AI時代を生きる
プログラム開発の進め方

～プログラマーはAIの操縦士へ ― AIを味方につける開発の未来～

@watabin ← YouTubeチャンネル登録をお願いします。

はじめに

皆さん、こんにちは。
本日は
「AI時代を生きるプログラム開発の進め方」
というテーマでお話しします。
AIの進化が目覚ましい昨今、
プログラム開発の仕事はどう変わっていくのか、
そして私たちがどう適応していくべきかについて、
一緒に考えていきましょう。

AI時代のプログラミングはどうなるか

AIがプログラミングの「**書き手**」としての役割を担い、開発プロセスが劇的に効率化されます。

- **コーディングの自動化**: AIが指示に基づいてコードを自動生成するようになります。プログラマーはゼロからコードを書くのではなく、AIが生成したコードを**レビュー**し、**修正**する役割が中心になります。
- **デバッグと最適化**: AIがコードのバグを検出し、より効率的な解決策を提案します。これにより、開発にかかる時間や労力が大幅に削減されます。
- 「**ノーコード・ローコード**」**の普及**: 専門的な知識がなくても、視覚的な操作でアプリやシステムを開発できるツールがさらに普及し、誰でもプログラミング的な思考で物作りができる時代になります。

第1部
プログラム開発の基礎知識

プログラマーの役割と仕事内容を知る

プログラム、プログラミング、プログラマーの違い

まず、基本的な言葉の整理です。

- **プログラム**：
  コンピュータへの指示書、命令の集まりです。スマートフォンアプリやウェブサイトなど、私たちが使うデジタル製品はすべてプログラムで動いています。
- **プログラミング**：
  その指示書を「書く行為」そのものです。
- **プログラマー**：
  プログラミングを仕事として行う「人」です。

プログラム言語の種類

プログラム言語は、コンピュータに指示を出すための「言葉」です。現在まで、数千種のコンピュータ言語がありますが、主要なものでも約50〜100種あり、Python、Java、JavaScriptなどが代表的です。それぞれ得意な分野が異なります。

- **Python**：AI、データ分析に強い。
- **JavaScript**：Webサイトの動きを作る。
- **Java**：大規模システムやAndroidアプリ開発に強い。

開発の基本プロセス

プログラム開発の一般的な流れを見てみましょう。

企画→要件定義→設計→プログラミング→テスト→リリース

これは「**ウォーターフォールモデル**」という伝統的な開発手法で、一つ一つの工程を順番に完了させるのが特徴です。これに進捗管理と各項でレビューが入ります。

新しい開発手法「アジャイル開発」

ウォーターフォールモデルに対し、変化の激しい現代では「**アジャイル開発**」が主流になりつつあります。

- 全体を小さな機能単位に分け、短いサイクル（**スプリント**）で開発とテストを繰り返します。
- これにより、仕様変更にも柔軟に対応できます。
- スマホのゲームアプリは殆どがアジャイル開発を採用。

第2部
AIが変える
プログラム開発の未来

AIの登場により生じる変化点

AIはプログラマーの仕事をどう変えるか？

AIは、単純なコーディング作業を自動化し、開発を劇的に効率化します。

- **コードの自動生成**：AIが指示に基づいてプログラムのコードを自動で作成します。
- **バグの自動修正**：AIがコードの不具合を見つけ、解決策を提案します。 これにより、プログラマーは「コードを書く人」から、「**AIを操る人**」へと役割が変わります。

プログラマーに求められる新しいスキル

AI時代、プログラマーに求められるのは、より高度で人間的なスキルです。

- **課題解決能力**：
  顧客の課題を深く理解し、AIを使ってどう解決するかを考える力。
- **AIへの指示力**：
  AIが意図した通りのコードを生成するように、的確な指示（**プロンプト**）を出す力。
- **設計力**：
  AIが生成したコード全体を統合し、品質を管理する設計のスキル。
  単純なコーディングはAIに任せ、プログラマーはより上流工程や、人間ならではの創造的な仕事に集中するようになります。

AI時代の開発プロセス

AIの活用を前提とした新しい開発プロセスは、ウォーターフォールやアジャイルの考え方をベースにしつつ、AIの力を最大限に引き出す形になります。

- **企画・要件定義**：
  これは人間が担当する最も重要な部分です。
- **設計**：
  AIに生成させる部分と人間が手作業で行う部分を明確にします。
- **プログラミング**：
  AIがコードを生成し、人間がそれをレビュー・修正します。

まとめ

AIは強力なパートナー。そして未来へのメッセージ。

AIは脅威ではなく「強力なパートナー」

AIはプログラマーの仕事を奪うのではなく、私たちの**強力なパートナー**となり、生産性を飛躍的に向上させてくれます。AIを使いこなすことで、より創造的で複雑な課題に挑戦できるようになります。

未来へのメッセージ

未来のプログラマーは、AIを使いこなし、人間ならではの思考力や創造性を発揮できる人材です。 ぜひ、今日お話しした内容を参考に、AI時代を生き抜くためのスキルを磨いていってください。

ご清聴
ありがとうございました。

2025年9月8日

わたびん

一旦ここまで、まだ続きがあります。

付録

用語など色々と補足します

プログラム

「プログラム」（Program）とは、コンピュータを動かすための**指示書**や**命令の集まり**です。 コンピュータは人間のように自分で考えて行動できません。そのため、「このボタンが押されたら画面に『こんにちは』と表示しなさい」といった具体的な命令を、人間がコンピュータに分かるように記述する必要があります。この命令の集まりがプログラムです。スマートフォンアプリ、ウェブサイト、家電製品に組み込まれたソフトウェアなど、私たちが普段利用しているあらゆるデジタル機器は、プログラムによって動いています。

プログラミング

「プログラミング」（Programming）とは、コンピュータに実行させたい処理を、特定のルールに従って記述し、**プログラムを作成する行為**そのものです。 これは、プログラム言語（例：Python、Java、JavaScriptなど）を使ってコードを書く作業を指します。いわば、設計図（プログラム）を**組み立てる工程**です。単にコードを書くだけでなく、目的の動作を実現するために論理的な思考を組み立てたり、エラーを修正したりする作業も含まれます。

プログラマー

「プログラマー」（Programmer）とは、プログラミングを行う**人**、つまり**職業**のことです。 システムエンジニア（SE）などが設計した仕様書に基づき、実際にコードを書いてプログラムを構築します。プログラマーは、アプリケーション、ウェブサービス、組み込みシステムなど、多岐にわたる分野で活躍しており、コンピュータの力を最大限に引き出す専門家といえます。

主要な
プログラミング言語と
その特徴

主要言語の紹介

Python 🐍

- **特徴**: シンプルで読みやすい文法が最大の特徴。初心者にも学習しやすく、多様なライブラリが用意されているため、非常に幅広い分野で利用されています。
- **主な用途**: Webアプリケーション開発（Django, Flask）、AI・機械学習、データ分析、自動化スクリプトなど。

Java ☕

- **特徴**: 「一度書けば、どこでも動く（Write Once, Run Anywhere）」という理念のもと開発された言語です。特定のOSに依存しない汎用性の高さが強みです。
- **主な用途**: Androidアプリ開発、大規模な企業向けシステム開発、Webアプリケーション開発など。

JavaScript ✨

- **特徴**: Webブラウザ上で動作する唯一のプログラミング言語です。Webサイトに動きをつけたり、ユーザーの操作に応じて表示を変えたりするのに不可欠な言語です。
- **主な用途**: Webサイトのフロントエンド開発、サーバーサイド開発（Node.js）、Webアプリケーション開発など。

C言語 / C++ ⚙️

- **特徴**: コンピュータのハードウェアに近い、低水準なプログラミングが可能です。実行速度が非常に速く、OSや組み込みシステムの開発によく使われます。C++はC言語にオブジェクト指向の概念を取り入れた言語です。
- **主な用途**: OS、組み込みシステム、ゲーム開発、ロボット制御、高性能計算など。

PHP 🐘

- **特徴**: Webサイトのサーバーサイド開発に特化した言語で、HTMLに直接コードを埋め込むことができます。WordPressなどの多くのWebサイトで利用されています。
- **主な用途**: Webサイト・Webサービスのサーバーサイド開発。

Ruby 💎

- **特徴**: 「楽しさ」と「生産性」を重視して開発された日本生まれの言語です。シンプルで直感的な文法が特徴で、特にWeb開発フレームワーク「Ruby on Rails」と組み合わせて使われます。
- **主な用途**: Webアプリケーション開発。

Swift 🐦

- **特徴**: Apple社が開発した、安全で高速なプログラミング言語です。Objective-Cに代わり、iOSアプリやmacOSアプリ開発の標準言語となっています。
- **主な用途**: iPhone/iPadアプリ、Macアプリ開発。

環境設定（AIに結構助けられます）

プログラムはAIで作成するとして出来たプログラムを動かすには記述したプログラム言語を動かす**環境が必要**なります。

これをやらないと動作確認や評価テストができません。

個人で使っているパソコンはOS周りの個人設定が個々で微妙に違うので、本やサイトで調べた通りに**インストールしても動かない時**があります。これはプログラムの実行環境が調整しきれ無かった為です。

この手のトラブルはAIを利用すると殆どの場合、改善できます。やり方はChatGPTやGeminiへ不具合状況やエラーコードをプロンプトに入力、そして出てきた指示通りに対処する。これだけです。

纏めると、サイトで一連のインストール方法を確認。トラブル時は**ChatGPTにエラーコードを丸投げ**し対処。

用語説明、他

説明を色々と綴ります

OS (Operating System) 🖥️

OS（オペレーティングシステム）は、コンピュータ全体の基本的な管理を行う基盤ソフトウェアです。ユーザーがコンピュータを操作したり、他のソフトウェアを動かしたりするための環境を提供します。代表的なものに、Windows、macOS、iOS、Androidなどがあります。OSがなければ、コンピュータはただの箱に過ぎません。

ソフトウェアとアプリ 

- **ソフトウェア**は、コンピュータを動かすための命令やデータの総称です。OS、アプリケーション、ゲームなど、コンピュータ上で動作するすべてのプログラムが含まれます。
- **アプリ**は、特定の目的のために作られたソフトウェアのことです。スマートフォンのアプリ（アプリケーションの略）が代表的ですが、PC上で動くExcelやWordなどもアプリケーションソフトウェアの一種です。一般的に、**アプリはソフトウェアの一種**であり、よりユーザーに近い、特定の機能を持つプログラムを指すことが多いです。

マクロ 🤖

マクロは、一連の操作を自動化するために記録・作成されたプログラムです。特に、Microsoft ExcelやWordなどのアプリケーション内で、繰り返し行う作業を効率化するために使われます。ユーザーは一度操作を記録するだけで、次回からはボタン一つで同じ操作を自動実行できます。

API (Application Programming Interface) 🤝

APIは、あるソフトウェアが、別のソフトウェアと連携して情報をやり取りするための**窓口**や**規約**です。例えば、ウェブサイトで地図を表示する際に、Google MapsのAPIを利用することで、複雑な地図データを自社で作成することなく、簡単に組み込むことができます。APIは、異なるソフトウェア同士を連携させ、新たなサービスを生み出すために不可欠な技術です。

バッチ ⚙️

バッチ（Batch）は、複数の処理をひとまとまりにし、自動的に連続して実行するプログラムです。ユーザーが一つ一つの指示を出すことなく、あらかじめ決められた手順で一括処理を行います。主に、夜間にデータを集計・更新する処理や、大量のファイル変換など、人間が手動で行うと手間がかかる作業に使われます。

シェルスクリプト 

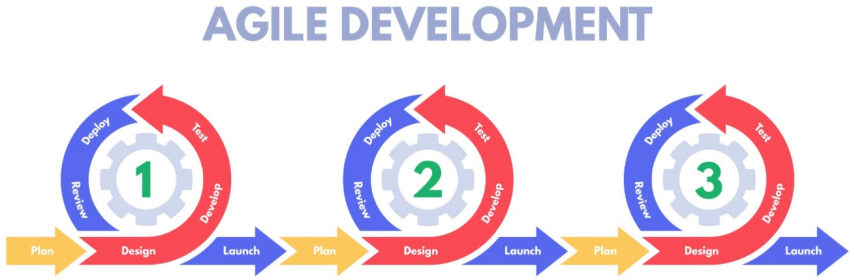
シェルスクリプトは、OSの**シェル**（コマンドを入力してOSを操作するプログラム）を介して、一連のコマンドを自動実行するためのプログラムです。Windowsのコマンドプロンプトや、macOS、Linuxのターミナルなどで使われます。ファイルの一括処理やシステムの定期的なメンテナンスなど、OSの機能を活用した自動化を行う際に非常に便利です。

プログラマーとSEの役割の違い

職種	主な役割	特徴
システムエンジニア（SE）	**設計者**	顧客の要望をヒアリングし、要件定義やシステムの基本設計を行う**上流工程**を担当します。
プログラマー	**実装者**	SEが作成した設計書に基づいて、実際にプログラムを**コーディング**する**下流工程**を担当します。

SEはシステムの「**全体像**」を描く仕事であり、プログラマーはそれを「**具現化**」する仕事です。
SEには顧客との折衝やマネジメント能力、プログラマーには高いコーディングスキルや論理的思考力が求められます。

アジャイル開発とは



アジャイル開発（Agile Development）は、変化に柔軟に対応することを重視した開発手法です。全体を一度に作り上げるのではなく、**小さな機能単位で「開発→テスト→レビュー」というサイクルを繰り返し**、少しずつ完成に近づけていきます。この短いサイクルは「スプリント」や「イテレーション」と呼ばれます。

アジャイル開発の主な特徴

- **変化への柔軟性**: 顧客からの仕様変更や追加要望に柔軟に対応できます。
- **短い開発サイクル**: 1〜4週間程度の短い期間で開発サイクルを繰り返します。
- **顧客との密な連携**: 開発の各段階で顧客とコミュニケーションを取り、フィードバックを素早く反映します。
- **実働するソフトウェアの優先**: ドキュメントよりも、実際に動くソフトウェアを重視します。

開発期間の比較

特徴	アジャイル開発	ウォーターフォールモデル
開発の流れ	開発サイクルを**繰り返し**ます。	企画からリリースまで、**一方向に進みます**。
仕様変更	**柔軟**に対応できます。	**原則として困難**です。後戻りにコストがかかります。
プロジェクト期間	短いサイクル（スプリント）の連続です。	完了までに数ヶ月〜年単位の**長い期間**を要します。
顧客の関与	開発中の各サイクルで**頻繁に**行われます。	主に**要件定義やレビュー時**に限られます。
成果物	サイクルごとに**動く機能**が完成します。	**最終段階**で完成品が納品されます。
リスク	早期に問題を発見し、**リスクを分散**させやすいです。	後戻りが難しく、**リスクが最後に顕在化**しやすいです。

AIを使いより良い
プログラム開発するために

良い指示（プロンプト）のポイント

1. 役割(ペルソナ)を与える 👨‍💻

AIに特定の役割を与えることで、期待する回答の質が向上します。「あなたは熟練のプログラマーです」や「あなたはユーザーへの説明が得意なシステムエンジニアです」といったように、**AIに具体的な役割**を担わせましょう。これにより、AIは専門家として、より的確で質の高い回答を生成しようとします。

例:

- **悪い例:** 「ゲームのコードを書いて。」
- **良い例:** 「あなたは、セキュリティを重視する熟練のゲーム開発者です。C++を使って、ユーザー認証システムの堅牢なコードを書いてください。」

2. ゴールを明確にする 🎯

どのような結果を求めているのか、最終的なゴールを具体的に伝えましょう。単に「プログラムを作って」ではなく、「**〇〇を解決するためのプログラム**」といったように、目的を明確にすることが大切です。

例:

- **悪い例:** 「データを分析するプログラムを作って。」
- **良い例:** 「Excelの売上データを読み込み、前年比の成長率を計算し、グラフとして表示するPythonプログラムを作成してください。」

3. 制約や条件を細かく指定する ⚙️

言語、バージョン、フレームワーク、ファイル形式、出力形式など、できるだけ詳細な条件を指示に含めましょう。AIは与えられた情報をもとに回答を絞り込むため、より精度の高い結果が得られます。

例:

- 「Python 3.10とPandasライブラリを使ってください。」
- 「出力はJSON形式でお願いします。」
- 「コードには、各関数の役割が分かるように、**docstring**（ドキュメンテーション文字列）を追加してください。」

4. 具体的な例を提示する 

AIに「〜のような形式で」「この例のように」と、アウトプットの**具体的なサンプル**を提示することは非常に有効です。これにより、AIは意図を正確に理解し、期待通りの形式で回答を生成しやすくなります。

例:

- 「以下に示すJSONフォーマットに従って、APIのレスポンスを生成してください。」
- 「関数のコメントは、JavaDocの形式でお願いします。」

5. なぜそれが必要かを伝える 💡

- 単に「コメントを書いて」と指示するだけでなく、「**後からメンテナンスする人がコードを理解しやすいように**、機能や引数の説明をコメントに追加してください」といったように、その指示の**背景にある理由**を伝えると、AIはより質の高い、文脈に沿った回答を生成します。
- AIとの協業は、人間同士のコミュニケーションと同様に、**いかに明確で丁寧な指示を出すか**が成功の鍵となります。これらのアドバイスを参考に、ぜひAIを効果的に活用してみてください。

AIによる
プログラミングの
個別指導

プログラムの学習にはAIサポートが最適

1. プログラムの「なぜ」をAIに質問する 🤔

初心者にとって、既存のコードや新しい概念を理解するのは時間がかかります。AIは、その「なぜ」を瞬時に、かつ分かりやすく解説してくれます。

- **コードの解説:** 難しいコードの断片を貼り付けて「このコードは何をしていますか？」「この変数は何のためにありますか？」と質問する。
- **エラーメッセージの解説:** エラーメッセージをそのままAIに貼り付けて、**「このエラーの原因と解決策を教えてください。初心者にも分かるように具体的に説明してください」**と聞く。
- **概念の学習:** 「オブジェクト指向プログラミングを初心者向けに分かりやすく教えてください」「APIとは何ですか？例を挙げて説明してください」といったように、基本的な概念を質問する。

2. コードを「読む力」をAIと養う 

プログラマーにとって、自分でコードを書く力だけでなく、他人のコードを理解する力も非常に重要です。AIを使い、レビューの練習をしましょう。

- **リファクタリングの提案:** 自分が書いたコードをAIに見せて「このコードをより効率的に、あるいは読みやすくするにはどうすればいいですか？」と質問する。
- **ベストプラクティスを学ぶ:** 「Pythonでファイルを扱う際のベストプラクティスを教えてください」といったように、業界の慣習や推奨される書き方をAIに尋ねる。

3. 小さなプロジェクトを一緒に開発する 

AIは、小さなプロジェクトの良きパートナーになります。最初から複雑なものをAIに任せるのではなく、**段階的に**活用しましょう。

- **ステップ1: 計画を立てる**
  - AIに「簡単なToDoリストアプリを作りたいのですが、どのようなステップで進めればいいですか？」と尋ね、全体の開発計画を立ててもらう。
- **ステップ2: 機能を分割して実装する**
  - 「ToDoリストに項目を追加する機能のコードを、PythonとFlaskを使って書いてください」といったように、機能を一つずつAIに依頼し、生成されたコードを理解する。
- **ステップ3: 自分で考えて修正する**
  - AIが生成したコードをそのまま使うのではなく、**「なぜこのような書き方になっているのか」**を考え、自分で修正や追加を試みる。

4. メンターとの連携を怠らない 🤝

AIは素晴らしい学習ツールですが、人間とのコミュニケーションや実務で得られる経験を代替することはできません。

- **AIの限界を理解する:** AIは、最新の技術トレンドや、組織特有のルール、コードの文化までは把握できません。
- **AIの回答を鵜呑みにしない:** AIが生成したコードや解説が常に正しいとは限りません。必ず自分で検証する習慣をつけましょう。
- **人間のメンターに相談する:** AIで解決できない問題や、キャリアに関する悩みは、経験豊富な先輩プログラマーやメンターに積極的に相談することが大切です。

最後までご清聴
ありがとうございました。

2025年9月8日

わたびん

@watabin ← YouTubeチャンネル登録をお願いします。