Center for
HIGH PERFORMANCE COMPUTING
THE UNIVERSITY OF UTAH

Coding Agents at CHPC

Brandon Claridge

Center for High Performance Computing

University of Utah

Overview

Explain
Define what coding agents are, how they differ from chat-only assistants, and explain utility for scientific computing.

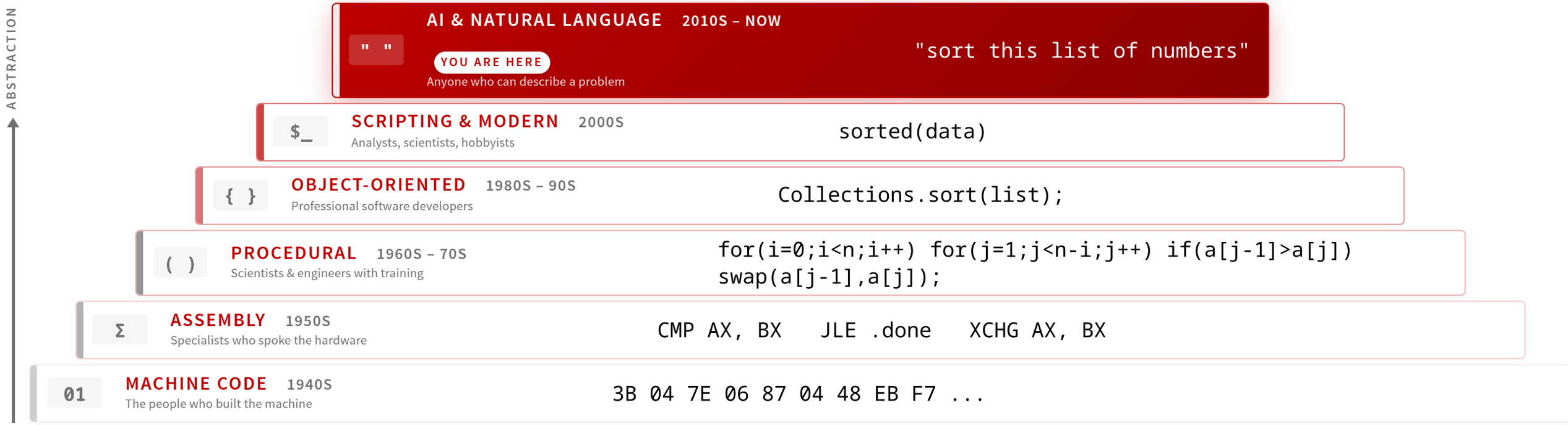
Tools
What coding agent tools are available at CHPC and where they fit.

Tutorial
Walk through a quick tutorial and share practical tips for getting started.

What are coding agents?

- An **LLM-based program** that can plan, write, inspect, explain, debug, and run code.
- Natural language prompts agent to edit files, execute shell commands, test output, docs, and feedback from the user.
- Compared with chat-only LLMs, agents are MUCH more powerful because they have direct access to your codebase/ information.

Programming languages context



Implications for scientific computing

Operational help

Generate test scripts, explain pipelines, spot syntax issues, and draft/execute SLURM submissions.

Faster development

Reduce friction for scripting, documentation, and debugging around scientific workflows.

Tool adoption

Lower the barrier to using complex tools such as workflow managers, containers, and schedulers.

Code adaptation

Translate examples from colleagues, local environments, other clusters, or other languages into local CHPC-friendly patterns.

Operational help

- Routine **file/data wrangling** tasks, like reorganizing outputs, renaming batches of files, and reshaping directory structure.
- Troubleshoot **failed jobs** (NO MORE COPY AND PASTING).
- **Monitor logs** from jobs and other processes.
- Ensure **SLURM jobs** run on the best partition/account/qos settings.
- Optimize **compute resource** use so jobs run efficiently without wasting allocation.
- Choose the right **software tools** for a given task.

Faster development

- Describe in detail your project or analysis and have the agent draft a script in a matter of **minutes instead of hours**.
- **Fix bugs** or quickly adjust analysis parameters.
- Generate comments and README file sections for a poorly documented script before **sharing with collaborators**.
- **Better understand** a project by having agent review codebase and asking questions.

Tool adoption

- **No more limitations** with steep learning curves preventing tool adoption!

- Describe your analysis pipeline in plain English and get a working **Snakemake or Nextflow skeleton** with proper SLURM executor configuration.

- Scale a script to **paralellize across multiple nodes** using MPI.

Code adaptation

- Convert a script from **one language to another**.
- Translate a local script into a **cluster-friendly version**.
- **Modernize legacy code** from your lab or collaborators.

Where are coding agents allowed?

ALLOWED

General Environment

Most coding agents are permitted in the GE for normal development, workflow assistance, and routine coding tasks.

NOT ALLOWED

Protected Environment

Coding agents are not currently permitted in the PE at this time. CHPC-hosted models should be available in the near future (end of summer).

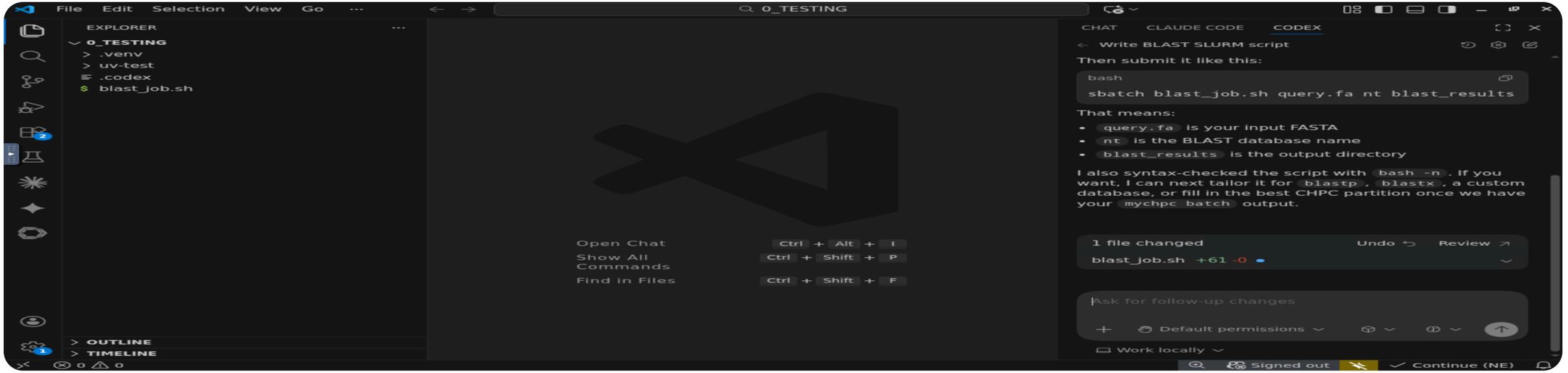
NOT ALLOWED

Citadel / Restricted Environment

Coding agents are not permitted in Citadel or other restricted environments. CHPC-hosted models should be available in the future.

Coding agents at CHPC

VSCode/ IDE extensions



Terminal User Interfaces (TUIs)

Coding agents at CHPC

VSCode/ IDE extensions

- User-installed extensions in Open OnDemand VSCode interactive desktop.
- Does not use CHPC skills (more on these later).
- Best for inline assistance, smaller edits, and more developer-centric.

Terminal User Interfaces (TUIs)

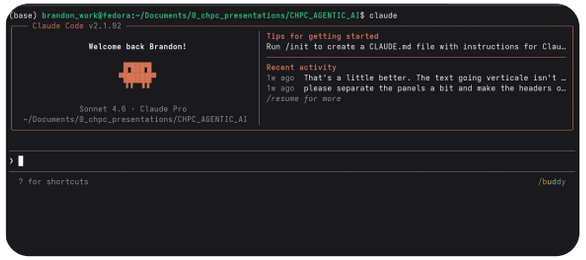
- CHPC modules available (OpenCode, Codex, Claude Code).
- Natively uses CHPC skills.
- Useful for those who already work in shells, SSH sessions, and remote development environments.

CHPC coding agent modules

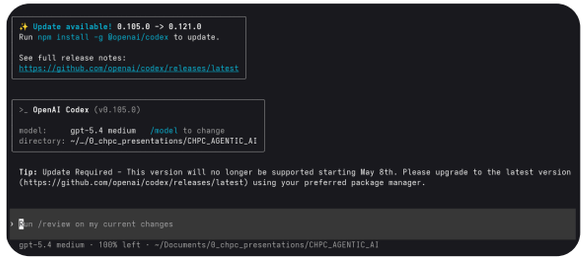
- **Environment-aware system prompt** so the agent knows cluster norms, mounted filesystems, and scheduler expectations.
- **CHPC-specific skills** for Slurm, modules, software stacks, storage, and common scientific workflow patterns.
- **Execution guardrails** so agents can help with commands and jobs without silently doing unsafe things.
- **Documentation grounding** so answers reflect CHPC policy and local best practices rather than generic Linux advice.

TUI modules

Claude Code



Codex



OpenCode

OpenCode and CHPC-hosted models

Model-agnostic interface

OpenCode is a terminal UI for coding agents that can connect to both CHPC-hosted and commercial models.

CHPC-hosted options

Current local choices include Qwen 3.5 27B and Gemma 4 31B for users who want an on-cluster model path.

Open-source tradeoff

These models are strong for most day-to-day coding work, but they are less reliable on larger, more complex tasks.

Still maturing

The service is still under development, so expect occasional downtime, rough edges, and periods of user overload.

Codex/ OpenAI subscriptions

- Codex models can be run with the **Codex TUI** and in **OpenCode**.
- Access to Codex is provided by personal subscriptions, including **free and paid** plans, as well as **university subscriptions**.
- Authorization requires browser interactivity from the terminal; **device code authorization is not available for university accounts**.

Skills

01 **Reusable context**
Markdown instruction files the agent loads on demand.

02 **Concrete guidelines**
Exact steps, commands, and decision rules.

03 **Cluster guardrails**
Enforce CHPC policy and prevent wasted allocation.

04 **Auto-invoked**
Triggered by task context or explicitly called (/skills).

EXAMPLE SKILL

`job-triage.md`

skill_example.md

````
name: job-triage
description: Use when a SLURM job has failed, was cancelled, won't start, or produced unexpected output on CHPC. Diagnoses the root cause from job logs and sac
ct output — covers OOM kills, walltime exceeded, invalid partition/account, missing modules, shared library errors, NFS path issues, and OOD/RStudio/Jupyter se
ssion failures. Distinct from slurm-utilization-audit, which tunes jobs that ran but were inefficient.
---

# CHPC Job Triage

Use this skill to diagnose why a SLURM job failed, was cancelled, or won't start. Always gather evidence first, then work through the diagnostic tree. If the r
oot cause cannot be determined, escalate to the helpdesk.

---

## Step 1: Gather Evidence

Ask for the job ID if it is missing. Do not start diagnosing until you have at least the `sacct` output or the user has pasted an error message.

Before interpreting missing or incomplete Slurm output, verify that you are querying the correct cluster. If `sacct` returns only the header, a job appears to
be missing, or `squeue`/`scontrol` cannot find the job, retry with explicit cluster selection first.

```bash
# Job state, exit code, and resource usage
sacct -j JOBID --format=JobID,JobName,State,ExitCode,Reason,Elapsed,ReqMem,MaxRSS,AllocCPUS,NodeList -X

# If the job may be on a different cluster, query it explicitly
sacct -M CLUSTER -j JOBID --format=JobID,JobName,State,ExitCode,Reason,Elapsed,ReqMem,MaxRSS,AllocCPUS,NodeList -X
squeue -M CLUSTER -j JOBID
scontrol -M CLUSTER show job JOBID

# Last 50 lines of job output (adjust filename as needed)
tail -50 slurm-JOBID.out

# Full job details — especially useful for pending jobs
scontrol show job JOBID
```

---

## Step 2: Diagnostic Tree

### A. Job never started

#### "Invalid account or account/partition combination"
- Run `mychpc batch` to show all valid partition/account/QOS combinations for this user
- Copy `--partition`, `--account`, and `--qos` values exactly from `mychpc batch` output — do not guess or construct them
- Verify the allocation is active for the current quarter

#### "Node count specification invalid" or `Reason=ReqNodeNotAvail`
- Single-job node limits: ~32 nodes (notchpeak), 24 (kingspeak), 106 (lonepeak)
- Check chpc.utah.edu news page for scheduled maintenance
- Suggest a different partition or waiting out maintenance

#### Job stuck in PENDING indefinitely
- If the job seems to be missing or not found, rule out a wrong-cluster query first by retrying the Step 1 commands with `-M CLUSTER`
- Check `scontrol show job JOBID` → look at the `Reason=` field
````

ONE SKILL · EVERY AGENT · REUSABLE ACROSS RUNS

CHPC skills

job-triage
Helps diagnose why a SLURM job failed, stalled, or produced unexpected output.

partition-selection
Guides users to the best partition, account, and QOS choices for a SLURM job.

slurm-utilization-audit
Reviews completed jobs for CPU or GPU efficiency and suggests concrete tuning changes.

software-environments
Helps load modules and build Python, conda, or container-based environments on CHPC.

storage-placement
Advises where inputs and outputs should live based on size, I/O pattern, and retention.

Demonstration: modules and Codex authentication

- available coding agent TUIs
- University subscription authorization via Open OnDemand interactive desktop

Demonstration: job troubleshooting

- find and resolve bug with existing batch script

Demonstration: simple bioinformatics example

- align `12S_rRNA` sequences for 8 species
- infer an evolutionary tree

Tips

- **Plan mode** - discussion with agent to provide context, clarify goals, set parameters, etc.
  - Conserve tokens by planning with a chat-based LLM that outputs a markdown file.
- **Build/execute mode** - allows agent to edit code and call tools.
- **Beware of context rot!!** - saturating model context with too much information will result in more hallucinations and other degredation.
- `/compact` - compacts the context to reduce context rot.

Tips

- `/connect` - connect to a commercial provider in OpenCode.
- `/model` - change model.
- `@<path to file/dir>` - specify files (with autocomplete).
- `! bash` - execute verbatim bash commands inside the TUI.
- `/init` - initialize project, create summary markdown file.

Where do coding agents still fall short?

Scientific software is underrepresented

Model quality is often stronger in mainstream software development than in niche research stacks.

Context saturation hurts performance

Long sessions can accumulate stale assumptions, contradictory state, and missed details.

Execution does not equal understanding

An agent may produce runnable commands while still misunderstanding scientific intent.

Permissions matter

On shared systems, autonomy has to be constrained by policy, auditability, and resource controls.

Active development

What CHPC is actively building next for coding-agent workflows.

Protected Environment support
Likely release of an OpenCode module with CHPC-hosted models in the Protected Environment.

Open-source model rollout
Continued deployment of leading open-source models.

Stronger CHPC skills
Increasing capabilities of CHPC models with more skills and additional training on CHPC documentation.

More training coverage
More focus in training presentations on agentic coding, including fundamentals and practical use.

Questions & Discussion

How have you been using coding agents?

WINS

Where has it helped?

GAPS

Where has it fallen short?

QUESTIONS?