ペアプログラミングチャレンジ！

Pythonで「数当てゲーム」を開発しよう！

ミッション

今日のミッションは、ペアの友達と協力して、Pythonで **数当てゲーム** を開発すること！

これまでに学んだ **変数、データ型、演算、** `input()` **、** `print()` **、そしてf-string** などの知識をフル活用しよう。

このチャレンジで目指すこと：

- 論理的な思考力と問題解決能力を養う。
- ペアプログラミングを通じて、協力して開発するスキルを体験する。
- プログラムを作る楽しさを実感する！

ゲームの基本ルール

1. 出題者（プログラム）が、1から30までの範囲でランダムな「秘密の数字」を心の中で（変数に）持ちます。
2. 回答者（ユーザー）は、その「秘密の数字」を予想してキーボードから入力します。
3. プログラムは、回答者の予想が「秘密の数字」と比べて...
    - 大きすぎるか
    - 小さすぎるか
    - それともピッタリ正解か
      をヒントとして教えてくれます。
4. 回答者は、ヒントを元に、正解するまで予想を繰り返します。

ステップ１：ゲームの土台を作ろう

1. プロジェクト開始！

まず、VSCodeで新しいPythonファイルを作成しよう。

ファイル名は `guess_the_number.py` のような、分かりやすい名前にすると良いね！

2. 「秘密の数字」を準備

プログラムの冒頭で、「秘密の数字」を決めよう。

`secret_number` という変数に、1から30の間のお好きな整数を代入してね。

（ペアで相談して、最初は固定の数字でテストすると開発しやすいよ！）

```
# filepath: guess_the_number.py
# 「秘密の数字」をここで設定（1〜30の範囲で！）
secret_number = 15  # 例：最初は15にしてみよう。後で変えてテストしてね！
challenge_count = 0 # ユーザーが何回挑戦したかを数える変数


```

---

3. ユーザーからの最初の「予想」を受け取る

次に、ユーザーに数字を予想してもらう部分を作ろう。

*   `input()` 関数を使って、「1から30の数字を予想して入力してください：」のような案内メッセージを表示する。
*   ユーザーが入力した内容を、`user_guess_str` のような変数に保存する。
*   `input()` で受け取ったものは**文字列**なので、数値として比較するためには `int()` 関数で**整数**に変換する必要があるね！変換後の数値は `user_guess_num` のような変数に入れよう。
*   予想したら、`challenge_count` を1増やしておこう。

コードのヒント：

ステップ２：ヒントを出そう！ - 判定ロジックの設計

ここがゲームの心臓部！ユーザーの予想 (`user_guess_num`) と「秘密の数字」 (`secret_number`) を比べて、適切なヒントを出すロジックを考えよう。

考えるべき3つのケース：

1. 予想が「秘密の数字」と **等しい** 場合（正解！）
2. 予想が「秘密の数字」より **小さい** 場合
3. 予想が「秘密の数字」より **大きい** 場合

使う道具：

ヒントの出し方 - アイデア１：比較結果を観察する

まず、比較演算子がどんな結果を返すか見てみよう。

`True` (真) か `False` (偽) が返ってくるはずだ。

```
# filepath: guess_the_number.py
# ... (ユーザーの予想を受け取る処理の後) ...

# デバッグ用：入力された値と秘密の数字を確認
print(f"（デバッグ情報：あなたの予想 {user_guess_num}, 秘密の数字 {secret_number}）")
```

これらの `True` / `False` を見て、どのメッセージを表示すべきか判断できるかな？

例えば、`is_correct` が `True` なら、「おめでとう！正解！」と表示したいよね。

ヒントの出し方 - アイデア２：条件に応じたメッセージ（`if` 文なしで挑戦！）

`if` 文（条件分岐）は次の授業で詳しく学ぶけど、今回はそれを使わずに、どうやったら適切なヒントだけを表示できるか、ペアで話し合って工夫してみよう！

例えば、こんな風に考えられるかな？

- もし `is_correct` が `True` だったら、「おめでとう！正解です！{challenge_count} 回で当たりました！」と表示する。
- もし `is_too_small` が `True` だったら、「残念！もっと大きい数字だよ。」と表示する。
- もし `is_too_large` が `True` だったら、「残念！もっと小さい数字だよ。」と表示

ステップ３：ゲームを完成させよう（発展課題）

ステップ２までで、1回の予想に対するヒントは出せるようになったはず。
さらにゲームとして面白くするために、以下の機能に挑戦してみよう！

- **繰り返し挑戦**: 正解するまで、何度もユーザーに予想を入力させ、ヒントを出し続けるようにするにはどうしたらいいかな？
  （ヒント： `while` ループというものを使うと実現できるけど、これは次々回の内容。まずは、正解するまでプログラムの実行と修正を繰り返す形で体験してみよう。）
- **挑戦回数の制限**: 例えば「5回以内に当てないとゲームオーバー！」のようなルールを追加してみよう。 `challenge_count` が役立つはず！

これらの発展課題は、ペアで自由にアイデアを出し合って、どこまで実装できるかチャ

このチャレンジで使う主なPythonの道具たち

- **変数**: `secret_number`, `user_guess_str`, `user_guess_num`, `challenge_count` など
- **データ型**: `int` (整数), `str` (文字列)
- **型変換**: `int()` (文字列を整数に変換)
- **算術演算子**: `+` (主に回数カウントで)
- **比較演算子**: `==` (等しい), `<` (より小さい), `>` (より大きい)
- **関数**:
  - `input()` (ユーザーからのキーボード入力を受け取る)

ペアプログラミングの進め方 - 成功の秘訣！

- **役割分担と交代**:
  - **ドライバー**: 実際にキーボードを操作してコードを書く人。
  - **ナビゲーター**: ドライバーの書くコードを見ながら、アイデアを出したり、間違いを指摘したり、次の手順を考えたりする人。
  - 15分〜20分くらいで役割を交代しながら進めよう！
- **声に出して考える**:「次はこうしてみようか？」「このエラーは何だろう？」と、常に二人でコミュニケーションを取りながら進めるのが大事。
- **エラーを恐れない**: プログラミングにエラーはつきもの！エラーメッセージをよく読んで、何が原因か二人で推理しよう。解決できた時の達成感は格別だよ！

さあ、チャレンジ開始！

準備はいいかな？
ペアの友達と知恵を出し合い、最高の「数当てゲーム」を開発しよう！

困ったときは？

- まずは二人でじっくり考えてみる。
- エラーメッセージを翻訳サイトで調べてみる。
- それでも分からなければ、遠慮なく先生やTAにヒントを求めてね！

Good luck and have fun! 💻✨