Coding for non-coders

Just specify what you want and enjoy the music

Yogesh Haribhau Kulkarni (PhD)

Published in Technology Hits

5 min read · Just now

Photo by Steinar Engeland on Unsplash

Machine learning has become integral to software applications over the past decade, but building ML systems requires significant specialized expertise. Declarative programming paradigms for ML aim to make the technology more accessible. In declarative ML, developers specify what they want the system to do

rather than the procedural steps to implement it. This approach provides simpler abstractions that open up ML to a wider range of software engineers.

What Is Declarative Programming?

Declarative programming focuses on specifying the logic and desired outcomes of a program rather than step-by-step instructions. It centers on expressions and declarations to define rules that are then interpreted by the system. This contrasts with imperative styles that require manually coding algorithms and control flows. Key traits include:

- High-level abstractions that hide complex implementation details
- Immutable data structures
- Incorporation of domain-specific languages

By separating logic from control flow, declarative programming enables clearer, more maintainable code.

Some key challenges in developing machine learning systems

- **Challenge**: Too many decisions to get right. Building ML models involves lots of complex choices that can go wrong. **Solution**: Defaults and automation to guide developers, instead of difficult manual tuning.
- **Challenge**: Focusing too much on new models without improving old ones. Teams build new models without evaluating why old ones failed. **Solution**: Standard testing and comparisons of models to understand gaps. This allows focusing on data and evaluation rather than coding.
- **Challenge**: Silos between specialist teams cause duplication. Disjoint teams like entity disambiguation and intent classification don't collaborate and duplicate work. **Solution**: Shared code and interfaces for different components to increase collaboration and reuse.
- **Challenge**: Most developers lack ML expertise for low-level coding. **Solution**: Abstractions so most developers don't have to do specialized ML coding.
- **Challenge**: Development takes too long because of many cycles. **Solution**: Faster iterations with automation and abstraction to achieve better quality faster.
- **Challenge**: Too few people can work directly with complex ML systems. **Solution**: Different interfaces for more stakeholders to participate and leverage

ML.

The core issue is complexity that blocks productivity. The solutions aim to simplify and automate to make ML development more accessible.

Here comes, Declarative ML Systems

Internal ML platforms at tech leaders like Uber, Apple, and Meta pioneered declarative ML systems to empower non-specialists to utilize ML. These platforms allow developers to specify inputs, outputs, and constraints on a model while automating the underlying training and deployment pipeline. Benefits include:

1. **Reduced Decision Overhead**: Smart defaults and automation around hyperparameters, model selection, etc. let engineers focus on high-value supervision.
2. **Rapid Iteration**: Higher abstractions and automation accelerate each cycle to achieve higher quality faster.
3. **Accessibility**: Enables usage across skill levels with configurable controls.

In essence, declarative ML separates model objectives from the means of implementation, freeing developers to iterate without coding pipelines.

Salient tenets

1 — No reinventing basic components. Just specify input and output data to generate a complete ML pipeline. No need to code low-level parts like normalization, tokenization, etc. yourself.

2 — Customize as needed over smart defaults. The system provides good defaults to get started quickly. Additional tweaks like lowering text or adding regularization are simple config changes.

3 — Abstracts away infrastructure hassles. No more wasting time debugging infrastructure issues. Built on top of scalable distributed frameworks, so you focus on the machine learning instead of fighting with CUDA and clusters.

Declarative ML systems handle the heavy lifting of building robust and scalable ML pipelines so developers avoid reinventing the wheel and spending time on dev ops. Smart defaults jumpstart projects while still allowing customization. The goal is minimizing overhead so domain experts can concentrate on the machine learning tasks rather than infrastructure.

My own projects

I have long been an enthusiastic proponent of declarative programming, leveraging it across my work from early on. Originally applying it for geometric modeling applications, I later expanded use into my current focus areas of AI, machine learning, natural language processing, and large language models.

A few examples that demonstrate the power of declarative abstractions

- **Bot for Bot** (more details):
  Enables anyone to generate a custom chatbot application simply by specifying a configuration file with basic parameters like name, data sources, and index locations. The system handles all the complex NLP pipeline and model building automatically.

```
{
 "APP_NAME": "MyApp",
 "DOCS_INDEX":"/fullpath/to/docs.index",
 "FAISS_STORE_PKL":"/fullpath/to/faiss_store.pkl",
 "FILES_PATHS": [
 "/fullpath/to/file1.csv",
 "/fullpath/to/file2.txt",
 "/fullpath/to/file3.pdf"
 ]
}
```

- **Fine-tuning LLMs using Ludwig** (more details)
  Allows fine-tuning large pretrained language models without writing any model code through a declarative yaml configuration. Defines features like model architecture and training data while the underlying platform trains the model.

```
qna_tuning_config_yaml= yaml.safe_load("""
input_features:
 - name: Question
 type: text
 encoder:
 type: auto_transformer
 pretrained_model_name_or_path: meta-llama/Llama-2-7b-hf
 trainable: false
 preprocessing:
 cache_encoder_embeddings: true
```

```
output_features:
 - name: Answer
 type: text
""")
```

The declarative approach in these projects and others removes the heavy lifting from users so they can concentrate on domains tasks rather than implementation details. By radically simplifying the interfaces to leverage state-of-the-art techniques, declarative programming unlocks advanced AI capabilities to a much broader audience.

Pros and Cons

Declarative programming brings notable advantages but also some limitations in AI:

Pros:

- Enhances readability and comprehension
- Concise, intuitive representations
- Encourages modular, reusable components

Cons:

- Can introduce computational overhead
- Limits low-level control and optimization
- Not suited for all applications

Despite drawbacks for select specialized use cases, declarative techniques will continue expanding AI accessibility.

The future is bright for opening up AI's immense potential to a broader audience of builders through declarative abstractions. As models become exponentially more powerful, putting them in more hands responsibly is pivotal.

References

Declarative Machine Learning Systems

PDF In the past 20 years ML (machine learning) has progressively moved from an academic endeavor to a pervasive…

queue.acm.org

4 Reasons Why Declarative ML Makes Sense for Engineers

Machine learning is starting to go mainstream, graduating out of the research lab and making its way into products. In…

opendatascience.com

Declarative Programming — Lark

Click pic below or visit LinkedIn to know more about the author

Artificial Intelligence | Future | Advice | Declarative Programming | Generative Ai Tools

Edit profile

Written by Yogesh Haribhau Kulkarni (PhD)

1.2K Followers · Editor for Technology Hits

PhD in Geometric Modeling | Google Developer Expert (Machine Learning) | Top Writer 3x (Medium) | More at https://www.linkedin.com/in/yogeshkulkarni/

More from Yogesh Haribhau Kulkarni (PhD) and Technology Hits

Yogesh Haribhau Kulkarni (PhD) in Analytics Vidhya

Want to generate Social Media content automatically?

Using Crew AI to build a team of autonomous agents just for that

8 min read · 6 days ago

17

Yogesh Haribhau Kulkarni (PhD) in ILLUMINATION Videos and Podcasts

Yoganidra: A Deep Sleep Without Sleeping

Finding moments of calmness even during daytime

3 min read · Dec 18, 2023

292 5

Yogesh Haribhau Kulkarni (PhD) in ILLUMINATION Videos and Podcasts

Change in India's Foreign Policy Doctrine

Jotting from discussion between Dr Ankit Shah and Sanjay Dixit

3 min read · Jan 1

73 2

Yogesh Haribhau Kulkarni (PhD) in Technology Hits

Specs for 'Knowledge as a Service' (KaaS) project

Product Requirements Document with hints of Implementation

3 min read · Dec 8, 2023

22 1

See all from Yogesh Haribhau Kulkarni (PhD)

See all from Technology Hits

Recommended from Medium

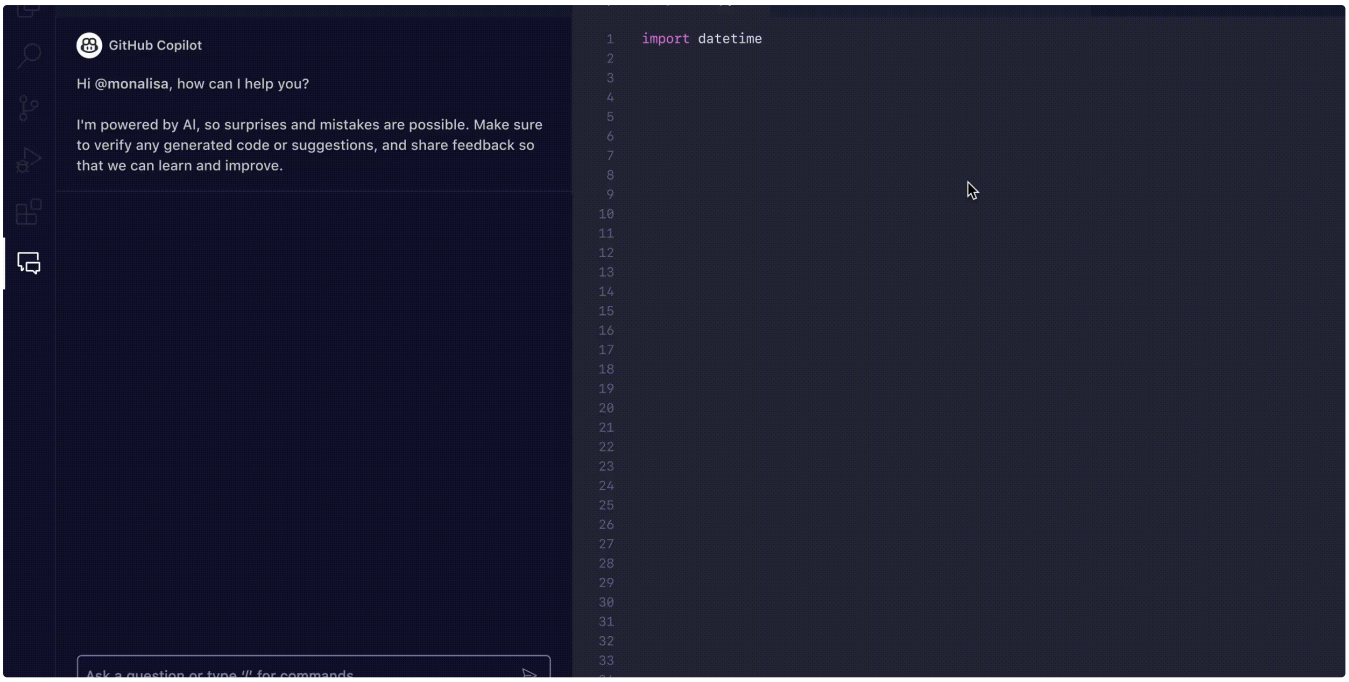
Ignacio de Gregorio in Towards AI

Is Mamba the End of ChatGPT As We Know It?

The Great New Question

✨ · 8 min read · Jan 11

4K 40



Jacob Bennett in Level Up Coding

The 5 paid subscriptions I actually use in 2024 as a software engineer

Tools I use that are cheaper than Netflix

· 5 min read · Jan 4

2.9K 42

Lists

AI Regulation

6 stories · 279 saves

ChatGPT

23 stories · 402 saves

ChatGPT prompts

34 stories · 983 saves

Generative AI Recommended Reading

52 stories · 631 saves

James Presbitero Jr. in Practice in Public

These Words Make it Obvious That Your Text is Written By AI

These 7 words are painfully obvious. They make me cringe. They will make your reader cringe.

4 min read · Jan 1

16.4K 469

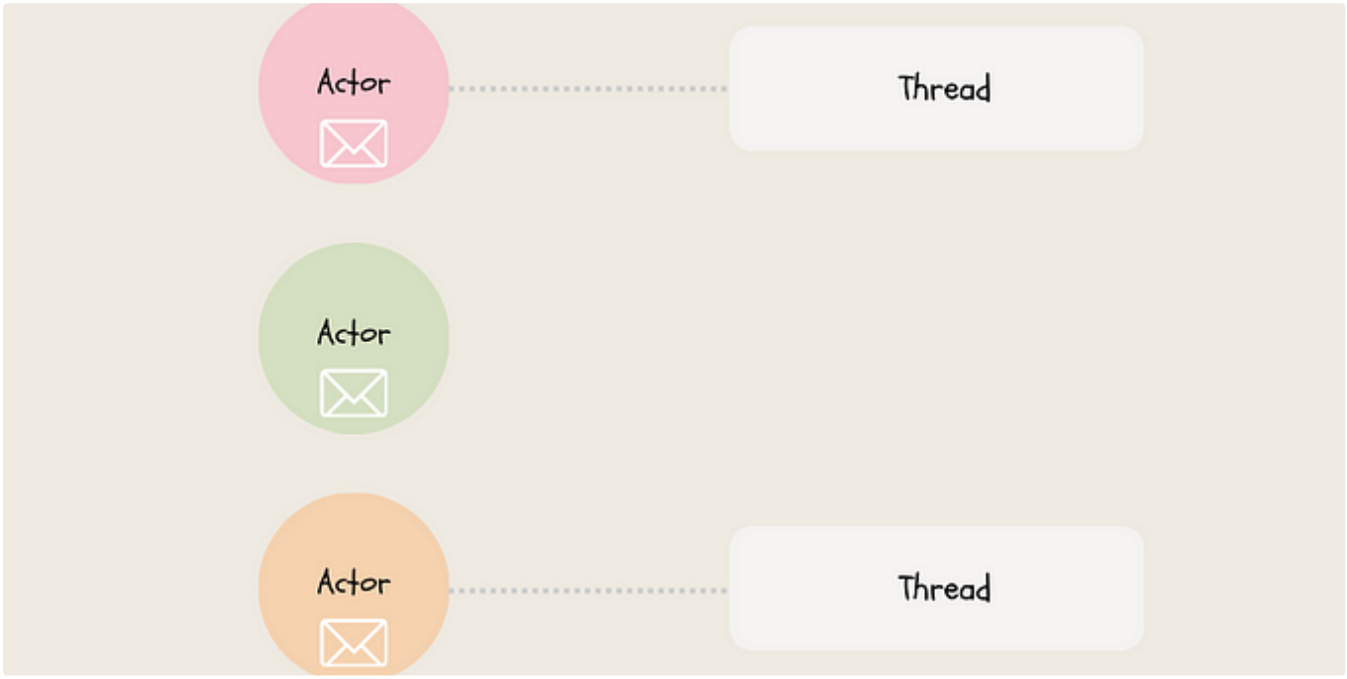
Ravi M in Python in Plain English

“Python GUI Magic: Elevate Your User Experience with these Top 20 Stunning Libraries”

“Unlock the Power of Python’s Diverse GUI Packages—From Tkinter to CefPython, Unleash Creative Interfaces and Seamless User…

✦ · 6 min read · Dec 31, 2023

191 3



Nidhey Indurkar

How did PayPal handle a billion daily transactions with eight virtual machines?

I recently came across a reddit post that caught my attention: ‘How PayPal Scaled to Billions of Transactions Daily Using Just 8VMs’…

7 min read · Jan 1

2.1K 25

Siavash Yasini in Towards Data Science

How to Write Memory-Efficient Classes in Python

Three tricks to prevent your data project from memory overflow

· 7 min read · 5 days ago