IntuitionLabs

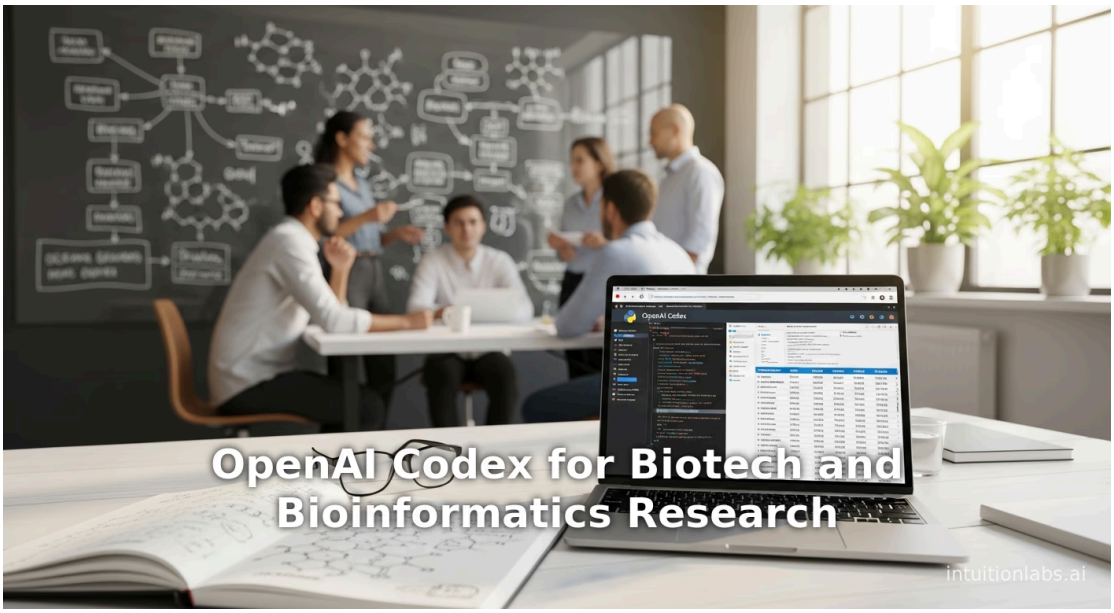
INTUITIONLABS / RESEARCH & PRACTICAL GUIDANCE

OpenAI Codex for Biotech and Bioinformatics Research

> Build practical AI for pharma and biotech with IntuitionLabs. We help life-science teams turn complex information and workflows into useful software, governed knowledge systems and AI tools.

Published by IntuitionLabs · 2026-04-11 · openai codex · bioinformatics · biotech research · ai coding agents · computational biology · data analysis · python programming

Original article: https://intuitionlabs.ai/articles/openai-codex-biotech-bioinformatics-research



In this article

1. Executive Summary
2. Introduction and Background
3. Technical Overview of OpenAI Codex
4. Using Codex in Bioinformatics and Biotech Workflows
5. Workflow Integration
6. Data Analysis Evidence
7. Case Studies and Real-World Examples
8. GeneGPT: Domain-Specific Codex for Genomics
9. AI-Assisted Research (Dog Vaccine Example)
10. Bridge Informatics Pharma Case (Pipeline Refactoring)
11. Detailed Use Cases in Biotechnology
12. Genomics and Sequence Analysis
13. Proteomics and Structural Biology
14. Synthetic Biology and Lab Automation
15. Drug Discovery and Computational Biology
16. Performance, Limitations and Best Practices
17. Discussion and Future Directions
18. Conclusion

Executive Summary

OpenAI's Codex is an advanced AI-driven coding agent that dramatically accelerates software development tasks, including those crucial to biotechnology research. It integrates with ChatGPT and programming environments to write, test, and debug code based on natural-language prompts. This report provides a comprehensive analysis of how Codex can be leveraged for scientific research in biotech. We review the historical context of AI in biology, the technical capabilities of Codex, and practical applications in genomics, proteomics, and drug discovery. We include data from academic studies and industry reports showing how Codex and similar models can automate common bioinformatics tasks (e.g. sequence analysis, pipeline development) and yield functional code even for non-programmers (pmc.ncbi.nlm.nih.gov). For example, an educational pilot used ChatGPT 3.5 to help biology students generate genomic-analysis scripts. This is historical evidence about LLM-assisted scripting, rather than a direct evaluation of the current Codex agent (Delcher et al.). We analyze these and other case studies, compare Codex to earlier tools like GitHub Copilot, and discuss Codex's strengths (multi-file memory, code execution, debugging) and the need to validate its non-deterministic output. Findings from earlier Codex models are treated as historical evidence rather than current-product benchmarks (pubs.rsc.org). We also cover future implications: Codex promises to transform how biotech researchers code data analyses (freeing scientists to focus on biology above syntax (pubs.rsc.org)), but safety and reproducibility must be ensured. In summary, Codex is poised to become a powerful collaborator in biotech R&D, dramatically speeding up coding tasks while requiring careful oversight (openai.com) (pubs.rsc.org).

Introduction and Background

The intersection of advances in artificial intelligence and biotechnology has grown increasingly significant. Modern biotech research—spanning genomics, proteomics, drug discovery, and synthetic biology—relies heavily on computational analysis. Historically, biologists have depended on specialized software and programming by bioinformaticians to process vast biological datasets. However, programming expertise can be a barrier for domain scientists. The advent of large language models (LLMs) and AI coding assistants promises to bridge this gap. Notably, OpenAI's Codex (building on GPT architectures) can generate code from natural language descriptions, enabling researchers to automate routine tasks without writing every line themselves (pubs.rsc.org) (pmc.ncbi.nlm.nih.gov).

Historical Context. Over the past decade, AI tools have gradually entered scientific workflows. Early efforts applied machine learning to problems like protein folding (e.g., AlphaFold) and genomics (predicting regulatory elements), while NLP models began assisting literature review. In programming, OpenAI's GPT-3 and later Codex (2021–2022) marked breakthroughs by generating functional code from text prompts (pubs.rsc.org). In parallel, code-assistants like GitHub Copilot (2021) helped engineers auto-complete code. These tools, however, lacked the ability to run or test code comprehensively. In 2025, OpenAI introduced Codex as a cloud-based coding agent integrated into ChatGPT (the "research preview" launched May 2025 (openai.com)), capable of executing tasks and managing multi-file projects (bridgeinformatics.com) (help.openai.com). This evolution arrived at a time when biotech data volumes – from next-generation sequencing, high-throughput assays, and multi-omics – were exploding, creating an urgent need for more scalable coding solutions.

What is OpenAI Codex? Codex is an AI agent specialized for software engineering tasks. OpenAI's May 2025 launch used codex-1, a version of o3 optimized for software engineering (OpenAI launch announcement). That is a historical launch model, rather than a fixed current core model. Current OpenAI documentation recommends using Codex with the latest general-purpose models (OpenAI documentation). Codex can inspect files, make edits, and run installed tools locally through its CLI, while Codex Cloud provides separate task workspaces (Codex CLI; Codex Cloud).

Codex (and its evolution into GPT-5.3-Codex (openai.com)) thus represents a new paradigm: a conversational coding assistant with memory across files and automated execution capabilities. Bridge Informatics notes that in pharmaceutical and biotech contexts, Codex is "more than a code completion tool, [it] is an autonomous, cloud-based coding assistant that can execute scripts, troubleshoot errors, validate outputs, and even propose changes across multiple files and repositories" (bridgeinformatics.com). Compared to earlier AI aids like ChatGPT's basic code interpreter or GitHub Copilot, Codex enables end-to-end software workflows: one can ask it to refactor pipelines, test edge cases, and output a validated script for human review, all in one session (bridgeinformatics.com) (help.openai.com).

This report explores how such capabilities can support scientific research in biotechnology. We cover technical details of Codex's design, illustrate specific use cases (from sequence analysis to lab automation), analyze data and studies on its performance, and discuss implications for researchers.Throughout, we cite peer-reviewed research, industry case studies, and official documentation to ground our discussion.

Technical Overview of OpenAI Codex

Model Architecture and Performance. OpenAI announced GPT-5.3-Codex on February 5, 2026, reporting then-leading results on SWE-Bench Pro and Terminal-Bench. Those are release-time benchmark claims, rather than a statement of current model leadership (OpenAI announcement). As checked on October 27, 2026, OpenAI's code-generation documentation recommends the latest general-purpose models for Codex and identifies GPT-5.3-Codex as deprecated (OpenAI documentation).

Codex is a coding-agent product with CLI, IDE, web and other interfaces. Access to models through the API is a separate integration option: OpenAI's February 2026 prompting guide explicitly documents `gpt-5.3-codex` in the API (Codex Prompting Guide). For new integrations, consult current model guidance rather than treating this historical model as the default (code-generation documentation).

Features and Tools. Codex is available through coding-agent interfaces including the CLI, IDE, and web. Code Interpreter is a separate tool for writing and running Python; it is not a Codex mode (OpenAI code-generation documentation; Code Interpreter). OpenAI also provides a dedicated Codex app (for Windows/MacOS) which can spawn multiple Codex agents working in parallel on separate projects (help.openai.com). Additionally, Codex offers a CLI and IDE extensions (e.g. for VSCode, Cursor.ai, Windsurf) allowing in-situ interaction: "Starting from a prompt or spec, Codex navigates your repo to edit files, run commands, and execute tests" (help.openai.com). It can both pair with a developer locally (via prompt-and-response) or run tasks unattended in the cloud (each task isolated in a sandbox) (help.openai.com). Notably, Codex has a feature to automate code reviews: it can be set to review pull requests on GitHub repositories, providing feedback similar to senior engineers (help.openai.com).

Codex leverages internet access for research tasks: users can enable an "internet browsing" mode enabling Codex to fetch documentation or APIs beyond its training data. Internally, Codex has memory of multiple files and can revise code across them in one session, a leap beyond simple code completion. It is trained to follow instructions closely and can "iteratively run tests until success" (openai.com).

Coding Languages and Libraries. Codex supports virtually all mainstream programming languages encountered in biotech: Python, R, MATLAB, C/C++, etc. In practice, Python is most common in scientific contexts, and Codex is proficient with libraries like NumPy, pandas, scikit-learn, BioPython, and others in its training corpus. It even understands domain-specific pseudo-code or query languages; for example, code queries to NCBI APIs (as in GeneGPT (pmc.ncbi.nlm.nih.gov)) or BLAST command-line arguments. Codex's multi-language training allows it to suggest code for diverse tasks, from data parsing to simulation scripting.

Performance Characteristics. Codex's outputs are probabilistic. A 2022 study of an earlier Codex model reported task-specific results that varied across single and repeated attempts (pubs.rsc.org). Those historical results do not establish the performance of current Codex models. In that 2022 study, the earlier Codex model could generate syntactically valid code that did not always follow best scientific practices (pubs.rsc.org), underscoring the importance of reviewing generated code in its intended context. Codex can catch some of its own mistakes by writing code and unit tests, but direct testing by the user remains advised. The effectiveness of Codex depends heavily on prompt phrasing; precise, detailed instructions yield much better outcomes.

Table 1 distinguishes Codex execution interfaces from the separate Code Interpreter tool, based on documentation checked on October 27, 2026. It is an interface summary, not a comparative performance benchmark:

Tool / Model	Code Generation	Code Execution / Testing	Multi-File/Memory	Context/Knowledge	Notes
Codex CLI	Coding agent	Runs local development tools	Inspects and edits repository files	Depends on the selected model	Works against a local repository (OpenAI)
Codex Cloud	Coding agent	Runs tasks in a cloud workspace	Uses the task's project workspace	Depends on the selected model	Each task has its own workspace (OpenAI)
Code Interpreter	Python tool used by a model	Writes and runs Python	Separate tool configuration	Depends on the calling model	A Python execution tool, not a Codex mode (OpenAI)

Table 1. Execution interfaces and tool roles. Capabilities depend on configuration; this table makes no performance or exclusivity claim.

Using Codex in Bioinformatics and Biotech Workflows

Biotechnology research often involves coding tasks in data analysis and automation. Codex can support many such tasks:

- **Data Parsing and Transformation.** Converting and cleaning large biological data files (e.g. FASTQ, FASTA, VCF, PDB). For example, Codex can generate Python scripts to strip unwanted characters from sequences, reverse-complement DNA strings, or reformat data into standard formats (pmc.ncbi.nlm.nih.gov) (pubs.rsc.org).
- **Pipeline Development.** Building end-to-end workflows for sequencing analysis (quality control, alignment, variant calling, etc.). Codex can scaffold pipeline code by stitching together tools like samtools, BWA, and summarizing outputs, even generating shell scripts or Nextflow/Snakemake configurations. Bridge Informatics anticipates tasks like "refactor a pharmacogenomics pipeline … output a clean, validated script" being handed to Codex (bridgeinformatics.com).
- **Statistical Analysis and Visualization.** Writing analysis scripts in R or Python (pandas, SciPy) for experiments: computing differential expression, plotting results, or running machine learning models. Codex can use natural language to generate code that produces plots or statistical tests (e.g. ANOVA, t-tests) of experimental data (pubs.rsc.org).
- **Database Access and Bio-APIs.** Querying scientific databases via APIs. As GeneGPT demonstrates, Codex can be prompted to formulate API calls to NCBI or elsewhere, parse results, and integrate them into answers (pmc.ncbi.nlm.nih.gov). A researcher could ask, "fetch and count all human genes on

chromosome 17 from Ensembl" and Codex could generate code using Ensembl's REST API. GeneGPT achieved a GeneTuring score of 0.83 by using Codex to call NCBI tools (pmc.ncbi.nlm.nih.gov), exemplifying this capability.

- **Machine Learning and Modeling.** While advanced modeling might exceed simple prompts, Codex can still scaffold ML code. For instance, generating Python code to train a classification on omics data or to define deep learning models in TensorFlow/PyTorch, given appropriate prompts about layer architectures and loss functions.
- **Lab Automation and Robotics.** In principle, Codex can write control scripts for lab instruments (e.g. gating microfluidic devices, controlling pipetting robots, data collection loops). For example, a user might prompt Codex: "Write Python code to control a Biomek liquid handler to pipette sample columns A1 through A12 into a 96-well plate, using XYZ library," and Codex could produce a valid script. This is supported by its generality and execution capability. (Although direct case studies are sparse, similar usage is envisioned for coding tasks across disciplines (bridgeinformatics.com) (help.openai.com).)
- **Documentation and Reporting.** Beyond code, Codex can draft README files, documentation, or reports summarizing code, saving researcher time. In drug development, it can even help format regulatory reporting code (e.g. data submissions), as Bridge Informatics suggests, handling "clinical data wrangling" and submissions (bridgeinformatics.com).

Critical for these uses is the iterative nature of Codex: researchers can refine prompts and ask the agent to debug or optimize code. Codex's ability to maintain context means it can adjust a pipeline codebase when requirements change.

Workflow Integration

Researchers can access Codex through:

- **Codex Cloud.** Sign in to ChatGPT and select a published cloud environment for the project. On the web or desktop app, choose **Work in > Cloud**, then **Select environment**. Start a task and describe the coding work; review the resulting changes and test results before integrating them (OpenAI Codex Cloud documentation).
- **Codex CLI and IDE Extensions.** For advanced use, the Codex CLI/extension can be used in local development environments. For instance, in VSCode, a researcher could open an existing project, then use a prompt like "Refactor the variant_analysis.py script to improve performance and add unit tests," and Codex will edit files and run tests. This mode is analogous to having an AI collaborator integrated into the codebase (help.openai.com).
- **Notebook Integration.** While not official, one can copy Codex-generated code into Jupyter notebooks or similar to execute and visualize results immediately, iterating in that interactive environment.

Best Practices. OpenAI's guidance emphasizes writing clear, specific prompts to get reliable code (pmc.ncbi.nlm.nih.gov) (help.openai.com). Researchers should review and test all generated code. Suggestions include asking Codex to produce unit tests or verifying outputs on known data (pubs.rsc.org).

Because Codex's underlying model may not know lab-specific constraints or newer libraries, user oversight is vital.

Data Analysis Evidence

Delcher et al. describe an informal pilot course at the University of South Alabama with 14 students: six graduate students, seven undergraduates, and one medical student auditing the course. Students used ChatGPT 3.5 to generate Python scripts for genomic tasks. The authors explicitly state that the teaching method was not formally assessed. These observations provide indirect historical evidence about LLM-assisted scripting, rather than a controlled evaluation of the current Codex agent (Delcher et al.):

- **DNA Sequence Manipulation:** All students generated working code to remove non-nucleotide characters and reverse-complement a DNA sequence using ChatGPT 3.5 (Delcher et al.).
- **Sequence Fragmentation:** A majority generated working code to divide a sequence into 100-nt fragments in FASTA format. Students whose own attempts failed were expected to obtain and run a classmate's working code; shared completion should be distinguished from independently generated solutions (Delcher et al.).
- **Counting Reads in Large FASTA:** All students generated working code to count FASTA reads, although three of the 12 code submissions were markedly more efficient. These are ChatGPT 3.5 observations, not Codex-agent results (Delcher et al.).
- **Complex BLAST Parsing:** Three students generated code that successfully extracted read names from BLAST output and corresponding sequences from FASTA files. Working solutions were then shared with the class. The paper does not give a task-specific denominator for that result (Delcher et al.).

Overall, the pilot suggests that ChatGPT 3.5 can help nonprogrammers generate scripts for genomic analysis, with direct validation of outputs and sharing of working code. Because the teaching method was not formally assessed, the results should be treated as informal classroom observations and indirect historical evidence, rather than a performance evaluation of the current Codex agent (Delcher et al.).

Table 2: Example Tasks from the Educational Case Study and ChatGPT 3.5 Outcomes

Task	ChatGPT 3.5 Outcome	Citation
Remove non-nucleotides & reverse-complement DNA	All students generated working code	Delcher et al.
Split a DNA sequence into 100-nt fragments (FASTA)	A majority generated working code; other students were expected to run a classmate's solution	Delcher et al.
Count reads in a large FASTA file	All students generated working code; efficiency varied among submissions	Delcher et al.
Extract read names from BLAST output & corresponding sequences from FASTA	Three students generated working code, which was shared with the class	Delcher et al.

Table 2. Qualitative classroom observations reported by Delcher et al. (2025), using ChatGPT 3.5. The pilot did not formally assess the teaching method and was not an evaluation of the current Codex agent.

Beyond education, emerging research shows Codex-like models aiding scientific inquiry more broadly. For example, OpenAI reports that GPT-5 (the next-generation model) significantly improved a molecular cloning protocol – the model introduced a novel recipe that increased cloning efficiency by **79×** compared to baseline (openai.com). While this specific experiment focused on wet-lab AI reasoning (not code-writing), it illustrates the power of AI-suggested improvements in biotech workflows. Combined with Codex's coding accelerations, future research could see integrated pipelines where AI suggests experimental protocols and generates the analysis code to interpret them.

Case Studies and Real-World Examples

GeneGPT: Domain-Specific Codex for Genomics

Researchers introduced **GeneGPT** as a method that prompts the historical Codex model `code-davinci-002` with NCBI API documentation and demonstrations, then detects and executes API calls during decoding. This uses in-context learning rather than a reported domain fine-tuning procedure and is distinct from the modern Codex coding-agent product. In the paper's historical GeneTuring evaluation, GeneGPT scored **0.83**, compared with **0.44 for New Bing** and **0.12 for ChatGPT**. The approximately **88%** improvement over New Bing is a relative score improvement, not a percentage-point gain or a measure of current Codex-agent performance (Jin et al.).

AI-Assisted Research (Dog Vaccine Example)

High-profile anecdotes illustrate AI's emerging role in biotech. OpenAI CEO Sam Altman recounted a story of a tech founder (Paul Conyngham) who, faced with Rosie's cancer, used ChatGPT to help devise a plan (tech.yahoo.com). Guided by AI, he sequenced the dog's tumor DNA, pored through mutation data, and even consulted structural models (AlphaFold was involved) to design a **custom mRNA immunotherapy** candidate (tech.yahoo.com). While Codex itself didn't generate the vaccine sequence, ChatGPT (a close relative) acted as a "smart research assistant" suggesting hypotheses and literature (tech.yahoo.com). This anecdote highlights the broader trend: biologists (or bio-entrepreneurs) are using LLMs and AI coding agents as research tools. Such tools can analyze genomic data, propose experiments, and even write scripts – tasks that previously needed expert teams. It underscores the message that Codex and siblings can accelerate discovery in life sciences.

Bridge Informatics Pharma Case (Pipeline Refactoring)

Bridge Informatics' August 2025 primer discusses the potential of Codex for bioinformatics, regulatory and research programming. This is a proposed workflow discussion, rather than documentation of an implemented pharmaceutical pipeline, measured results, or a merged pull request (Bridge Informatics). Any generated changes should be reviewed before integration.

Detailed Use Cases in Biotechnology

In this section, we delve deeper into specific domains of biotech and how Codex can be applied, with examples and citations.

Genomics and Sequence Analysis

Genomics research is arguably the most code-intensive area of biotech, making it ripe for Codex. Common genomics tasks Codex can assist with include:

- **DNA/RNA Sequence Processing.** Cleaning raw sequencing outputs (FASTQ/FASTA) is routine. Researchers might prompt Codex: *"Write Python code to trim all bases below quality 20 from FASTQ reads using Biopython, and write trimmed reads to a new file."* Codex can leverage libraries like Biopython to implement trimming logic. As shown in the educational study, ChatGPT-produced code handled trimming (removing non-nucleotide symbols) and reverse-complementing easily (pmc.ncbi.nlm.nih.gov).
- **Alignment and Assembly Pipelines.** Codex can script workflows invoking BWA or Bowtie2 for alignment, breaking pipelines into steps. For example, *"Create a shell script to run a BWA alignment of sample.fastq to human_g1k_v37, then use samtools to sort and index the SAM file."* Codex will produce a multi-command script.
- **Variant Calling and Annotation.** After alignment, variant callers like GATK or FreeBayes are used. Codex can generate code to run GATK, then annotate VCFs. For instance, *"Generate a Python script that loads sample.vcf, filters variants with depth < 10 and quality < 30, and outputs a summary table of remaining SNPs."* Even if Codex makes minor mistakes, it can often provide a solid template for filtering and summarization.
- **Motif and Pattern Search.** Searching for DNA/RNA motifs (e.g. promoter regions) can be coded by Codex given a description. Example prompt: *"Find all occurrences of the promoter motif TATA [A/T]A [A/T] [A/G] in the given sequence file and output their positions."* Codex can use regex or scanning algorithms. Students in the case study had ChatGPT generate code to count occurrences of k-mers across files .
- **BLAST and Homology Analyses.** Codex can automate calling NCBI BLAST (via command line or API) and parsing results. In the student project, ChatGPT-generated code extracted unique read names from BLAST output and retrieved corresponding sequences (pmc.ncbi.nlm.nih.gov). A researcher might similarly ask Codex to: *"Run BLASTN of all FASTA reads in sample.fasta against the P22 phage genome, then count how many hits each read has."* Codex would output a script using NCBI BLAST+ command-line tools and parse its TSV or JSON output.

Case Study: Students Analyzing RNA-Seq Data

The RNA-seq class project used ChatGPT 3.5 to generate Python scripts for analyzing Salmonella sequencing data. Students generated code to count FASTA reads, supporting reads-per-million calculations. This classroom example concerns LLM-assisted scripting, rather than a direct evaluation of the current Codex coding agent (Delcher et al.).

For the more complex BLAST task, students attempted to extract unique read names from an Excel output file and retrieve the corresponding sequences from the original FASTA file. Three students generated working code for these tasks and shared it with the rest of the class. Independently generated solutions should therefore be distinguished from completion using classmates' code (Delcher et al.).

Tools and Libraries

Codex knows popular bioinformatics libraries. For example, `BioPython` (for sequence I/O), `pysam` (for SAM/BAM), `Bioconductor` and `data.table` in R, or `scikit-bio`. A prompt like *"Using BioPython, get GC content of each sequence in a FASTA file"* yields Python code with `SeqIO` and string methods. It can also use general tools: e.g., producing bash loops with `grep` and `awk` for simple patterns. CODex's ability to incorporate best practices (like streaming files to avoid memory overload) depends on prompt detail. Users can refine by follow-ups, e.g. "Make the code memory-efficient."

Proteomics and Structural Biology

Beyond nucleic acids, Codex can also support protein and small-molecule analysis:

- **Protein Sequence and Structure.** Calculating protein properties (isoelectric point, molecular weight) from FASTA inputs is straightforward in Python. Codex can import Bio.SeqUtils and generate such scripts. For structural data, prompts like *"Write code to read a PDB file, calculate distances between all cysteine pairs, and highlight disulfides"* can be answered using libraries (e.g. MDAnalysis or BioPython PDB module). Even elegantly invoking AlphaFold or RoseTTAFold from Python is possible in principle (e.g. "write code to call AlphaFold on sequence X"), though running those requires GPU backend, so Codex might sketch a subprocess call.
- **Mass Spectrometry Data.** Codex can help parse spectral data if given format hints. For example: *"Using pyOpenMS, create a Python script that reads a mzML file, extracts all peaks above intensity 1000, and plots their m/z distribution."* It may produce code with pyOpenMS or similar libraries if prompted correctly.
- **Molecular Docking and Screening.** A user might ask for code to automate a docking workflow: e.g., "generate an AutoDock Vina docking script to test ligand.sdf against target.pdb." Codex can write the shell or Python wrapper calling Vina, exhaustively sampling.

While literature examples in proteomics are fewer, these tasks follow the same pattern: give Codex a domain-specific problem, and it will often produce working analysis code.

Synthetic Biology and Lab Automation

In synthetic biology, experiments often require both simulation and physical execution steps. Codex may assist in designing genetic circuits (producing pseudocode or Python) or controlling lab equipment:

- **Gene Circuit Design.** A prompt could be *"Write a COBRAbios Python script to simulate flux balance in an E. coli metabolic network with a new gene knockout."* Codex would use COBRApy library to load a genome-scale model, knock out a gene, and run FBA. Results (growth rate, pathway fluxes) would be printed. This sort of metabolic engineering task can be largely automated by Codex as long as the language description is clear.
- **CRISPR Guide Design.** For instance, *"Generate Python code to design gRNA sequences targeting gene X with no off-targets in genome Y"*. Codex might use Biopython to fetch gene sequences and a CRISPR library (though likely incomplete knowledge). Alternatively, hooking to a CRISPR API through Codex's code-writing could be done (similar to GeneGPT).
- **Automating Lab Instruments.** One could imagine prompts such as *"Using Opentrons Python API, write a protocol to transfer 100 µL from wells A1–A8 of a deep-well plate to wells B1–B8 of a 96-well plate"*. Codex would script the Opentrons deck and pipette instructions. Indeed, Opentrons has an official Python API, and Codex's broad training should include examples. This is a concrete example of Codex bridging in silico plans to hands-on experiments. Such integration is forward-looking but technically feasible given Codex's capability to output instrument code.

Drug Discovery and Computational Biology

Codex can contribute to early-phase drug research tasks:

- **Virtual Screening Pipelines.** Generating workflows to screen compound libraries (e.g. calling OpenEye OMEGA or RDKit). Prompt: *"Write a bash script to use RDKit to filter SDF for molecules with MW < 500 and logP < 5, then output smiles."* Codex can chain RDKit commands in Python or shell.
- **Chemoinformatics Data Analysis.** Statistical analysis of assay results (linear regression for a dose-response curve, logistic regression for activity classification). Codex can create scripts in R or Python (using scikit-learn) to model this data.
- **"Wet lab" Planning.** Not coding per se, but by generating code to simulate experiment design. e.g. simulating population growth under inhibitors.
- **Documentation for Regulatory Submission.** Codex can format data analysis outputs or create compliance reports if prompted with relevant standards (like converting data frames to CSV with specific columns).

These tasks leverage Codex's general coding power; specifics (e.g. accurate pharmacology models) still need expert knowledge.

Performance, Limitations and Best Practices

Accuracy and Reliability. Hocky & White reported task-specific results for an earlier Codex model; those historical findings do not quantify current Codex performance (pubs.rsc.org). For research purposes, every piece of AI-generated code must be validated on test cases. Peer scientists advise that Codex is best used as a rapid prototyping tool: it can draft substantial portions of code quickly, but humans should verify logic and results.

Relevant validation checks include indexing behavior, prompt interpretation, and library-version compatibility. In its evaluation of an earlier Codex model, the RSC perspective reported that the model could produce code that “does not follow best scientific practice” for complex tasks (pubs.rsc.org). However, RSC also found that even when mistakes occur, using Codex encourages scientists to articulate task requirements clearly (writing good prompts) and review results critically (pubs.rsc.org). Overall, they concluded Codex makes domain scientists *better programmers* by removing boilerplate work (pubs.rsc.org).

Security and Bias. Relying on large models and code-generation poses some risks. Execution location depends on the deployment: local Codex sessions run tools on the user’s computer, while cloud tasks run remotely (Codex environments). Local execution does not mean prompts, files, or outputs remain on-device: Codex sends inputs to OpenAI for inference (OpenAI deployment guidance). Before using confidential data, review the actual deployment, account and workspace data controls; OpenAI documents different applicable settings for ChatGPT sign-in and API-key authentication (authentication guidance). Additionally, Codex’s training data includes a mix of licensed, public, and user-submitted code, raising potential IP and licensing concerns for generated code (pubs.rsc.org). OpenAI’s policies aim to mitigate this by disallowing certain uses, but organizations should still review legal aspects.

Scientific Best Practice. Any AI-assistance should augment, not replace, scientific judgment. Researchers should use Codex to automate routine coding and focus on interpreting results. It is recommended to maintain version control (e.g. Git) of auto-generated code and to perform code reviews. Also, documenting prompts and outputs is good practice for reproducibility. The NIH GeneGPT team’s success was partly due to *in-context* documentation: including API docs in the prompt (pmc.ncbi.nlm.nih.gov). Similarly, a research team using Codex should embed comments and references in code to ensure interpretability later.

Discussion and Future Directions

OpenAI’s Codex is at the frontier of AI-assisted programming, and its implications for biotech research are profound.

Accelerating Innovation. OpenAI reports that GPT-5 improved end-to-end cloning efficiency **79-fold** in a wet-lab experiment, with the result confirmed across experimental replicates (OpenAI). This is a protocol-efficiency result; it does not measure elapsed experiment time or Codex coding productivity. Any claim about Codex’s effect on a particular bioinformatics workflow requires a separate evaluation.

Accessibility and Education. The ChatGPT 3.5 pilot suggests a role for LLM-assisted scripting in teaching biology students, but its teaching method was not formally assessed. It provides indirect historical evidence, rather than a direct assessment of Codex (Delcher et al.). In the future, biologists with minimal coding background may routinely rely on AI to do heavy lifting, shifting training priorities toward prompt engineering and result interpretation. This can broaden participation in computational biology. However, it also means curricula may need to adapt, teaching students how to critically use AI tools along with traditional methods.

Integration with Lab Automation. Looking ahead, we expect deeper integration between coding assistants and laboratory rigs. A “digital lab assistant” could use Codex to translate experimental designs into instrument protocols. Combined with robotics and automated experimentation platforms, this could create closed loops: design (via AI), execute (automated lab), analyze (AI), iterate faster than human cycles allow.

Multi-Modal AI Agents. Codex itself already spans code and text. Future AI agents (like GPT-5.3 and beyond) will further blur the line between code, data, and reasoning. The ongoing trend is toward unified agents: per OpenAI's announcement, GPT-5.3-Codex now supports "all work in the software lifecycle" including writing PRDs, doing user research, and analyzing data in spreadsheets (openai.com). In biotech, this could mean an AI that not only writes analysis code but also understands lab notebooks, generates slides for reports, and even converses with humans about results.

Challenges and Risks. However, there are open challenges. **Reliability:** Ensuring scientific correctness is paramount; an AI's plausible-sounding answer might be subtly wrong. **Reproducibility:** Black-box AI models complicate reproducibility of results. Journals and regulators may require clear documentation of AI involvement in research. **Security:** The Dog-vaccine example highlights biosecurity concerns: powerful AI could conceivably suggest how to engineer pathogens if misused. OpenAI itself conducted GPT-5 tests on benign tasks with strict controls (openai.com), but the community must develop guidelines for safe AI use in biology.

Compare with Other Pathway Developments. It's worth noting that OpenAI is not alone: Google's DeepMind recently launched "AlphaGenome" (June 2025), a new DNA-sequence model for variant effect prediction, and others are pursuing tool-enabled LLMs. The ecosystem will likely feature multiple AI tools for biotech. Codex is one of several coding-agent options. GitHub also documents a Copilot cloud agent that edits repository files and runs tests and linters (GitHub documentation). These feature descriptions do not establish comparative performance superiority. (Google's "AlphaEvolve" project hints at AI-assisted algorithm discovery, but its biotech focus is not documented publicly as of 2025).

Conclusion

OpenAI Codex is a transformative technology for biotechnology research. As an AI coding agent, it bridges the gap between natural-language scientific questions and executable code. It allows researchers to offload rote programming tasks—such as data cleaning, pipeline scripting, and basic analysis—unlocking their time for higher-level insight. The educational example used ChatGPT 3.5 and provides indirect historical evidence about LLM-assisted genomic scripting. Its teaching method was not formally assessed, so it does not empirically confirm the performance of the current Codex coding agent (Delcher et al.).

The combination of Codex's **multi-file context**, **built-in execution/runtime**, and **iteration capabilities** sets it apart from earlier tools (bridgeinformatics.com) (help.openai.com). It is essentially like having a 24/7 junior developer who understands biology-specific needs. However, AI-generated code still requires validation. Historical studies of earlier Codex models document the importance of critically reviewing generated code (pubs.rsc.org). Ethically and practically, researchers must ensure AI assistance boosts reliability, not undermines it, by thoroughly validating results and assessing biases.

Looking ahead, the trajectory is clear: Codex will become more capable (already GPT-5.3 is faster and smarter), and its adoption in biotech will grow. We will see more case studies of rapid prototyping in genomics, automated experimental design, and integrated lab-AI cycles. As OpenAI's research suggests, AI can dramatically speed up biological research even in the lab (openai.com). Combined with Codex's coding

prowess, the future research lab may look very different: a collaboration between human biologists and AI agents. The implications reach from fundamental science to healthcare innovation, making tools like Codex vital to the next generation of biotech breakthroughs.

The classroom findings should remain distinct from claims about the current Codex agent: the reported tool was ChatGPT 3.5, and the authors describe an informal pilot whose teaching method was not formally assessed (Delcher et al.). Proposed workflows and historical model studies should not be presented as measured outcomes for the current product.

Source links

Links cited in this article, in order of first appearance. Full addresses are provided for readers using extracted text.

1. https://pmc.ncbi.nlm.nih.gov/articles/PMC12290243
2. https://pubs.rsc.org/en/content/articlehtml/2022/dd/d1dd00009h
3. https://openai.com/index/accelerating-biological-research-in-the-wet-lab
4. https://openai.com/sl-SI/index/introducing-codex
5. https://bridgeinformatics.com/ai-coding-agents-in-bioinformatics-a-first-look-at-openais-codex
6. https://help.openai.com/en/articles/11369540-using-codex-with-your-chatgpt-plan
7. https://openai.com/index/introducing-codex
8. https://developers.openai.com/api/docs/guides/code-generation
9. https://learn.chatgpt.com/docs/codex/cli
10. https://learn.chatgpt.com/docs/cloud
11. https://openai.com/index/introducing-gpt-5-3-codex
12. https://developers.openai.com/cookbook/examples/gpt-5/codex_prompting_guide
13. https://developers.openai.com/api/docs/guides/tools-code-interpreter
14. https://pmc.ncbi.nlm.nih.gov/articles/PMC10153281
15. https://www.unsw.edu.au/news/2025/06/paul-turns-to-ai-to-save-his-dog-from-terminal-cancer
16. https://tech.yahoo.com/ai/chatgpt/articles/someone-used-chatgpt-custom-mrna-155515225.html
17. https://learn.chatgpt.com/docs/environments/modes
18. https://learn.chatgpt.com/docs/hipaa-configuration
19. https://learn.chatgpt.com/docs/auth
20. https://docs.github.com/en/copilot/concepts/copilot-surfaces/copilot-on-github

THE PUBLISHER

About IntuitionLabs

IntuitionLabs is an AI consulting, custom software development and data engineering firm serving pharmaceutical, biotechnology, medical-device and other life-science organizations. We work with clinical, regulatory, medical-affairs, commercial, quality and IT teams to connect technology decisions with the work people need to accomplish.

AI consulting and adoption

Our AI enablement services cover readiness assessments, use-case selection, governance and policies, team workshops, adoption measurement and ongoing advisory support. We help organizations structure the information layer behind AI: source material, context, permissions and maintained knowledge that make generated answers useful and reviewable. Private LLM inference and hosted AI options support teams evaluating how to operate AI with appropriate control over their data and infrastructure.

Software, data and life-science workflows

IntuitionLabs develops custom software for pharma and biotech, integrates enterprise systems, and builds data engineering and business intelligence solutions. Areas of focus include AI agents, regulatory research, medical writing, medical affairs, CMC information, competitive intelligence and clinical-document workflows. Our eTMF intelligence work includes cross-system reconciliation and inspection-readiness support.

Enterprise platforms and regulated delivery

We provide Veeva services, application support, managed services, integrations and custom applications, alongside enterprise content work involving platforms such as Egnyte. For regulated workflows, our services include GxP enablement, computer-system validation and software development addressing 21 CFR Part 11 requirements. The applicable controls, validation responsibilities and acceptance criteria are defined for each engagement.

Work with IntuitionLabs

Explore AI enablement, pharma and biotech software development, data engineering and BI, and Veeva services. Contact IntuitionLabs to discuss your workflow, information sources and implementation needs.

IntuitionLabs publishes educational research to help life-science teams make informed technology decisions. Coverage of a product or organization does not imply a client relationship, endorsement or partnership.

Website: https://intuitionlabs.ai

This article is educational. Verify consequential medical, legal, financial, regulatory and technical decisions with appropriate professionals and the cited primary sources. Company services are described separately from the article's research findings.