

July 30, 2020

# CSV の読み込み 1

沼田 賢一

富山県立大学 情報基盤工学講座

1. はじめに
2. 提案手法
3. おわりに

July 30, 2020

## 本研究の背景

近年、企業などでは世間に溢れる様々な情報を収集し、ビッグデータと呼ばれる非常に巨大で複雑なデータの集合として扱うことが増えてきている。また、ビッグデータを扱うには様々な処理や解析によって情報を取捨選択し、自分たちに必要な形で保管する必要がある。しかし、ビッグデータの情報量は膨大で、人の手で全て解析するのは困難であるため、一般にプログラミング言語を用いて機械に処理させるのが一般的である。このため、ビッグデータを扱うためにはプログラミングの知識や技術が必要不可欠であり、プログラミングに触れたことがない人には扱いづらいものになっている。

## 目的

プログラミング初心者でも扱いやすいビジュアルプログラミング言語を使い、ブロック 1 つ 1 つに処理を対応させることでブロックを並べるだけで処理できるプログラムを作成することができるようにする。プログラミングができない人でもビッグデータを扱うことができるようにする。

## ビジュアルプログラミング言語

プログラムをテキストで記述するのではなく、視覚的なオブジェクトで記述するプログラミング言語のこと。視覚的でわかりやすいものが多いため、プログラムの組み立て方を学ぶのに有効であると注目されている。

## ビジュアル言語

### ブロックタイプ



テキスト言語の論理に近い

例 Scratch・MakeCode...

### フロータイプ



フローチャートの

例 MESHアプリ...

### 独自ルールタイプ



独自の考え方

例 Viscuit...

## ブロックタイプの VPL

機械学習（人工知能・AI）を使って課題を解決するクラウドサービスの MAGELLAN BLOCKS（BLOCKS）や教育用作られ様々なアプリケーションに応用して使われている Blockly などがある。  
応用して使われているサービスとして Scratch や MakeCode が存在する。

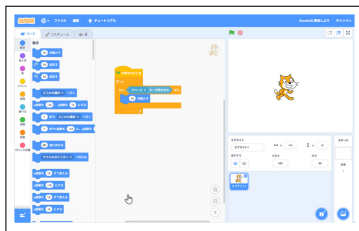


Figure: scratch

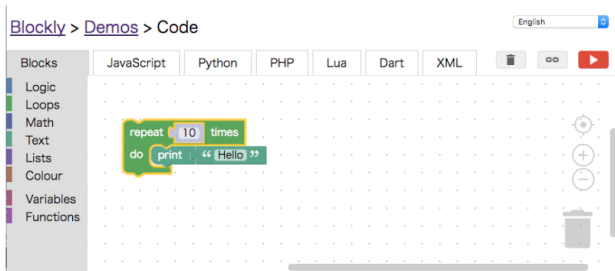


Figure: makecode

## Blockly

Google が提供しているビジュアルプログラミング言語のライブラリ。簡単な記述で自分だけのビジュアルプログラミング言語を作ることができる。

また、作成したブロックは JavaScript や python, PHP, Lua, Dart などのプログラミング言語にエクスポートすることができる。

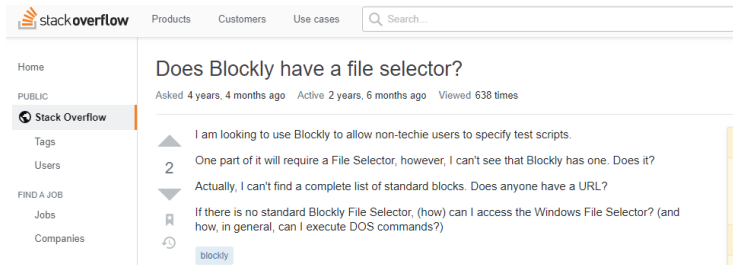


## Blockly の問題点

ビックデータを扱うとき、外部からデータ (CSV) を読み込む必要があるが blockly のデモコードには外部からのファイルの読み込みが実装されていない。

## CSV の読み込み方

デモコードでは、実装されていないが Blockly をもとに作られている Scratch では CSV の読み込みが実装されている。また、stackoverflow で読み込み方を質問していたので参考にして作る。



The screenshot shows a Stack Overflow page for the question "Does Blockly have a file selector?". The page header includes the Stack Overflow logo, navigation links (Products, Customers, Use cases), and a search bar. The left sidebar shows the "Stack Overflow" link selected. The question is marked as "PUBLIC" and was asked 4 years, 4 months ago, with 638 views. The first answer, by user "2", states that while a File Selector is needed, it's not in Blockly, and suggests using a URL or the Windows File Selector. A second answer asks if there's a standard Blockly File Selector and how to access the Windows File Selector or execute DOS commands. A "blockly" tag is visible at the bottom.

## カスタムブロックとは

Blockly では、もともとあるブロックの他にユーザが好きなブロックを作成することができる。

CSV ファイルを読み込むためには、カスタムブロックを自作する必要がある。

## カスタムブロックを作る前に..

js でローカルファイルを読み込む方法は、調べると色々出てくる。しかし、カスタムブロックに実装する方法はない。カスタムブロックの構成を理解していないと簡単には導入できない。

## カスタムブロックの構成

1. ブロックの定義
2. コードの生成
3. ブロックのカテゴリーと配置決め

## Blockly Developer Tools

簡単にカスタムブロックを作成する支援ツールとして、Blockly Developer Tools がある。  
この支援ツールは blockly を用いて、ブロックを作ることができる。



## 1. ブロックの定義

作成したいブロックの外観とブロックに接続する数値やテキストをここで定義する。

外観は、ブロックの色やブロックの接続 (構文ブロックと値ブロック), 表示する文字等がある。

また、ブロック内の空きに何を入力 (input) or 出力 (output) とするか決める。



Figure: block の種類



Figure: console log に結果を出力する関数

## 2. コードの生成

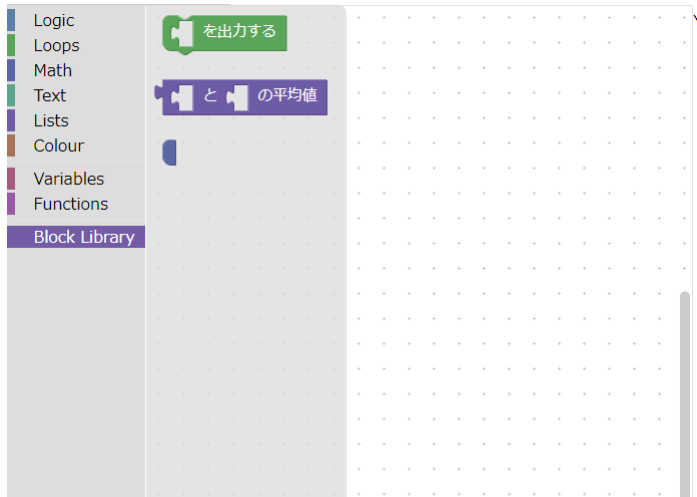
コードの生成では、ブロックの動作の定義を行う。例えば、平均値を出すブロックを作成するときは、平均を出す計算部分をここで書く。

```
Blockly.JavaScript['average'] = function(block) {  
  var value_v1 = Blockly.JavaScript.valueToCode(block, 'v1', Blockly.JavaScript.ORDER_ATOMIC);  
  var value_v2 = Blockly.JavaScript.valueToCode(block, 'v2', Blockly.JavaScript.ORDER_ATOMIC);  
  // TODO: Assemble JavaScript into code variable.  
  var code = '(' + value_v1 + '+' + value_v2 + ')/2';  
  // TODO: Change ORDER_NONE to the correct strength.  
  return [code, Blockly.JavaScript.ORDER_NONE];  
};
```

## 3. ブロックのカテゴリーと配置決め

作ったブロックをどこのカテゴリーに入れるかを決める。

はじめに  
VPL  
進捗具合  
おわりに



The screenshot shows the VPL Block Library interface. On the left is a vertical sidebar with a list of categories, each preceded by a small colored square: Logic (blue), Loops (green), Math (dark blue), Text (teal), Lists (purple), Colour (brown), Variables (maroon), Functions (dark purple), and Block Library (highlighted in a darker purple). To the right of the sidebar is a large grid area where blocks are placed. Two blocks are visible in the top-left of the grid: a green block labeled 'を出力する' (Output) and a purple block labeled 'と の平均値' (Average of ... and ...). The grid itself is composed of many small dots, and a vertical scrollbar is visible on the right side of the grid area.

## 実装手順

- 1 ブロックの役割と見た目を確定する.
- 2 ブロックの見た目と役割のための動作を実装する.

## ブロックの役割と見た目

作りたいのは、CSV を読み込ませてリストにしてくれるブロック.

すでにあるブロックとして、変数の Variables ブロックがある.  
このブロックは、型の指定がないのでこの変数に読み込んだ CSV を入れることにした.

→読み込んだ CSV を何度も読み込む必要がなくなる.

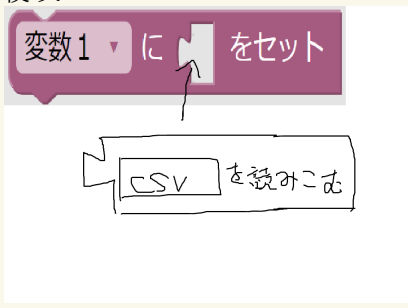
→変数名を変更できることができる.



## ブロックの役割と見た目

作りたいブロックの役割に変数名の設定が含まれていないので、目標とするブロックの見た目は以下のようなものにした。

実際に使うときは、以下のように変数のブロックとセットにして使う。



## ブロックの役割と見た目

実際に作ったブロックの外観と、外観作成のためのコードが以下になります。

**\*\* ファイルを選択してください \*\*** を読み込み

```
Blockly.JavaScript['import file'] = function(block) {  
  //var value_csv = Blockly.JavaScript.valueToCode(block, 'csv',  
  var value_csv = Blockly.JavaScript.valueToCode(block, 'csv', Bl  
  // TODO: Assemble JavaScript into code variable.  
  var code = '...';  
  // TODO: Change ORDER_NONE to the correct strength.  
  return [code, Blockly.JavaScript.ORDER_NONE];  
};
```

## ブロックの役割と見た目

```
Blockly.Blocks['import file'] = {
  init: function() {
    let fileInput = new Blockly.FieldTextInput('** ファイルを選択してください');
    this.appendDummyInput()
      .appendField(fileInput);
    //this.appendField(fileInput, 'FIELDNAME');
    //fileInput.showEditor_=(function(){alert("Intercepted");});
    fileInput.showEditor_=(()=>{
      var input = document.createElement('input');
      input.type = 'file';
      input.accept = '.csv';
      input.click();
    })

    this.appendDummyInput()
      .appendField("を読み込み");
    this.setInputsInline(true);
    //this.setOutput(true, "Array");
    this.setColour(285);
    this.setTooltip("");
    this.setHelpUrl("");
  }
};
```

## ブロックの役割のための動作作成

作ったブロックには、テキストの入力フィールドがあります。  
ここをクリックすることで、クリックをイベントハンドラとして  
`showEditor_`関数を使えるようになります  
この関数が起動すると `input` タグを作成。  
`type` を `file`, `accept` を `csv` のみにして `input` タグが動き、ファイルの  
読み込みダイアログがでるようにした。  
(こうすることで不可視の入力ボックスをつくることができた。)

```
showEditor_
```

```
保護された  
showEditor_ ()
```

フィールドクリックが呼び出され、クリックハンドラが定義されている場合は、ハンドラを呼び出します。



## 今回の進捗

- 1 ブロックの作成と，読み込みまでできるようにした

## 今後の課題

- 1 ファイルを読み込んだあとに，ファイル名を取得して，作ったブロックのファイルを選択してください，をファイル名に変更したい
- 2 csv をリストに変換
- 3 csv のファイルの書き方によらないものを作るか検討する。