

生成AIツールを利用したフラッシュカード作成事例 (1/8)

生成AIはさまざまなプログラミング言語でコードを書くことに長けており、そのような機能は幅広く活用されています。

ただし、以下のようなデメリットがあります。

- ・生成AIは確率的なモデルのため、同じ命令（プロンプト）でも出力されるものが違うことがある。
- ・安定性や一貫性のある出力をさせるためには、各生成AIツールの特性やツールの設定方法、プロンプトでの指示の方法について詳しく知る必要がある。

そこで、既に動作が確認できているフラッシュカードのサンプルプログラムを活用し、NotebookLMを使用して簡単かつ安全に生成AIツールを使ってプログラムを作成する方法をご紹介します。

NotebookLMについて

NotebookLMは、Googleが開発したAI搭載の研究 アシスタントです。アップロードしたソース（PDF、テキスト、画像、音声ファイル、ウェブサイト、Youtube動画など）をもとに、AIが要約、整理・分析、説明、質問への回答などを生成します。

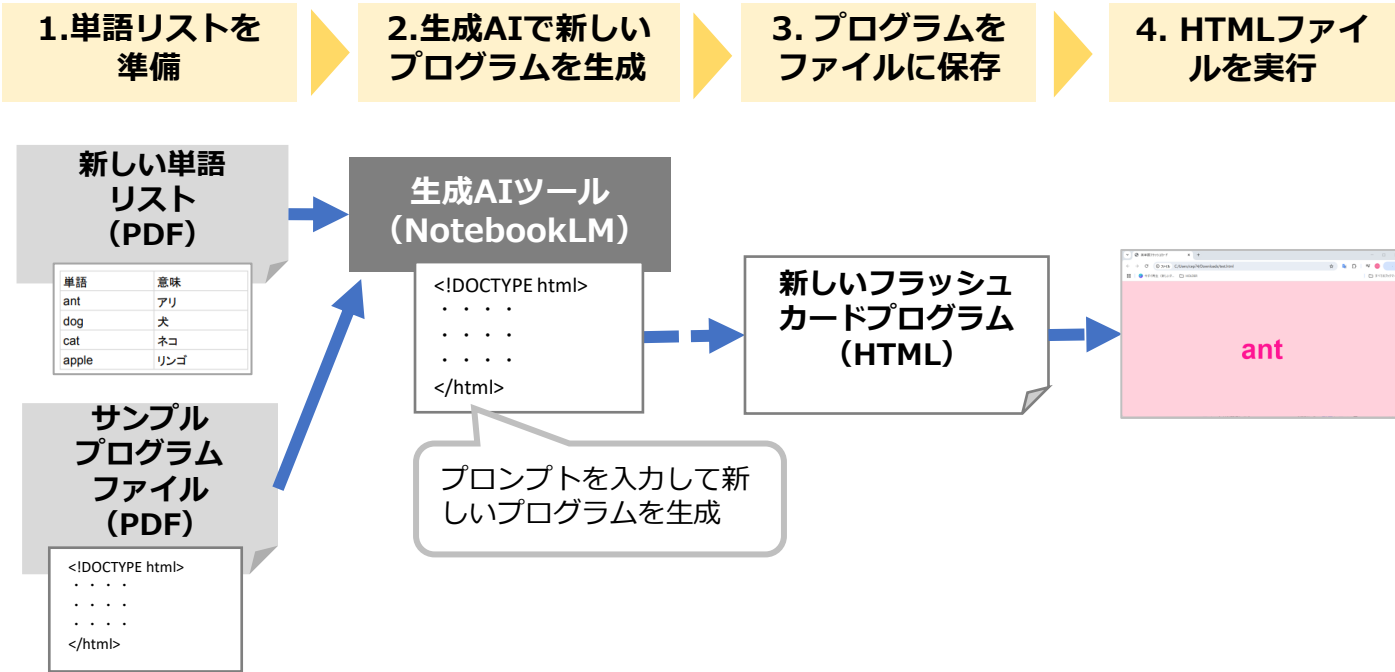
ソースとなる情報が誤って解釈される可能性があるため、回答が正しいかどうか再確認する必要があります。

- ・利用可能年齢は、**18歳以上**（先生のみ利用可能）です。（2025年7月15日現在）
- ・利用にはGoogleアカウントが必要です。
- ・個人データ（アップロードしたソース、質問、回答など）はNotebookLMのトレーニングには使用されないとされています。

※AIツールの利用に関しては、各自治体の方針やルールを必ずご確認ください。

生成AIツールを利用したフラッシュカード作成事例 (2/8)

プログラム作成の流れ



サンプルのフラッシュカードプログラムの動作イメージ



生成AIツールを利用したフラッシュカード作成事例 (3/8)

サンプルのフラッシュカードプログラム

```
<!DOCTYPE html>
<html lang="ja">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>単語フラッシュカード</title>
  <style>
    body {
      display: flex;
      justify-content: center;
      align-items: center;
      min-height: 100vh;
      margin: 0;
      font-family: 'Arial', sans-serif;
      overflow: hidden; /* スクロールバーを非表示にする */
    }
    .flashcard-container {
      width: 100vw;
      height: 100vh;
      display: flex;
      justify-content: center;
      align-items: center;
      font-size: 10vw; /* 画面幅に合わせて文字サイズを調整 */
      font-weight: bold;
      color: #333;
      text-align: center;
      transition: background-color 0.5s ease;
      cursor: pointer;
      position: absolute;
      top: 0;
      left: 0;
    }
    .pink-bg {
      background-color: #FFD1DC; /* ピンク */
    }
    .blue-bg {
      background-color: #ADD8E6; /* 水色 */
    }
    .text-pink {
      color: #FF1493; /* 濃いピンク */
    }
    .text-blue {
      color: #0000FF; /* 青 */
    }
  </style>
</head>
<body>

  <div id="card1" class="flashcard-container pink-bg"
  onclick="showNextCard(1)">
    <span class="text-pink">desert</span>
  </div>
  <div id="card2" class="flashcard-container blue-bg"
  style="display:none;" onclick="showNextCard(2)">
    <span class="text-blue">砂漠</span>
  </div>

  <div id="card3" class="flashcard-container pink-bg"
  style="display:none;" onclick="showNextCard(3)">
    <span class="text-pink">beach</span>
  </div>
  <div id="card4" class="flashcard-container blue-bg"
  style="display:none;" onclick="showNextCard(4)">
    <span class="text-blue">砂浜</span>
  </div>

  <div id="card5" class="flashcard-container pink-bg"
  style="display:none;" onclick="showNextCard(5)">
    <span class="text-pink">ocean</span>
  </div>
  <div id="card6" class="flashcard-container blue-bg"
  style="display:none;" onclick="showNextCard(6)">
    <span class="text-blue">大洋</span>
  </div>

  <div id="card7" class="flashcard-container pink-bg"
  style="display:none;" onclick="showNextCard(7)">
    <span class="text-pink">valley</span>
  </div>
  <div id="card8" class="flashcard-container blue-bg"
  style="display:none;" onclick="showNextCard(8)">
    <span class="text-blue">谷</span>
  </div>

</body>
</html>
```

```
<script>
  let currentIndex = 1;
  const totalCards = 8;

  function showNextCard(clickedIndex) {
    // 現在表示されているカードを非表示にする
    const currentCard = document.getElementById('card' +
currentCardIndex);
    if (currentCard) {
      currentCard.style.display = 'none';
    }

    // 次のカードのインデックスを計算
    currentIndex++;
    if (currentIndex > totalCards) {
      currentIndex = 1; // 最後のカードまで行ったら最初に戻る
    }

    // 次のカードを表示する
    const nextCard = document.getElementById('card' +
currentCardIndex);
    if (nextCard) {
      nextCard.style.display = 'flex'; // flexを使って中央寄せを維持
    }
  }
</script>
</body>
</html>
```

生成AIツールを利用したフラッシュカード作成事例 (4/8)

サンプルのフラッシュカードプログラム概要

```
<!DOCTYPE html>
<html lang="ja">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>単語フラッシュカード</title>
  <style>
    body {
      display: flex;
      justify-content: center;
      align-items: center;
      min-height: 100vh;
      margin: 0;
      font-family: 'Arial', sans-serif;
      overflow: hidden; /* スクロールバーを非表示にする */
    }
    .flashcard-container {
      width: 100vw;
      height: 100vh;
      display: flex;
      justify-content: center;
      align-items: center;
      font-size: 10vw; /* 画面幅に合わせて文字サイズを調整 */
      font-weight: bold;
      color: #333;
      text-align: center;
      transition: background-color 0.5s ease;
      cursor: pointer;
      position: absolute;
      top: 0;
      left: 0;
    }
    .pink-bg {
      background-color: #FFD1DC; /* ピンク */
    }
    .blue-bg {
      background-color: #ADD8E6; /* 水色 */
    }
    .text-pink {
      color: #FF1493; /* 濃いピンク */
    }
    .text-blue {
      color: #0000FF; /* 青 */
    }
  </style>
</head>
<body>

  <div id="card1" class="flashcard-container pink-bg"
  onclick="showNextCard(1)">
    <span class="text-pink">desert</span>
  </div>
  <div id="card2" class="flashcard-container blue-bg"
  style="display:none;" onclick="showNextCard(2)">
    <span class="text-blue">砂漠</span>
  </div>

  <div id="card3" class="flashcard-container pink-bg"
  style="display:none;" onclick="showNextCard(3)">
    <span class="text-pink">beach</span>
  </div>
  <div id="card4" class="flashcard-container blue-bg"
  style="display:none;" onclick="showNextCard(4)">
    <span class="text-blue">砂浜</span>
  </div>

  <div id="card5" class="flashcard-container pink-bg"
  style="display:none;" onclick="showNextCard(5)">
    <span class="text-pink">ocean</span>
  </div>
  <div id="card6" class="flashcard-container blue-bg"
  style="display:none;" onclick="showNextCard(6)">
    <span class="text-blue">大洋</span>
  </div>

  <div id="card7" class="flashcard-container pink-bg"
  style="display:none;" onclick="showNextCard(7)">
    <span class="text-pink">valley</span>
  </div>
  <div id="card8" class="flashcard-container blue-bg"
  style="display:none;" onclick="showNextCard(8)">
    <span class="text-blue">谷</span>
  </div>

</body>
</html>
```

レイアウト、色、
文字サイズ等の設定

単語リスト

```
<script>
  let currentCardIndex = 1;
  const totalCards = 8;

  function showNextCard(clickedIndex) {
    // 現在表示されているカードを非表示にする
    const currentCard = document.getElementById('card' +
currentCardIndex);
    if (currentCard) {
      currentCard.style.display = 'none';
    }

    // 次のカードのインデックスを計算
    currentCardIndex++;
    if (currentCardIndex > totalCards) {
      currentCardIndex = 1; // 最初からやり直す
    }

    // 次のカードを表示する
    const nextCard = document.getElementById('card' +
currentCardIndex);
    if (nextCard) {
      nextCard.style.display = 'flex'; // flexを使って中央寄せを維持
    }
  }
</script>
</body>
</html>
```

カードの動き

生成AIツールを利用したフラッシュカード作成事例 (5/8)

1. 単語リストを
準備

2. 生成AIで新しい
プログラムを生成

3. プログラムを
ファイルに保存

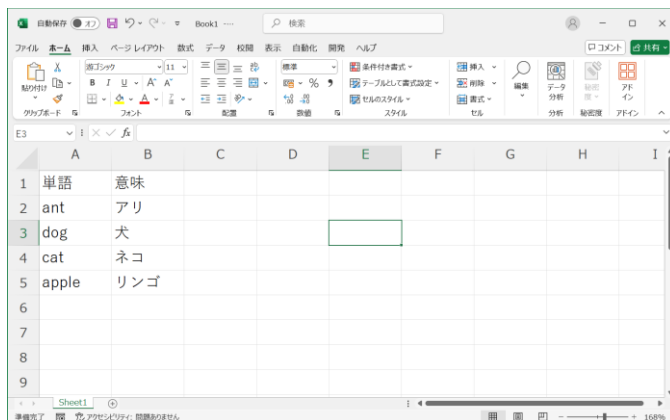
4. HTMLファイ
ルを実行

1. 単語リストを準備する

- ① Excelなどの表計算アプリを起動して、「単語」「意味」のセットで一行ずつ入力します。

※注意

単語の数が多いほど、プログラム生成に時間がかかります。
単語の数は、最大100件程度を推奨します。



- ② PDF形式でファイルに保存します。

生成AIツールを利用したフラッシュカード作成事例 (6/8)

1. 単語リストを
準備

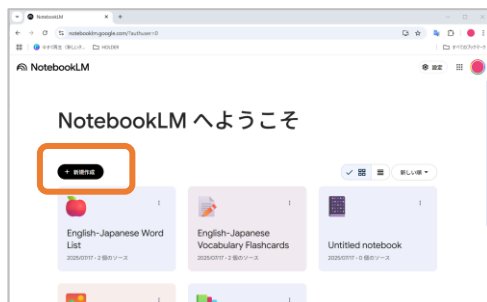
2. 生成AIで新しい
プログラムを生成

3. プログラムを
ファイルに保存

4. HTMLファイ
ルを実行

2. 生成AIで新しいプログラムを生成する

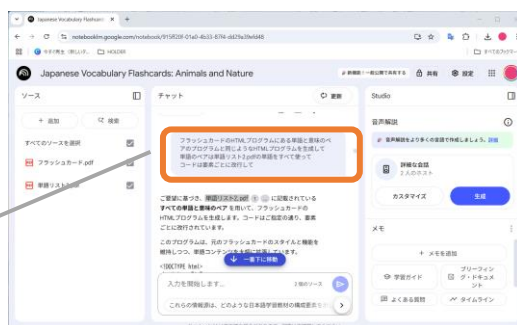
① NotebookLMを起動して新規作成します。



② 「ソースをアップロード」からテンプレートのプログラムファイル（フラッシュカード.pdf）と、1で出力した単語リストのファイル（PDF）を指定してアップロードします。



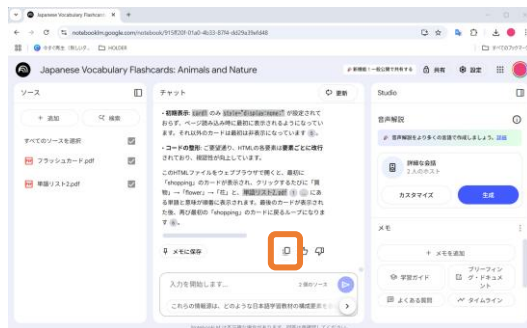
③ HTMLプログラムを生成する命令をプロンプトに入力します。



(プロンプト例)

フラッシュカードのHTMLプログラムにある単語と意味のペアのプログラムと同じようなHTMLプログラムを生成して単語のペアは単語リスト2.pdfの単語をすべて使ってコードは要素ごとに改行して

④ 生成されたHTML形式のテキストをコピーします。



生成AIツールを利用したフラッシュカード作成事例 (7/8)

1. 単語リストを準備

2.生成AIで新しい プログラムを生成

3. プログラムを ファイルに保存

4. HTMLファイルを実行

3. プログラムをファイルに保存する

- ① テキストエディタ（メモ帳など）を開き、コピーしたテキストを貼り付けます。

※注意

テキストを貼り付けた後、以下のHTMLタグの範囲以外の文字は削除します。

```
<!DOCTYPE html> ←プログラムの始まり
. . . . .
. . . . .
. . . . .
</html> ←プログラムの終わり
```

```

<div id="card5" class="flashcard-container pink-bg" style="display:none;" onclick="showNextCard(5)">
  <span class="text-pink">ocean</span>
</div>
<div id="card6" class="flashcard-container blue-bg" style="display:none;" onclick="showNextCard(6)">
  <span class="text-blue">次洋</span>
</div>
<div id="card7" class="flashcard-container pink-bg" style="display:none;" onclick="showNextCard(7)">
  <span class="text-pink">valley</span>
</div>
<div id="card8" class="flashcard-container blue-bg" style="display:none;" onclick="showNextCard(8)">
  <span class="text-blue">谷</span>
</div>

<script>
let currentCardIndex = 1;
const totalCards = 8;

function showNextCard(clickedIndex) {
  // 現在表示されているカードを非表示にする
  const currentCard = document.getElementById('card' + currentCardIndex);
  if (currentCard) {
    currentCard.style.display = 'none';
  }

  // 次のカードのインデックスを計算
  currentCardIndex++;
  if (currentCardIndex > totalCards) {
    currentCardIndex = 1; // 最後のカードまで行ったら最初に戻る
  }

  // 次のカードを表示する
  const nextCard = document.getElementById('card' + currentCardIndex);
  if (nextCard) {
    nextCard.style.display = 'flex'; // flexを使って中央寄せを維持
  }
}
</script>
</body>
</html>

```

- ② 名前を付けて保存をする時に、ファイルの種類を指定せず、ファイル名に「フラッシュカード.html」のように拡張子にhtmlを付けてファイルを保存します。

Figure 1: Screenshot of a Windows File Explorer window showing the 'Documents' folder. The 'Files and folders' pane on the left shows a list of files and folders. The 'Main pane' shows a table of files with columns for 'Name', 'Last modified', and 'Type'. The file 'flexCard(8).txt' is selected. The 'Address bar' shows the path 'C:\Users\user\Documents'. The 'Status bar' at the bottom shows '104 行, 1 列', '100%', 'Windows (CRLF)', and 'UTF-8'.

生成AIツールを利用したフラッシュカード作成事例 (8/8)

1. 単語リストを
準備

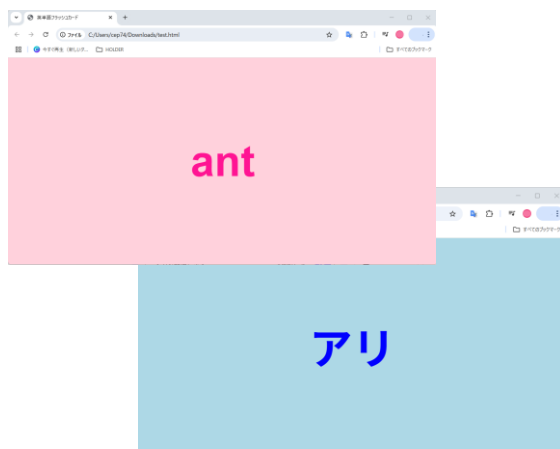
2. 生成AIで新しい
プログラムを生成

3. プログラムを
ファイルに保存

4. HTMLファイ
ルを実行

4. HTMLファイルを実行する

- ① 保存したHTMLファイルをダブルクリックするとブラウザで実行され、新しい単語リストの単語が表示されます。
- ② カードをクリックすると、意味のカードが表示されます。クリックしていくと、単語、意味が順番に表示されます。



※注意事項

・ NotebookLMは、ソースとなる情報が誤って解釈される可能性があるため、回答が正しいかどうか再確認する必要があります。

・ 作成したHTMLプログラムは、お使いになる前に必ず動作確認を行ってください。

・ サンプルプログラムや、プロンプトの入力例は、あくまでサンプルですので、必ずしも同じ動作を保証するものではありません。