1. Agile is a type of software development that revolves around more than just frameworks, models, or practices. Agile is a type of software development approach that focuses on the ability to adapt to change, as well as the interaction between the working developers and how that interaction takes place. Self-organization is of extreme importance with agile development. Customer requirements are not compatible with agile as well, as contract negotiation is not a key value.
2. The quote, “I follow the Agile methodology. I find it very easy. It is a well-defined collection of steps and decisions that I simply need to follow. There is not much social interaction or thinking needed from my side except for the coding,” is incorrect in its entirety. For one, the Agile methodology does not involve a concrete set of planning or documentation. Social interaction is also an integral component of this methodology; independent coding is not something one would see very often.
3. The 12 Principles of Agile:
   1. The early delivery of software to the client followed by consistent updates and added features is top priority.
   2. Adaptability is extremely important, with a change in requirements even late in development being completely acceptable.
   3. Release function software over a preferably short period of time, ranging from a few weeks to a few months.
   4. Cooperation and collaboration among developers and other business workers is a daily essential during project development.
   5. Developers of a project must be motivated and trusted to complete the work they were assigned, and should be accommodated with any and all resources they require.
   6. Communication across the development team is best done via face-to-face conversation.
   7. A software’s current functionality is the best measurement of its development progress and resulted functionality.
   8. A steady, unchanging pace in development speed is extremely important in the Agile development process; any changes should not cause significant change in such.
   9. Focus on good design and technical quality should remain constant.
   10. Simplicity is crucial; any work needing development should be maximized at all times.
   11. Self-organized development teams output the best results in software quality.
   12. Self-reflection within the team should occur regularly. Any and all behaviors needing change should be acted upon immediately.
4. The quote, “Lines of code is a valid measure progress since we, the developers, are spending time and resources to write it,” is not entirely accurate when referring to Agile development. According to the seventh principle, a software’s measure of progress is best represented in said software’s current functionality. In other words, the measurement of progress is best recorded in how close the software works in comparison to the customer’s feedback.
5. In its most basic definition, to maximize the amount of work not done means to make continuous efforts for improvement, no matter how small the possibilities for such might be. Different tasks are differed to exterior sources for the purpose of having as many hands on deck as possible. While many may consider this cheating, I suppose I would consider it to be a prioritization of efficiency. The Agile process is about getting as much progress done as possible, which this accomplishes.
6. Extreme programming (XP) is a framework labeled under the Agile software development umbrella. The processes under XP involve changing requirements and the ability to adapt to the risks provided by deadlines. This framework isn’t utilized in most software development scenarios, but can still be extremely useful.
7. Extreme programming, as previously mentioned, is a framework of Agile, and are therefore not entirely comparable. For example, while both describe methodologies that heavily involve the client during the development process, extreme programming does not involve any middle-man between the two. In other words, extreme programming involves direct contact between developer and client, whereas most Agile frameworks do not.
8. The 12 practices of extreme programming are as follows: on-site customer, the planning game, short releases, metaphor, collective ownership, continuous integration, coding standards, sustainable pace, testing, simple design, refactoring, and pair programming. These practices are very similar to the 12 principles of Agile in many ways, namely the methodology in which they follow. Some of the listed practices are also seen in the Agile principles, such as continuous integration, sustainable pace, and simple design. Both XP and Agile center around the idea of a collaborative effort among a team’s developers, and working together to produce the best software possible.
9. I believe that to follow extreme programming means to be following Agile at its core. XP is in fact a framework of Agile itself, and to try and follow XP without following the summative methodology of Agile itself would be incorrect. XP is designed to be agile. A rigid structure would not work with the defined practices.
10. Extreme programming is in fact iterative, as functioning software is designed to be release over short increments of time typically spanning one-to-two weeks. This time span is, in fact, shorter than the amount of time defined by the standard Agile principles.
11. User stories are extremely critical for a development team following the XP methodology, as it allows for said team to shift its primary focus on what the client wants fixed the most. These values are extremely important within the XP model.
12. I believe that pair programming can be especially helpful if and only if an extra set of hands isn’t needed elsewhere. Working in a pair allows for ideas to be bounced back and forth, and can result in some very efficient progress in what is being worked on. Two developers can have two very different approaches to a single problem, and finding the best, most efficient solution is of the upmost importance.