1. Sit Together

¿Qué? Trabajar en proximidad con otros para facilitar la comunicación y colaboración.

¿Cómo?

1. **Grupo de estudio:** Organizar reuniones de estudio presenciales o virtuales con compañeros de clase.
2. **Espacio compartido:** Estudiar en la biblioteca o en un espacio común donde pueda interactuar con otros estudiantes.
3. **Clases en grupo:** Asistir a clases y talleres en grupo para discutir y resolver dudas conjuntamente.
4. **Ejemplo adicional:** Crear grupos de estudio en línea si no es posible reunirse físicamente.

2. Whole Team

¿Qué? Incluir a todas las partes interesadas en el proceso de estudio y trabajo.

¿Cómo?

1. **Participación activa:** Involucrar a todos los miembros del grupo de estudio en las discusiones y decisiones.
2. **Reuniones con profesores:** Programar reuniones regulares con profesores para recibir orientación y feedback.
3. **Colaboración:** Trabajar con compañeros de clase para asegurarse de que todos entiendan los temas y progresen juntos.
4. **Ejemplo adicional:** Invitar a tutores o exalumnos para compartir sus experiencias y consejos.

3. Informative Workspace

¿Qué? Crear un espacio de trabajo que ofrezca información útil de manera visible.

¿Cómo?

1. **Tablero de tareas:** Utilizar un tablero (físico o digital) para visualizar las tareas y fechas de entrega.
2. **Calendario de estudio:** Tener un calendario visible con fechas importantes y horarios de estudio.
3. **Notas y recordatorios:** Colocar notas y recordatorios sobre temas clave y conceptos importantes en el área de estudio.
4. **Ejemplo adicional:** Utilizar herramientas como Trello o Asana para gestionar tareas y plazos.

4. Energized Work

¿Qué? Mantenerse energizado y motivado durante el estudio.

¿Cómo?

1. **Descansos regulares:** Tomar descansos regulares durante las sesiones de estudio para mantener la concentración.
2. **Ambiente positivo:** Crear un ambiente de estudio agradable con buena iluminación y ventilación.
3. **Ejercicio físico:** Incorporar actividad física en la rutina diaria para aumentar la energía y reducir el estrés.
4. **Ejemplo adicional:** Escuchar música motivadora o relajante mientras estudias para mantenerte energizado.

5. Pair Programming

¿Qué? Trabajar en parejas para mejorar la calidad del trabajo y el aprendizaje.

¿Cómo?

1. **Estudio en parejas:** Estudiar en parejas para discutir y resolver problemas juntos.
2. **Revisión de trabajos:** Intercambiar trabajos con un compañero para revisarlos y ofrecer retroalimentación.
3. **Proyectos conjuntos:** Realizar proyectos en parejas para compartir conocimientos y habilidades.
4. **Ejemplo adicional:** Practicar ejercicios de programación o resolución de problemas en parejas.

6. Stories

¿Qué? Utilizar historias para definir y entender el trabajo a realizar.

¿Cómo?

1. **Historias de usuario:** Escribir historias de usuario para definir qué se espera aprender o lograr en cada tema.
2. **Metas claras:** Establecer metas claras para cada sesión de estudio basadas en las historias de usuario.
3. **Progreso:** Evaluar el progreso basándose en cómo se completan estas historias.
4. **Ejemplo adicional:** Utilizar casos de estudio o ejemplos prácticos para entender mejor los conceptos teóricos.

7. Weekly Cycle

¿Qué? Planificar y revisar el trabajo semanalmente.

¿Cómo?

1. **Planificación semanal:** Planificar las tareas y objetivos de estudio para cada semana.
2. **Revisión semanal:** Revisar el progreso al final de cada semana y ajustar el plan si es necesario.

3. **Tareas pequeñas:** Dividir las tareas grandes en actividades manejables que puedan completarse en una semana.
4. **Ejemplo adicional:** Utilizar reuniones de grupo semanales para revisar y ajustar los planes de estudio.

8. Quarterly Cycle

¿Qué? Planificar a largo plazo y revisar trimestralmente.

¿Cómo?

1. **Metas trimestrales:** Establecer metas a largo plazo para cada trimestre académico.
2. **Revisión trimestral:** Revisar el progreso al final de cada trimestre y hacer ajustes según sea necesario.
3. **Evaluaciones:** Programar evaluaciones y autoevaluaciones trimestrales para medir el progreso.
4. **Ejemplo adicional:** Utilizar el final de cada trimestre para reflexionar sobre lo aprendido y planificar el próximo ciclo.

9. Slack

¿Qué? Dejar tiempo adicional en el horario para adaptarse a cambios y sorpresas.

¿Cómo?

1. **Tiempo extra:** Incluir tiempo adicional en el calendario para imprevistos y repaso.
2. **Flexibilidad:** Ser flexible con el plan de estudio para adaptarse a cambios en el horario o nuevas prioridades.
3. **Días de descanso:** Programar días sin estudio para relajarse y recuperarse.
4. **Ejemplo adicional:** No sobrecargar el horario, dejando margen para emergencias o necesidades personales.

10. Ten Minute Build

¿Qué? Automatizar y optimizar procesos para ser más eficiente.

¿Cómo?

1. **Resúmenes rápidos:** Crear resúmenes rápidos y eficientes de cada tema estudiado.
2. **Revisiones rápidas:** Realizar revisiones rápidas de los apuntes y materiales después de cada clase.
3. **Herramientas digitales:** Utilizar herramientas digitales para tomar y organizar notas de manera eficiente.
4. **Ejemplo adicional:** Desarrollar una rutina de estudio que permita empezar rápidamente y ser productivo en poco tiempo.

11. Continuous Integration

¿Qué? Integrar y revisar el trabajo continuamente para identificar y corregir errores temprano.

¿Cómo?

1. **Revisiones continuas:** Revisar y actualizar los apuntes y trabajos constantemente para mantenerlos correctos y completos.
2. **Feedback frecuente:** Buscar retroalimentación constante de profesores y compañeros sobre el progreso.
3. **Pruebas frecuentes:** Realizar autoevaluaciones y pruebas frecuentes para identificar áreas de mejora.
4. **Ejemplo adicional:** Utilizar herramientas de seguimiento del progreso para mantenerse al día y corregir el curso rápidamente.

12. Test-First Programming

¿Qué? Definir objetivos y criterios de éxito antes de empezar a trabajar.

¿Cómo?

1. **Objetivos claros:** Establecer objetivos claros y específicos para cada sesión de estudio.
2. **Criterios de éxito:** Definir criterios de éxito para cada tarea o tema antes de comenzar.
3. **Autoevaluaciones:** Realizar autoevaluaciones al final de cada sesión para verificar el cumplimiento de los objetivos.
4. **Ejemplo adicional:** Utilizar rúbricas o listas de verificación para asegurarse de que se cumplan todos los criterios de éxito.

13. Incremental Design

¿Qué? Diseñar y mejorar el trabajo de manera incremental.

¿Cómo?

1. **Estudio progresivo:** Dividir el estudio en etapas y abordar cada una de manera incremental.
2. **Mejoras continuas:** Mejorar y refinar los métodos de estudio y los materiales de manera continua.
3. **Iteración:** Iterar sobre los proyectos y tareas, mejorándolos en cada ciclo.
4. **Ejemplo adicional:** Aplicar el enfoque incremental a la preparación para exámenes, revisando y mejorando el conocimiento con cada repaso.

b)

Principio	Objetivo	Ejemplos prácticos
1. Sit Together	Facilitar la comunicación y colaboración directa entre los miembros del equipo.	Colocar a todo el equipo en el mismo espacio físico, áreas de trabajo abiertas.
2. Whole Team	Involucrar a todos los roles necesarios en el desarrollo de software, incluyendo clientes.	Incluir desarrolladores, testers, diseñadores y clientes en el equipo.
3. Informative Workspace	Crear un espacio de trabajo que proporcione información clave y fomente la transparencia.	Tableros Kanban, gráficos de burndown, pantallas mostrando el estado de la CI.
4. Energized Work	Asegurar que el equipo trabaje en un entorno que promueva la energía y la motivación.	Fomentar un equilibrio entre la vida laboral y personal, ofrecer descansos regulares.
5. Pair Programming	Promover la calidad del código y el aprendizaje mediante la programación en parejas.	Dos desarrolladores trabajando juntos en una sola estación de trabajo.
6. Stories	Utilizar historias de usuario para capturar requisitos de manera que sean comprensibles y manejables.	Escribir historias de usuario con criterios de aceptación claros, involucrar a clientes en su creación.
7. Weekly Cycle	Planificar y realizar entregas de trabajo en ciclos semanales para obtener retroalimentación rápida.	Reuniones de planificación semanales, entregas incrementales al final de cada semana.
8. Quarterly Cycle	Establecer metas a largo plazo y revisar el progreso trimestralmente.	Planificación de objetivos trimestrales, revisiones cada tres meses.
9. Slack	Incluir tiempo adicional en las iteraciones para manejar imprevistos y fomentar la creatividad.	Asignar tiempo para aprendizaje, experimentación o resolver problemas inesperados.
10. Ten Minute Build	Garantizar que el sistema pueda ser construido completamente en 10 minutos o menos.	Automatizar el proceso de construcción, optimizar scripts de build.

11. Continuous Integration	Integrar y probar el código frecuentemente para detectar errores lo antes posible.	Configurar un servidor de CI, hacer commits pequeños y frecuentes.
12. Test-First Programming	Escribir pruebas antes de desarrollar el código para asegurar que se cumplen los requisitos.	Practicar TDD (Test Driven Development), escribir pruebas unitarias antes del código.
13. Incremental Design	Mejorar y evolucionar el diseño del sistema de forma continua y adaptativa.	Refactorizar el código regularmente, realizar mejoras de diseño iterativas.