Strive
SCHOOL

Surprise ~~🎉~~

JetBrains

M1 + M2

Build Week - Day 2

M1-2 Extra: Python + Math

/ In the first four weeks, we explored Python, Pandas, Matplotlib and some areas of Calculus, Statistics and Probability

/ We took a lot of things in, now we need to make them settle, to recap and to make them natural before moving to the ML part

/ Many of you asked for MORE time to work on the projects

Why?

- To make this tech “solid” before moving to the ML part
- To improve our understanding of git and GitHub
- To recap all the topics from M1 and M2

What [Notion]

Build Week

How?

Build Week

1 week-long-project

- We need to work with a “complete” project, from the import to presenting it
- We need to learn by doing, step by step, piece by piece
- Try to organize to create the right code and tasks organization
- It WILL BE very challenging!

Teams

/ You are gonna work in teams

/ You should now right now how to do it

/ If you have any problem with your teammates, just report it to us so that we can arrange / swap find somebody else more suitable

/ This won't be pair programming, you're gonna be working probably on separate stuff but you'll be helping each other learning by doing and sharing

All day tutoring

/ Tutors and TAs will be available each day from kick off to debrief

/ Each team can ask for help both on Discord or by setting up a meeting with calendly

/ Please be prepared for the meeting:

> Be ready to explain which is the problem
> Prepare a list of questions you want to ask
> Push your code's latest version BEFORE asking for help, so that the tutor can check immediately

Today

09:00	
	Kick Off
09:30	
	{ CODE && Tutoring }
13:00	
	Lunch
14:00	
	{ CODE && Tutoring }
17:00	
	Debrief / Q&A
18:00	

Demo Day

/ On Friday you're gonna present your projects

/ It does not matter if they are finished or not, it's a benchmark / challenge

/ Suggestions:

> Finish stuff. Don't try to do EVERYTHING from the beginning, just focus on one part, finish and move on

> Learn to collaborate with GitHub. Versioning, branching and merging will be part of your daily job as a developer. Learning now will be a big plus for you all

In a nutshell

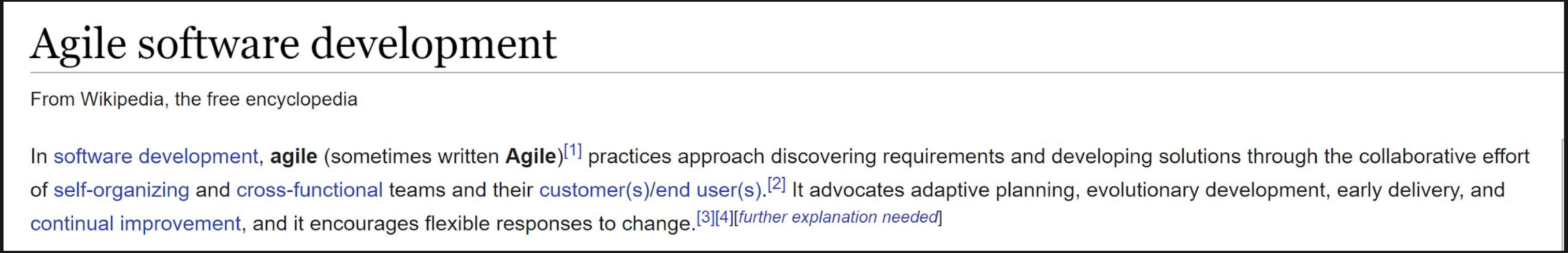
/ 1-week-long-project

/ In Team

/ No lectures || new topics

/ Friday ⇒ Demo!

What?



Agile software development

From Wikipedia, the free encyclopedia

In software development, **agile** (sometimes written **Agile**)[1] practices approach discovering requirements and developing solutions through the collaborative effort of self-organizing and cross-functional teams and their customer(s)/end user(s).[2] It advocates adaptive planning, evolutionary development, early delivery, and continual improvement, and it encourages flexible responses to change.[3][4][*further explanation needed*]

/ Agile frameworks: SCRUM - Daily Standup Meetings

/ 10 minute-ish time-boxed event where the development team to plan for the next 24 hours

/ "The Scrum Field Guide" by Mitch Lacey: "what's my level of confidence in reaching the Sprint Goal?"

What?

/ The three common questions in a daily standup:

1. What did you do yesterday?
2. What will you do today?
3. Is there anything blocking your progress?

What?

/ The team gets on the same page on what's been done and may discover:

- what still needs to be done
- do our plans for today change based on yesterday's results?

/ The team will know if they are on track or something needs to be changed

/ Teammates get the change to help each other removing those blocking situations

Why?

/ Agile software development, compared to traditional software engineering, uses a dynamic system that adapts quickly to changes and reduces the leap-of-faith required before any evidence of value can be obtained.

/ Breaks development into small increment, that minimize the amount of up-front planning and design.

/ Developed in a short time frame called Sprint that typically last two weeks

Why?

/ A non-agile method could potentially cause the initial prediction to be very wrong and make it difficult if not impossible to change direction when the project is already too far in the development phase.

Common pitfalls

/ MISALIGNMENT: deviate too much, talking about non relevant issues that will make teammates check-out mentally and lose potentially relevant updates.

/ TOO LENGTHY: people start rambling after a while or start a cafeteria-like conversation resulting in an ineffective meeting.

Common pitfalls

/ PROBLEM-SOLVING DURING STAND UP: people could start trying to solve problems during the standup instead of afterwards.

/ INCONVENIENT TIME: could happen in an unfortunate time, while resolving important issues. It's also comes with a time cost to have the whole team present even if they have different schedules or time-zones.

Common pitfalls

/ CAN MAKE INTROVERTS UNCOMFORTABLE: some people overshare informations and take pride in that, while others will have the opposite effect. They may shy away, not go into enough details or not share anything at all.

/ NOT LISTENING TO TEAMMATES: it’s common for some to rehearse what they need to say and miss out on valuable information by not paying attention.

Common pitfalls

/ SKIPPING STANDUPS: not having a consistent meeting cadence can lead to people skipping or forgetting the standups.

/ NOT RAISING BLOCKING ISSUES: if somebody is too embarrassed or uncomfortable during the meeting they'll probably omit that they are in trouble.

In the end..

/ Someone may think standups are a waste of time. If that’s the case it’s probably because they are not done properly.

/ They are a common practice in tech companies either with an on-site or remote team.

/ You’ll get to appreciate the benefits over time, especially if you work remotely, it’s a great tool to be in sync with everyone in your team, without leaving anyone out.

Q&A

Don’t be SHY

Strive
SCHOOL