

## プログラミング演習④(Google Apps Script) + ChatGPT

Google Apps Script (以降 GAS) は Google 社が提供しているプログラミング言語である。Google メールや Google スプレッドシートなどの Google アプリの操作を自動化したりすることができるだけでなく、WebAPI を利用して、数多くのアプリと連携させることができる。GAS はベースが JavaScript ので、基本的な構文等を学習するときは JavaScript の記事を読めばよい。ここでは GAS の環境を使って、より実戦的なプログラム作成の手順を説明する。

### 【課題】

テスト結果を HRNO を 4 桁の数字で表したファイル名にして PDF ファイルで書き出したものを図 1 のように同じフォルダに入れておく（ファイル名の例；1 年 1 組 1 番のファイルは「1101.pdf」）。また、スプレッドシートを作り HRNO と個別のメールアドレスとファイル名を入れる（図 2）。このデータを読み取り、個別にテスト結果の PDF ファイルをメールで送信するプログラムを作りなさい。なお、このプログラムを使ってスパムメールなどを送信する行為は法に触れる。



図 1 ファイル

	A	B	C
名前	HRNO	メールアドレス	ファイル名
1	1101	20241101@gl.xxx-xxxxxxx.ed.jp	1101.pdf
2	1102	20241102@gl.xxx-xxxxxxx.ed.jp	1102.pdf
3	1103	20241103@gl.xxx-xxxxxxx.ed.jp	1103.pdf
4	1104	20241104@gl.xxx-xxxxxxx.ed.jp	1104.pdf
5	1105	20241105@gl.xxx-xxxxxxx.ed.jp	1105.pdf

図 2 GAS 個別ファイル配布ファイル

始めは、図書館で GAS の本を借りたり、ネットで検索したりする。図書館で本を借りてきて、GAS についての基礎知識を習得したり、目的にあった内容の記事を見つける。

本には、GAS のスクリプトには、Google ドライブ上にファイルを作るスタンドアロン型と、スプレッドシートなどのファイルを作りおいて作るコンテナバインド型の二つがあると書かれており、今回は図 2 の対応表がスプレッドシートで作られているので、そのファイルに組み込むコンテナバインド型の GAS を作った方がいいと考える。

頭 1 のファイルを作っておき、同じフォルダに入れる。GAS 個別ファイル配布ファイルを開き、本に書いてある通りの操作をする。

A	B	C
1 HRNO	メールアドレス	ファイル
2 1101 20241101@gl.xxx-xxxxxx.ed.jp		1101.pdf
3 1102 20241102@gl.xxx-xxxxxx.ed.jp		1102.pdf
4 1103 20241103@gl.xxx-xxxxxx.ed.jp		1103.pdf
5 1104 20241104@gl.xxx-xxxxxx.ed.jp		1104.pdf
6 1105 20241105@gl.xxx-xxxxxx.ed.jp		1105.pdf

プログラムを書く、準備ができる

無題のプロジェクトをクリックして、ファイル名を帰る

本を参考にして、文字列を出力するプログラムを書く

```

1 function myFunction() {
2   console.log("Hello, World!")
3 }

```

以下の実行ログに文字列が表示される。

実行ログ		
14:46:40	お知らせ	実行開始
14:46:39	情報	Hello, World!
14:46:41	お知らせ	実行完了

基本的なことは学習済みとして、目的のプログラムを作ることにする。

検索をして、それらしいプログラムを探すと下記のようなページが検索される。

**Gmailを一斉送信するとき個別に添付ファイルを送りたい。GAS(Google Apps Script)**

♡ 58  
金太郎 #プログラミング #学校 #国際政治経済  
2022年3月18日 22:09

スプレッドシートにあるメールリストに一斉送信したい。BCCでは送りたくない。本文にはそれぞれの所属と名前を入れたい。添付ファイルも付けたい。

(参照) Note Gmail を一斉送信するとき個別に添付ファイルを送りたい。GAS(Google Apps Script)  
(<https://note.com/edup/n/nfe8259c9bfe4>)

他にも下記のようなページがあるので一通り目を通す。

(参考)

初めての GAS(Google Apps Script)|スプレッドシートの内容を本文に差し込んで、添付ファイルも付けて、Gmail一斉送信してみた。  
<https://note.com/edup/n/n855b4e6f3be9>

スプレッドシートの内容を本文に差し込んで、添付ファイルも付けて、Gmail一斉送信してみた。【改訂版】

<https://note.com/edup/n/n1f1021f7a0e4>

それらを参考にしてプログラムを作る

```
1  function sendMail(){
2
3  const sheet = SpreadsheetApp.getActiveSheet();
4  const lastRow = sheet.getLastRow();
5
6  const subject = '1学期期末 試験結果';
7
8  for(let i = 2; i <= lastRow; i++){
9
10    const address = sheet.getRange(i, 2).getValue();
11    const fileName = sheet.getRange(i, 3).getValue();
12    const body = "HRNO" + syozoku + "\n"
13
14    テストの結果をメールに添付しました。
15    ご確認ください。
16  ;
17
18  var tempu = DriveApp.getFilesByName(fileName).next();
19
20  GmailApp.sendEmail(address, subject, body, {
21    name: 'T.Matsuura',
22    attachments: [tempu]});
23
24 }
```

実行ボタンを押して実行する



エラーが出てきて動かず、プロジェクトも保存できない・

The screenshot shows the Google Sheets script editor interface. The top menu bar includes '実行' (Run), 'デバッグ' (Debug), and 'myFunction' dropdown. A blue button labeled '実行ログ' (Execution Log) is highlighted. The code editor contains the following script:

```
14 テストの結果をメールに添付しました。
15 ご確認ください。
16 ;
17
18 var tempu = DriveApp.getFilesByName(fileName).next();
19
20 GmailApp.sendEmail(address, subject, body, {
21   name: 'T.Matsuura',
22   attachments: [tempu]});
23
24 }
```

The execution log at the bottom shows:

15:00:56 エラー myFunction を実行しようとしましたが、プロジェクトを保存できませんでした。

構文エラー: SyntaxError: Unexpected end of input 行: 24 ファイル: コード.gs [コピー](#)



右側にブレークポイントを設定するように出ているので、ソースの左側をクリックしてブレークポイントを設定する。

The screenshot shows the Google Sheets script editor with a detailed explanation of breakpoints.

Left panel (Message Box):

ブレークポイントが設定されています。  
デバッグを開始するには、プロジェクトに少なくとも1つのブレークポイントまたは「debugger」ステートメントを追加してください。

Right panel (Script Editor):

Breakpoint icon (purple dot) circled in red, indicating it's set at line 8.

```
1 for(let i = 2; i <= lastRow; i++){
2   const address = sheet.getRange(i, 2).getValue();
3 }
```

A callout box points to the breakpoint icon with the text: 'ブレークポイントが設定されています'.

同じ状態になるが、どこがおかしいか示してくれる。



ここで、自分でチェックしてもよいのだが、ここでは、そのデバッグ作業を ChatGPT を使ってやることにする。

ChatGPT を検索して、ChatGPT のサイトにアクセスをする。

The screenshot shows a Google search results page. The main search term is 'ChatGPT'. Below the search bar, three suggestions are listed: 'ChatGPT - Google 検索', 'chatgpt login', and 'OpenAI'. The 'ChatGPT - Google 検索' suggestion is highlighted.

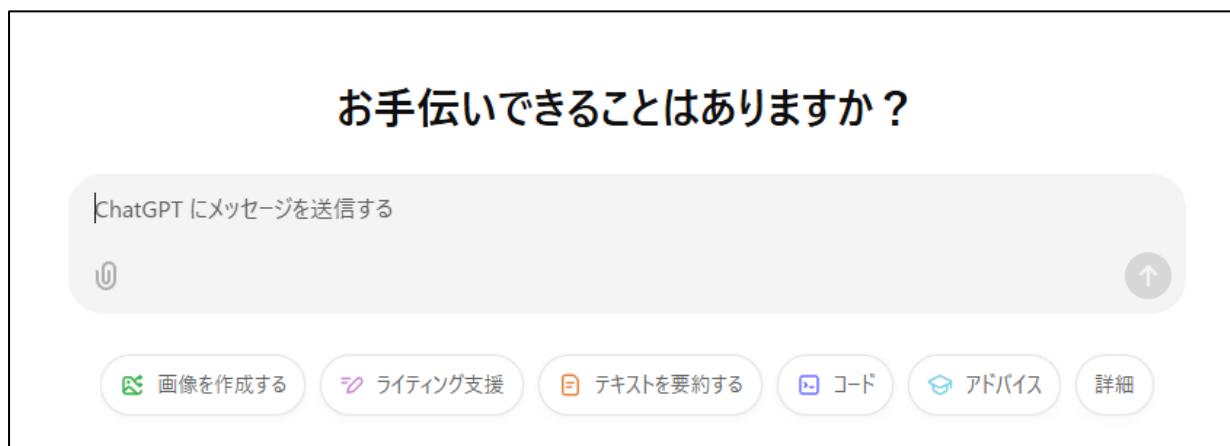
The screenshot shows the OpenAI ChatGPT landing page. It features the OpenAI logo and the text 'ChatGPT'. Below the heading, a paragraph reads: '無料でかんたんに使えるChatGPTに質問するだけで、文章作成や学び、アイデア出しをお手伝いします。今すぐ始める（新しいウィンドウで開く）アプリをダウンロードする。' To the right, there is a small image of a smartphone displaying the ChatGPT interface.

今すぐ始めるボタンを押す



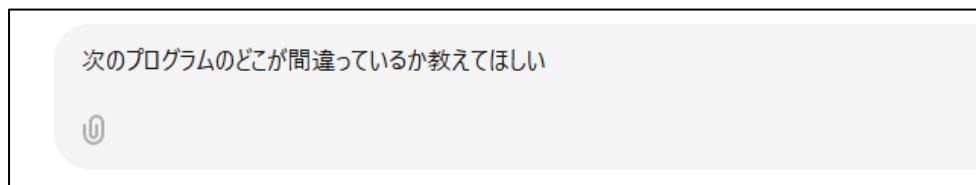
The screenshot shows the official ChatGPT landing page. At the top, there's a navigation bar with 'OpenAI' on the left and links for '研究' (Research), '製品' (Products), '安全性' (Safety), and '企業情報' (Enterprise Information) on the right. Below this is another row with 'ChatGPT' on the left and '無料版とPlus' (Free and Plus), 'Team', 'Enterprise', and 'Education' on the right. The main headline 'ChatGPT' is centered above the text '答えを手に。ひらめきを感じて。生産性を新たな次元へ。' (Get answers at your fingertips. Feel the spark of inspiration. Take productivity to a new dimension.). Below the headline is a subtext: '無料でかんたんに使えるChatGPTに質問するだけで、文章作成や学び、アイデア出しをお手伝いします。' (Ask questions directly to the free and easy-to-use ChatGPT, and it will help you with article creation, learning, and idea generation.). At the bottom of the main section are two buttons: '今すぐ始める' (Start now) and 'アプリをダウンロードする' (Download the app).

## お手伝いできることはありますか？



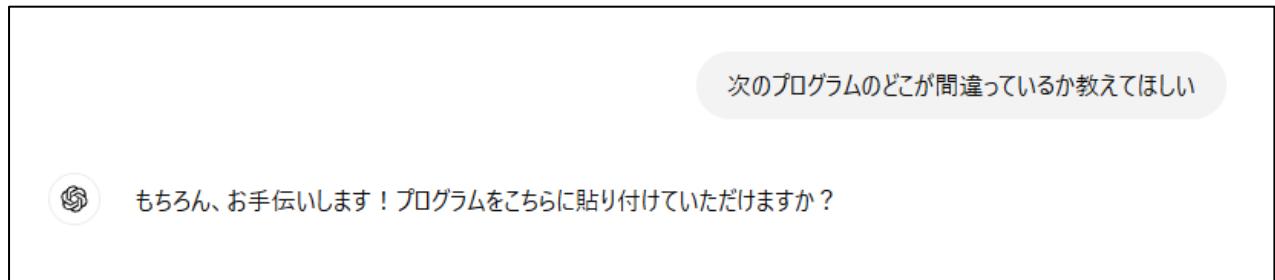
A large input field for messaging ChatGPT. Below it are several circular buttons with icons and text: '画像を作成する' (Create image), 'ライティング支援' (Writing support), 'テキストを要約する' (Summarize text), 'コード' (Code), 'アドバイス' (Advice), and '詳細' (Details). There are also small 'Up' and 'Down' arrows in the corners of the message area.

お手伝いできることはあるか聞いてくるので、ChatGPT に質問をする



An input field containing the text '次のプログラムのどこが間違っているか教えてほしい' (Please tell me where the following program is wrong). Below the input field is a small 'Up' arrow.

答えてくれるので、対話をしながら解決していく



The AI responds with a message: 'もちろん、お手伝いします！プログラムをこちらに貼り付けていただけますか？' (Of course, I'll help you! Please paste the program here). The AI's icon is a blue robot head.

プログラムコードを選択してコピーして、メッセージを送信するに貼り付ける。

```
function sendMail(){

const sheet = SpreadsheetApp.getActiveSheet();
const lastRow = sheet.getLastRow();

const subject = '1 学期期末 試験結果';

for(let i = 2; i <= lastRow; i++){

    const address = sheet.getRange(i, 2).getValue();
    const fileName = sheet.getRange(i, 3).getValue();
    const body = "HRNO" + syozoku + "\n"

    テストの結果をメールに添付しました。
    ご確認ください。
    ;

    var tempu = DriveApp.getFilesByName(fileName).next();

    GmailApp.sendEmail(address, subject, body, {
        name: 'T.Matsuura',
        attachments: [tempu]});
}
}
```

修正すべき場所を指摘してくれる。

⚙️ プログラムを見たところ、いくつかの問題があります。以下に修正点を説明します。

- 1. syozoku 変数が定義されていない**

コード内の `HRNO + syozoku + "\n"` という部分で `syozoku` という変数が使われていますが、これがどこにも定義されていません。`syozoku` の値を取得する方法（例えばシートから取得する）を記述する必要があります。
- 2. body の構文エラー**

`body` の宣言部分でテンプレートリテラル（バッククオート）を使っていますが、構文が不完全です。バッククオートで始まっているのに、閉じるバッククオートが不足しています。
- 3. DriveApp.getFilesByName(fileName).next() の使用**

`DriveApp.getFilesByName(fileName).next()` でファイルを取得しているが、`next()` はファイルが1つしか見つからない場合にしか使えません。もしファイルが見つからなかった場合、エラーが発生します。ファイルが存在しない場合に備えてチェックを入れるべきです。

## 修正したコード

以下は修正されたコードです：

```
javascript Copiar código  
  
function sendMail() {  
  const sheet = SpreadsheetApp.getActiveSheet();  
  const lastRow = sheet.getLastRow();  
  const subject = '1学期期末 試験結果';  
  
  // 各行をループ処理  
  for (let i = 2; i <= lastRow; i++) {  
    const address = sheet.getRange(i, 2).getValue(); // メールアドレス  
    const fileName = sheet.getRange(i, 3).getValue(); // ファイル名  
    const syozoku = sheet.getRange(i, 4).getValue(); // 例: 所属の情報を取得する場合  
  
    // メール本文を構築  
    const body = `HRNO: ${syozoku}\nテストの結果をメールに添付しました。`;  
  
    // ファイルを取得  
    const files = DriveApp.getFilesByName(fileName);  
  
    if (files.hasNext()) {  
      const file = files.next(); // 最初のファイルを取得  
      GmailApp.sendEmail(address, subject, body, {  
        name: 'T.Matsuura',  
        attachments: [file] // 添付ファイル  
      });  
    } else {  
      Logger.log(`ファイルが見つかりません: ${fileName}`);  
    }  
  }  
}
```

## 主な変更点：

1. `syozoku` をシートの列から取得する部分を追加しました。例えば、`sheet.getRange(i, 4).getValue()` と  
していますが、これはシートの4列目に所属の情報がある場合です。適宜変更してください。
2. `body` のテンプレートリテラルが正しく閉じられていなかったため修正しました。
3. `DriveApp.getFilesByName(fileName)` を使ってファイルを取得する際に、ファイルが存在するかをチェックするコードを追加しました。

これでプログラムが正常に動作するはずです。



これらを参考にしてプログラムを修正する。

```
1  function sendMail() {
2      const sheet = SpreadsheetApp.getActiveSheet();
3      const lastRow = sheet.getLastRow();
4      const subject = '1学期期末 試験結果';
5
6      // 各行をループ処理
7      for (let i = 2; i <= lastRow; i++) {
8          const hrno = sheet.getRange(i, 1).getValue(); // HRNO
9          const address = sheet.getRange(i, 2).getValue(); // メールアドレス
10         const fileName = sheet.getRange(i, 3).getValue(); // ファイル名
11
12         // メール本文を構築
13         const body = `HRNO: ${hrno}\nテストの結果をメールに添付しました。\\nご確認ください。`;
14
15         // ファイルを取得
16         const files = DriveApp.getFilesByName(fileName);
17
18         if (files.hasNext()) {
19             const file = files.next(); // 最初のファイルを取得
20             GmailApp.sendEmail(address, subject, body, {
21                 name: 'T.Matsuura',
22                 attachments: [file] // 添付ファイル
23             });
24         } else {
25             Logger.log(`ファイルが見つかりません: ${fileName}`);
26         }
27     }
28 }
29 }
```

```

<コード>
function sendMail() {
  const sheet = SpreadsheetApp.getActiveSheet();
  const lastRow = sheet.getLastRow();
  const subject = '1 学期期末 試験結果';

  // 各行をループ処理
  for (let i = 2; i <= lastRow; i++) {
    const hrno = sheet.getRange(i, 1).getValue(); // HRNO
    const address = sheet.getRange(i, 2).getValue(); // メールアドレス
    const fileName = sheet.getRange(i, 3).getValue(); // ファイル名

    // メール本文を構築
    const body = `HRNO: ${hrno}\nテストの結果をメールに添付しました。ご確認ください。`;

    // ファイルを取得
    const files = DriveApp.getFilesByName(fileName);

    if (files.hasNext()) {
      const file = files.next(); // 最初のファイルを取得
      GmailApp.sendEmail(address, subject, body, {
        name: 'T.Matsuura',
        attachments: [file] // 添付ファイル
      });
    } else {
      Logger.log(`ファイルが見つかりません: ${fileName}`);
    }
  }
}

```

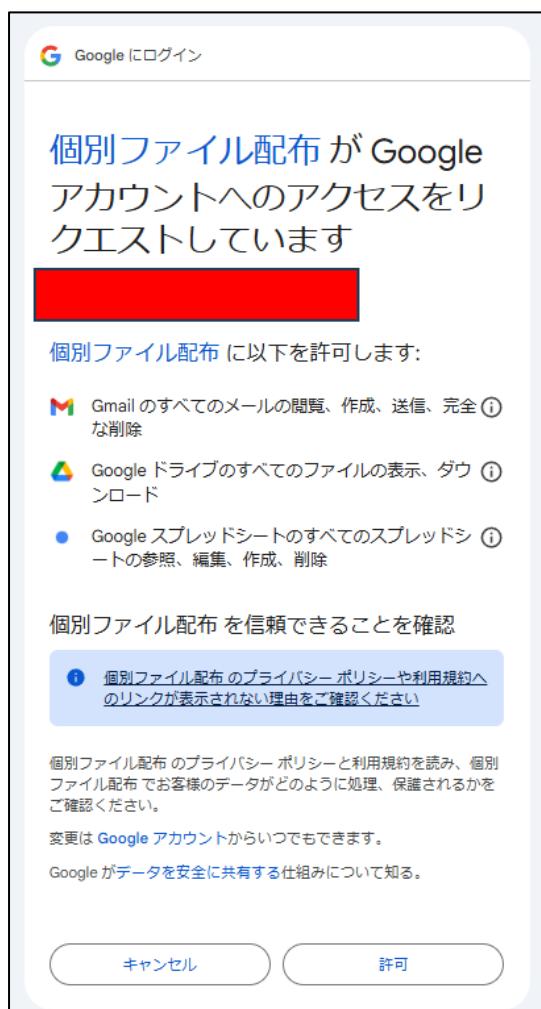
実行をして、権限を確認ボタンを押す



自分のアカウントをクリックする



許可ボタンを押す



エラーが出るので、Googleで検索をする

### 実行ログ

15:30:24	お知らせ	実行開始
15:30:27	エラー	Exception: Access denied: DriveApp. sendMail @ <a href="#">コード.gs:20</a>

Google エラーメッセージ Exception: Access denied: DriveApp. ×

すべて 動画 ショッピング 画像 ニュース 書籍 ウェブ :もっと見る

はてなブログ  
<https://takeda-san.hatenablog.com> entry, 2021/07/17 :

**GASでGoogle Driveのファイルを削除する(できないものもある)**  
2021/07/17 — 権限不足で例外が出てしました。実行結果。Exception: Access denied:  
DriveApp.trashFile @ main.gs:3. Webの画面からなら削除できるんです ...

teratail[テラテイル]  
<https://teratail.com> questions :

**GASでスプレッドシートのリンク共有をしたい。エラー内容**  
2020/12/10 — 「Access denied GAS」で検索して色々な記事を読みましたが、残念ながら解決できませんでした。accessに代入した構文もpermissionに代入した構文も間違つ ...

Google for Developers  
<https://developers.google.com> guides, troubleshooting :

**トラブルシューティング | Apps Script**  
ライブラリに同時にアクセスしているユーザーが多いことが原因である可能性があります。このエラーを回避するには、次のいずれかの方法を試してください。ライブラリ ...

検索してみると、アクセスしているユーザーが多いとエラーになるようなので、ファイルを開いているタグを閉じてもう一度実行すると成功する。

### 実行ログ

15:37:41	お知らせ	実行開始
15:37:53	お知らせ	実行完了

Gmail を見て確認をする

The screenshot shows a Gmail inbox with five messages from the sender '自分' (Self) with subject '1学期期末 試験結果 - HRNO: 1105 テストの結果をメールに添付しました。ご確認ください。'. Each message includes a PDF attachment labeled '1105.pdf', '1104.pdf', '1103.pdf', '1102.pdf', and '1101.pdf' respectively.

件名	添付ファイル
1学期期末 試験結果 - HRNO: 1105 テストの結果をメールに添付しました。ご確認ください。	PDF 1105.pdf
1学期期末 試験結果 - HRNO: 1104 テストの結果をメールに添付しました。ご確認ください。	PDF 1104.pdf
1学期期末 試験結果 - HRNO: 1103 テストの結果をメールに添付しました。ご確認ください。	PDF 1103.pdf
1学期期末 試験結果 - HRNO: 1102 テストの結果をメールに添付しました。ご確認ください。	PDF 1102.pdf
1学期期末 試験結果 - HRNO: 1101 テストの結果をメールに添付しました。ご確認ください。	PDF 1101.pdf

このままでも課題は達成できているのだが、改良する部分などを教えてくれるので、次のようにメッセージを送る。

ありがとう、うまくいきました。

改良して次のようなコードにしました。

```
function sendMail() {
  const sheet = SpreadsheetApp.getActiveSheet();
  const lastRow = sheet.getLastRow();
  const subject = '1学期期末 試験結果';

  // 各行をループ処理
  for (let i = 2; i <= lastRow; i++) {
    const hrno = sheet.getRange(i, 1).getValue(); // HRNO
    const address = sheet.getRange(i, 2).getValue(); // メールアドレス
    const fileName = sheet.getRange(i, 3).getValue(); // ファイル名

    // メール本文を構築
    const body = HRNO: ${hrno}\nテストの結果をメールに添付しました。\\nご確認ください。;

    // ファイルを取得
    const files = DriveApp.getFilesByName(fileName);

    if (files.hasNext()) {
      const file = files.next(); // 最初のファイルを取得
      GmailApp.sendEmail(address, subject, body, {
        name: 'T.Matsuura',
        attachments: [file] // 添付ファイル
      });
    } else {
      Logger.log(ファイルが見つかりません: ${fileName});
    }
  }
}
```





改良したコードはとても良い形になっています！ファイルの存在チェックやメール本文の構築が正しく行われており、コードもシンプルでわかりやすいです。特に、`HRNO` の取得をシートの1列目に変更し、ファイル名やメールアドレスを取得する部分も整理されていますね。

もしさらに改良できる点があるとすれば、以下の点を考慮することができます。

## 1. エラーハンドリングの強化

`GmailApp.sendEmail()` や `DriveApp.getFilesByName()` などの外部サービスを使う場合、予期しないエラーが発生することがあります。エラーハンドリングを追加することで、問題が発生してもコードが途中で止まらないようにすることができます。

例えば、`try-catch` 構文を使ってエラーをキャッチする方法があります。

## 2. 処理時間の長さを考慮

もしシートの行数が多い場合、Google Apps Script の制限に引っかかることがあります（例えば、1回の実行時間が6分を超えるとタイムアウトします）。その場合、スプレッドシートのデータをパッチ処理する方法も考慮した方が良いかもしれません。

以下は、エラーハンドリングを追加した改良版のコードです：

```
javascript □ コードをコピーする
function sendMail() {
  const sheet = SpreadsheetApp.getActiveSheet();
  const lastRow = sheet.getLastRow();
  const subject = '1学期期末 試験結果';

  // 各行をループ処理
  for (let i = 2; i <= lastRow; i++) {
    const hrno = sheet.getRange(i, 1).getValue(); // HRNO
    const address = sheet.getRange(i, 2).getValue(); // メールアドレス
    const fileName = sheet.getRange(i, 3).getValue(); // ファイル名

    // メール本文を構築
    const body = `HRNO: ${hrno}\nテストの結果をメールに添付しました。\\nご確認ください。`;

    // ファイルを取得
    const files = DriveApp.getFilesByName(fileName);

    if (files.hasNext()) {
      const file = files.next(); // 最初のファイルを取得

      try {
        GmailApp.sendEmail(address, subject, body, {
          name: 'T.Matsuura',
          attachments: [file] // 添付ファイル
        });
        Logger.log(`メールを送信しました: ${address}`);
      } catch (error) {
        Logger.log(`エラーが発生しました（メール送信失敗）: ${address} - ${error.message}`);
      }
    } else {
      Logger.log(`ファイルが見つかりません: ${fileName} - 行: ${i}`);
    }
  }
}
```

## 改良点：

1. エラーハンドリング： `GmailApp.sendEmail()` を `try-catch` で囲んで、送信エラーが発生した場合でも処理が続行できるようにしました。エラーメッセージも `Logger.log()` で記録します。
2. ログ出力：メール送信が成功した場合にもログを出して、送信状況がわかるようにしました。

これにより、実行中の問題に対するフィードバックが得られやすくなり、エラーが発生した場合にもどの行で問題が発生したかを特定しやすくなります。

その他、コードがうまく動作している場合はそのまま問題ありません！



と、改善案を提示してくれるので、今までのコードはどこかに保存しておき、コードを鵜が湧きして実行する。

(コード)

```
function sendMail() {
  const sheet = SpreadsheetApp.getActiveSheet();
  const lastRow = sheet.getLastRow();
  const subject = '1 学期期末 試験結果';

  // 各行をループ処理
  for (let i = 2; i <= lastRow; i++) {
    const hrno = sheet.getRange(i, 1).getValue(); // HRNO
    const address = sheet.getRange(i, 2).getValue(); // メールアドレス
    const fileName = sheet.getRange(i, 3).getValue(); // ファイル名

    // メール本文を構築
    const body = `HRNO: ${hrno}\nテストの結果をメールに添付しました。ご確認ください。`;

    // ファイルを取得
    const files = DriveApp.getFilesByName(fileName);

    if (files.hasNext()) {
      const file = files.next(); // 最初のファイルを取得

      try {
        GmailApp.sendEmail(address, subject, body, {
          name: 'T.Matsuura',
          attachments: [file] // 添付ファイル
        });
        Logger.log(`メールを送信しました: ${address}`);
      } catch (error) {
        Logger.log(`エラーが発生しました (メール送信失敗) : ${address} - ${error.message}`);
      }
    } else {
      Logger.log(`ファイルが見つかりません: ${fileName} - 行: ${i}`);
    }
  }
}
```

(操作画面)

```
1  function sendMail() {
2      const sheet = SpreadsheetApp.getActiveSheet();
3      const lastRow = sheet.getLastRow();
4      const subject = '1学期期末 試験結果';
5
6      // 各行をループ処理
7      for (let i = 2; i <= lastRow; i++) {
8          const hrno = sheet.getRange(i, 1).getValue(); // HRNO
9          const address = sheet.getRange(i, 2).getValue(); // メールアドレス
10         const fileName = sheet.getRange(i, 3).getValue(); // ファイル名
11
12         // メール本文を構築
13         const body = `HRNO: ${hrno}\nテストの結果をメールに添付しました。\\nご確認ください。`;
14
15         // ファイルを取得
16         const files = DriveApp.getFilesByName(fileName);
17
18         if (files.hasNext()) {
19             const file = files.next(); // 最初のファイルを取得
20
21             try {
22                 GmailApp.sendEmail(address, subject, body, {
23                     name: 'T.Matsuura',
24                     attachments: [file] // 添付ファイル
25                 });
26                 Logger.log(`メールを送信しました: ${address}`);
27             } catch (error) {
28                 Logger.log(`エラーが発生しました（メール送信失敗）: ${address} - ${error.message}`);
29             }
30         } else {
31             Logger.log(`ファイルが見つかりません: ${fileName} - 行: ${i}`);
32         }
33     }
34 }
```

## 実行ログ

16:04:05	お知らせ	実行開始
16:04:07	情報	メールを送信しました: .jp
16:04:09	情報	メールを送信しました: .jp
16:04:11	情報	メールを送信しました: .jp
16:04:13	情報	メールを送信しました: .jp
16:04:15	情報	メールを送信しました: .jp
16:04:16	お知らせ	実行完了

<input checked="" type="checkbox"/>	☆ 自分	1学期期末 試験結果 - HRNO: 1105 テストの結果をメールに添付しました。ご確認...	16:04
<input type="checkbox"/>	☆ 自分	1学期期末 試験結果 - HRNO: 1104 テストの結果をメールに添付しました。ご確認...	16:04
<input type="checkbox"/>	☆ 自分	1学期期末 試験結果 - HRNO: 1103 テストの結果をメールに添付しました。ご確認...	16:04
<input type="checkbox"/>	☆ 自分	1学期期末 試験結果 - HRNO: 1102 テストの結果をメールに添付しました。ご確認...	16:04
<input type="checkbox"/>	☆ 自分	1学期期末 試験結果 - HRNO: 1101 テストの結果をメールに添付しました。ご確認...	16:04

これで、どこで送信エラーが出たかわかるようになった。煩わしければ、元のコードで実行してもよい。

別の日に、ネット検索と ChatGPT を使ってやってみる。

次のプログラムのどこが間違っているか教えてほしい



もちろんです！プログラムの内容を教えていただければ、間違っている部分を確認して指摘します。コードをこちらに貼り付けてください。

間違ったコードを貼り付ける

```
function sendMail(){

const sheet = SpreadsheetApp.getActiveSheet();
const lastRow = sheet.getLastRow();

const subject = '1学期期末 試験結果';

for(let i = 2; i <= lastRow; i++){

const address = sheet.getRange(i, 2).getValue();
const fileName = sheet.getRange(i, 3).getValue();
const body = "HRNO" + syozoku + "%n"

テストの結果をメールに添付しました。
ご確認ください。
;

var tempu = DriveApp.getFilesByName(fileName).next();

GmailApp.sendEmail(address, subject, body, {
  name: 'T.Matsuura',
  attachments: [tempu]});
}
}
```

すると、ChatGPT がどこがおかしいか答えてくれるので、修正する。

このコードにはいくつかの潜在的な問題点があります。以下に修正点とともに指摘します。

### 1. syozoku 変数が定義されていない

javascript

コードをコピーする

```
const body = "HRNO" + syozoku + "\n"
```

`body` 変数の作成時に使用されている `syozoku` が定義されていません。この変数が何を表すかに応じて、適切に定義する必要があります。例えば、もしシートの列に所属情報（`syozoku`）があるのであれば、その列の値を取得する必要があります。

修正案:

javascript

コードをコピーする

```
const syozoku = sheet.getRange(1, 4).getValue(); // 例えは4列目に所属情報がある場合
const body = "HRNO " + syozoku + "\n" + "テストの結果をメールに添付しました。\\nご確認ください。
```

### 2. テンプレートリテラルの不適切な使用

javascript

コードをコピーする

```
const body = "HRNO" + syozoku + "\n"
```

```
-
```

テストの結果をメールに添付しました。  
ご確認ください。

```
^;
```

`body` の部分でテンプレートリテラル（バッククオト）を使っているように見えますが、正しい位置で閉じられていないため、エラーが発生します。テンプレートリテラルの構文が混亂しているので、これを修正します。

修正案:

javascript

コードをコピーする

```
const body = `HRNO ${syozoku}\nテストの結果をメールに添付しました。\\nご確認ください。`;
```

### 3. DriveApp.getFilesByName(fileName).next() のエラーチェック

DriveApp.getFilesByName(fileName).next() は、ファイルが見つからない場合にエラーを引き起こす可能性があります。ファイルが存在しない場合に備えてエラーチェックを追加すると良いでしょう。

修正案:

```
javascript コードをコピーする
var files = DriveApp.getFilesByName(fileName);
if (files.hasNext()) {
  var tempu = files.next();
  GmailApp.sendEmail(address, subject, body, {
    name: 'T.Matsuura',
    attachments: [tempu]
  });
} else {
  Logger.log('ファイルが見つかりません: ' + fileName);
}
```

### 4. メール送信処理の最適化

GmailApp.sendEmail の送信処理は、メールアドレスや添付ファイルが存在することを前提としているため、これらのデータが正しく取得されていることを確認する必要があります。現在のコードで不安な部分があれば、チェックを追加することをお勧めします。

#### 修正後のコード

```
javascript コードをコピーする
function sendMail() {
  const sheet = SpreadsheetApp.getActiveSheet();
  const lastRow = sheet.getLastRow();

  const subject = '1学期期末 試験結果';

  for (let i = 2; i <= lastRow; i++) {
    const address = sheet.getRange(i, 2).getValue(); // メールアドレス
    const fileName = sheet.getRange(i, 3).getValue(); // ファイル名
    const syozoku = sheet.getRange(i, 4).getValue(); // 所属情報 (仮に4列目としています)

    const body = `HRNO ${syozoku}\nテストの結果をメールに添付しました。\nご確認ください。`;

    var files = DriveApp.getFilesByName(fileName);
    if (files.hasNext()) {
      var tempu = files.next(); // ファイルを取得
      GmailApp.sendEmail(address, subject, body, {
        name: 'T.Matsuura',
        attachments: [tempu] // 添付ファイル
      });
    } else {
      Logger.log('ファイルが見つかりません: ' + fileName); // エラーログ
    }
  }
}
```

## まとめ

- `syozoku` の変数を適切に定義し、テンプレートリテラルの使い方を修正しました。
- `DriveApp.getFilesByName()` のファイル検索結果に対するエラーチェックを追加しました。
- その他、コードの可読性と安定性を向上させるために微調整を加えました。

これで、正常に動作するはずです。

`shozoku`が読み取れるようにしましたが、エラーになりました。

次のどこが間違っていますか？

```
function sendMail(){  
  
const sheet = SpreadsheetApp.getActiveSheet();  
const lastRow = sheet.getLastRow();  
  
const subject = '1 学期期末 試験結果';  
  
for(let i = 2; i <= lastRow; i++){  
  
    const shozoku = sheet.getRange(i, 1).getValue();  
    const address = sheet.getRange(i, 2).getValue();  
    const fileName = sheet.getRange(i, 3).getValue();  
    const body = "HRNO" + syozoku + "%n"  
  
    テストの結果をメールに添付しました。  
    ご確認ください。  
};  
  
var tempu = DriveApp.getFilesByName(fileName).next();  
  
GmailApp.sendEmail(address, subject, body, {  
    name: 'T.Matsuura',  
    attachments: [tempu]});  
} ↓  
}
```

エラーの原因は主に2つあります。以下にそれぞれの問題とその解決方法を示します。

### 1. `syozoku` という変数が定義されていない

コードの中で `syozoku` という変数が使われていますが、定義されていません。一方、`shozoku` という変数は正しく定義されています。おそらく、`shozoku` を使いたいところで `syozoku` と書いてしまったため、エラーが発生しています。

修正案: `syozoku` を `shozoku` に修正します。

### 2. テンプレートリテラルの誤った使用

次に、`body` 変数の設定でテンプレートリテラル（バッククオート）を使おうとしている部分が誤っています。正しくは、バッククオートで囲った部分を1行で書くべきですが、コードでは誤った位置で改行を行っているため、エラーが発生しています。

修正案: テンプレートリテラルの構文を修正します。

## 修正後のコード

```
javascript □ コードをコピーする
function sendMail(){
  const sheet = SpreadsheetApp.getActiveSheet();
  const lastRow = sheet.getLastRow();

  const subject = '1学期期末 試験結果';

  for(let i = 2; i <= lastRow; i++){
    const shozoku = sheet.getRange(i, 1).getValue(); // shozoku (所属) の取得
    const address = sheet.getRange(i, 2).getValue(); // メールアドレスの取得
    const fileName = sheet.getRange(i, 3).getValue(); // ファイル名の取得

    // body をテンプレートリテラルで作成
    const body = `HRNO ${shozoku}\nテストの結果をメールに添付しました。\\nご確認ください。`;

    // ファイルを取得
    var tempu = DriveApp.getFilesByName(fileName).next();

    // メール送信
    GmailApp.sendEmail(address, subject, body, {
      name: 'T.Matsuura',
      attachments: [tempu]
    });
  }
}
```

## 修正内容:

- 変数名の修正: `syozoku` → `shozoku` に変更。
- テンプレートリテラルの修正: `body` の作成部分で正しくパッククオート (`) を使用。

この修正でエラーが解消され、正常に動作するはずです。



GASを使って個別のファイルを配信するコードを書いて

- ⑤ Google Apps Script (GAS) を使って個別のファイルを配信するコードを作成するために、以下の要素を考慮したコードを提供します。

### 前提

- Google スプレッドシートに、各受信者のメールアドレス、ファイル名、その他必要な情報が記載されていると仮定します。
- 添付するファイルは、Google Drive に保存されているファイルで、ファイル名を元に取得します。
- 各受信者に個別にメールを送信し、ファイルを添付します。

### 必要なカラム

- A列: 受信者のメールアドレス
- B列: 添付するファイル名

<コード>

```
function sendIndividualFiles() {
    // スプレッドシートとシートの取得
    const sheet = SpreadsheetApp.getActiveSpreadsheet().getActiveSheet();
    const lastRow = sheet.getLastRow(); // 最終行の取得

    // メールの件名
    const subject = '個別のファイルを送信します';

    // 各行ごとにループ
    for (let i = 2; i <= lastRow; i++) {
        // 各セルの値を取得
        const email = sheet.getRange(i, 1).getValue(); // A列：メールアドレス
        const fileName = sheet.getRange(i, 2).getValue(); // B列：ファイル名

        // メール本文
        const body = '指定されたファイルを添付して送信します。ご確認ください。';

        // ファイルを Google Drive から取得
        const files = DriveApp.getFilesByName(fileName);
        if (files.hasNext()) {
            const file = files.next(); // 最初のファイルを取得

            // メール送信（添付ファイル付き）
            GmailApp.sendEmail(email, subject, body, {
                name: 'T.Matsuura', // 送信者名
                attachments: [file] // 添付ファイル
            });

            Logger.log('ファイルを送信しました: ' + email + ', ' + fileName);
        } else {
            Logger.log('ファイルが見つかりません: ' + fileName);
        }
    }
}
```

## 説明

### 1. スプレッドシートのデータ取得:

- `sheet.getRange(i, 1).getValue()` でA列から受信者のメールアドレスを取得。
- `sheet.getRange(i, 2).getValue()` でB列からファイル名を取得。

### 2. ファイルの検索:

- `DriveApp.getFilesByName(fileName)` で Google Drive 上のファイルを検索。ファイル名が一致する最初のファイルを取得します。

### 3. メール送信:

- `GmailApp.sendEmail()` を使用して、メールアドレスにファイルを添付して送信。
- メールの件名と本文は指定通り設定しています。

### 4. エラーハンドリング:

- ファイルが見つからない場合は、`Logger.log()` でログを記録します。

## 使用方法

1. Google スプレッドシートにメールアドレスとファイル名を入力してください。
2. 上記のスクリプトを Google Apps Script のエディタに貼り付けて実行します。

これで、スプレッドシートに記載されたメールアドレスに対して、個別にファイルを添付してメールを送信できます。



スプレッドシート(※ここでは、ファイル名を 1100 から 1104 にしている)

A	B	C
HRNO	メアド	ファイル名
1100	matsuura-puj@gl.pen-kanagawa.ed.jp	1100.pdf
1101	matsuura-uxm@gl.pen-kanagawa.ed.jp	1101.pdf
1102	matsuura-puj@gl.pen-kanagawa.ed.jp	1102.pdf
1103	matsuura-uxm@gl.pen-kanagawa.ed.jp	1103.pdf
1104	matsuura-puj@gl.pen-kanagawa.ed.jp	1104.pdf

編集したデータ

```
function sendIndividualFiles() {
  // スプレッドシートとシートの取得
  const sheet = SpreadsheetApp.getActiveSpreadsheet().getActiveSheet();
  const lastRow = sheet.getLastRow(); // 最終行の取得

  // メールの件名
  const subject = '1学期中間 試験結果';

  // 各行ごとにループ
  for (let i = 2; i <= lastRow; i++) {
    // 各セルの値を取得
    const email = sheet.getRange(i, 2).getValue(); // B列:メールアドレス
    const fileName = sheet.getRange(i, 3).getValue(); // C列:ファイル名

    // メール本文
    const body = '試験結果を添付しました。ご確認ください。';

    // ファイルを Google Drive から取得
    const files = DriveApp.getFilesByName(fileName);
    if (files.hasNext()) {
      const file = files.next(); // 最初のファイルを取得

      // メール送信(添付ファイル付き)
      GmailApp.sendEmail(email, subject, body, {
        name: 'T.Matsuura', // 送信者名
        attachments: [file] // 添付ファイル
      })
    }
  }
}
```

```

    });

    Logger.log('ファイルを送信しました: ' + email + ', ' + fileName);
} else {
    Logger.log('ファイルが見つかりません: ' + fileName);
}
}

}
}

```

Apps Script    GAS個別ファイル配布

デプロイ ▾ | ② | ■■■

ファイル	AZ	+	◀	⟳	実行	DEBUG	sendIndividualFiles	実行ログ
<pre> コード.gs function sendIndividualFiles() {     // スプレッドシートとシートの取得     const sheet = SpreadsheetApp.getActiveSpreadsheet().getActiveSheet();     const lastRow = sheet.getLastRow(); // 最終行の取得      // メールの件名     const subject = '1学期中間 試験結果';      // 各行ごとにループ </pre>								

実行ログ

14:10:02	お知らせ	実行開始
14:09:32	情報	ファイルを送信しました: matsuura-puj@gl.pen-kanagawa.ed.jp, 1100.pdf
14:09:35	情報	ファイルを送信しました: matsuura-uxm@gl.pen-kanagawa.ed.jp, 1101.pdf
14:09:37	情報	ファイルを送信しました: matsuura-puj@gl.pen-kanagawa.ed.jp, 1102.pdf
14:09:41	情報	ファイルを送信しました: matsuura-uxm@gl.pen-kanagawa.ed.jp, 1103.pdf
14:09:44	情報	ファイルを送信しました: matsuura-puj@gl.pen-kanagawa.ed.jp, 1104.pdf
14:10:17	お知らせ	実行完了

<input type="checkbox"/> ☆ 自分	1学期中間 試験結果 - 試験結果を添付しました。ご確認ください。
<input type="checkbox"/> ☆ 自分	1学期中間 試験結果 - 試験結果を添付しました。ご確認ください。
<input type="checkbox"/> ☆ 自分	1学期中間 試験結果 - 試験結果を添付しました。ご確認ください。

<input type="checkbox"/> ☆ T.Matsuura	1学期中間 試験結果 - 試験結果を添付しました...	14:09
	<a href="#">PDF 1103.pdf</a>	
<input type="checkbox"/> ☆ T.Matsuura	1学期中間 試験結果 - 試験結果を添付しました...	14:09
	<a href="#">PDF 1101.pdf</a>	

```
ありがとう、うまいきました。  
改良して次のようなコードにしました。  
function sendIndividualFiles() {  
  // スプレッドシートとシートの取得  
  const sheet =  
    SpreadsheetApp.getActiveSpreadsheet().getActiveSheet();  
  const lastRow = sheet.getLastRow(); // 最終行の取得  
  
  // メールの件名  
  const subject = '1 学期中間 試験結果';  
  
  // 各行ごとにループ  
  for (let i = 2; i <= lastRow; i++) {  
    // 各セルの値を取得  
    const email = sheet.getRange(i, 2).getValue(); // B列 : メールアドレス  
    const fileName = sheet.getRange(i, 3).getValue(); // C列 : ファイル名  
  
    // メール本文  
    const body = '試験結果を添付しました。ご確認ください。';
```

```
  // ファイルをGoogle Driveから取得  
  const files = DriveApp.getFilesByName(fileName);  
  if (files.hasNext()) {  
    const file = files.next(); // 最初のファイルを取得  
  
    // メール送信（添付ファイル付き）  
    GmailApp.sendEmail(email, subject, body, {  
      name: 'T.Matsuura', // 送信者名  
      attachments: [file] // 添付ファイル  
    });  
  
    Logger.log('ファイルを送信しました: ' + email + ', ' + fileName);  
  } else {  
    Logger.log('ファイルが見つかりません: ' + fileName);  
  }  
}
```

素晴らしいですね！コードの改良と整理がとても良い感じです。コードは非常にシンプルで、目的に対して効果的に動作しているように見えます。

もしさらに改善したい点があるとしたら、以下の提案を考慮してみてください。

## 1. メール送信時のエラーハンドリングの追加

- もしメール送信に失敗した場合（例えば、送信先のメールアドレスが無効だった場合など）、エラーハンドリングを追加して、どの受信者に送信できなかったかを確認できるようにするとより良いでしょう。

javascript

□ コードをコピーする

```
try {
  GmailApp.sendEmail(email, subject, body, {
    name: 'T.Matsuura', // 送信者名
    attachments: [file] // 添付ファイル
  });
  Logger.log('ファイルを送信しました: ' + email + ', ' + fileName);
} catch (error) {
  Logger.log('送信エラー: ' + email + ', ' + error.message);
}
```

## 2. メール送信後の確認

- メールが送信されたかどうかをさらに明確にログとして残すことも有用です。メール送信後、`GmailApp.getMessagesForThreads()` や `GmailApp.getUserLabels()` を使って、送信済みメールを確認する方法もありますが、単純にログに送信完了メッセージを記録しておくことでも十分です。

## 3. ファイル名のトリミング（オプション）

- ファイル名に余計なスペースや改行が含まれている場合、それが原因で `DriveApp.getFilesByName(fileName)` が正しく動作しないことがあります。`fileName` を事前にトリミングすることで、この問題を避けられる場合があります。

javascript

□ コードをコピーする

```
const fileName = sheet.getRange(i, 3).getValue().trim(); // ファイル名の前後スペースを削除
```

#### 4. ログ出力をより詳細に

- どの行でエラーが発生したかをログに残すと、デバッグが楽になります。例えば、行番号や受信者の名前を記録することができます。

javascript

コードをコピーする

```
Logger.log(`行 ${i} の処理開始: メール送信先: ${email}, 添付ファイル名: ${fileName}`);
```

#### 5. スプレッドシートのデータ検証

- メールアドレスやファイル名が空でないか、または正しい形式であることを確認するバリデーションを加えると、より堅牢なコードになります。

javascript

コードをコピーする

```
if (!email || !fileName) {
  Logger.log(`行 ${i} のデータが不完全です。メールアドレス: ${email}, ファイル名: ${fileName}`);
  continue; // 次の行に進む
}
```



#### 改良版コード（上記提案を反映）

javascript

コードをコピーする

```
function sendIndividualFiles() {
  // スpreadsheetとシートの取得
  const sheet = SpreadsheetApp.getActiveSpreadsheet().getActiveSheet();
  const lastRow = sheet.getLastRow(); // 最終行の取得

  // メールの件名
  const subject = '1学期中間 試験結果';

  // 各行ごとにループ
  for (let i = 2; i <= lastRow; i++) {
    // 各セルの値を取得
    const email = sheet.getRange(i, 2).getValue().trim(); // B列: メールアドレス
    const fileName = sheet.getRange(i, 3).getValue().trim(); // C列: ファイル名

    // メール本文
    const body = '試験結果を添付しました。ご確認ください。';
  }
}
```

```
// データの検証
if (!email || !fileName) {
  Logger.log(`行 ${i} のデータが不完全です。メールアドレス: ${email}, ファイル名: ${fileName}`);
  continue; // 次の行に進む
}

// ファイルをGoogle Driveから取得
const files = DriveApp.getFilesByName(fileName);
if (files.hasNext()) {
  const file = files.next(); // 最初のファイルを取得
```

```
// メール送信 (添付ファイル付き)
try {
  GmailApp.sendEmail(email, subject, body, {
    name: 'T.Matsuura', // 送信者名
    attachments: [file] // 添付ファイル
  });
  Logger.log(`行 ${i}: ファイルを送信しました: ${email}, ${fileName}`);
} catch (error) {
  Logger.log(`行 ${i}: 送信エラー: ${email}, ${error.message}`);
}

} else {
  Logger.log(`行 ${i}: ファイルが見つかりません: ${fileName}`);
}
}
```

## まとめ

- エラーハンドリングとログ出力を強化することで、どこで問題が発生しているのかをより迅速に把握できるようになります。
- データの検証を加えることで、誤ったデータによるエラーを未然に防げます。

これでさらに強化されたコードになるかと思います。引き続き、うまくいくことを願っています！



### <改良コード>

```
function sendIndividualFiles() {
    // スプレッドシートとシートの取得
    const sheet = SpreadsheetApp.getActiveSpreadsheet().getActiveSheet();
    const lastRow = sheet.getLastRow(); // 最終行の取得

    // メールの件名
    const subject = '1 学期中間 試験結果';

    // 各行ごとにループ
    for (let i = 2; i <= lastRow; i++) {
        // 各セルの値を取得
        const email = sheet.getRange(i, 2).getValue().trim(); // B 列：メールアドレス
        const fileName = sheet.getRange(i, 3).getValue().trim(); // C 列：ファイル名

        // メール本文
        const body = '試験結果を添付しました。ご確認ください。';

        // データの検証
        if (!email || !fileName) {
            Logger.log(`行 ${i} のデータが不完全です。メールアドレス:${email}, ファイル名:${fileName}`);
            continue; // 次の行に進む
        }

        // ファイルを Google Drive から取得
        const files = DriveApp.getFilesByName(fileName);
        if (files.hasNext()) {
            const file = files.next(); // 最初のファイルを取得

            // メール送信（添付ファイル付き）
            try {
                GmailApp.sendEmail(email, subject, body, {
                    name: 'T.Matsuura', // 送信者名
                    attachments: [file] // 添付ファイル
                });
                Logger.log(`行 ${i}: ファイルを送信しました: ${email}, ${fileName}`);
            } catch (error) {
                Logger.log(`行 ${i}: 送信エラー: ${email}, ${error.message}`);
            }
        } else {
            Logger.log(`行 ${i}: ファイルが見つかりません: ${fileName}`);
        }
    }
}
```

これでも、同じように送信ができる。