Agile Practices
For everybody - not just developers
Time Yourselves!

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

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The top row MUST ALSO sum to 9.

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

Side-Effects

Constraints

Side-Effects

SIDE-EFFECTS

GOAL
Idea #1:
The work is in the thinking, not the typing.
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.

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.
THIS IS TOO MUCH
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.
So, what do you really do?
About 2 Weeks

Pick some small period of time.

*(scrum/xp example)*
Pick *only* enough work for the period.
Make it work end-to-end.
Plan → Develop/Test

Very short period limits exposure
Idea #3: We don’t “do more.” We don’t “go faster.” We deliver sooner.
CONTINUOUS DEPLOYMENT
The **magic** seems to be in the task slicing.
Oh, great. MORE tasks to assign and manage!
“Cross-functional teams have all competencies needed to accomplish the work without depending on others not part of the team.”

-Scrum Guide 2013
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
Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
<table>
<thead>
<tr>
<th>Available</th>
<th>Working</th>
<th>Done</th>
</tr>
</thead>
<tbody>
<tr>
<td>clean sanctuary</td>
<td>clean</td>
<td></td>
</tr>
</tbody>
</table>

People do the parts they want to do
Or the parts they want to learn.
We let them.
If I don’t push them, won’t they stop working?
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!
It's Official: Micromanaging Kills Productivity

BY JILL KRASNY @JILLKRASNY

New research suggests the more you try to dictate how and when employees work, the less they will accomplish.
3 FACTORS LEAD TO BETTER PERFORMANCE & PERSONAL SATISFACTION...

AUTONOMY
MASTERY
PURPOSE

PURPOSE

Daniel H. Pink
DRiVE
The Surprising Truth About What Motivates Us
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.
Brains per Task > Tasks per Brain
Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
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.
a 2:27
b 1:48
c 2:53
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.
"It is not the strongest of the species that survives, nor the most intelligent that survives. It is the one that is most adaptable to change".

Charles Darwin
What about estimating, planning poker, story points?
Backlog (project context)
Most “important”
Backlog

To Do
Ermigersh! Unclaimed Capacity!
But all our stories are too big for a sprint!
FEATURE

Story Splitting

Story
Story
Story
Story
Story
Story
Story
Story
Story
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
First release
"walking skeleton"
Second Release
Third Release
Fourth Release
Do you finish one story before starting the next?
Story slices *will* compete for priority.
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?
<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><strong>Capacity</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
Learning & experimenting takes *time*, and we are always in a time crunch!
I was told to expect *hyper-productivity*. 
Few implementations of Scrum achieve the hyperproductive state for which Scrum was designed (5-10 times normal performance).

Those that do all implement variations on XP engineering practices and many, like the CMMI Level 5 implementation of Scrum (which has not achieved the hyperproductive state yet) drive the whole implementation from a lean perspective.

http://www.scruminc.com/origins-of-scrum/
IF you see the first scrum at Easel Corporation implemented by Jeff Sutherland, it incorporated all the activities of extreme programming’s engineering practices.

All the most high-performance teams use Scrum and XP together. It is hard to get a Scrum with extreme velocity without XP engineering practices.

How about increasing brains-per-puzzle?

Could you learn all the ways unique add up to a given sum?

Could you write a solver program?

Could you write a spreadsheet that auto-summed?

How else would you "Adapt"?
import itertools

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

for total, sequence in sorted(combinations):
    print(total, sequence)
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.
SAFE SOFTWARE DEVELOPMENT
Continuous attention to technical excellence and good design enhances agility.