ナンプレ問題集ヒント機能付き

著者：関 勝寿　公開日：2022 年 8 月 19 日　ソース：https://sekika.github.io/2022/08/19/sudoku/

オンラインのナンプレ問題集にヒントを表示する機能をつけた。H キーを押すことで、その場面でどのように考えれば良いかというヒントが表示される。解き方を理解するまではヒントを多めに表示して、次第にヒントの表示を少なくし、さらにより高いレベルの問題に挑戦することで、数独の解き方を理解できる。

この問題集は、数独（ナンプレ）を解析する解独 (Kaidoku) のウェブ版である。解独はコマンドラインからナンプレの問題を解析・作成するプログラムであり、オンライン問題集は解独が作成した問題を難易度別にまとめたものである。解独は「人間が数独を解くための手順」をなるべく再現できるようにしている。使われている解法に記されているように、「単独候補数字」「単独候補マス」のような基本的な解法が適用できるかどうかを試して、その適用ができなければ様々な高度な解法を順番に試して、探索は最後の手段としている。コンピュータにとっては探索だけで解く方が簡単に解けるが、人間にとってより解きやすい方法を探索しているということになる。

0.1. ブラウザで Python を実行するための技術メモ

解独は Python で書かれているため、ブラウザで直接実行できなかった。したがって、オンライン版には解析機能をつけていなかった。しかし、Pyodide を使えばブラウザで Python が動かせることがわかったので、この技術によってヒント表示機能をつけることができた。

局面の文字列をクリップボードにコピーする機能はすでにつけていたので、あとはそのクリップボードの局面からヒントを返すようなモジュールを作成して呼び出せばいいだけなので簡単だろうと思ったが、いくつか苦労したところがあるので Pyodide プログラミングの参考のためにここに記録する。

まずは、解独にモジュールを作成して、Kaidoku 1.0.0 として PyPI に公開した。モジュールの使い方はサンプルスクリプトのように簡単なもので、局面の文字列を引数としてインスタンスを生成すればヒントの文字列がインスタンスのプロパティとして設定される。

次に JavaScript から Pyodide を読み込む。micropip で kaidoku を読み込む時に、pip の依存パッケージを読み込むとうまくいかず、依存パッケージは不要なので Micropip API に書かれているように deps=False を指定して `await micropip.install("kaidoku", deps=False);` とするとエラーになる。JavaScript から呼び出しているのだから、Python と同じ引数の指定方法ではだめだ。しかし、どうやって deps のオプションを指定するのだろうか？と考えて、よくわからなかったので deps は 3 番目の引数だから 3 番目が false になっていればいいのかな、と思って `await micropip.install("kaidoku", false, false);` としたところうまくいった。

変数のやりとりについてはここに書かれているように pyodide.registerJsModule を使った。Python の変数を JavaScript に読み込むときには、ここに書かれているように pyodide.globals.get を使った。

以上の仕組みでなんとか動いたのだが、最初はヒントを表示する関数 hint() のところに async func hint() として pyodide を読み込んで実行するようにしていたため、ヒントボタンを押すたびに pyodide を読み込むことになる。読み込みには時間がかかるので、できれば読み込みは 1 回だけバックグラウンドで読み込むこととしたい。そこで、pyodide をグローバル変数として、JavaScript を読み込んだ時に最初に loadpyodide() として読み込むこととした。

```
async function loadpyodide() {
  pyodide = await loadPyodide();
  await pyodide.loadPackage("micropip");
  const micropip = pyodide.pyimport("micropip");
  await micropip.install("kaidoku", false, false);
}
```

すると、ヒントを表示する関数の中では、すでに pyodide が読み込まれているグローバル変数 pyodide を使って pyodide の実行ができる。これでうまくいくと思ったのだが、今度は JavaScript で

```
let js_namespace = { pos : current };
pyodide.registerJsModule("js_namespace", js_namespace);
```

として、Python で from js_namespace import pos として読み込ませた pos 変数が、2 回目以降の実行の時に更新されない、という現象が生じた。JavaScript 側では js_namespace は更新されていて Python 側では pos 変数が更新されないことから、pyodide をグローバル変数としたため、2 回目の pyodide の実行時に import で pos が更新されなかったためと推測された。そこで、Python で pos 変数の取得後に `del sys.modules["js_namespace"]` で js_namespace モジュールを削除することで、次に js_namespace をインポートする時に pos 変数が更新されることとなった。

このように、Python で開発したモジュールを JavaScript から Pyodide で読み込んで実行するという一連の流れを試すことができた。