Agile Practices
For everybody - not just developers
Time Yourselves!

WOOT WOOT!!
How did you do?
This time, the rightmost column MUST sum to 15.

WOOT WOOT!!
The top row MUST ALSO sum to 9.

WOOT WOOT!!
The diagonal MUST ALSO sum to 18.

WOOT WOOT!!
Still just 9 digits and 4 lines, folks.

Why does it take longer now?
EFFORT: writing 9 digits

RISK: of new entry violating the constraints
Estimate how long this will take you.
UNCERTAINTY

EFFORT

RISK
Idea #1: The work is in the thinking, not the typing.
import itertools

combinations = {
    (sum(x), x)
    for x in itertools.combinations(range(1, 10), 3)
}

for total, sequence in sorted(combinations):
    print(total, sequence)
Could you write a spreadsheet that auto-summed?

How about increasing brains-per-puzzle?

Could you write a solver program?

How else would you “Adapt”?
Manifesto for Agile Software Development

Decision-making criteria.

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.
the right, we value the items on the left more.

Kent Beck          James Grenning          Robert C. Martin
Mike Beedle        Jim Highsmith          Steve Mellor
Arie van Bennekum  Andrew Hunt            Ken Schwaber
Alistair Cockburn  Ron Jeffries           Jeff Sutherland
Ward Cunningham     Jon Kern               Dave Thomas
Martin Fowler       Brian Marick

Twelve Principles of Agile Software

Become a Signatory
Principles behind the Agile Manifesto

We follow these principles:

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

Business people and developers must work together daily throughout the project.

Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Working software is the primary measure of progress.

Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

Continuous attention to technical excellence and good design enhances agility.

Simplicity—the art of maximizing the amount of work not done—is essential.

The best architectures, requirements, and designs emerge from self-organizing teams.

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.
Working software is the primary measure of progress.

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
Idea #2: Agile is about delivering value.
Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
Planning & Design: 2 Months
Code: 2 Months
Test: 5 Weeks
Release
How can you do a project in a couple of weeks?

These must be trivial projects!
100% of system
80% done
80% of system
100% done
Manager’s View

About 2 Weeks

If you are only working in 2 week periods...

Only plan 2 weeks worth.

Only deliver 2 weeks worth.

See what can happen in 2 weeks.
Plan → Develop/Test

Very short period limits exposure
Idea #3:

We don’t “do more.”
We don’t “go faster.”
We deliver sooner.
Seven-minute Sprint
Retrospective
Thin slicing?
Great, now I have EVEN MORE tasks to assign and manage!
We didn’t assign...

... we tend to not assign.
The Bee-Watcher-Watcher watched the Bee-Watcher. He didn’t watch well. So another Hawtch-Hawtcher had to come in as a Watch-Watcher-Watcher!
And today all the Hawtchers who live in Hawtch-Hawtch are watching on Watch-Watcher-Watchering-Watch, Watch-Watching the Watcher who’s watching that bee. You’re not a Hawtch-Watcher. You’re lucky, you see!
If I don’t push them, won’t they stop working?
Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
work. We start from the presumption that our people are talented and want to contribute. We accept that, without meaning to, our company is stifling that talent in myriad unseen ways. Finally, we try to identify those impediments and fix them.
No one (not even the Scrum Master) tells the Development Team how to turn Product Backlog into increments of Potentially releasable functionality.

-- The Scrum Guide
Pillory is Not an Agile Value
Brains per Task > Tasks per Brain
Pair Programming Illuminated

Laurie Williams and Robert Kessler
Business people and developers must work together daily throughout the project.

Customer collaboration over contract negotiation
Responding to change over following a plan

Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
What about estimating, planning poker, story points?
Not today.
Manager’s View

About 2 Weeks

If you are only working in 2 week periods...

Only plan 2 weeks worth.

Only deliver 2 weeks worth.

See what can happen in 2 weeks.
Easily said, but how do I decide how much the team can do in two weeks?
Idea #4:

Capacity (velocity) is a consequence, not a choice.
Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
Backlog (project context)
Most “important”
Ermigersh! Unclaimed Capacity!
Crossword Creators!
Goal: help students identify english words by first letter.

- Common word-starting letter
- Intersects needed (as clues)
- Long words (5+) need 2 or more intersects
- 2-7 Words Per Puzzle
- Must be fully playable at all times
- Good UX for students
But all our stories are too big for a sprint!
Shrink XL Stories to Fit

- Defer alternate paths, edge cases, or error cases
- Defer supporting fields
- Defer side effects
- Stub dependencies
- Split operationally (for example, CRUD)
- Defer nonfunctional aspects
- Verify against audit trail
- Defer variant data cases
- Inject dummy data
- Ask the customer
<table>
<thead>
<tr>
<th>Release</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>First Release</td>
<td>&quot;Walking skeleton&quot;</td>
</tr>
<tr>
<td>Second Release</td>
<td></td>
</tr>
<tr>
<td>Third Release</td>
<td></td>
</tr>
<tr>
<td>Fourth Release</td>
<td></td>
</tr>
</tbody>
</table>
Do you finish one story before starting the next?
Story slices *will* compete for priority

Later:
Idea #5:

A feature is “done” when it’s not the most important thing anymore.
We can shrink or drop stories, but can’t we also increase capacity?
YOU KEEP USING THAT WORD
I DO NOT THINK IT MEANS WHAT YOU THINK IT MEANS
<table>
<thead>
<tr>
<th>To Increase</th>
<th>You Must:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Speed</strong></td>
<td>Increase Effort</td>
</tr>
<tr>
<td></td>
<td>Cut corners</td>
</tr>
<tr>
<td></td>
<td>Take chances</td>
</tr>
<tr>
<td></td>
<td>Abandon early</td>
</tr>
<tr>
<td><strong>Capability</strong></td>
<td>Develop skills</td>
</tr>
<tr>
<td></td>
<td>Increase Knowledge</td>
</tr>
<tr>
<td></td>
<td>Improve tools</td>
</tr>
<tr>
<td></td>
<td>Share work efficiently</td>
</tr>
<tr>
<td></td>
<td>Reduce waste</td>
</tr>
</tbody>
</table>
At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.
Idea #6:

Agile methods are empirical
Idea #7:

Agile is an alternative, advantageous ruleset (AKA: “cheating”).
Simplicity--the art of maximizing the amount of work not done--is essential.
Idea #8:

If something is hard, we do it more often and automate it to death.
Continuous attention to technical excellence and good design enhances agility.