**Leaderboards**

For benchmarking the performance of various LLMs and SLMs on a specific task (code generation in our case).

* **Big Code Models Leaderboard** evaluates on HumanEval (Python) + irrelevant for me MultiPL-E (C++, Java, and JavaScript): <https://huggingface.co/spaces/bigcode/bigcode-models-leaderboard>
* **EvalPlus** evaluates using [HumanEval+](https://github.com/evalplus/humanevalplus_release) version 0.1.10; [MBPP+](https://github.com/evalplus/mbppplus_release) version 0.2.0. Models are ranked according to pass@1 using greedy decoding: <https://evalplus.github.io/leaderboard.html>
* **LiveCodeBench** - holistic and contamination-free evaluation of coding capabilities of LLMs: <https://livecodebench.github.io/leaderboard.html>. Optional – to estimate the usefulness for the Praxis.
* **CanAICode results**: <https://huggingface.co/spaces/mike-ravkine/can-ai-code-results> .
* **Awesome Code LLM:** <https://github.com/huybery/Awesome-Code-LLM>
* **SEAL Leadervoard** – evaluates multiple coding languages: <https://scale.com/leaderboard/coding>. Optional – to estimate the usefulness for the Praxis
* **Vellum LLM Leaderboard** evaluates on HumanEval among other metrics: <https://www.vellum.ai/llm-leaderboard>. Optional – to estimate the usefulness for the Praxis

**Evaluation Datasets**

Used to evaluate the quality of the generated code

1. **HumanEval:** popular benchmark for evaluating code generation models - contains programming problems with corresponding unit tests that can be used to verify the correctness of generated solutions.  
   Source: <https://github.com/openai/human-eval/tree/master/data>
2. **MBPP (Mostly Basic Python Problems):** designed to evaluate code generation models on Python programming tasks. It consists of a large number of Python problems with a set of unit tests that assess the correctness of the generated Python code.  
   Source: <https://github.com/google-research/google-research/tree/master/mbpp>
3. **LiveCodeBench Dataset:** holistic and contamination-free evaluation of coding capabilities of LLMs  
   Source: <https://huggingface.co/livecodebench>

**Evaluation Code**

* **My code** - Testing LLMs on the Code Generation Task: <https://github.com/agnedil/Praxis>
* **EvalPlus** evaluation code: <https://github.com/evalplus/evalplus/>
* **LiveCodeBech** evaluation code: <https://github.com/LiveCodeBench/LiveCodeBench>

**Training Datasets**

Used to fine-tune SLMs to improve their code generation capabilities

1. **Tested-143k-Python-Alpaca**

Description: Python dataset with 143,327 examples of code that passed automatic tests to ensure high quality.

Link: <https://huggingface.co/datasets/Vezora/Tested-143k-Python-Alpaca>

1. **CodeFeedback-Filtered-Instruction**

Description: a curated collection of code instruction queries extracted from open-source code instruction tuning datasets. It significantly advances code generation capabilities by integrating execution and iterative refinement functionalities.

Link: <https://huggingface.co/datasets/m-a-p/CodeFeedback-Filtered-Instruction>

1. **Magicoder-Evol-Instruct-110K**

Description: A decontaminated version of [evol-codealpaca-v1](https://huggingface.co/datasets/theblackcat102/evol-codealpaca-v1). Decontamination was done in the same way as StarCoder ([bigcode decontamination process](https://github.com/bigcode-project/bigcode-dataset/tree/main/decontamination)). See [Magicoder paper](https://arxiv.org/abs/2312.02120).

Link: <https://huggingface.co/datasets/ise-uiuc/Magicoder-Evol-Instruct-110K>

1. **Python-code-dataset-500k**

Description: a summary and reformat pulled from GitHub code. 500K examples to be cleaned first. Cleaning can be done using an SLM.

Link: <https://huggingface.co/datasets/jtatman/python-code-dataset-500k>

1. **Just-write-the-code-Python-GenAI-143k**

Description: The entire dataset of 230k examples of AI and Machine Learning python code retrieved from public repositories on GitHub. It is a prototype and needs to be cleaned.

Link: <https://huggingface.co/datasets/guidevit/Just-write-the-code-Python-GenAI-143k> and <https://huggingface.co/datasets/guidevit/Just-write-the-code-Python-GenAI-230k>

1. **Tiny codes**

**1.6 M short and clear code snippets** that can help LLM models learn how to reason with both natural and programming languages.

Link: <https://www.sonarsource.com/learn/llm-code-generation/>

**Small Language Models**

In addition to the ones whose results are submitted.

1. **DeepSeek-Coder-V2** – among the best small language models for code generation. Description and usage examples: <https://github.com/deepseek-ai/DeepSeek-Coder-V2>.
2. [CodeGemma-7B-it](https://huggingface.co/google/codegemma-7b-it) (HumanEval – 61%)
3. **SmolLM**

**Code**

* Main repo that I am using: <https://github.com/openai/human-eval>
* May be helpful: <https://github.com/abacaj/code-eval/tree/main>
* Leaderboard: <https://huggingface.co/spaces/bigcode/bigcode-models-leaderboard>
* Pass @k explained: <https://deepgram.com/learn/humaneval-llm-benchmark>
* Replicate LLMs: <https://replicate.com/pricing> + deploy your own model.

*# pip install -q datasets*

from datasets import load\_dataset

*# Languages: "python", "js", "java", "go", "cpp", "rust"*

ds = load\_dataset("bigcode/humanevalpack", "python")["test"]

ds[0]

**ADVANCED LLM TECHNIQUES**

LLM agents and multi-agent systems: <https://www.linkedin.com/posts/areganti_if-you-want-to-learn-how-to-build-llm-activity-7247795075475193857-b2iR/>

OpenAI lightweight agentic library SWARM: <https://www.linkedin.com/posts/philipp-schmid-a6a2bb196_this-came-unexpected-openai-released-swarm-activity-7250841965519368192-oJ35/>

<https://github.com/andrewyng/aisuite> - aisuite makes it easy for developers to use multiple LLM through a standardized interface

Install LLMs locally: <https://www.iphones.ru/iNotes/kak-ustanovit-neyroset-tipa-chatgpt-pryamo-na-mac-i-polzovatsya-ey-bez-interneta-besplatno>

## Hugging Face Releases SmolTools: A Collection of Lightweight AI-Powered Tools Built with LLaMA.cpp and Small Language Models: <https://www.marktechpost.com/2024/11/06/hugging-face-releases-smoltools-a-collection-of-lightweight-ai-powered-tools-built-with-llama-cpp-and-small-language-models/>

LLM on CPU: <https://www.linkedin.com/posts/gabriele-venturi_ai-localllm-1bit-ugcPost-7252959485252554752-8dnk/>

**AGENTS**

Qwen agent: <https://github.com/QwenLM/Qwen-Agent>

Recommended agentic library: <https://www.linkedin.com/posts/mehdiallahyari_multi-agent-rag-system-activity-7273554473375940608-mHhf?utm_source=share&utm_medium=member_desktop>

Another one talked about: <https://www.linkedin.com/feed/update/urn:li:activity:7273016722708643841?utm_source=share&utm_medium=member_desktop>

<https://ai.pydantic.dev/#instrumentation-with-pydantic-logfire>