AI エディタ Windsurf の活用

Windsurf は、AI 機能を統合したコードエディタです。VS Code をベースにしているため、VS Code と同様の操作性で利用できます。

1．メリットと機能

- **無料で利用可能**: 学生や個人開発者向けに無料プランが提供されています。
- **VS Code ベースで、操作法が VS Code と類似**: 全てはありませんが、VS Code の多くの拡張機能をそのまま利用できます。
- **Windsurf Tab**: Tab キーを活用したコード補完および提案機能があります。
- **Cascade 機能**：コード生成や実行を支援する AI 機能があります。

2．Cascade 機能

Cascade はコード生成や実行を支援する AI 機能です。

- **開始方法**: Ctrl＋L キー（同時押し）で Cascade パネルを開きます。
- **使用例**: 例えば、「Python で折れ線グラフのサンプルコードを作成して」のように日本語でリクエストできます。

3．無料プランの機能（2025 年 6 月現在）

- **プロンプトクレジッ*:** 月 25 クレジット（GPT-4.1 プロンプト約 100 回分に相当）。
- **AI モデル**: GPT-4.1（0.25 クレジット/プロンプト）、Claude 3.7 Sonnet（1 クレジット/プロンプト）、DeepSeek-V3-0324（無料）など、複数のモデルを利用できます。
- **その他機能**: 無制限の Fast Tab、SWE-1 Lite などの機能があります。

4．API Key 要件

- **API Key 不要**: サードパーティの API キーは不要です。
- **設定**: Windsurf アカウントの作成のみが必要です（無料で可能）。
- **利用開始**: Windsurf のアカウント登録で利用開始できます。

5．Cursor と Windsurf の比較

Claude 3.7 Sonnet の利用回数では、Cursor 無料版は月 50 回、Windsurf 無料版は月約 25 回

利用可能です。ただし Windsurf では、無料（0 クレジット）のモデル（DeepSeek-V3-0324 など）が無制限で利用できる点が特徴です。

6．Windsurf の起動と初回設定

1. Windsurf を起動します（スタートメニューまたは「windsurf」コマンドを使用)。
2. 「Get Started」をクリックします。
3. VS Code から設定を引き継ぎたい場合は「Import from VS Code」を選択し、そうでない場合は「Start fresh」を選択します。
4. 設定を続行します。
5. 「Log in to Windsurf」の画面で、「Sign up」をクリックしてアカウントを新規作成します。この登録情報は覚えておき、次回からは「Log in」でログインしてください。Google アカウントを使用して Windsurf のアカウント登録も可能です。これでインストールと初期設定は完了です。
6. 動作確認のため、Ctrl＋L（同時押し） で Cascade を開き、「折れ線グラフを描くコードを出して」などと入力します。

7．Windsurf の推奨モデル（クレジット節約のため）

1. DeepSeek-V3-0324 (0 クレジット・恒久的)
2. SWE-1 (0 クレジット・期間限定)
3. SWE-1-lite (0 クレジット・軽量版)

8．DeepSeek-V3-0324 モデルの選択手順

1. Cascade パネル（Ctrl＋L（同時押し））を開きます。
2. 入力欄上部のモデル選択ドロップダウンをクリックします。
3. 使用モデル（例：「DeepSeek-V3-0324」）を選択します。

9．Windsurf をエディタとして活用する

ここでは、Windsurf で、ファイル作成、コード編集、実行を行います。

1. Windsurf の起動

Windsurf を起動します（スタートメニューなどを使用)。**起動直後にログインを求められた場合はログイン**します（通常は自動ログインされます)。**最初の起動では、サインイン（ID とパスワードの登録）が必要**です。サインインの際は、**Google アカウントの利用**を推奨します。

2. **Python と Windsurf のインストールコマンド**
まだインストールしていない人向けのガイドです（**インストール済みの人は実行不要**）。

1. **管理者権限でコマンドプロンプトを起動**します（Windows キー ＞ `cmd` と入力 ＞ 右クリック ＞ 「管理者として実行」）。
2. 以下のコマンドをそれぞれ実行します（`winget` コマンドは **1 つずつ実行**）。

```
winget install --scope machine --id Python.Python.3.12 --id Python.Launcher -e --silent
winget install --scope machine --id Codeium.Windsurf -e --silent
set "INSTALL_PATH=C:¥Program Files¥Python312"
echo %PATH% | find /i "%INSTALL_PATH%" >nul
if errorlevel 1 setx PATH "%PATH%;%INSTALL_PATH%" /M >nul
echo %PATH% | find /i "%INSTALL_PATH%¥Scripts" >nul
if errorlevel 1 setx PATH "%PATH%;%INSTALL_PATH%¥Scripts" /M >nul
set "NEW_PATH=C:¥Program Files¥Windsurf"
if exist "%NEW_PATH%" echo %PATH% | find /i "%NEW_PATH%" >nul
if exist "%NEW_PATH%" if errorlevel 1 setx PATH "%PATH%;%NEW_PATH%" /M >nul
```

3. **新規ファイルの作成**
1. 新規ファイルを作成します（メニューで「File」＞「New File」）。
2. ファイル名を設定します（例: `a.py`）。
このとき、拡張子を「.py」に設定します（Python ファイルとして認識させるため）。
3. ファイル作成を確定します（Enter キー ＞ 「Create File」をクリック）。

4. **プログラムの入力と実行**
1. Windsurf の `a.py` 画面で、以下のプログラムを入力します。

```
print(100 + 200)
```

2. プログラムを実行します（画面上部の三角形の実行ボタンをクリック）。
3. 実行結果として「300」が表示されることを確認します。
ターミナルがない場合には、「View」メニュー ＞ 「Terminal」。

print について: `print()` は Python の関数であり、括弧内の値や計算結果を画面に表示します。この例では、$100 + 200$ の計算結果である 300 が出力されます。

10. Windsurf の AI 機能の活用

ここでは、Windsurf で、本格的なプログラム実行（AI による画像分類を行うプログラム）AI との対話を行ってみます。

1. AI による画像分類プログラムのための必要なライブラリのインストール

管理者権限でコマンドプロンプトを起動（Windows キー ＞ `cmd` と入力 ＞ 右クリック ＞「管理者として実行」）し、**以下のコマンドを実行**します。

pip install tensorflow keras matplotlib numpy pillow japanize-matplotlib

pip について: Python のパッケージ管理システムです。インターネット上のライブラリを自動的にダウンロードし、インストールする機能を持っています。

ライブラリ: プログラミングにおいて、再利用可能なプログラムの集合体です。機械学習や画像処理などの複雑な処理を簡単に実装できます。

各ライブラリの役割:

- **TensorFlow:** Google が開発した機械学習フレームワークです。深層学習（ディープラーニング）の実装に使用されます。
- **Keras:** TensorFlow 上で動作する高レベルな深層学習ライブラリです。より簡単にニューラルネットワークを構築できます。
- **Matplotlib:** グラフや図表を作成するライブラリです。データの可視化に使用されます。
- **NumPy:** 数値計算ライブラリです。多次元配列の操作や数学関数を提供します。
- **Pillow:** 画像処理ライブラリです。画像の読み込み、編集、保存が可能です。
- **japanize-matplotlib:** Matplotlib で日本語を正しく表示するためのライブラリです。

2. 画像分類プログラムの実装

画像分類とは: コンピュータが画像を見て、その画像に写っているものが何かを自動的に判断する技術です。人工知能の代表的な応用分野の一つです。

1. 元のプログラムを削除します（Windsurf エディタの `a.py` 画面内）。
2. **`a.py` 画面**に、**以下のプログラムをコピーして貼り付けます。**

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
import os
import japanize_matplotlib

# CIFAR-10 データセットのダウンロードと読み込み
(x_train, y_train), (x_test, y_test) = ¥
```

```
    keras.datasets.cifar10.load_data()

# データの前処理
x_train = x_train.astype('float32') / 255.0
x_test = x_test.astype('float32') / 255.0
y_train = keras.utils.to_categorical(y_train, 10)
y_test = keras.utils.to_categorical(y_test, 10)

# クラス名の定義
class_names = ['airplane', 'automobile', 'bird', ¥
               'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck']

# CNN モデルの構築
model = keras.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', ¥
                  input_shape=(32, 32, 3)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')
])

# モデルのコンパイル
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# モデルの学習
history = model.fit(x_train, y_train,
                    epochs=5,
                    batch_size=32,
                    validation_data=(x_test, y_test))

# 予測と結果表示
predictions = model.predict(x_test[:5])
predicted_classes = np.argmax(predictions, axis=1)
true_classes = np.argmax(y_test[:5], axis=1)
```

```
# 結果の可視化
plt.figure(figsize=(15, 3))
for i in range(5):
    plt.subplot(1, 5, i+1)
    plt.imshow(x_test[i])
    plt.title(f'予測: {class_names[predicted_classes[i]]}¥n 実際: ¥
{class_names[true_classes[i]]}')
    plt.axis('off')
plt.tight_layout()
plt.show()

# 学習履歴の可視化
plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='訓練精度')
plt.plot(history.history['val_accuracy'], label='検証精度')
plt.title('モデル精度')
plt.xlabel('エポック')
plt.ylabel('精度')
plt.legend()
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='訓練損失')
plt.plot(history.history['val_loss'], label='検証損失')
plt.title('モデル損失')
plt.xlabel('エポック')
plt.ylabel('損失')
plt.legend()
plt.show()

print(f"テスト精度: {model.evaluate(x_test, y_test, verbose=0)[1]:.4f}")
```

プログラムの主要概念説明:

- **CIFAR-10 データセット:** 10 種類の物体（飛行機、自動車、鳥、猫、鹿、犬、カエル、馬、船、トラック）が写った 32×32 ピクセルの小さなカラー画像 60,000 枚で構成される標準的な画像認識用データセットです。
- **データの前処理:** 機械学習では、元のデータをモデルが学習しやすい形に変換する必要があります。ここでは画像の画素値を 0 から 1 の範囲に正規化し、正解ラベルをワンホットエンコーディング（one-hot encoding）という形式に変換しています。
- **CNN (畳み込みニューラルネットワーク):** 画像認識に特化したディープラーニングモデルです。畳み込み層（Conv2D）で画像の特徴を抽出し、プーリング層（MaxPooling2D）で情報を集約する構造を持っています。

- **エポック (epochs)**: 全訓練データを 1 回学習することを 1 エポックと呼びます。ここでは 5 エポック、つまり全データを 5 回繰り返し学習します。
- **バッチサイズ (batch_size)**: 一度に処理するデータの個数です。32 個ずつまとめて処理することで、学習の効率と安定性を向上させます。

3. **プログラムの実行と結果確認**

1. プログラムを実行します（実行ボタンを押してプログラムを開始）。
2. 実行完了まで数分待機します（機械学習の処理時間が必要です）。

実行時間について: 深層学習モデルの訓練は計算量が多いため、通常のプログラムより長い時間を要します。コンピュータの性能により数分程度かかる場合があります。

実行結果の確認手順:

1. 最初に AI による画像分類結果が表示されます（5 枚のテスト画像に対する予測結果と正解の比較）。
2. 画面を切り替えます（確認後、右上の「x」ボタンをクリックして次へ進みます）。
3. 学習曲線（精度と損失の変化）が表示されます（モデルの学習過程を可視化したグラフ）。
4. プログラム実行を終了します（確認後、右上の「x」ボタンをクリック）。

学習曲線の見方: 左のグラフは精度（正解率）の変化を、右のグラフは損失（誤差）の変化を示します。理想的には精度が上昇し、損失が減少する傾向を示します。

4. AI 対話機能の活用

AI 対話機能とは、Windsurf に搭載された人工知能アシスタント機能です。プログラムに関する質問や、コードの説明、改善提案などを自然言語で対話できます。

1. **AI 対話パネルの起動** (Ctrl + L（同時押し） キーを同時押し)。
2. 右側に AI 対話用パネルが開きます。

ショートカットキーについて: Ctrl + L（同時押し） は「Chat パネルを開く」という Windsurf の標準ショートカットキーです。

注意: AI 対話パネルが「Loading...」と表示されて使用できない場合は、Windsurf の再起動（一度 Windsurf を終了し、再起動）で解決する場合があります。これは AI サービスへの接続に時間がかかる場合に発生する現象です。

3. **AI との対話例**

AI への質問（右側の対話画面で以下の質問を試してください）：

- 「このプログラムの機能は？」 (Enter キー)

- 「このプログラムの使い方を具体的に教えて？」 (Enter キー)
- 「このプログラムを使って何が研究できるの？研究中間発表（研究目標、研究課題、取り組み、期待される成果、独自の工夫予定）のサンプルを簡潔に教えて。そのとき、社会課題の解決を考えて。」 (Enter キー)

AI との効果的な対話:

- 具体的で明確な質問をする。
- 専門用語が分からない場合は「初心者向けに説明して」と付け加える。
- 複数の質問がある場合は、一つずつ順番に聞く。
- AI の回答に対してさらに詳しく聞きたい場合は「もっと詳しく教えて」と追加質問する。

１１．Windsurf の重要機能一覧

インテリセンス（自動補完）機能

- **操作手順:** コード入力中に自動的に候補が表示され、Tab キーで選択します。
- **メリット:** タイプミス防止、開発効率の向上、関数名や変数名の正確な入力。

統合ターミナル機能

- **ターミナル起動:** 「View」メニュー ＞ 「Terminal」、または `Ctrl＋` キー。
- **メリット:** エディタを離れることなくコマンド実行、ライブラリインストールが可能。

AI コード生成機能

- **AI コード生成の起動:** `Ctrl＋I` キー、または AI パネルで「〇〇〇のコードを生成して」と依頼。
- **メリット:** 複雑なコードの自動生成、学習時間の短縮、コーディングパターンの習得。

AI コード解説機能

- **コード解説の実行:** コードを選択して右クリック ＞ 「Explain」、または AI パネルで質問。
- **メリット:** 理解困難なコードの詳細解説、学習効果の向上。

AI バグ修正支援機能

- **バグ修正支援の起動:** エラー箇所を選択して AI パネルで「このエラーを修正して」と依頼。
- **メリット:** デバッグ時間の短縮、エラー原因の理解促進。

ファイルエクスプローラー機能

- **ファイル選択:** 左側のサイドバーで自動表示され、フォルダやファイルをクリックで選択。
- **メリット:** 複数ファイルの効率的な管理、プロジェクト全体の把握。

デバッガー機能

- **デバッグの開始:** ブレークポイント設定でライン番号をクリックし、`F5` キーでデバッグ開始。
- **メリット:** 実行時の変数値確認、ステップ実行による詳細な動作解析。

問題検出機能

- **問題確認:** Problems パネルで自動表示される警告やエラーを確認。
- **メリット:** コンパイル前のエラー検出、コード品質の向上。