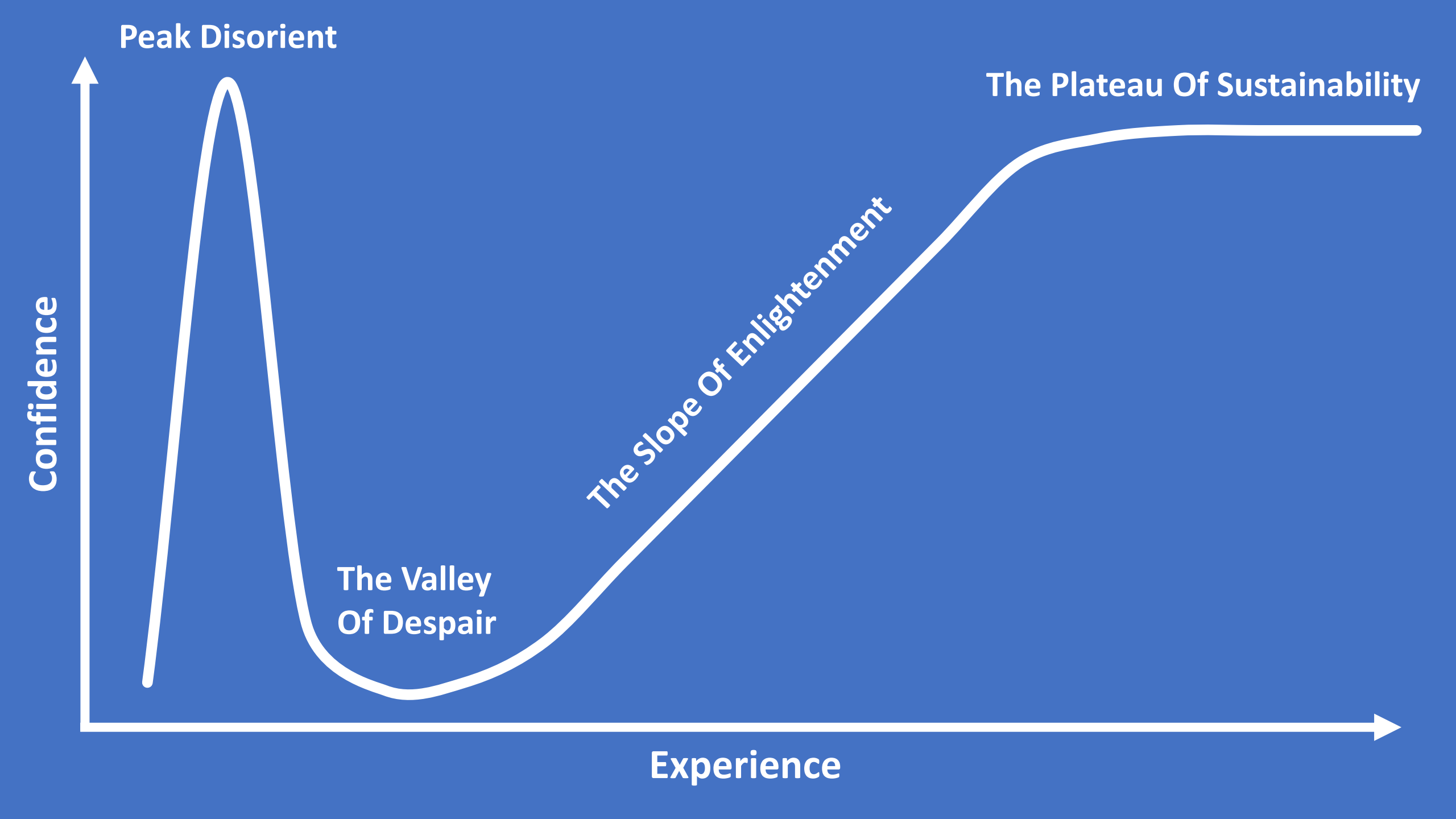
Getting Better at Getting Better

A toolbox for continuous learning in the tech industry

Mike Ritchie
mike@evolved.dev


Peak Disorient
The Plateau Of Sustainability
Confidence
The Slope Of Enlightenment
The Valley
Of Despair
Experience

Bootcamp Context #1

“A fast route into a tech career doesn’t mean a fast route through a tech career.”

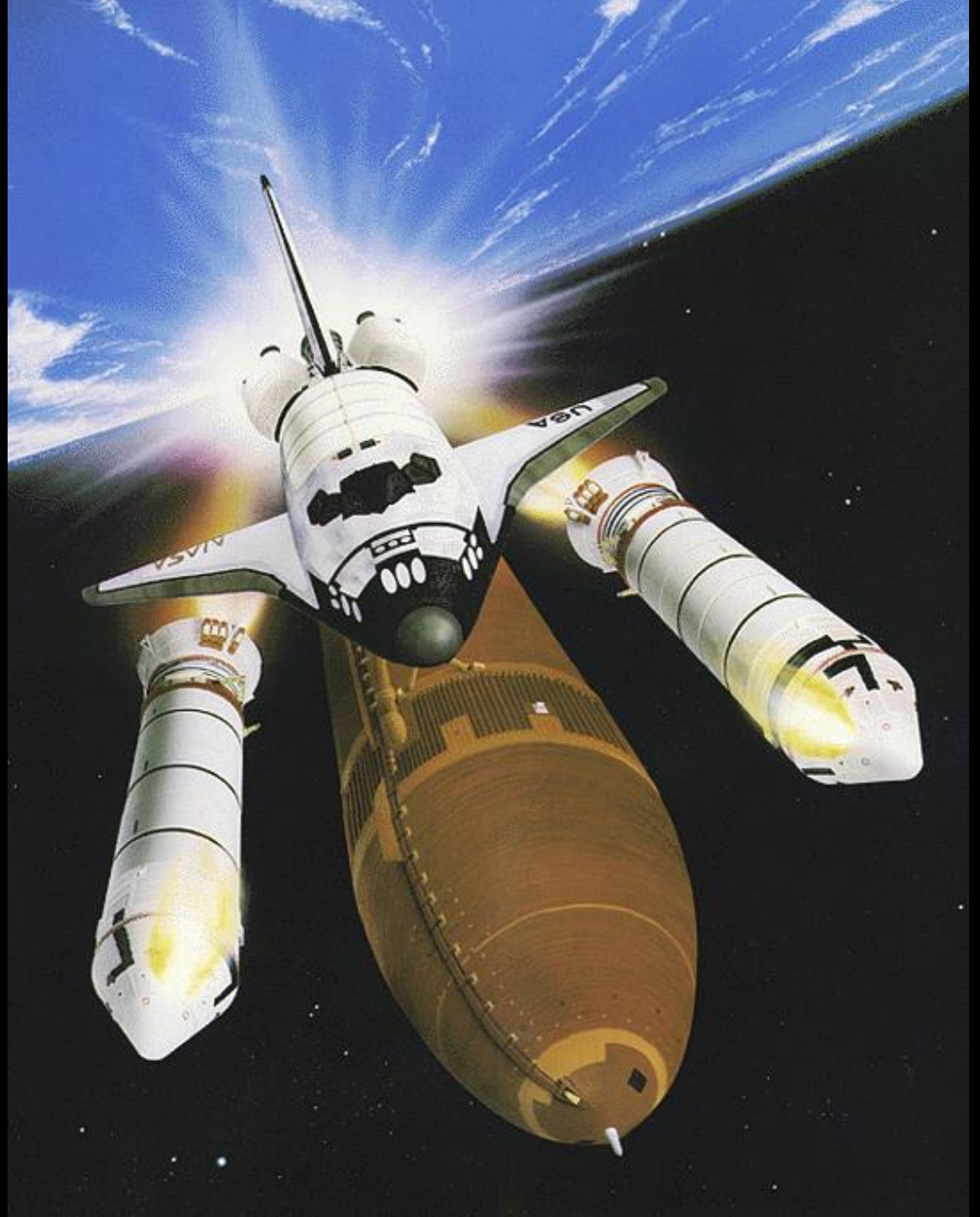


Image © NASA Goddard Space Flight Center

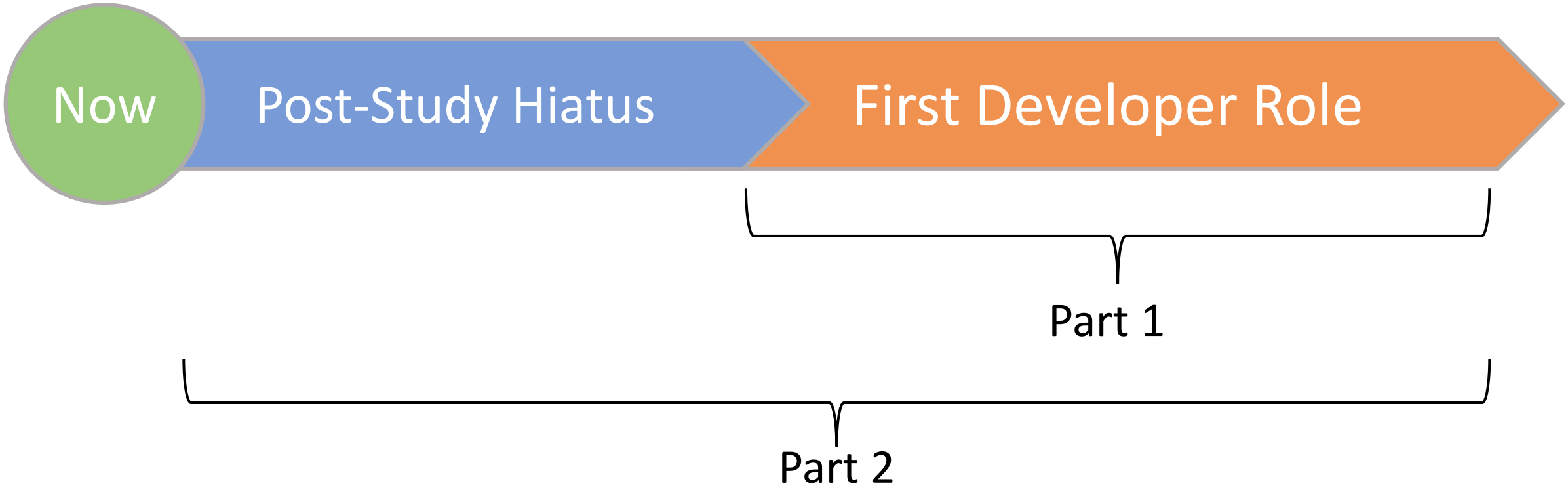
Bootcamp Context #2

“You won’t be able to recreate the bootcamp environment outside of the bootcamp.”

Bootcamp Context #3

“You have breadth. Depth is going to take time, effort, and tough calls.”

One Talk, Two Timeframes

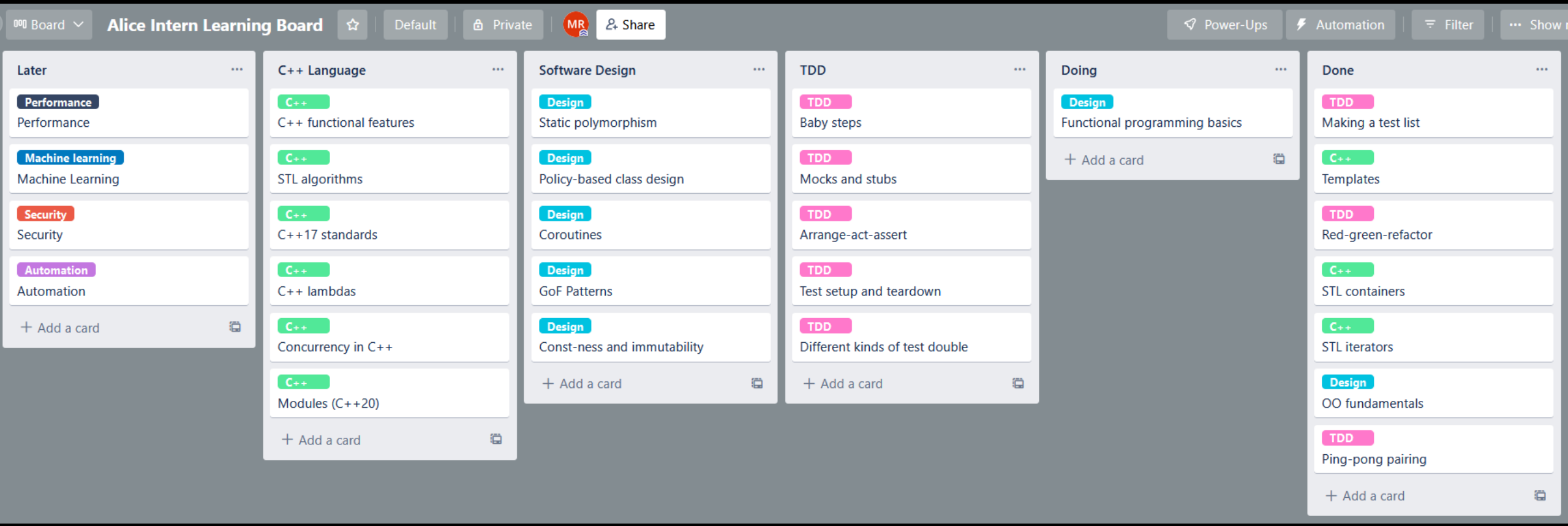


Developing situational awareness of your skills

Techniques to help focus on the things that matter most.

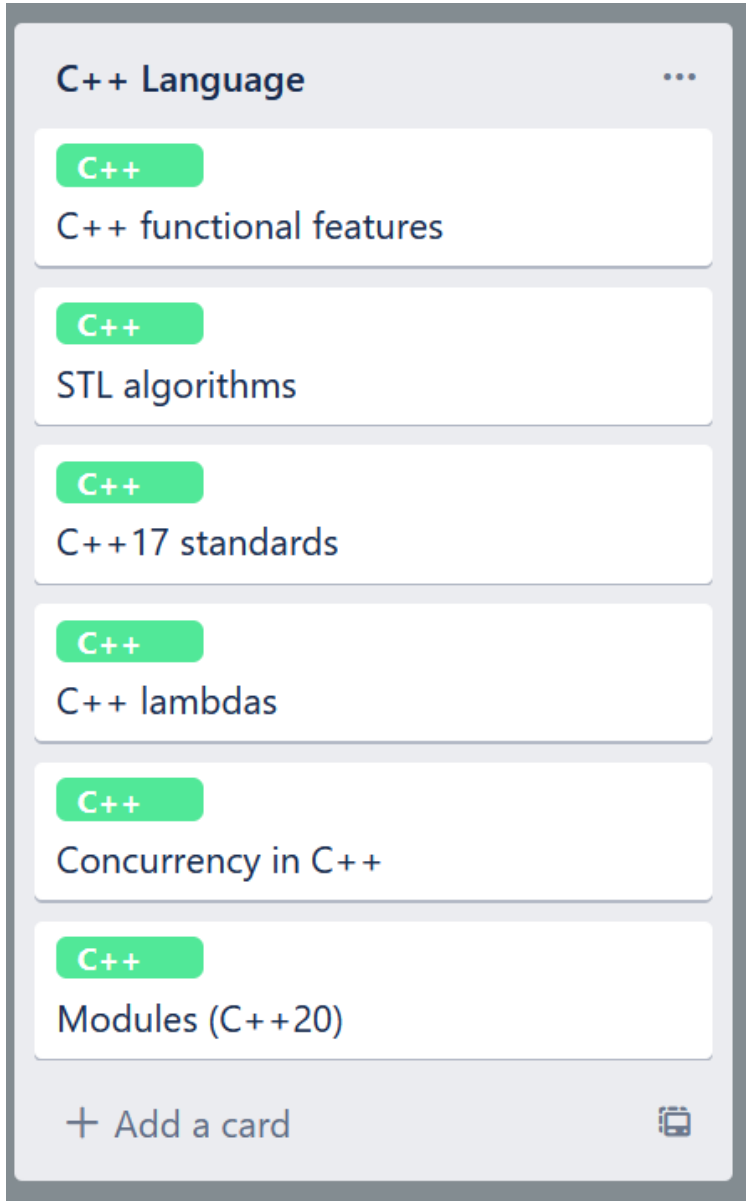
Differentiators: what makes “great”

Great engineers	Have an accurate perception of their own skills
	Are situationally aware of the skills landscape in their domain
	Learn intentionally, practice deliberately
	Value long-term portable skills most highly
	Prioritise what they work on in their learning time


Board
Alice Intern Learning Board
Default
Private
MR
Share
Power-Ups
Automation
Filter
Later
Performance
Performance
Machine learning
Machine Learning
Security
Security
Automation
Automation
Add a card
C++ Language
C++
C++ functional features
C++
STL algorithms
C++
C++17 standards
C++
C++ lambdas
C++
Concurrency in C++
C++
Modules (C++20)
Add a card
Software Design
Design
Static polymorphism
Design
Policy-based class design
Design
Coroutines
Design
GoF Patterns
Design
Const-ness and immutability
Add a card
TDD
TDD
Baby steps
TDD
Mocks and stubs
TDD
Arrange-act-assert
TDD
Test setup and teardown
TDD
Different kinds of test double
Add a card
Doing
Design
Functional programming basics
Add a card
Done
TDD
Making a test list
C++
Templates
TDD
Red-green-refactor
C++
STL containers
C++
STL iterators
Design
OO fundamentals
TDD
Ping-pong pairing
Add a card

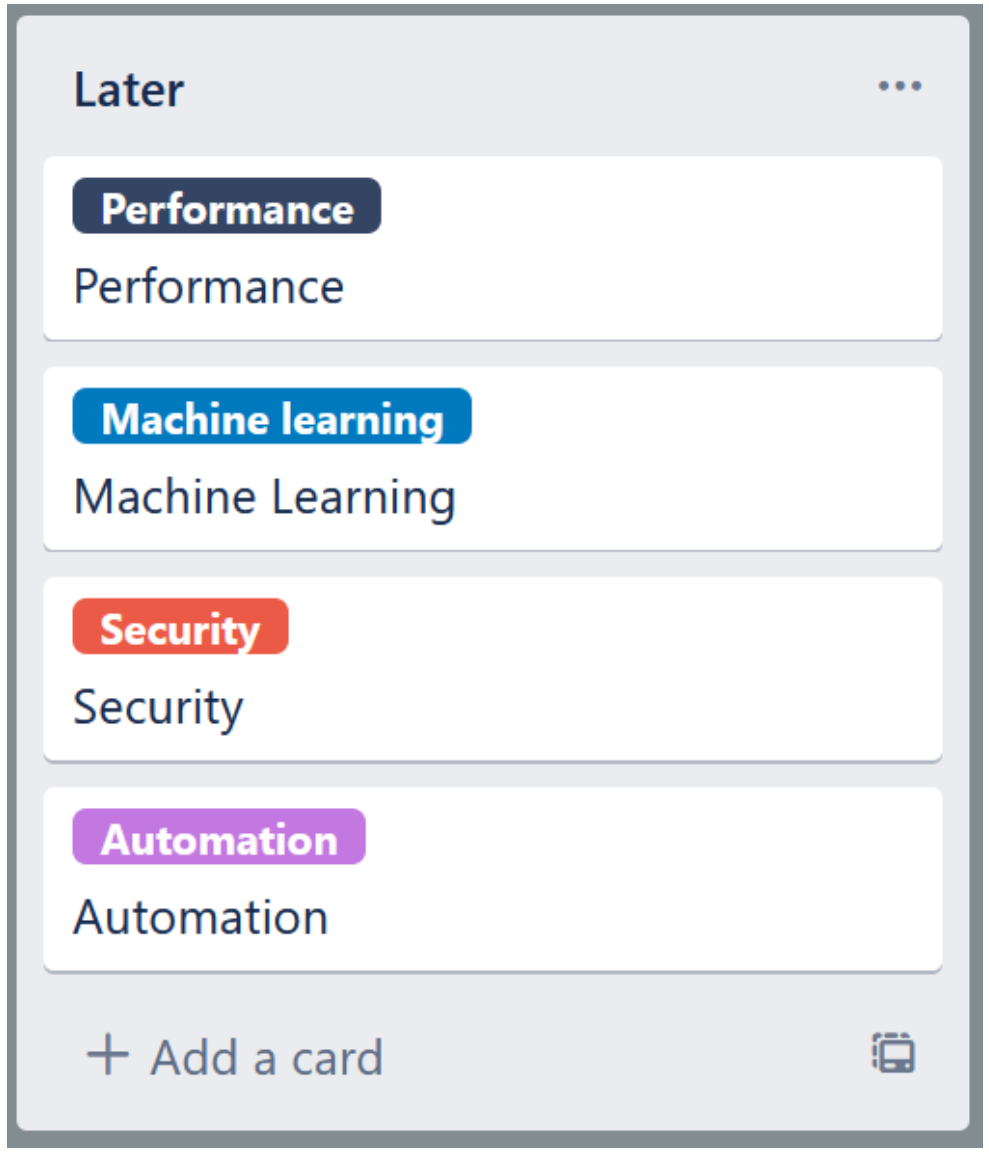
Three Focus Areas

Although the board *looks* like a lot at first glance, there are actually only three subjects that Alice is currently occupied with right now.



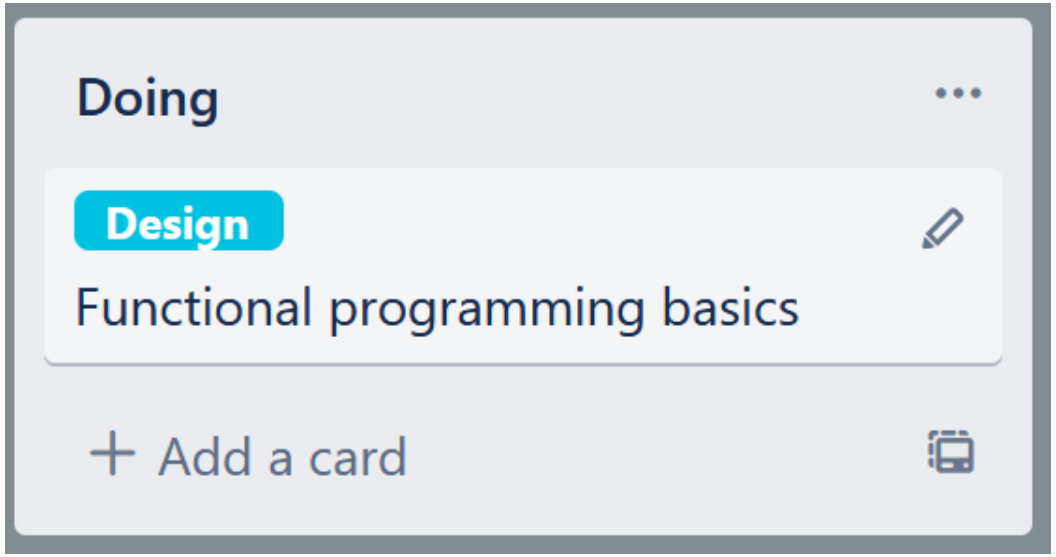
Decomposition

Each subject area is broken down into small, focussed, *achievable* chunks that Alice can work on without being overwhelmed.



Parked

Some big topics have been bookmarked for later. It’s not that these are unimportant, they’re just not important *right now*.



The Key To It All

Most importantly of all, she's working on a single card at a time. Two might be OK, but one is definitely best.

Your Initial Focus

Your first goal will be working to become a *productive individual contributor*:

1. What skills do team members need?
2. What level is “proficient” for those skills?
3. Where do I stand relative to “proficient”?
4. How soon will I be blocked by a skills gap?

Colleagues

Management

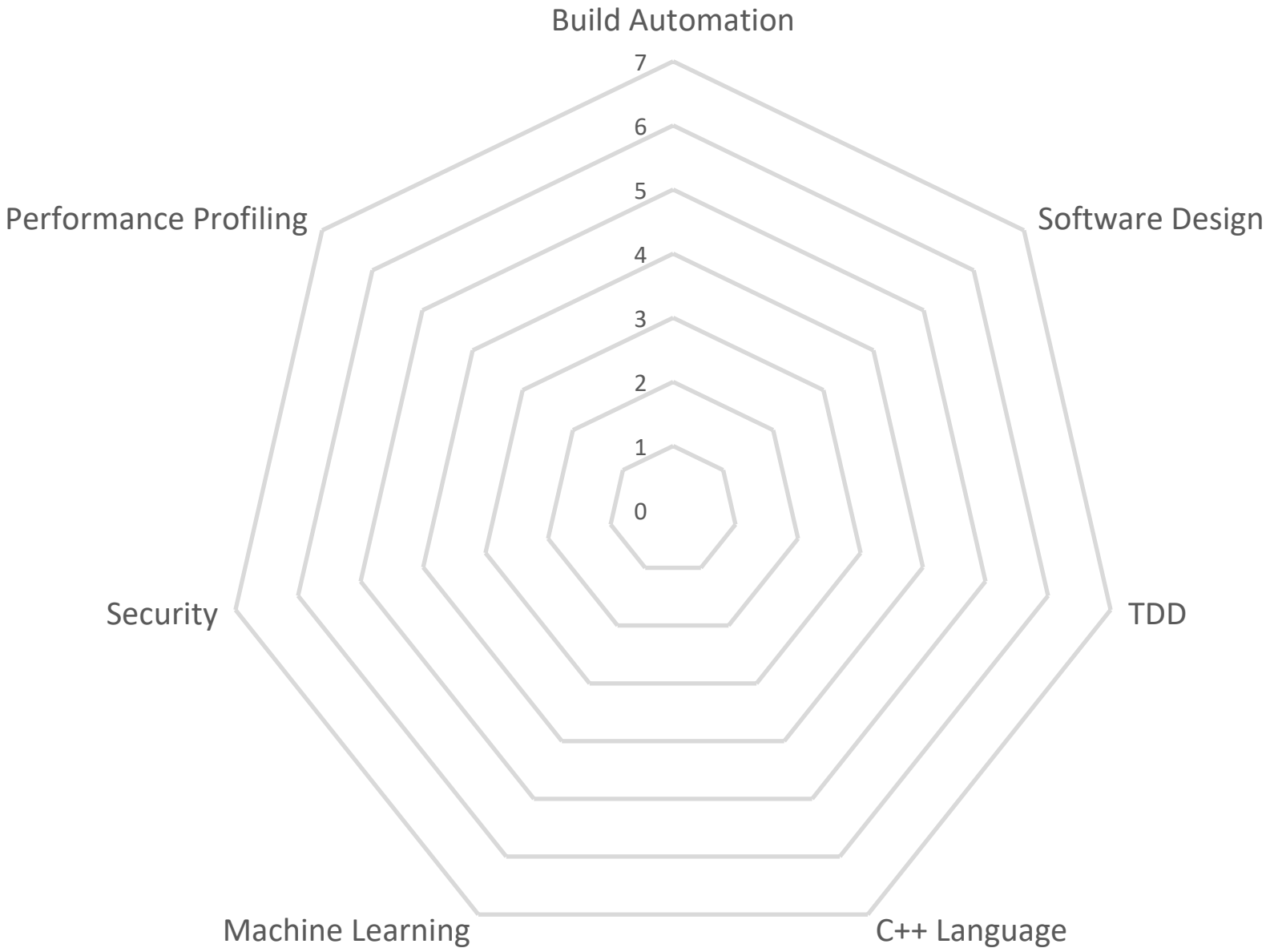
Tech Leadership

Mentoring

Your Initial Focus

Your first goal will be working to become a *productive individual contributor*:

1. What skills do team members need?
2. What level is “proficient” for those skills?
3. Where do I stand relative to “proficient”?
4. How soon will I be blocked by a skills gap?


Build Automation
Software Design
TDD
C++ Language
Machine Learning
Security
Performance Profiling
0
1
2
3
4
5
6
7

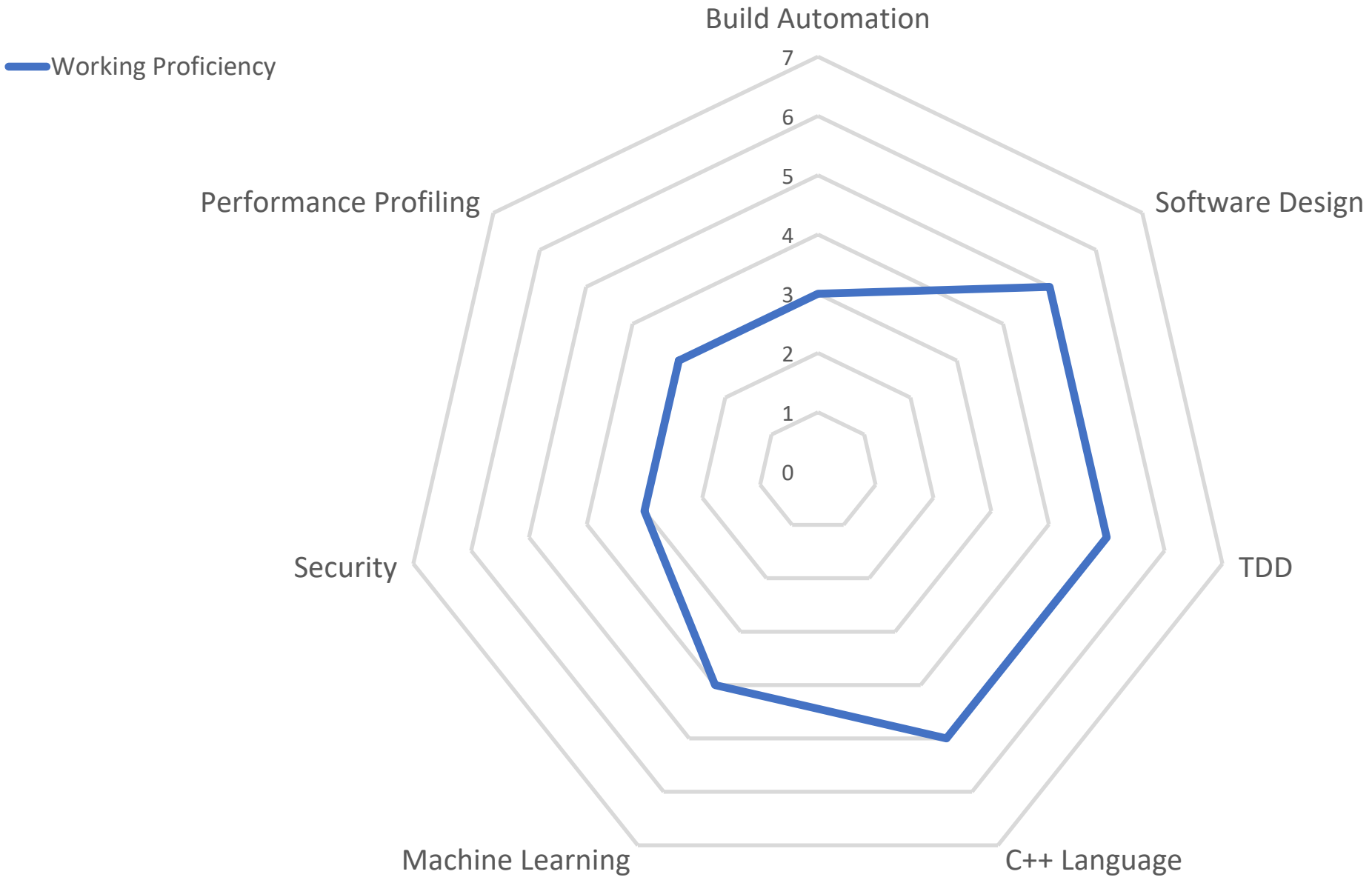
Characterising Skill Levels – Security Example

Level	Capability Expected
1	Understand why security is important, and the potential impact of insecurities in code
2	Recognise different types of vulnerability e.g. buffer overflows or reading uninitialized memory
3	Adopt programming practices to mitigate the most common types of vulnerability
4	Configuration and routine use of static analysis tools, remediating issues in own and others code
5	Configuration of dynamic memory and leak sanitizers in CI builds, remediating issues detected
6	Setting up fuzz testing for projects, integrating into CI builds, remediating issues detected
7	Team authority on secure programming, coaches others in the team on secure programming practices, defines security standards for new projects

Your Initial Focus

Your first goal will be working to become a *productive individual contributor*:

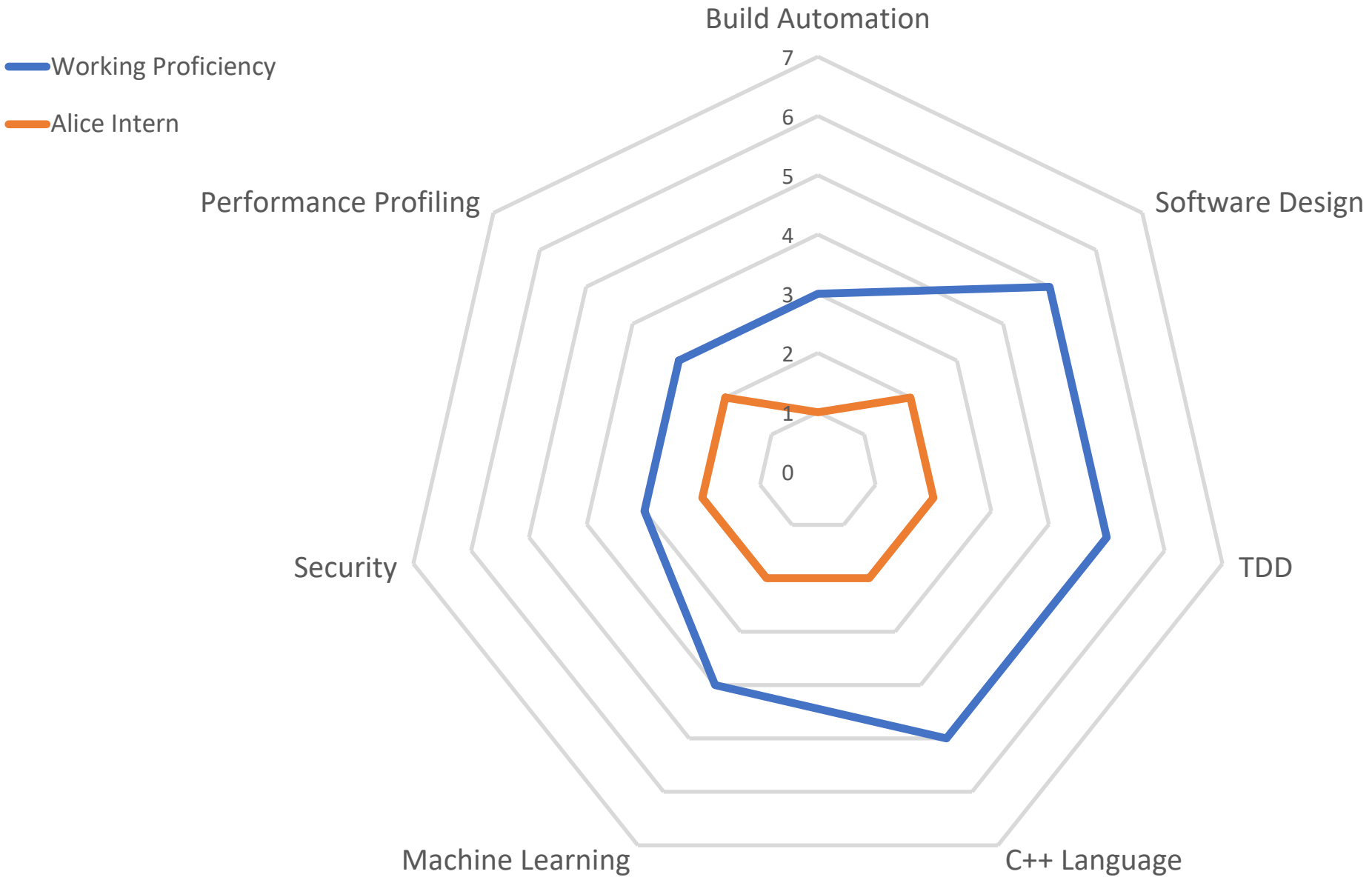
1. What skills do team members need?
2. What level is “proficient” for those skills?
3. Where do I stand relative to “proficient”?
4. How soon will I be blocked by a skills gap?

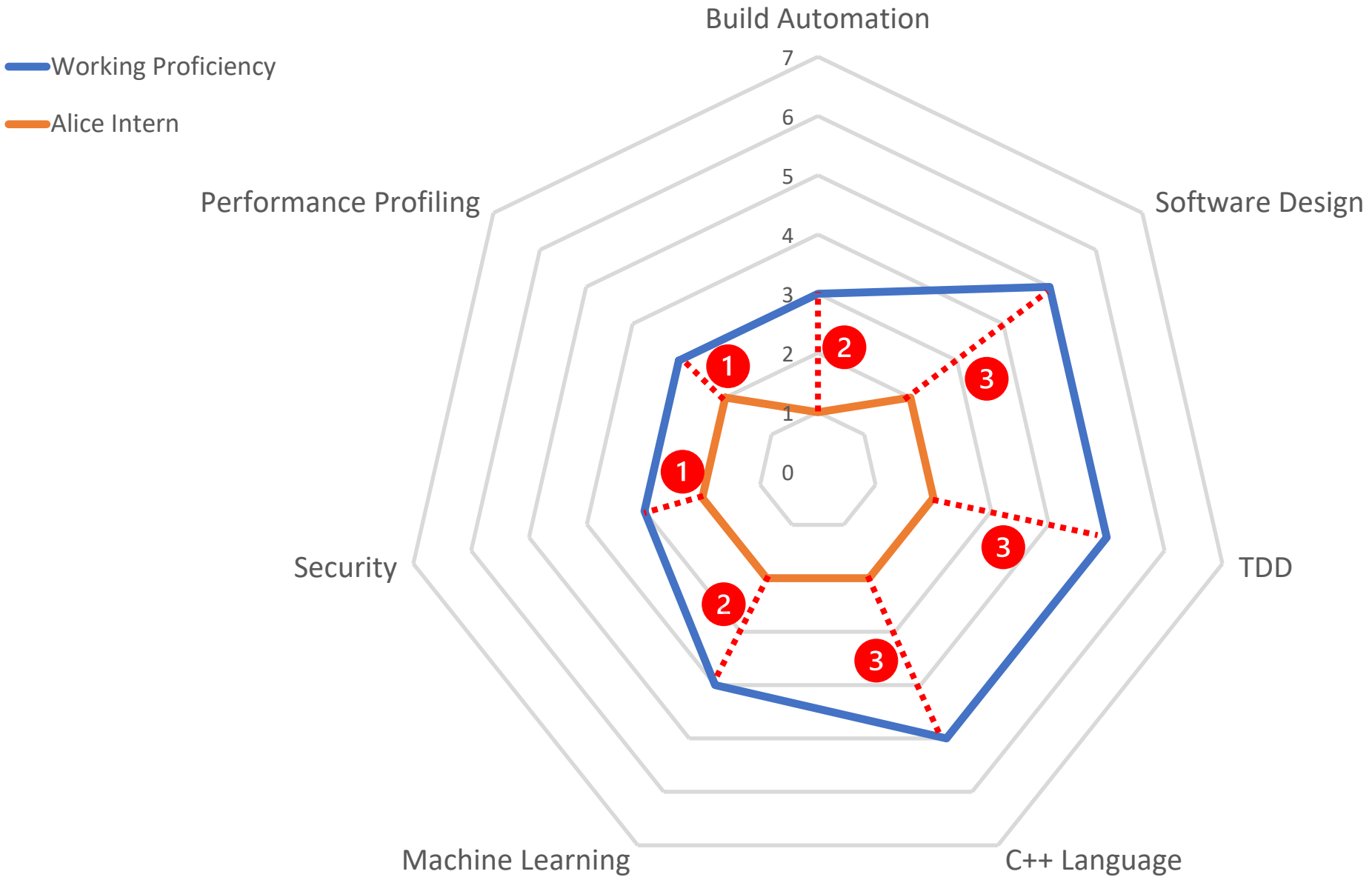

Working Proficiency
Build Automation
Software Design
TDD
C++ Language
Machine Learning
Security
Performance Profiling
7
6
5
4
3
2
1
0

Your Initial Focus

Your first goal will be working to become a *productive individual contributor*:

1. What skills do team members need?
2. What level is “proficient” for those skills?
3. Where do I stand relative to “proficient”?
4. How soon will I be blocked by a skills gap?

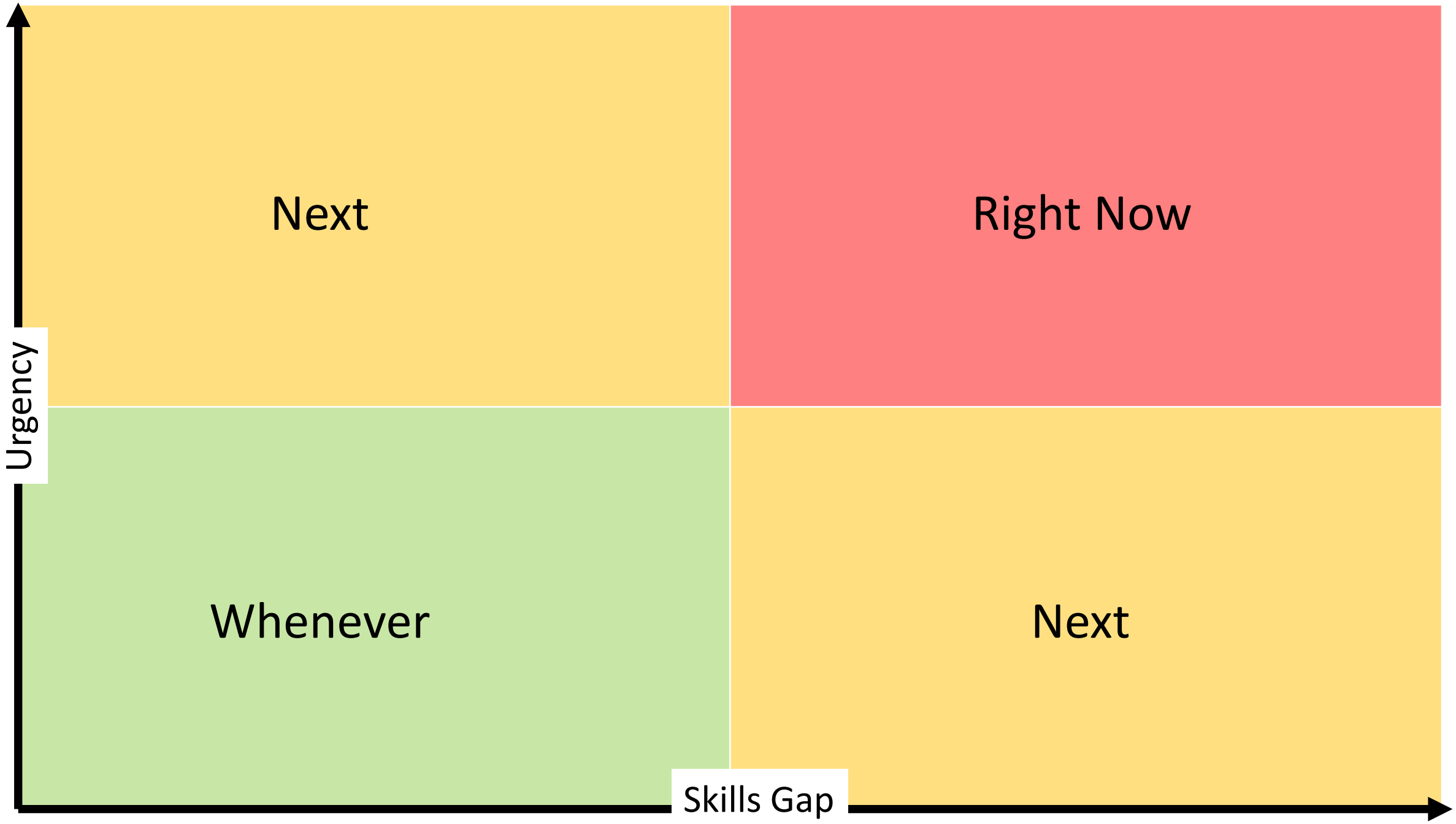

Working Proficiency
Alice Intern
Build Automation
Software Design
TDD
C++ Language
Machine Learning
Security
Performance Profiling
7
6
5
4
3
2
1
0

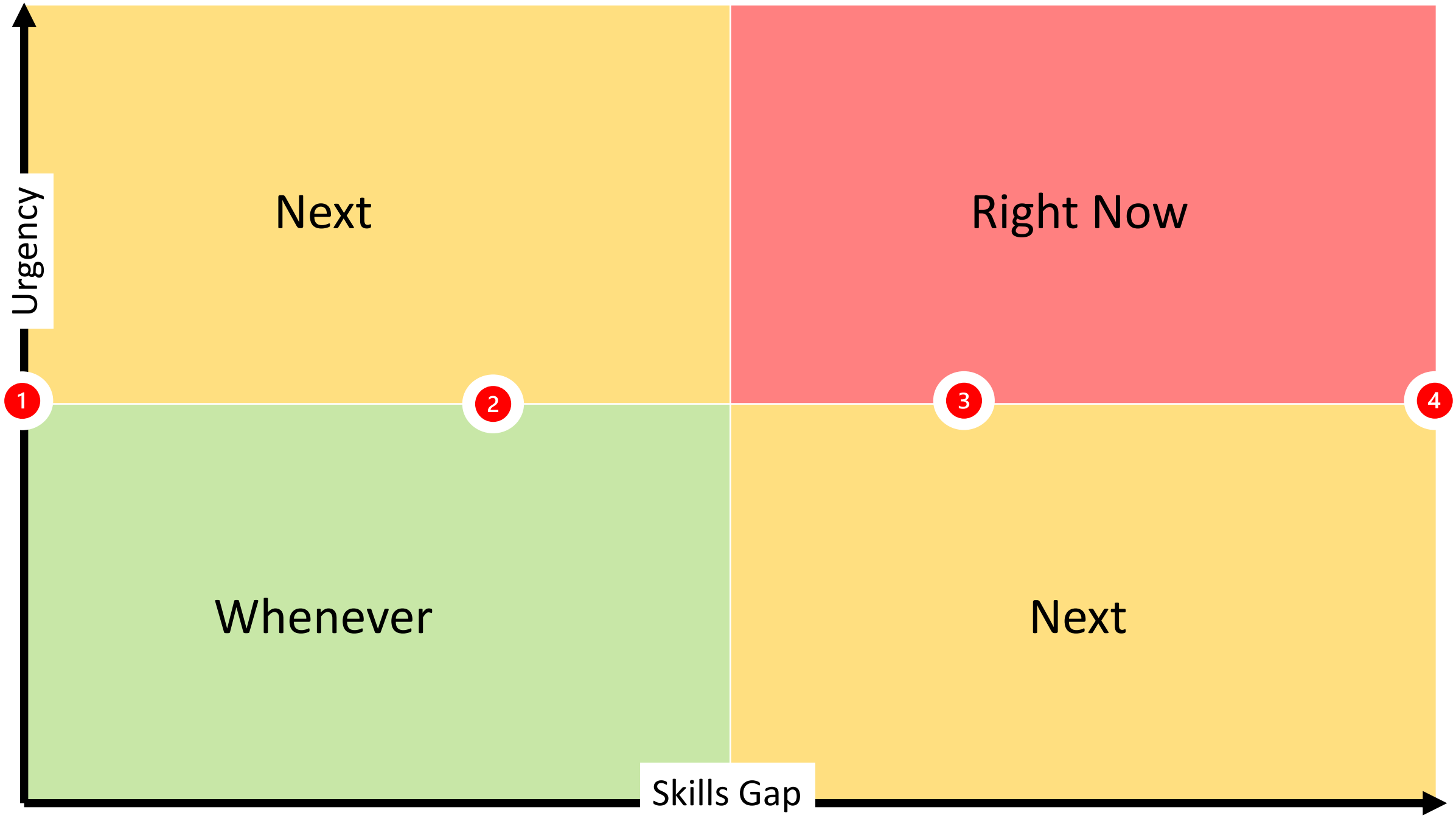

Working Proficiency
Alice Intern
Build Automation
Software Design
TDD
C++ Language
Machine Learning
Security
Performance Profiling
7
6
5
4
3
2
1
0
1
2
3
3
3
2
1

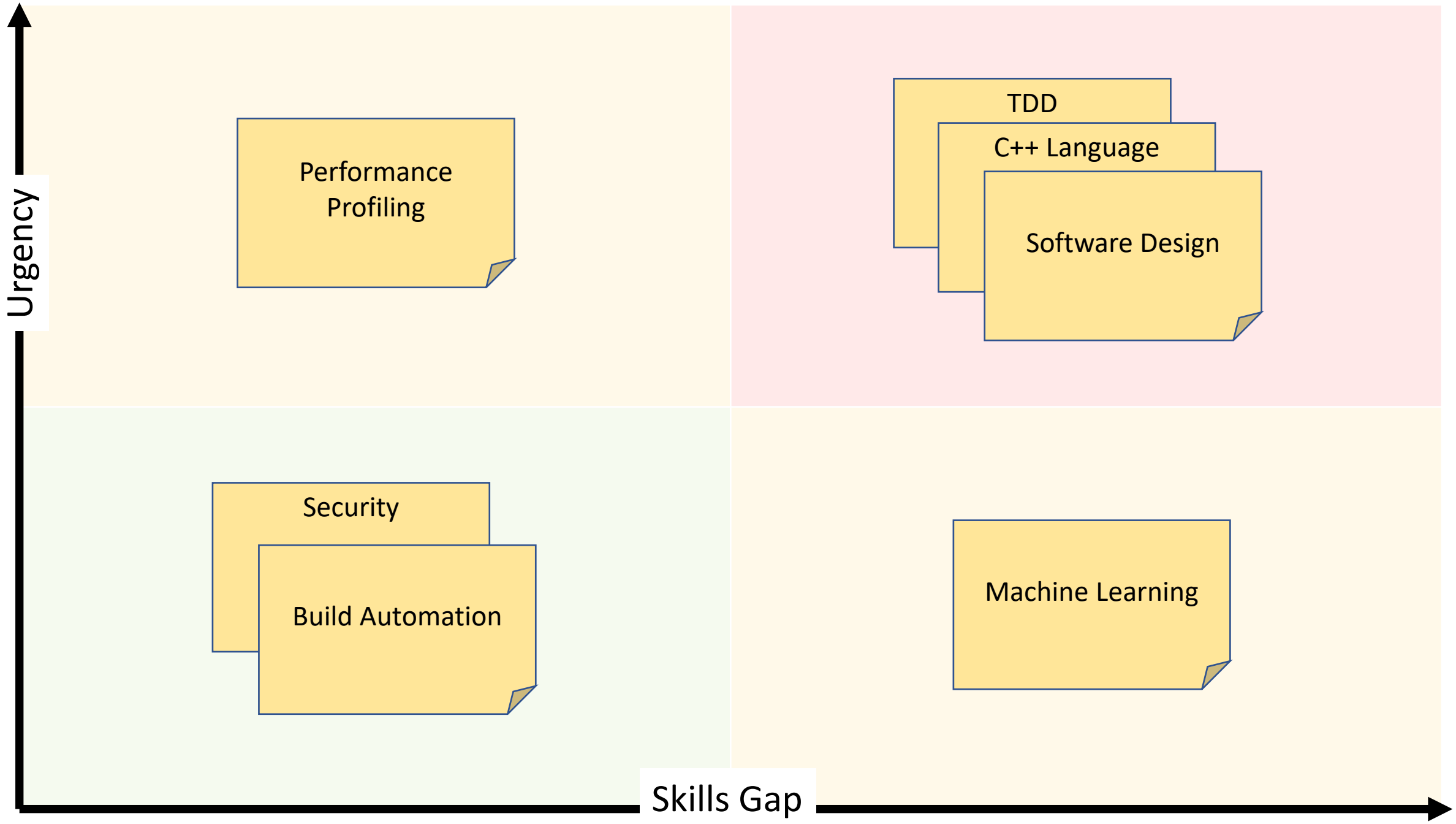
Your Initial Focus

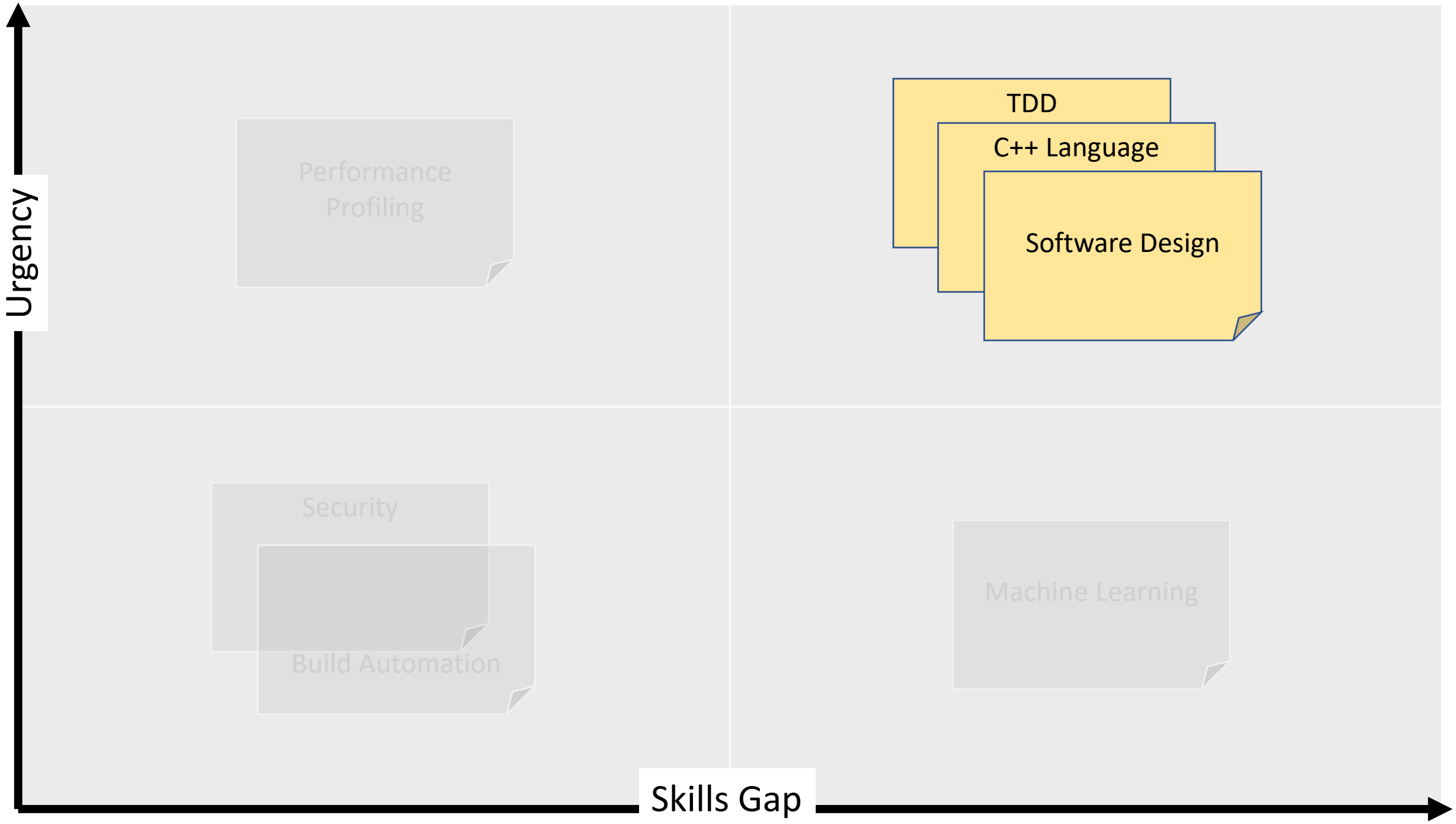
Your first goal will be working to become a *productive individual contributor*:

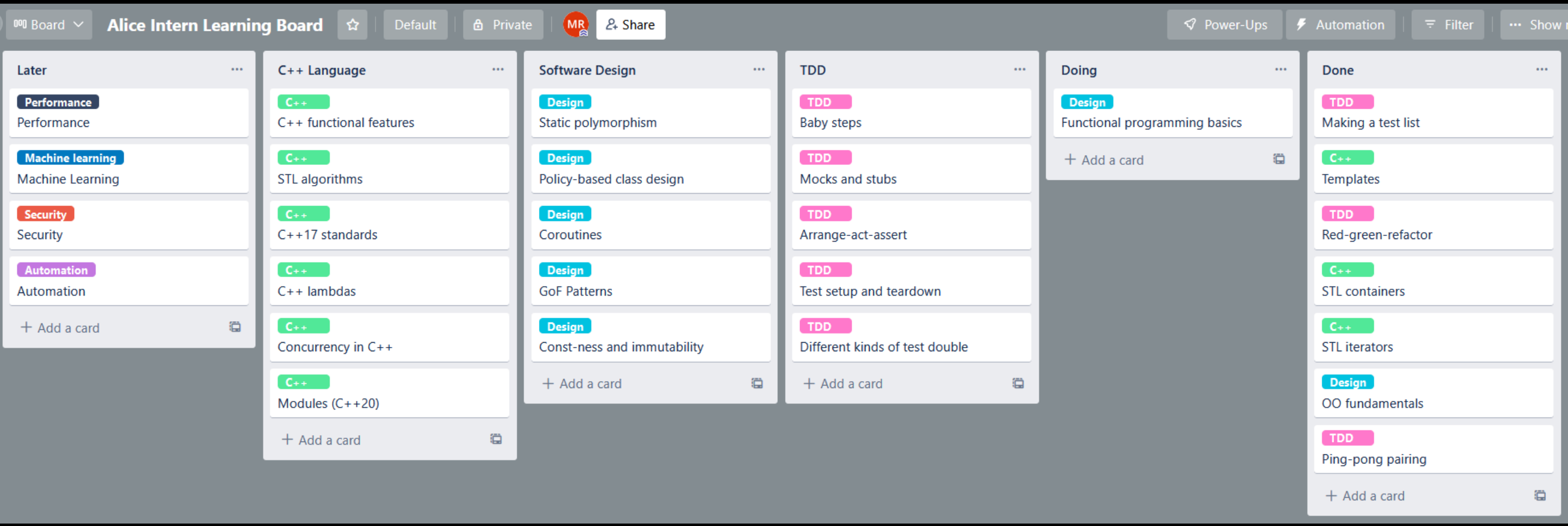
1. What skills do team members need?
2. What level is “proficient” for those skills?
3. Where do I stand relative to “proficient”?
4. How soon will I be blocked by a skills gap?


Next
Right Now
Whenever
Next
Urgency
Skills Gap


Urgency
Next
Right Now
Whenever
Next
Skills Gap
1
2
3
4


Urgency
Performance Profiling
TDD
C++ Language
Software Design
Security
Build Automation
Machine Learning
Skills Gap

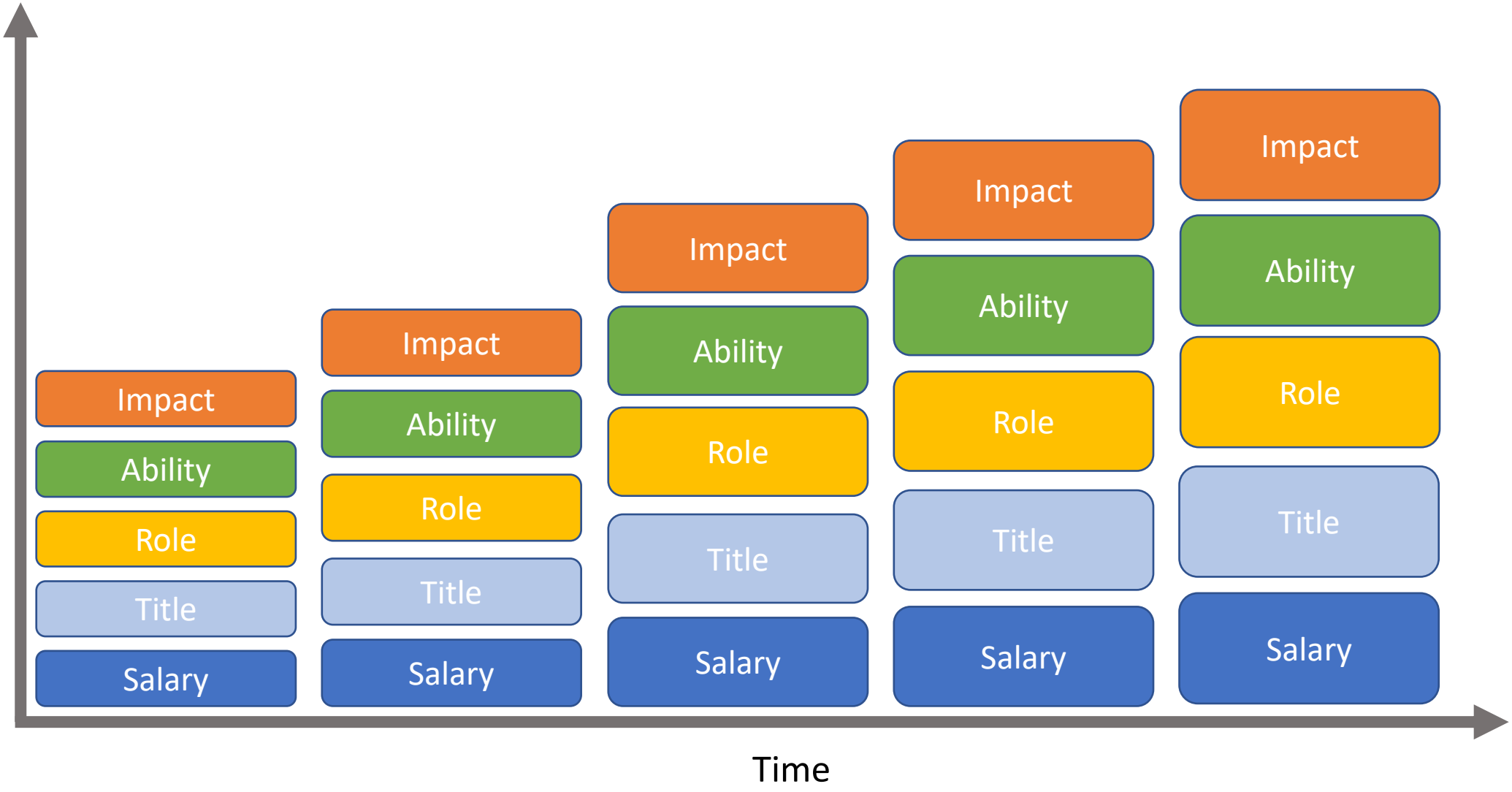

Urgency
Performance Profiling
TDD
C++ Language
Software Design
Security
Build Automation
Machine Learning
Skills Gap


Board
Alice Intern Learning Board
Default
Private
MR
Share
Power-Ups
Automation
Filter
Later
Performance
Performance
Machine learning
Machine Learning
Security
Security
Automation
Automation
Add a card
C++ Language
C++
C++ functional features
C++
STL algorithms
C++
C++17 standards
C++
C++ lambdas
C++
Concurrency in C++
C++
Modules (C++20)
Add a card
Software Design
Design
Static polymorphism
Design
Policy-based class design
Design
Coroutines
Design
GoF Patterns
Design
Const-ness and immutability
Add a card
TDD
TDD
Baby steps
TDD
Mocks and stubs
TDD
Arrange-act-assert
TDD
Test setup and teardown
TDD
Different kinds of test double
Add a card
Doing
Design
Functional programming basics
Add a card
Done
TDD
Making a test list
C++
Templates
TDD
Red-green-refactor
C++
STL containers
C++
STL iterators
Design
OO fundamentals
TDD
Ping-pong pairing
Add a card

Coaching and Mentoring

Career Mentoring

Once Upon a Time, In a Land Far Away



IMPACT

BANG

BOOM

SALARY

THWACK

POW

ABILITY

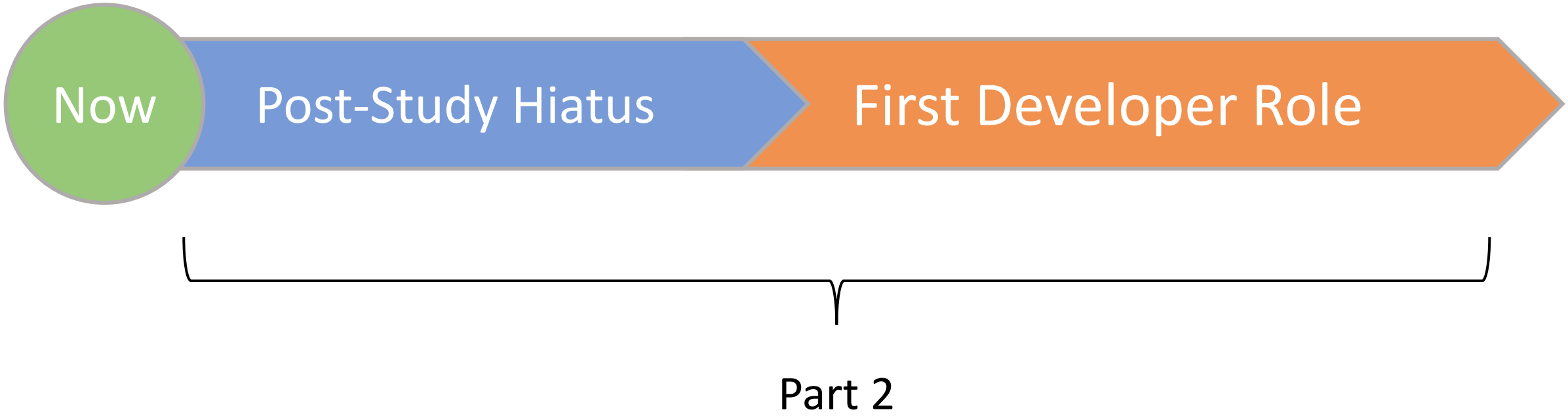
ROLE

PFOOM

TITLE

KABOOM

Part 2 – Learning and Practice Approaches



nothing

nothing *new*

Consolidate

- Solve bigger problems with what you already know
- Delve deeper into the frameworks you've used
- Extend your knowledge of your language + core lib
- Use the full capability of your test frameworks
- Make your tools work harder for you
- Be curious - peek under the hood!

Build Fluency

…in solving harder problems

…in software design

…in using your language to best effect

…in test-driven development

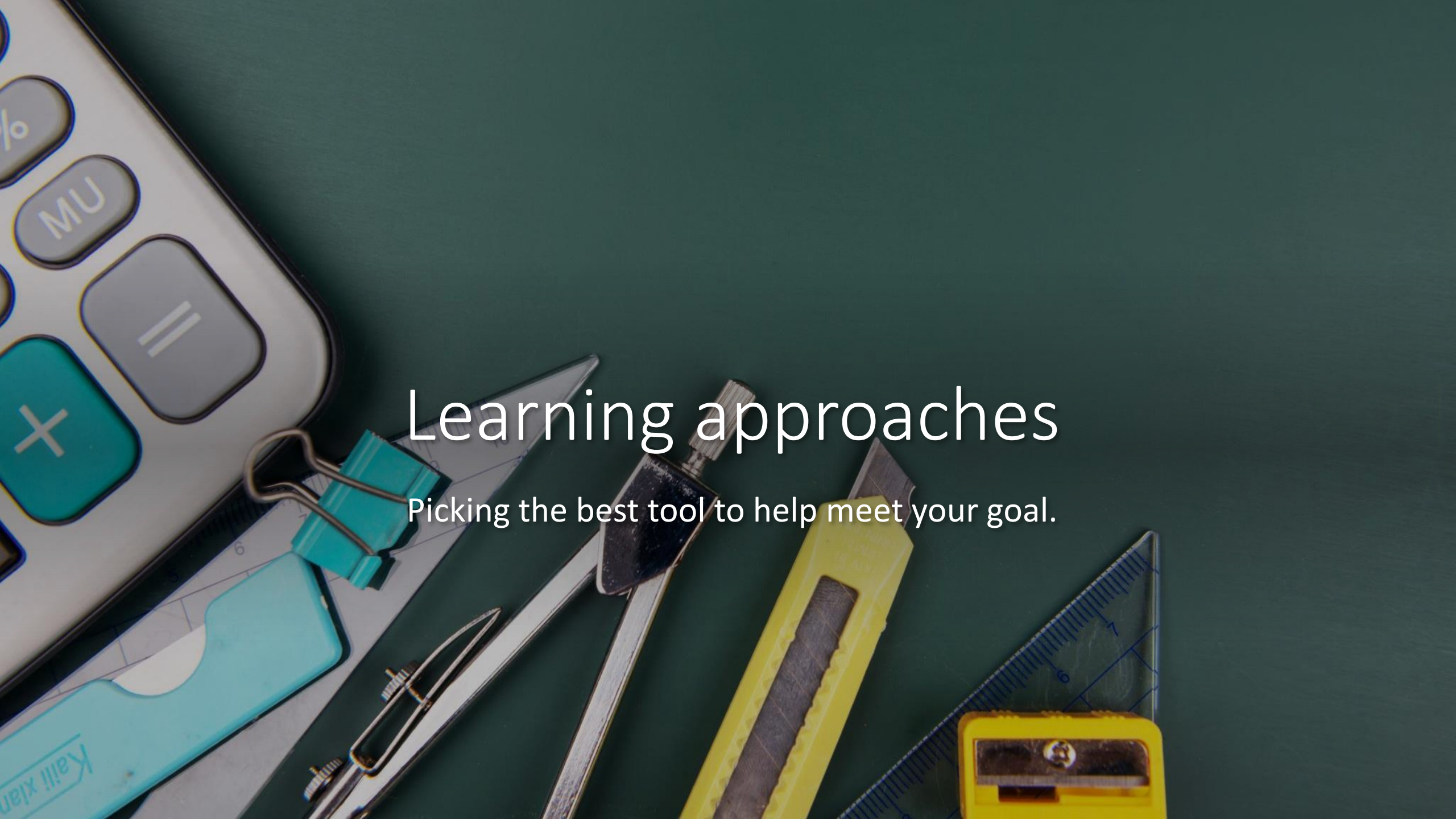
…in source control workflows

…in critiquing and refactoring code

…with your tools of choice

Connect

- Connect with companies, individuals and communities
- Scotland has some great meetups
- Go along to company-hosted events, chat with the hosts
- Go to meetups, let people know you're a new programmer
- Join meetup Slack/Discord channels if they have them



Learning approaches

Picking the best tool to help meet your goal.

Meet Professor Tanenbaum

- His entire professional life has been focused on education
- He specialises in low-level computing: computer architecture, operating systems, networking
- He has researched, taught, and collaborated with other experts in his field for decades
- He’s the author of many textbooks, some regarded as standards

Image CC-BY-SA, user Jantangring @ Wikipedia

Meet Professor Tanenbaum's Book

- First published in 1976
- Continuously improved, revised, and updated, and now in its 6th edition
- Exhaustively peer-reviewed by experts and editors
- Carefully structured for most effective learning by readers
- Field-tested in thousands of computer science and software engineering courses

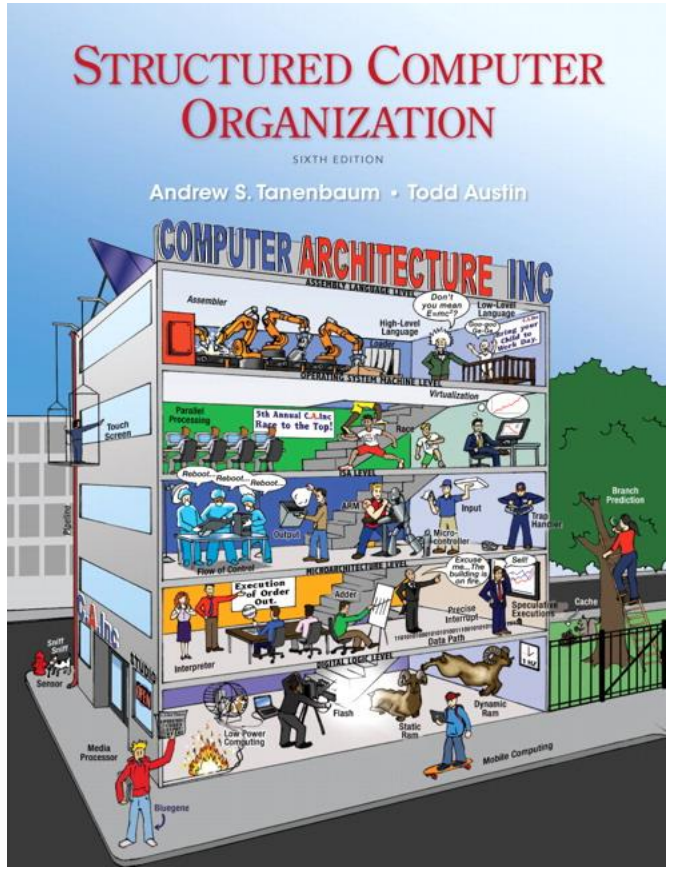


Image © Pearson

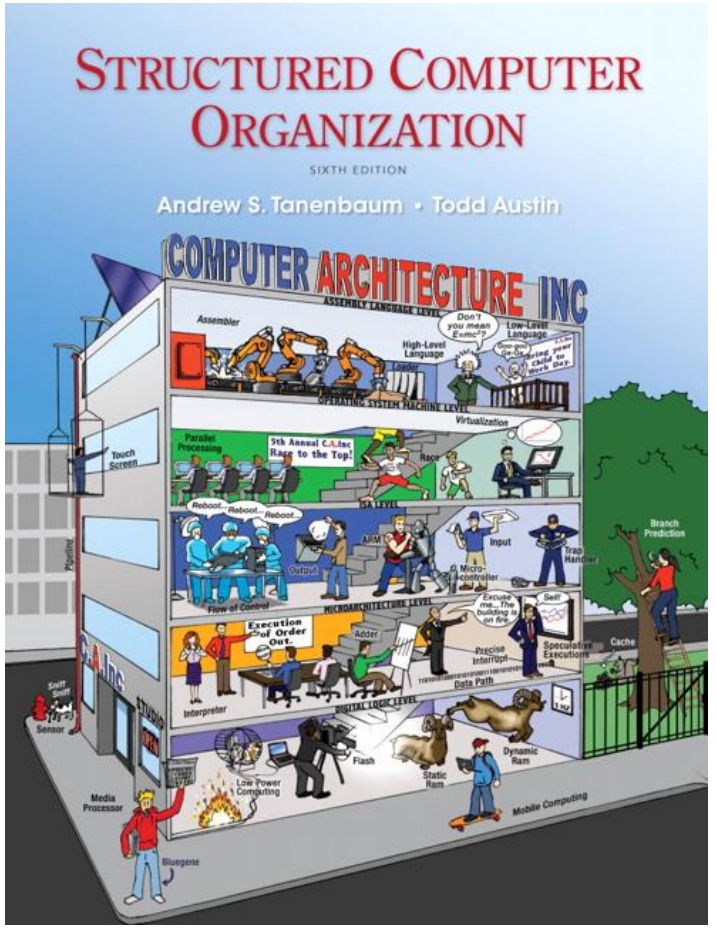
So….which of these do you think is the best option…

Google

how does a computer actually work lol😂

I'm Feeling Lucky

or



?

Image © Pearson

Adopting XYZ-lang: a hypothetical

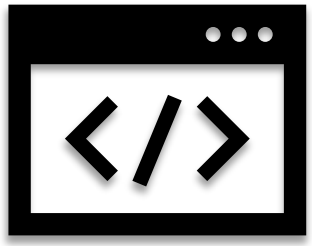
"Welcome to your new job! Oh, we've just adopted XYZ-lang here. Sorry, we forgot to mention that in the interview. It's great, you'll love it…"

“OMG I need to learn **XYZ-lang**…”

Books

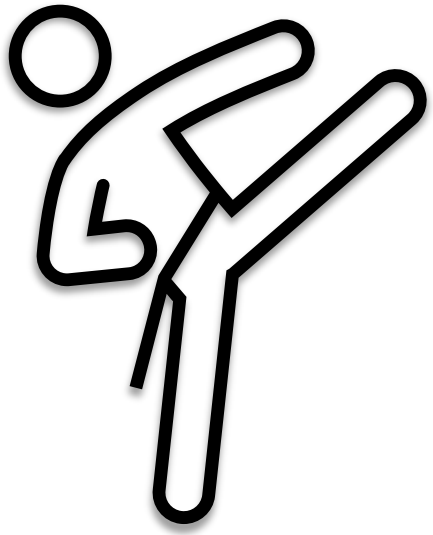
Online Courses

https://www.xyz-lang.org/

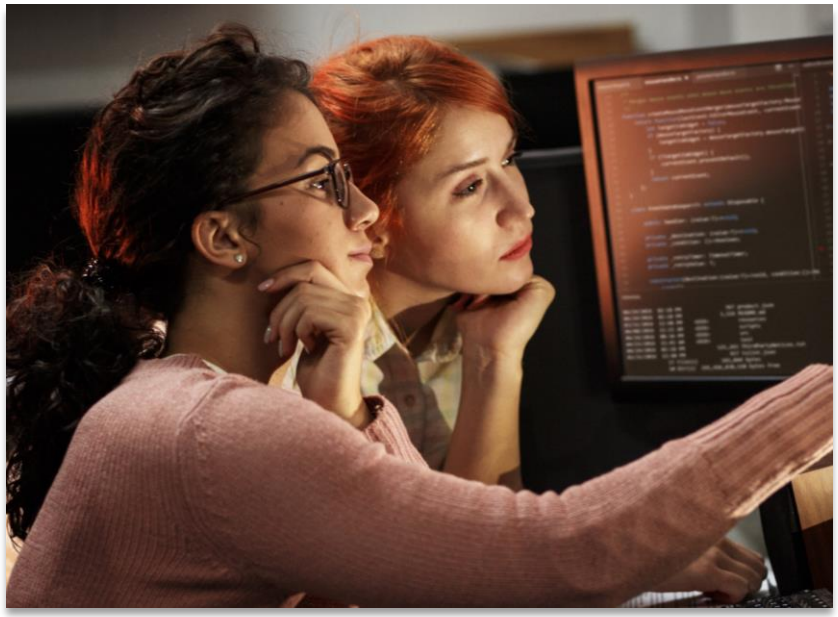


Official Site

“I wish I was more fluent in **XYZ-lang**…”



Kata Practice



Pair Programming

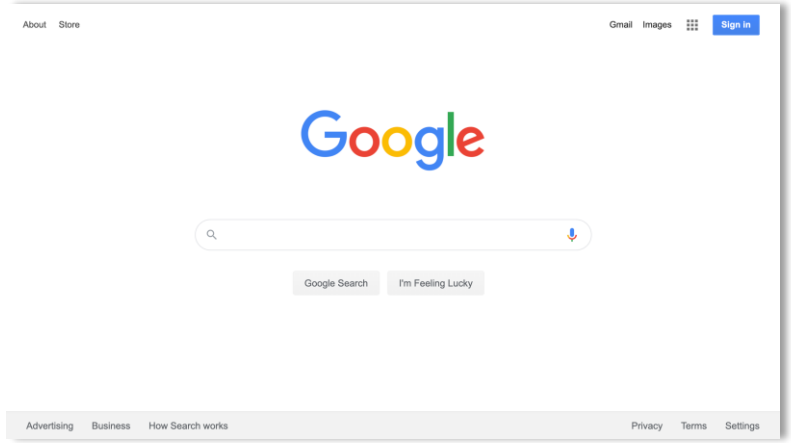
“I can work in **XYZ-lang**…but where now?”

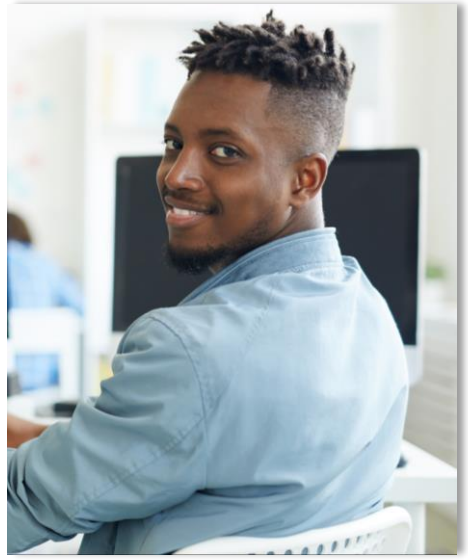
“Effective XYZ”

Advanced Courses

Conference Videos

“I can’t get this f🤬 thing working in **XYZ-lang**”





Q&A Sites

Web Searches

Colleagues

Yak Recognition: A Guide

Yak shaving

[MIT AI Lab, after 2000: orig. probably from a Ren & Stimpy episode.] Any seemingly pointless activity which is actually necessary to solve a problem which solves a problem which, several levels of recursion later, solves the real problem you're working on.

http://www.catb.org/jargon/html/

Full stack project with database, microservices, front end, user authentication and cloud deploy.

Build a microservice with a SQLite database, test using a REST client.

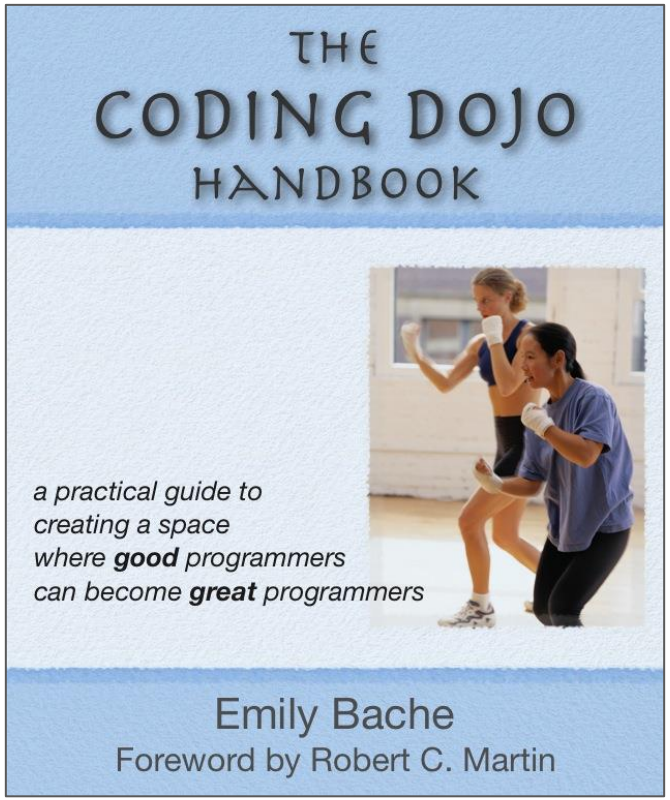
Simplest possible project, just code, unit tests and zero platforms or frameworks.


Full stack,
hairy yak

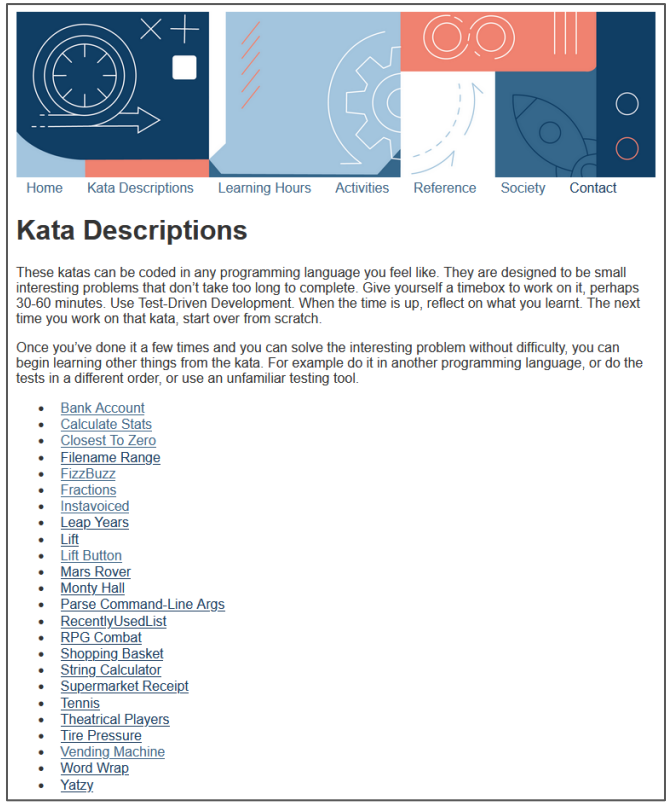
"You are bad at anything you don't practice."

-- Alex Bolboaca

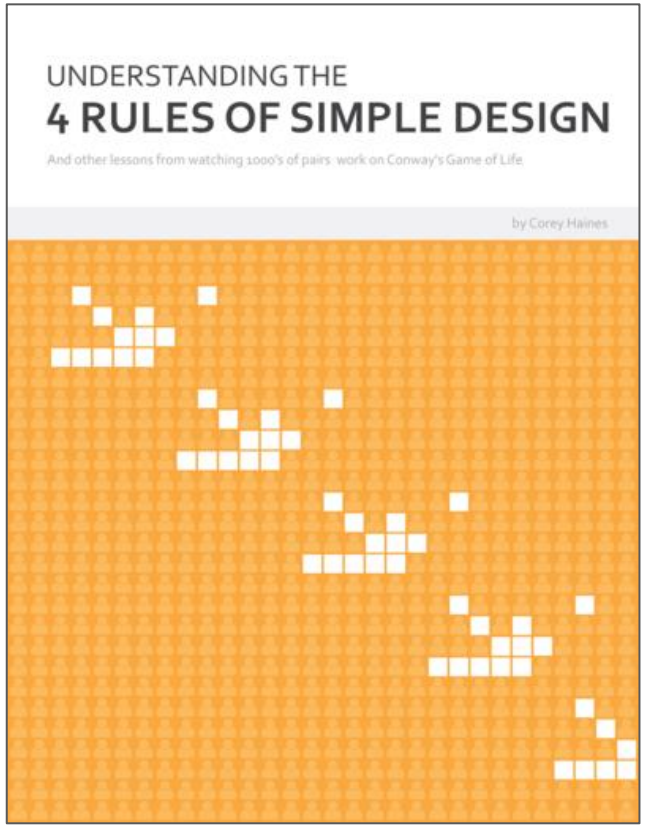
Deliberate Practice: yak-free experimentation



https://leanpub.com/codingdojohandbook



https://sammancoaching.org/kata_descriptions/index.html



https://leanpub.com/4rulesofsimpledesign

The goal of this program is to model a vending machine and the state it must maintain during its operation. How exactly the actions on the machine are driven is left intentionally vague and is up to the implementor.

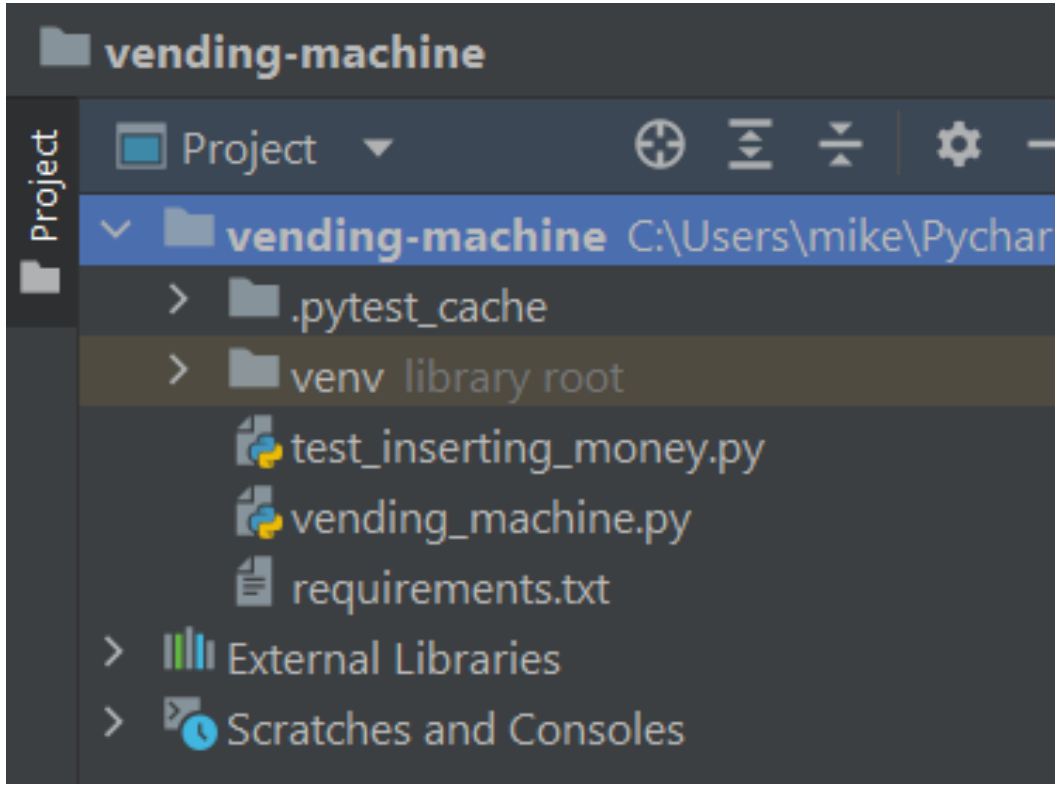
What does a vending machine do ?
The machine works like all vending machines : it takes money into a slot, then gives you items. The vending machine accepts money in the form of:
nickels (5 cents), **dimes** (10 cents), **quarters** (25 cents) and **dollars** (100 cents). Assume everything is a coin rather than a paper note for this kata, to keep things simple.

You must have at least 3 primary items that cost $0.65, $1.00, and $1.50. The user may hit a “coin return” button to get back the money they’ve entered so far. If you put more money in than the item’s price, you get change back.

Specification
The valid set of actions on the vending machine are:
NICKEL(0.05), DIME(0.10), QUARTER(0.25), DOLLAR(1.00) – insert money
COIN RETURN – returns all inserted money
GET-A, GET-B, GET-C – select item A ($0.65), B ($1), or C ($1.50)
SERVICE – a service person opens the machine and sets the available change and items

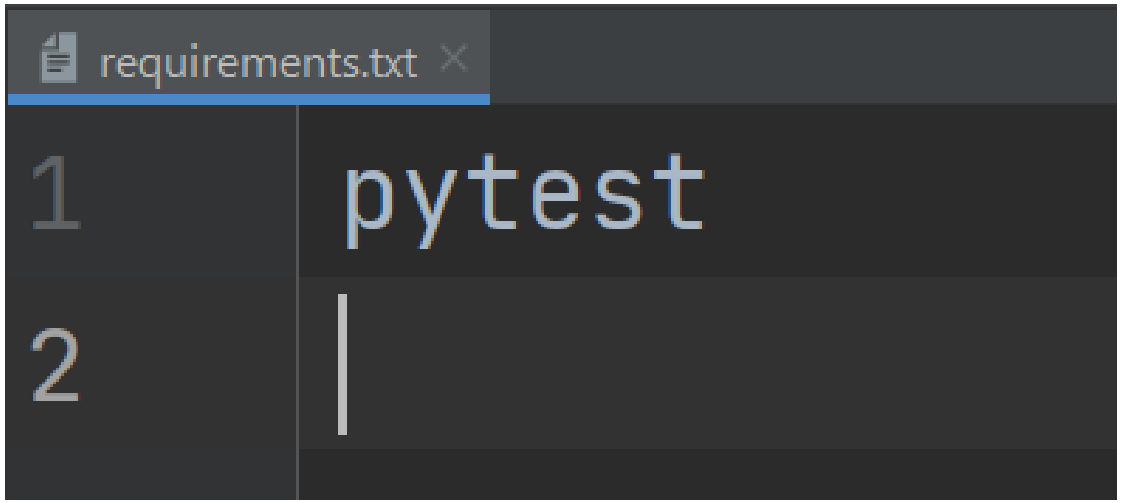
The valid set of responses from the vending machine are:
NICKEL, DIME, QUARTER – return coin
A, B, C – vend item A, B, or C

The vending machine must track the following state:
available items – each item has a count, a price, and a selector (A,B,or C)
available change – # of nickels, dimes, quarters, and dollars available
currently inserted money



Pure Vanilla Project

The most basic project structure your editor lets you create – for this Python example not even a main! We use the built-test runner only.



One Dependency

Try to limit your dependencies to only a unit test framework. If you're adding technology and frameworks here, you're getting it wrong.

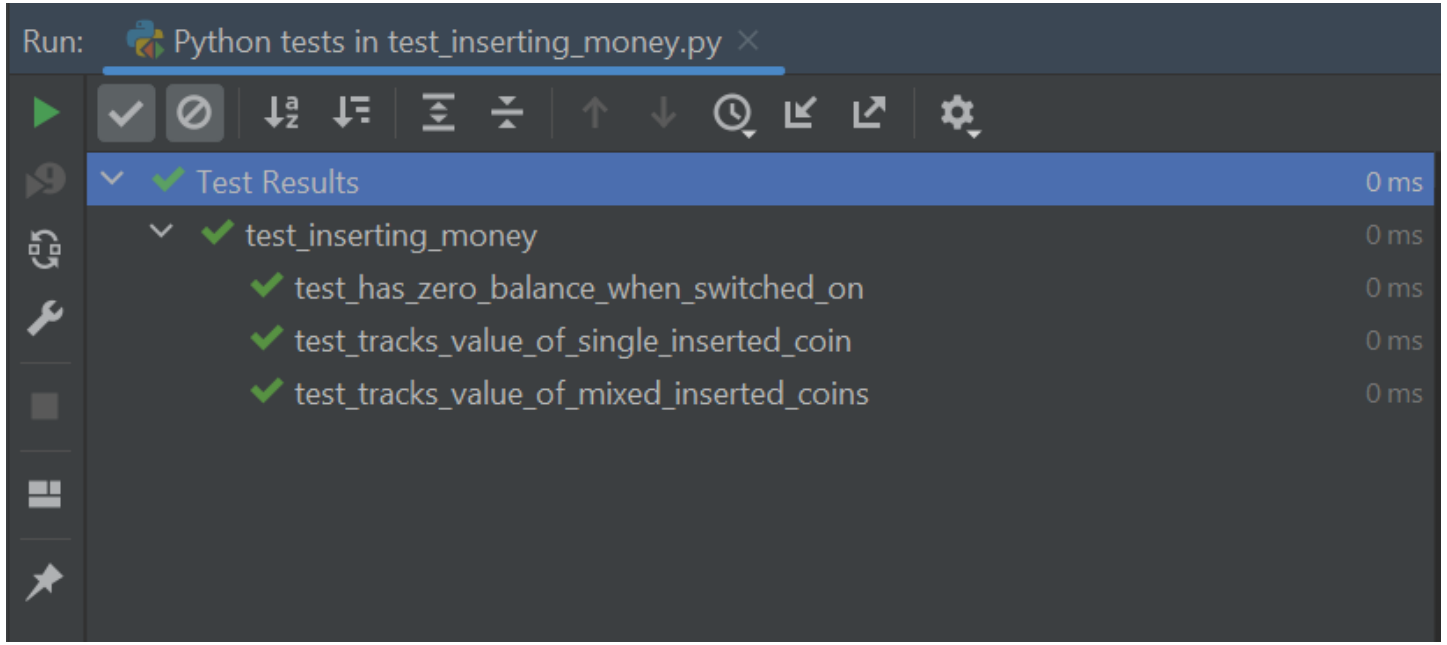
```python
def test_has_zero_balance_when_switched_on(machine):
    assert machine.get_balance() == 0


def test_tracks_value_of_single_inserted_coin(machine):
    machine.insert_money(Coin.NICKEL)
    assert machine.get_balance() == 5


def test_tracks_value_of_mixed_inserted_coins(machine):
    machine.insert_money(Coin.NICKEL, Coin.DIME,
                         Coin.QUARTER, Coin.DOLLAR)
    assert machine.get_balance() == 140
```

Baby Steps

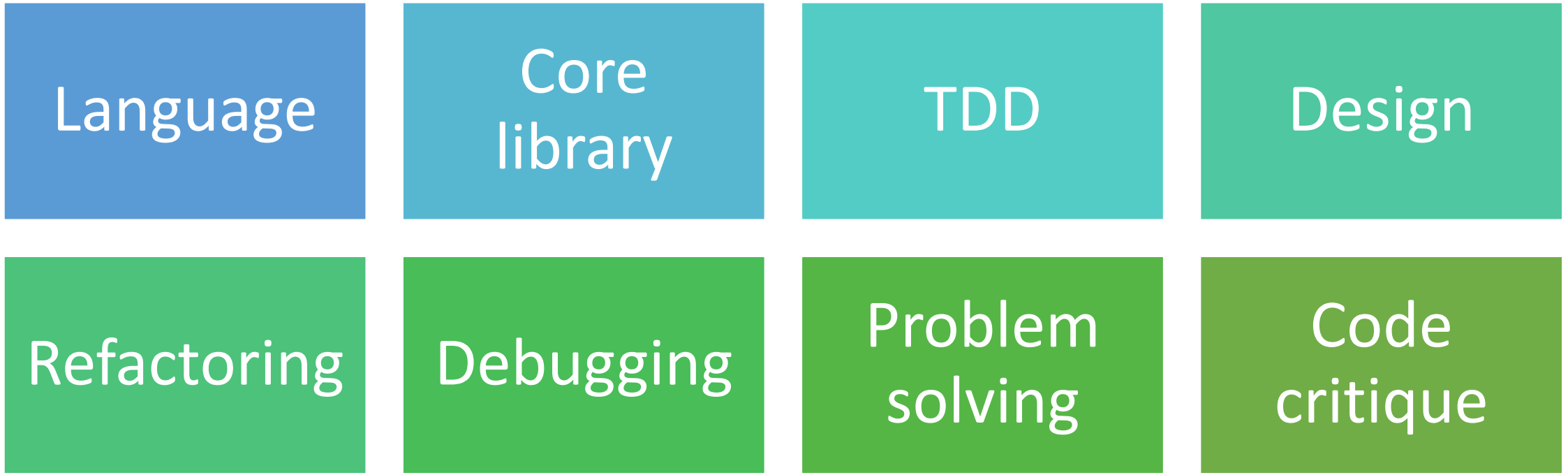
One test at a time, small increments, evolve your solution driven by those tests.

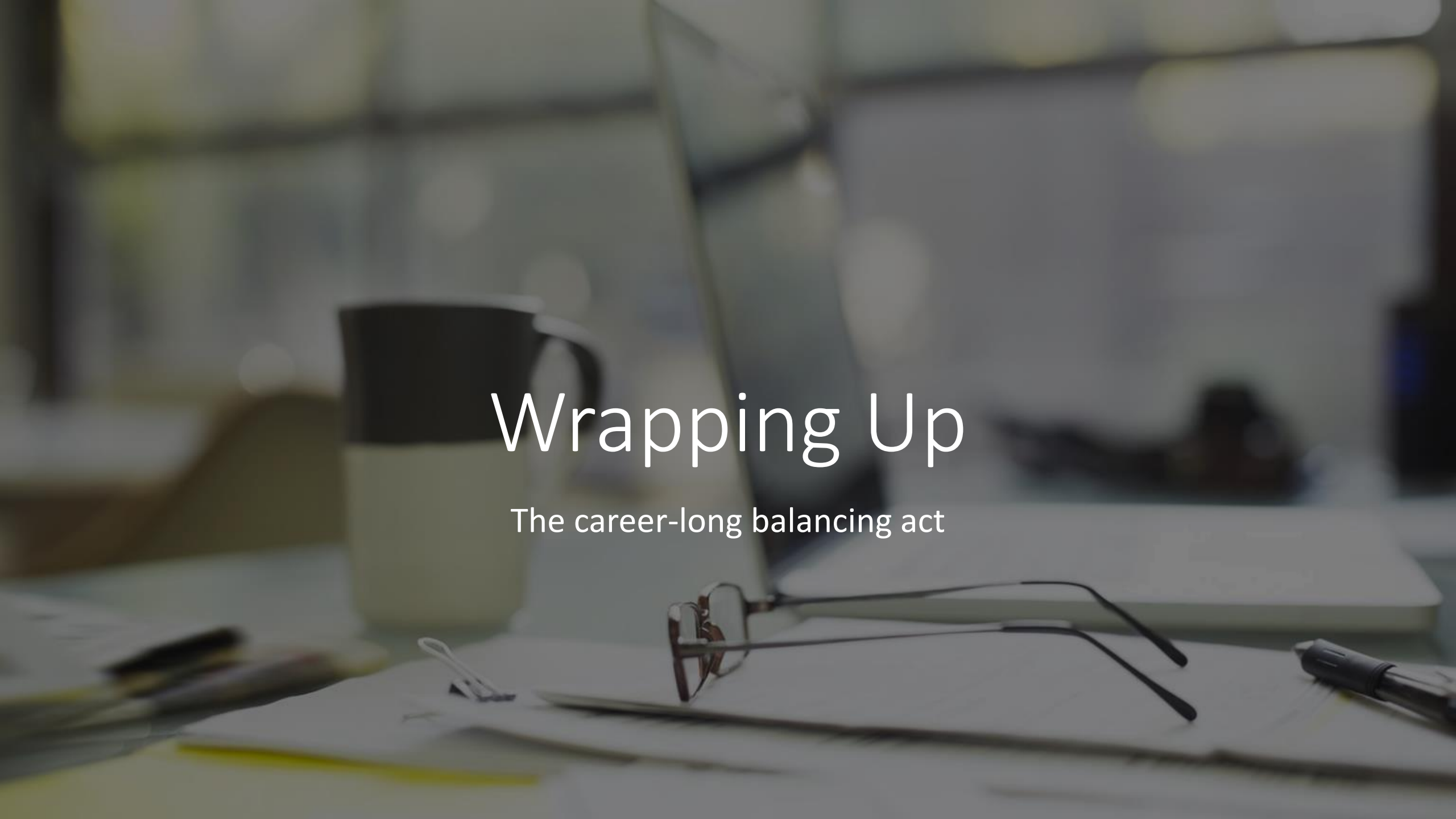


Rinse + Repeat

And that's it. TDD, software design, code, no platforms, no technologies, no frameworks, pure practice.

Deliberate Practice builds core fluency





Wrapping Up

The career-long balancing act

Making Work Work For You

- Make it your first priority to find mentors and coaches
- Pair program with experienced devs every chance you get
- Get other people invested in your development
- Actively seek out feedback. Feedback is your friend
- Ask for feedback on specifics, not generalities
- If you're stagnating, start the job hunt….

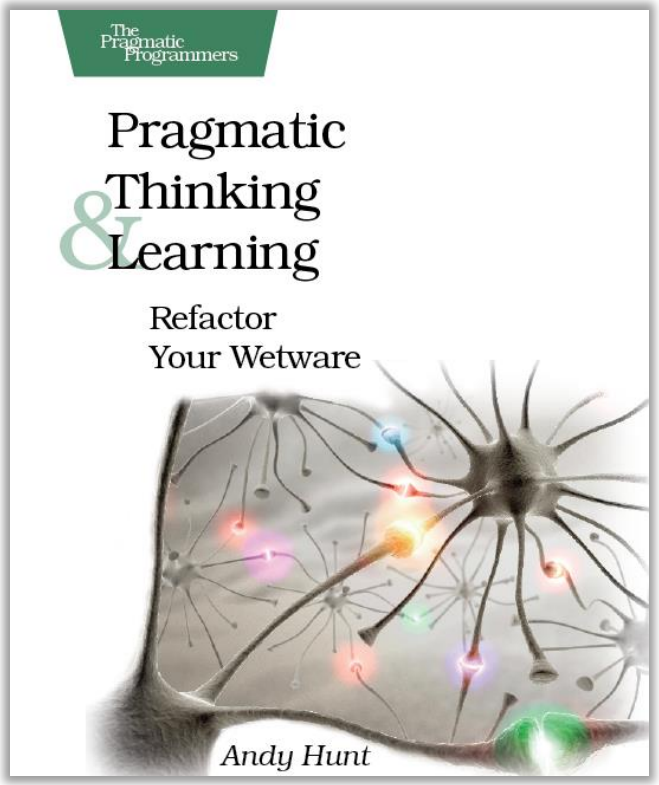
Sharpen the saw

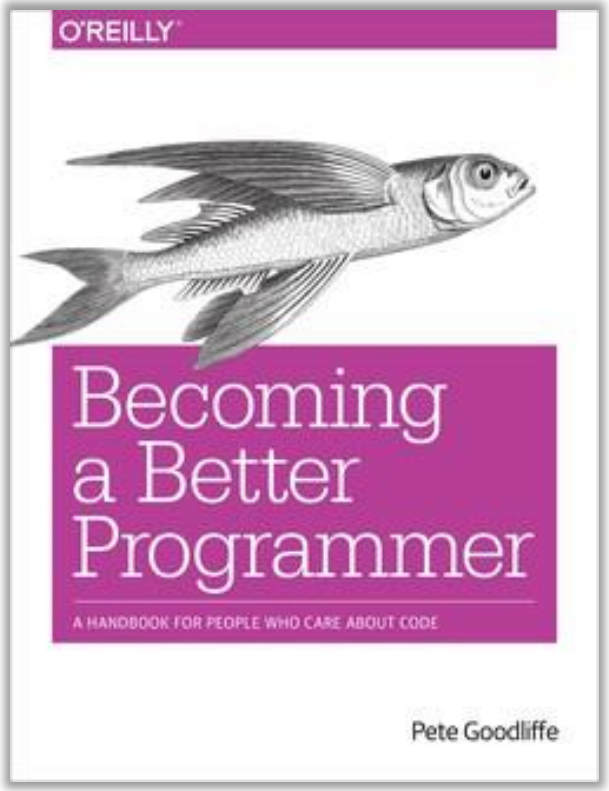
From Stephen. R. Covey's *The 7 Habits of Highly Successful People (1989).*

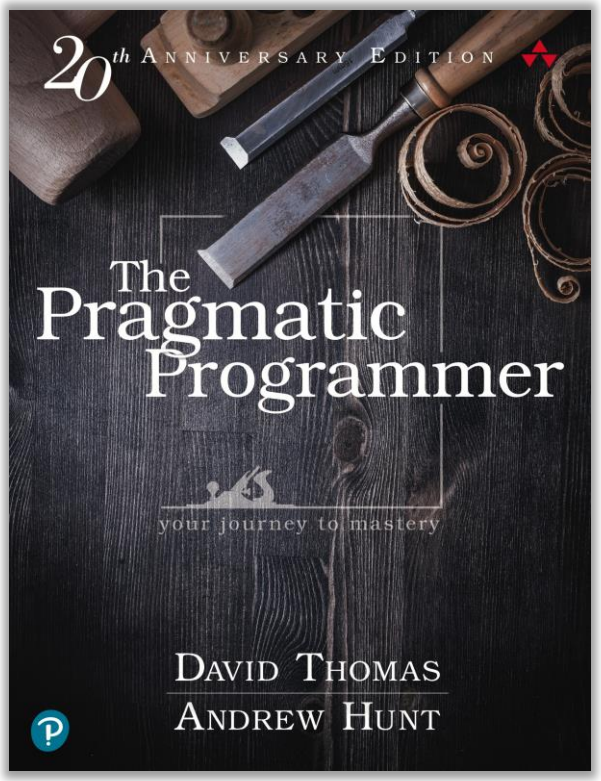
This captures the core idea behind deliberate, intentional learning.

"We must never become too busy sawing to take time to sharpen the saw."

Recommended reading







Questions please!