

Guiding novice Python programmers to navigate learning to code in an era of generative Artificial Intelligence



Imelda Smit

30 September 2026



Action Learning Action
Research Conference 2026
"Action Learning and Action
Research for Sustainable
and Inclusive Futures"
29 - 30 September 2026
Melbourne, Australia

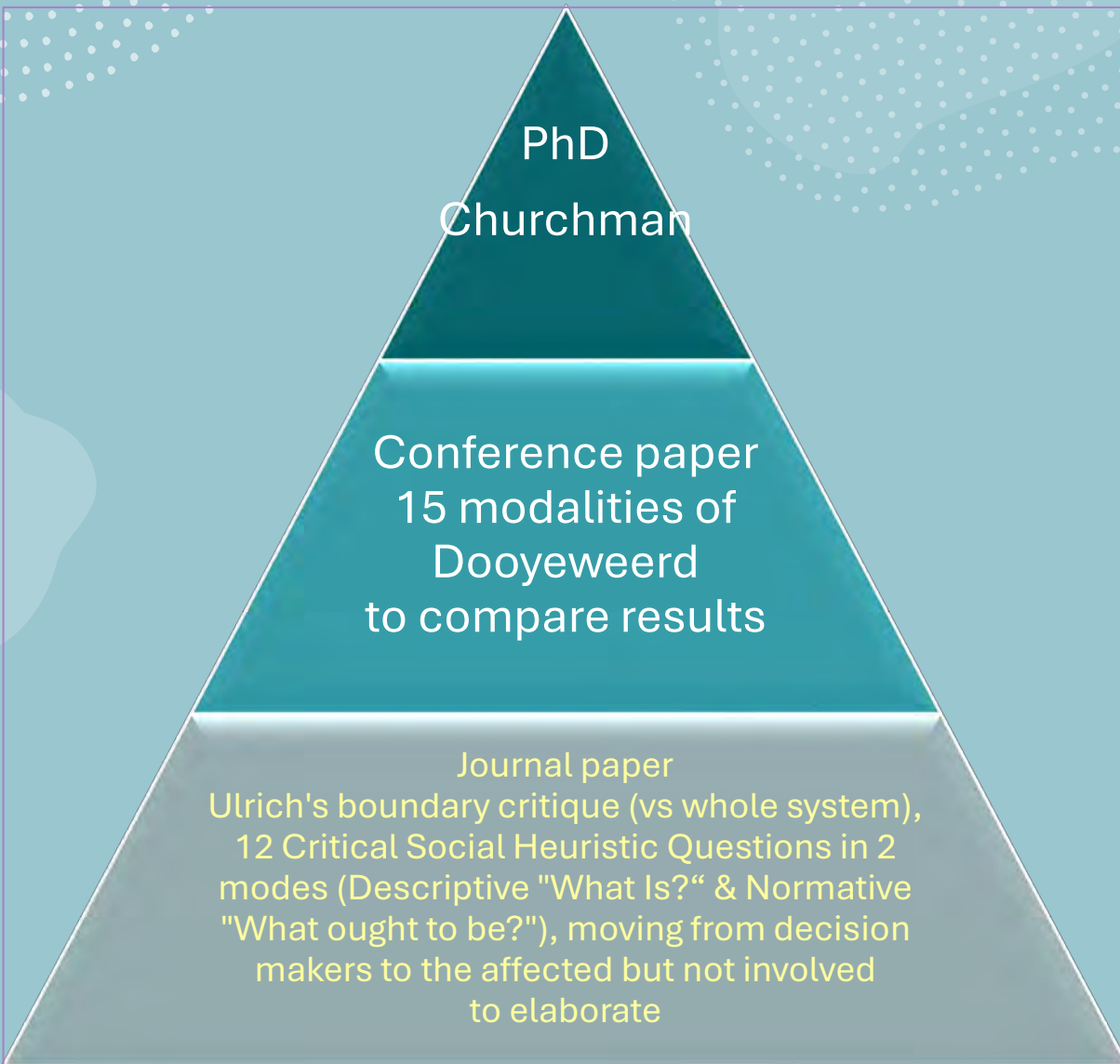
Discussion items

1. Methodology – Action Research (AR)
2. Focus & recent approaches used
3. New context
4. Suggested approach
5. Building on this work
6. References
7. Questions & Suggestions

Methodology – Action Research (AR)

Five-Stage AR iterative cycle
by Susman & Evered (1978)





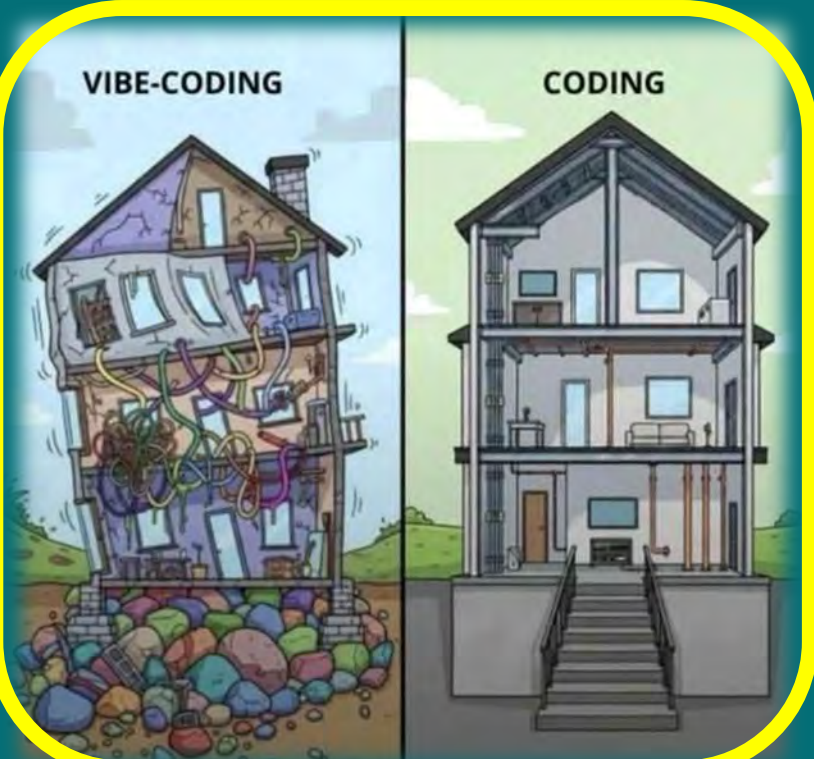
Churchman (1968), Dooyeweerd (1984), Ulrich (2005)

1

Diagnosing

Recent approaches,
different context

New context



4 Campuses: DC,
MC, PC, VC with
their stark
differences

Vanderbijlpark
Campus:
challenges

My research focus
was plagiarism

The genAI thing;
plagiarism, yes, but
now a more vicious
animal altogether

NWU view on genAI
use, AIITSA tool

My issues with
NWU view and tools

Churchman's approach

Consideration

→ Total system objectives

The overarching goals and overall performance measures of the entire system

Learn to code;

(1) Translating a problem into logic (**design**)

(2) Translating logic into expression (**coding**)

(3) Testing–Execution (run) & Debugging–Diagnosis (fix). Overcoming "The Wall"; a massive shift in how a person handles failure, the novice must develop the cognitive resilience to face continuous "**red error text**" without panicking, shutting down, or surrendering their autonomy to an AI oracle. Therefore, NOT to revert to an easy way out.

Consideration

→ System's environment

The external conditions, constraints, and factors outside the system's direct control that affect its performance

Our environment:

- Course Timelines & Deadlines – **Timetable**
- Grading Criteria and Metrics – **CodeGrade**
- Pedagogical Delivery – **Book, The Team** (Lecturers-as team-teachers, Student Assistants, Tutors, Supplemental Instructors)
- Strict Compiler & Syntax Rules – **Python** Language Syntax and IDEs
- Pervasive GenAI Tools – **AI availability** an inescapable environmental pressure
- Industry Job Expectations – Determine **skills** to be acquired, foundational
- Student's Physical Setup – Hardware, internet access & reliability, immediate physical study environment
- Cognitive Limits – Students' starting point, pressures to earn an income while studying

Consideration

→ Resources

The internal means and assets—such as money, personnel, time, and equipment—that the system utilizes to do its work

Talks to “the environment”:

- **Allotted Hours & Practice Blocks** – Budgeted against the course timeline to ensure concepts are given enough time to settle into long-term memory. Also, pacing –time to fix a mistake.
- **CodeGrade familiarisation** – Orientation of “how it works”.
- **Learning Materials** – The textbook, and other material the student actively selects to consume.
- **Development Environment** – The text editor, IDE extensions, and debugging tools used.
- **Prompting Power, GenAI as a Controlled Resource** – While the existence of GenAI is an environmental constraint, the choice to use it and how to prompt it is a resource; "copy-paste code generator" vs. "Socratic tutor that explains concepts line-by-line."
- **Asking Mentors/Peers for Help** – Clear up conceptual bottlenecks when internal cognitive resources are exhausted.
- **Focused Brainpower & Logic** – Deployed to break down problems, isolate bugs, and build mental code architecture.
- **Emotional Focus & Patience** – Spent to endure the anxiety of broken code without immediately quitting or resorting to cheating.

Churchman's approach

Consideration

→ Components

The distinct parts, subsystems, or product lines that perform specific activities and have their own local goals and performance measures (talks to the OBJs)



1. Problem Decomposition

<AI bypasses high-level logic>

Clear mental roadmap? The novice loses the ability to think like an architect.



2. Code Authoring

<AI generates syntax tokens>

Authoring syntactical expression? The novice remains illiterate in the programming language.



3. Testing & Execution

<AI glosses over edge-case flaws>

Verification for error-free execution? The novice suffers from automation bias, trusting broken systems.



4. Debugging & Diagnosis

<AI patches errors without explaining>

Diagnosing root causes? The novice panics easily and develops an illusion of competence.



Consideration

→ Management

The leadership and decision-making function that monitors system performance, sets policy, and allocates resources among components

Churchman's 5 Elements	The Management Control Strategy
1. Total Objective	The metric of success must be the student's understanding , not the software's execution.
2. Environment	Accept AI as a permanent environmental factor. Design tasks that assume the student has access to it.
3. Resources	Teach students to spend AI as an informational tutoring resource , rather than a cognitive substitute.
4. Components	Prevent GenAI from hyper-optimising code output at the expense of structural learning.
5. Management	Shift assessments to viva-voce code defences, peer reviews, and live debugging tasks.

Suggested approach

Churchman introduced "four enemies"—**Politics, Morality, Religion, and Aesthetics**, not being "enemies" maliciously, but rather representing powerful, non-rational human forces that bypass the comprehensive, top-down rationality of the "whole system" planner. When novice programmers rely on GenAI, they achieve short-term functional outcomes while introducing massive systemic imbalances. The four enemies expose the vulnerabilities of an AI-mediated approach to software systems:

Churchman (1979)

1. Politics (The Enemy of Immediate Implementation)

- GenAI acts as an hyper-efficient political ally for the novice, towards code that executes in order to **complete an assignment**, while also allowing them to bypass the **strict, structured "laws" of learning syntax, architecture, and proper debugging**. They satisfy the immediate political demand (the deadline) but introduce epistemic debt—building an operational system that they do not understand.

2. Morality (The Enemy of Individual Values over Total Goals)

- When novices lean heavily on AI, an ethical friction occurs between the human learner and the educational system. The total system's objective is to cultivate **deep critical thinking, problem-solving mastery, and authentic learning**. Unfortunately, a novice's immediate moral framework shifts toward self-preservation and efficiency, and they view AI as a "coping mechanism" to deal with cognitive overload, leading to an illusion of competence.

3. Religion (The Enemy of Unquestioned Authority and Transcendence)

- Novices frequently develop an almost religious reverence for GenAI, treating the large language model as an unquestioned, omniscient oracle. They suffer from automation bias, **accepting faulty logic or sub-optimal code frameworks blindly**. They bypass the **essential cognitive friction required to construct an accurate mental model**.

4. Aesthetics (The Enemy of the "Liveliness" of Chaos)

- Real programming is an art form rooted in **human intuition, architectural elegance, and the messy, iterative "joy of struggle"**. By robbing the novice of the painful but deeply human experience of getting stuck and finding an elegant, unique solution, AI replaces the **"art of coding"** with mechanical copy-pasting. The novice misses out on developing a personal coding signature, intuition, and the satisfaction of a hard-won victory.

Herman Dooyeweerd's framework explicitly consists of **15 distinct, irreducible modal aspects**.

Using Dooyeweerdian philosophy to "guide" Churchman's **four enemies** (Politics, Morality, Religion, and Aesthetics), as a powerful philosophical synthesis. Churchman warned that these four forces act as "enemies" because they reject the purely rational, mathematical logic of the whole-systems planner. Dooyeweerd provides an explanatory lens for *why* they do this: **the planner is guilty of "aspectual reductionism"** (trying to force higher normative realities into lower, analytical frameworks).

To understand how Herman Dooyeweerd's modal aspects can guide Churchman's **four enemies**, clarification of the "10 vs 15" distinction is needed. While Dooyeweerd originally outlined 15 discrete modalities, modern systems theorists (such as Andrew Basden) often group them into:

- 10 normative/human aspects when analysing socio-technical systems like software engineering.
- These 10 human aspects sit on top of the 5 foundational, non-human aspects (Numerical, Spatial, Kinematic, Physical, and Biotic).

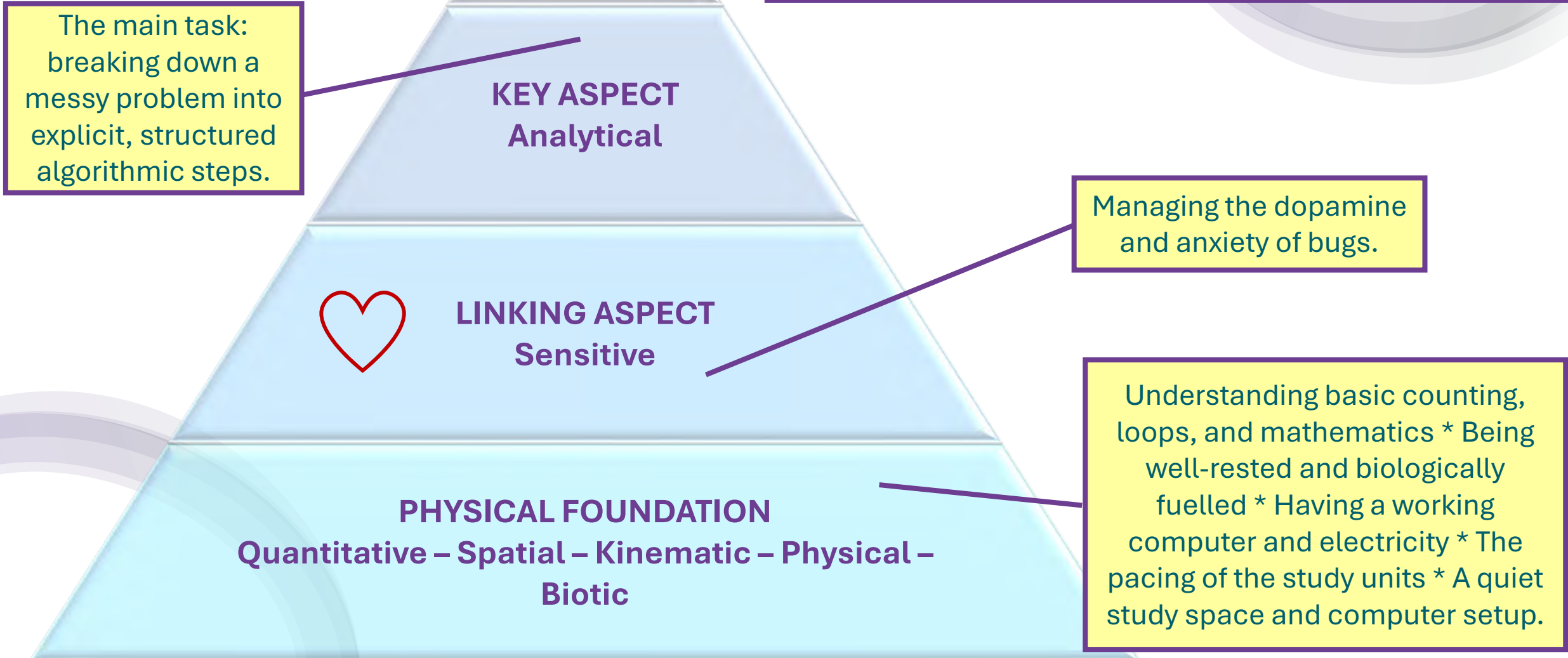
When a novice programmer uses GenAI, they create a systemic imbalance by leaning heavily on lower-level, mechanical aspects while neglecting higher-level human ones.

The **10 normative Dooyeweerdian modalities** guide, structure, and diagnose Churchman's **four enemies** within the context of a novice programmer's learning journey:

15 Dooyeweerdian modalities

Nr.	Aspect	Definition
1	Quantitative	Discrete amounts and numerical properties (how much or how many)
2	Spatial	Continuous space, extension, and geometry
3	Kinematic	Movement, flow, and transition
4	Physical	Energy, mass, and material forces
5	Biotic	Life functions, growth, and organic regulation
6	Sensitive	Senses, feeling, and emotional responsiveness
7	Analytical	Distinguishing objects from ground, logic, and conceptual analysis
8	Formative	Shaping, creativity, culture, history, and technology
9	Lingual	Symbolic communication and language
10	Social	Social interaction, community, and roles
11	Economic	Frugal, efficient use of scarce resources
12	Aesthetic	Harmony, beauty, fun, and surprise
13	Juridical	What is due to others, rights, and retribution
14	Ethical	Self-giving love, moral duty, and interpersonal openness
15	Pistic	Faith, vision, ultimate commitment, and trust

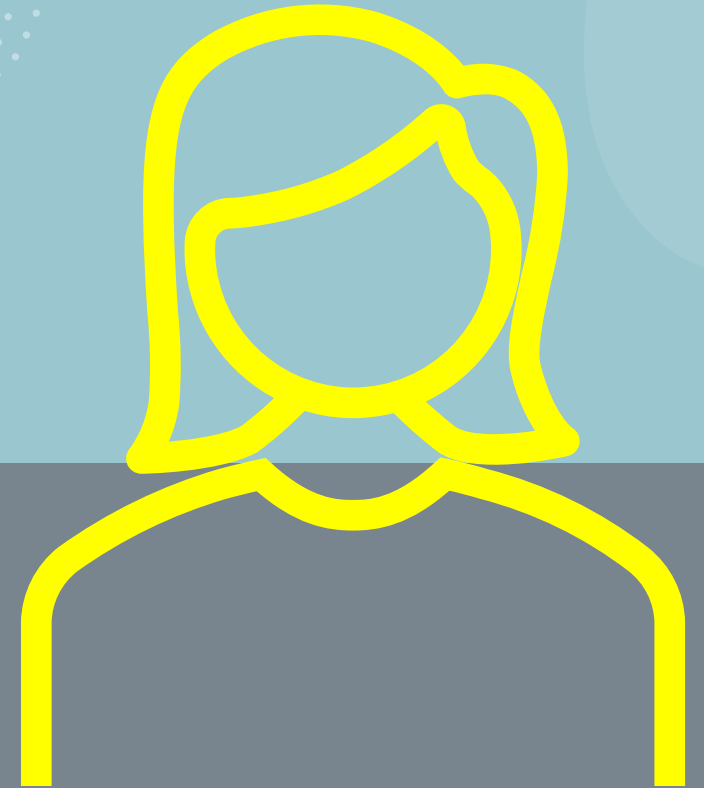
Suggested approach



Building on this work

A masters student uses a literature review
- to build guidelines for the use of genAI
when novice programmers learn to code

Methodology is Design Science Research,
not AR



Building on this work

A masters student utilises lightweight machine learning models to identify at-risk students



Building on this work

Own work; we started by looking into the metaphors students use for GenAI upon providing GenAI mental models of cognitive engagement, from extending and stretching thinking to outsourcing it (tool, tutor, facilitator, specialist, threat, liar), and now we are comparing what students claim to how they used the CodeGrade ChatBot.





I have drawn on the work of:

Basden, A. (2014). Understanding artefacts related to human aspects: The case of information technology and systems. *Koers*, 79(3), 01-10.

Basden, A. (2018). *The Foundations of Information Systems: Research and Practice*. London: Routledge Taylor & Francis Group.

Churchman, C. W. (1968). *Challenge to reason*. New York: Dell Publishing CO, INC.

Churchman, C. W. (1979). *The systems approach and its enemies*.

Dooyeweerd, H. (1984). *A New Critique of Theoretical Thought*.

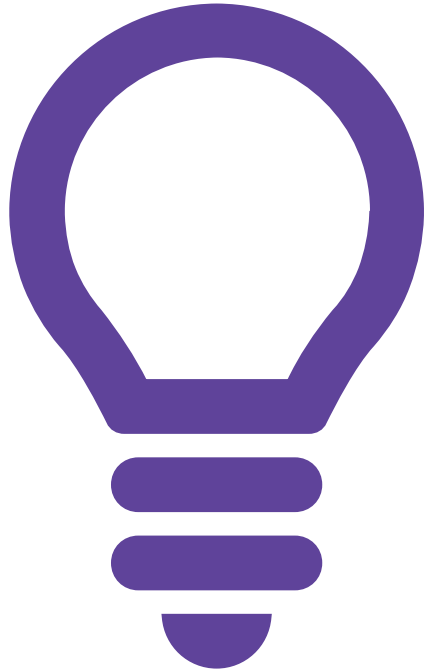
Volume I. Translated by David H Freeman and William S. Young (F. a. H. d. Jongste, Trans.). In (pp. 663). Ontario Canada: Paideia Press.

Malik, M. (2025). Co-Inference and Collaboration: Empowering Higher Education Staff and Students through a GenAI Literacy Development Framework Based on Mind-Metaphors. *Journal of Learning Development in Higher Education*(38).

Susman, G. I., & Evered, R. D. (1978). An assessment of the scientific merits of action research. *Administrative science quarterly*, 23(4), 582-603.

Ulrich, W. (2005). A brief introduction to critical systems heuristics (CSH). ECOSENSUS project site.

Guiding novice Python programmers to navigate learning to code in an era of generative Artificial Intelligence



Thank you



imelda.smit@nwu.ac.za