Notes from the Lecture/Paper on Installing and Using Claude Code

1. Installing Claude Code
- Search for Claude Code online.
- Take the provided installation command and run it in Terminal.
- After installation, typing `claude` opens Claude Code in the current folder.
- Recommended: Create a project folder before launching Claude Code.

2. Use in Cursor or Visual Studio Code (VSC)
- Open Claude Code within Cursor or VSC.
- Run the slash command `/ide` to manage IDE integrations.
- Enables Claude to reference open files and work seamlessly with the IDE.

3. Initialize Project
- Use `/init` command inside Claude.
- Scans entire project and records info in `claude.md`.
- `claude.md` acts as the project memory, providing context for chats.

4. Manage Context
- Similar to ChatGPT, long conversations add to context.
- When context reaches limit, use `/compact` to summarize while keeping key info.
- Use `/clear` to start a new chat, resetting history and context.

5. Change Models
- If subscribed to DeepAI, switch models between Opus and Sonnet.
- The default model uses Opus 4; Sonnet is used when nearing usage limits.

6. Change Modes (w/ planning)
- Shift + Tab toggles modes: default, auto accept, plan mode.
- Auto accept: Claude performs actions without asking.
- Plan mode: Claude generates a detailed plan, prompts for approval.

7. Add Files and Screenshots
- Drag files into Claude window to reference them by path.
- Use `@` command to autocomplete file references.
- Paste screenshots (Ctrl + V in terminal or Cmd + V in GUI).
- Dragging images updates file path references for Claude.

8. Terminal Tab
- Use Command + Esc to open Claude in a new terminal tab within VSC or Cursor.
- Treats Claude as a shell tab for quick access.

9. Custom Commands (slash commands)
- Define custom commands in `.claude/commands` folder as markdown files.
- Example: `joke_me.md` with prompt to tell a dad joke.

- Commands can accept arguments and are stored in project or home `.claude/commands`.
- Custom commands appear with slash (`/`) and execute immediately.

10. Coloring Cursor Commands
- Use `project_settings` command to change cursor color themes in VSC.
- Settings stored in `settings.json` files.
- Changes apply to all projects if stored globally.

1. Sub-agents: Multi-tasking
- Claude can run parallel subtasks, e.g., generate multiple UI designs simultaneously.
- Subtasks are created with specific prompts and stored in separate folders.
- Example: Generate five calculator interface designs with different themes.
- Subtasks update progress inside Claude interface.

12. YOLO Mode (Risky)
- Run Claude with `claude --dangerous` to bypass permissions.
- Use at your own risk; can cause unintended actions.
- Useful for advanced automation but caution advised.

13. Hooks (Event-based Triggers)
- Set hooks in `settings.json` for pre-tool, post-tool, notifications, sub-agent stops.
- Example: Play a sound file when Claude completes a task.
- Hooks run shell commands or scripts on specific events, enhancing automation.
- Example hook: Use `afplay` to play an audio file after task completion.

Notes with Data & Numbers
- Terminal commands like `claude`, `/init`, `/compact`.
- `claude.md` as project memory file.
- Run with `claude --dangerous` for YOLO mode.
- Hooks can execute commands like `afplay` to play sounds.

Examples Included
- Running `/init` to start a project.
- Using `/ide` to integrate with Cursor.
- Creating custom command `joke me.md` for dad jokes.
- Dragging files and screenshots into Claude for reference.
- Generating parallel UI designs with sub-agents.
- Setting hooks to play sounds after tasks.
- Changing cursor color theme via `project_settings`.
- Running Claude with `claude --dangerous`.