisor hierarchy.

Edited Task View

Edited run\_queries.py +54 -0 1

Ran background command Open

```
~/.../Google_AGY $ python3 /Users/m.e.barreto/Tools/Google_AGY/run_queries.py
```

Josh Zell	Jenny Vos
Kate King	Arnold Head
Kate King	Gerald Small
Kate King	Helga Pataki
Kim Grace	Jeff Chase
Lyle Leslie	Billie King
Lyle Leslie	Jon Kramer
Naveen Drew	Carl Reedy
Sammy Hall	Red Bacher

Ask every time ^ Exit code 0

## • Specification-first assignments (LSE ST207 – Databases – cont.)

### Unit Testing and Verification

SQL queries implemented and results displayed. Verified project grouping, top employees, and supervisor hierarchy. Data counts and foreign keys were previously verified during import. Now creating formal unit tests for automated validation.

#### Files Edited

test\_db.py Task

#### Progress Updates

Expand all

- 1 Creating and running Colab-compatible unit tests to verify query correctness and data counts.

### Final Presentation

Unit tests passed successfully. Database schema, data import, and queries have been verified for correctness and integrity. Now consolidating all work into a single Jupyter Notebook and preparing the final walkthrough.

```

test_db.py
1 import sqlite3
2 import unittest
3 import pandas as pd
4 import sys
5
6 class TestCompanyDatabase(unittest.TestCase):
7     @classmethod
8     def setUpClass(cls):
9         cls.db_path = '/Users/m.e.barreto/Tools/Google_AGY/company.db'
10        cls.conn = sqlite3.connect(cls.db_path)
11
12    @classmethod
13    def tearDownClass(cls):
14        cls.conn.close()
15
16    def test_table_counts(self):
17        """Verify that record counts match expected CSV row counts (excluding
18        header)"""
19        cursor = self.conn.cursor()
20
21        expected_counts = {
22            'employee': 40, # 41 lines in file - 1 header
23            'department': 6, # 7 lines - 1 header
24            'project': 11, # 12 lines - 1 header
25            'works_on': 48, # 49 lines - 1 header
26            'dependent': 11 # 12 lines - 1 header
27        }
28
29        for table, count in expected_counts.items():
30            cursor.execute(f"SELECT COUNT(*) FROM {table}")
31            actual_count = cursor.fetchone()[0]
32            self.assertEqual(actual_count, count, f"Count mismatch for {table}")
33
34    def test_query2_correctness(self):
35        """Verify Query 2: Top 5 employees by project count."""
36        query = """
37        SELECT e.Ssn, COUNT(w.Pno) as cnt
38        FROM employee e JOIN works_on w ON e.Ssn = w.Essn
39        GROUP BY e.Ssn ORDER BY cnt DESC, e.Lname ASC LIMIT 5;
40        """
41        df = pd.read_sql_query(query, self.conn)

```

Artifact for unit tests

## 10. Consolidation of all artifacts after successful unit test validation

I don't think you can simply cat all .py files into a final\_consolidation.txt file. I need a Python Jupyter notebook (extension .ipynb) and you need to remove any duplicate commands related to opening and closing database connections. Structure the code so all steps are properly separated in sections and make sure the entire code will work when imported to Google Colab. Replace all file paths with /content for compatibility.

### Final Presentation

Consolidated all code into a well-structured .ipynb file, removing duplicates, and ensuring compatibility with Google Colab by using /content paths. Created a final walkthrough for the project. Task completed.

#### Files Edited

Walkthrough company\_exercise.ipynb

#### Progress Updates

Expand all

- 1 Creating the Jupyter Notebook for Google Colab with consolidated and structured code.
- 2 Notifying the user of the completed notebook and walkthrough.

I have successfully consolidated the database exercise into a single Jupyter Notebook (company\_exercise.ipynb) and prepared a final walkthrough. The notebook is structured with clear sections (Schema, Import, Queries, Testing) and uses /content paths for full compatibility with Google Colab. I've also incorporated your feedback by setting the initial FK PRAGMA to OFF and removing duplicate database connections. Please review the final results.

company\_exercise.ipynb

Open

Walkthrough

Open

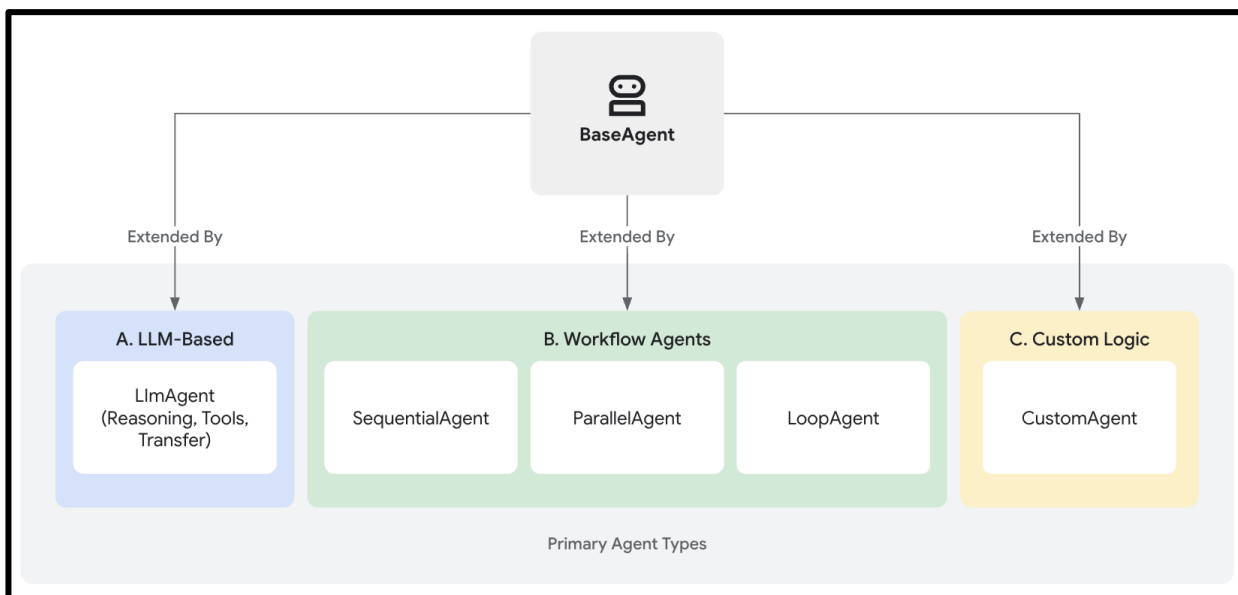
Final walkthrough of the database modelling exercise.

## 9. Definition of unit tests



# What we can do today

- **Multi-agent machine learning projects**  
(ongoing implementation for STxxx modules)



```
mle_pipeline_agent = agents.SequentialAgent(
    name="mle_pipeline_agent",
    sub_agents=[
        initialization_agent_module.initialization_agent,
        refinement_agent_module.refinement_agent,
        ensemble_agent_module.ensemble_agent,
        submission_agent_module.submission_agent,
    ],
    description="Executes a sequence of sub-agents for solving the MLE task.",
    after_agent_callback=save_state,
)

# For ADK tools compatibility, the root agent must be named `root_agent`
root_agent = agents.Agent(
    model=os.getenv("ROOT_AGENT_MODEL", "gemini-2.5-flash"),
    name="mle_frontdoor_agent",
    instruction=prompt.FRONTDOOR_INSTRUCTION,
    global_instruction=prompt.SYSTEM_INSTRUCTION,
    sub_agents=[mle_pipeline_agent],
    generate_content_config=types.GenerateContentConfig(temperature=0.01),
)
```

Event 5 of 5

Request: Greeter

Response: Coordinator, TaskExecutor, google\_search

content: { parts: [ { functionResponse: { id: 'adk-b89d81f3-2228-45ca-8022-c494d64b6ebb', name: 'transfer\_to\_agent', response: { result: null } } } ] }

role: "user"

InvocationId: "e-6143b271-56fd-406b-bc16-2f1fc8796020"

author: "Greeter"

actions: { stateDelta: {}, artifactDelta: { transferToAgent: "TaskExecutor", requestedAuthConfigs: {}, requestedToolConfirmations: {} } }

id: "c969f51d-b0d7-429e-8f5a-4ecc2e4557e3"

timestamp: 1767783189.004373

title: "functionResponse:transfer\_to\_agent"

SESSION ID: 59af099d-cc8f-454b-a55f-7e8dbab3664a

hi

transfer\_to\_agent

transfer\_to\_agent

Hello! I'm your Greeter assistant. How can I help you today?

what is the current time in london?

transfer\_to\_agent

transfer\_to\_agent

Type a Message...



# Closing thoughts

- **Reposition** programming education around **reasoning, specification, orchestration, and validation** of AI agent-mediated systems

## If AI Can Code, Why Are Big-Tech Founders Coding Again?

Zuckerberg moved his desk. Brin assembled a strike team. Vembu gave up the CEO title. Something deeper than nostalgia is driving the founders back to the terminal.

Faisal Haque. [12]

---

---

# References

- [1] BBC Bitesize. **Introduction to programming.** [\[link\]](#)
  - [2] M. Wyss. **Should You Still Learn to Code in 2026?** [Medium](#), February 2026
  - [3] L. Cress. **'Vibe coding' named word of the year by Collins Dictionary.** [BBC News](#), November 2025.
  - [4] Stack Overflow. **2025 Developer Survey.** <https://survey.stackoverflow.co/2025>
  - [5] J. Wiesinger, P. Marlow, V. Vuskovic. **AI Agents.** Google, September 2024
  - [6] H. Wang et al. **AI Agentic Programming: A Survey of Techniques, Challenges, and Opportunities.** September 2025, <https://arxiv.org/abs/2508.11126v2>
  - [7] R. Sapkota et al. **Vibe coding vs Agentic coding: Fundamentals and Practical Implications of Agentic AI.** May 2025. <https://arxiv.org/abs/2505.19443v1>
  - [8] McKinsey. **The state of AI in 2025: agents, innovation, and transformation,** November 2025.
  - [9] K. Feng et al. **Levels of autonomy for AI agents.** July 2025. <https://arxiv.org/abs/2506.12469v2>
  - [10] H. Zhao et al. **SageJavon: A scalable AI tutor for personalized programming learning.** Information Processing and Management, Volume 63, Issue 5, 2026, <https://doi.org/10.1016/j.ipm.2025.104605>
  - [11] M. Bochard et al. **LeafTutor: An AI Agent for Programming Assignment Tutoring.** December 2025. <https://arxiv.org/abs/2601.02375v1>
  - [12] Faisal Haque. **If AI Can Code, Why Are Big-Tech Founders Coding Again?** [Medium](#), April 2026
-

blank

---