実用アプリ

ローカルLLMでVS Code用コーディング・アシスタント作成

新連載

第1回　使用するローカルLLMの選定

氏森 充

● 連載主旨

本連載では，ローカルLLM（Large Language Models）と連携して動作するVisual Studio Code（以降，VS Code）の拡張機能を使って，完全にローカル環境で完結するコーディング・アシスタント（**図1**）を作成し，日常的に利用可能な水準まで仕上げます．

VS Code向けのLLM連携拡張機能は，現時点ではクラウドLLMを前提としたものが主流です．しかし，

- ソースコードの外部送信に対するセキュリティ懸念
- インターネットや外部サーバの状態が不安定になる懸念
- 利用量に応じたコスト増加

といった課題があり，業務用途や長期利用を考えると必ずしも最適とは言えません．そこで本連載では，ローカル環境で動作するLLMを用いてVS Code拡張機能を構築し，その実用性を検証します．

今回開発するアプリケーションは次の2点を必須条件とします．

- VS Code拡張機能として動作すること
- ローカルLLMと連携できること

本連載では，次の手順で開発と検証を進めます．

1. 使用するローカルLLMの選定
2. ローカルLLMと連携可能な拡張プラットフォームの構築
3. VS Code拡張機能として実装するコマンドの開発

● 今回のテーマ

今回は利用可能なローカルLLMを選定するための検証を行います．具体的には，今回開発するアプリケーション（コーディング・アシスタント）が，

- VS Code拡張機能として動作するか
- ローカルLLMと連携できるか

といった観点から，複数のローカルLLMを実機検証し，実運用に耐えうるモデルを絞り込みます．今回は，ローカルLLMを用いたVS Code拡張機能開発に向けた技術検証であると同時に，今後の連載で実装を進めていくための基礎検討を整理した位置づけとなります．

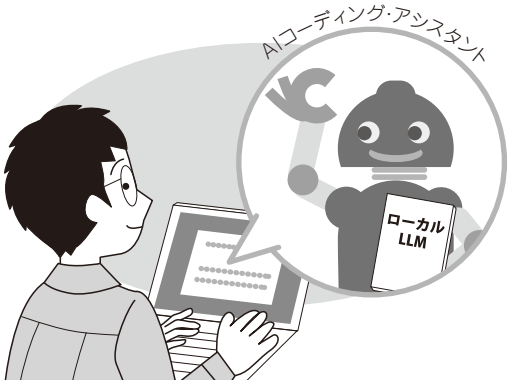

図1　VS Codeの拡張機能を使ってローカルで動く，コーディング・アシスタントを作る

ローカルLLMは公式ドキュメントや周辺情報が十分に整備されていない場合も多く，事前に適合性を判断するのは容易ではありません．そこで本稿では，候補となるローカルLLMに，先述の観点のプロンプトを入力して，その出力を検証する方針を採ります．

検証①
簡易なプログラムを作成させてみる

● 確認方法

ローカルLLMに，「おはようございます」というメッセージを表示させる簡易なVS Code拡張機能プログラムを作成させてみます．これによって，VS Code拡張機能について，ローカルLLMがどの程度理解しているのかを確認します．

● 確認用プロンプトの設計

候補となるローカルLLMに，次のようにロール・プロンプティング（Role Prompting：LLMに“役柄”を与える指示の仕方）を用いて役割を与えます．

Role：「あなたはVS Code拡張機能作成のスペシャリストです」