

# プログラミング

第11回  
配列

久保田 匠

# [準備]授業資料にアクセス

いつもの作業

- 久保田の授業ホームページに資料がアップロードされている。
- まずは「愛教大 数学」と検索してみよう。

Google 愛教大 数学

すべて 画像 動画 ショッピング ニュース 地図 書籍 : もっと見る ツール

愛知教育大学 数学教育講座

愛知教育大学 数学教育講座

2025年度愛知教育大学数学教育講座研究会は、2023/12/09(土) 13:00 から対面形式で開催いたします。

授業用ホームページ (久保田)

2025年度前期担当科目

	月曜	火曜	水曜	木曜	金曜
1限					
2限	確率統計II			確率統計II	
3限			線形数学演習I	確率統計II	
4限	4年ゼミ			(オフィスアワー)	
5限					

2025年度後期担当科目

	月曜	火曜	水曜	木曜	金曜
1限					
2限					
3限	科学リテラシー				プログラミング
4限	(オフィスアワー)	3年ゼミ		4年ゼミ	プログラミング
5限					

その他のコンテンツ → ● ●

数学教育講座 久保田匠 (自然科学棟 521 研究室)  
Email: skubota [at] uecc.aichi-edu.ac.jp

## プログラミング

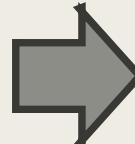
	内容	資料	コード
第1回	いろいろなプログラミング言語 VSCode のインストール	●	<a href="#">Prog_01-1</a>
第2回	Webページを構築する(HTML)	●	<a href="#">Prog_02-1</a>
第3回	Webページの見栄えを整える(CSS)	●	<a href="#">Prog_03-1</a> <a href="#">Prog_03-2</a>
第4回	JavaScriptに触れてみよう	●	<a href="#">Prog_04-1</a>
第5回	変数と演算	●, ★	(なし)
第6回	条件文	●, ★	(なし)
第7回	繰り返し(1)	●, ★	(なし)
第8回	繰り返し(2)		(なし)
第9回	繰り返し(3)		(なし)
第10回	オブジェクト		(なし)
第11回	配列	●	<a href="#">Prog_11-1</a> <a href="#">Prog_12-1</a>
第12回	ユーザー定義関数		<a href="#">Prog_12-1</a>
第13回	イベントハンドラ		(なし)
第14回	数式の表示(TeXについて)		<a href="#">Prog_14-1</a>
第15回	ウェブツールを開発してみよう		課題提出例

# [準備] コードの新規作成①

いつもの作業

- 授業用ホームページからサンプルコードをコピーしよう。

プログラミング			
	内容	資料	コード
第1回	いろいろなプログラミング言語 VSCode のインストール	●	<a href="#">Prog_01-1</a>
第2回	Webページを構築する(HTML)	●	<a href="#">Prog_02-1</a>
第3回	Webページの見栄えを整える(CSS)	●	<a href="#">Prog_03-1</a> <a href="#">Prog_03-2</a>
第4回	JavaScriptに触れてみよう	●	<a href="#">Prog_04-1</a>
第5回	変数と演算	●, ★	(なし)
第6回	条件文	●, ★	(なし)
第7回	繰り返し(1)	●, ★	(なし)
第8回	繰り返し(2)		(なし)
第9回	繰り返し(3)		(なし)
第10回	オブジェクト		(なし)
第11回	配列	●	<a href="#">Prog_11-1</a>
第12回	ユーザー定義関数		<a href="#">Prog_12-1</a>
第13回	イベントハンドラ		(なし)
第14回	数式の表示(TeXについて)		<a href="#">Prog_14-1</a>
第15回	ウェブツールを開発してみよう		<a href="#">課題提出例</a>



### Prog\_11-1

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Prog_11-1</title>
</head>

<body>
<!--
    演習のときに使う配列（適宜コピペして使ってください）
    let scores = [50, 70, 37, 90, 67];
    let array = [6, 1, 10, -3, 5, 23, 0, -12, 8, 4];
-->
</body>

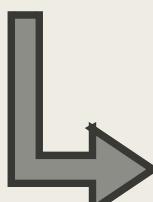
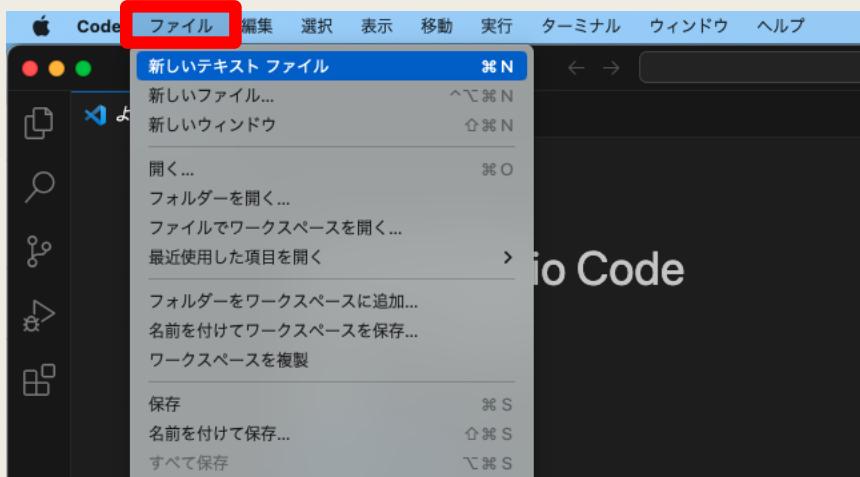
</html>
```

今日は「Prog\_11-1」を選択してください。

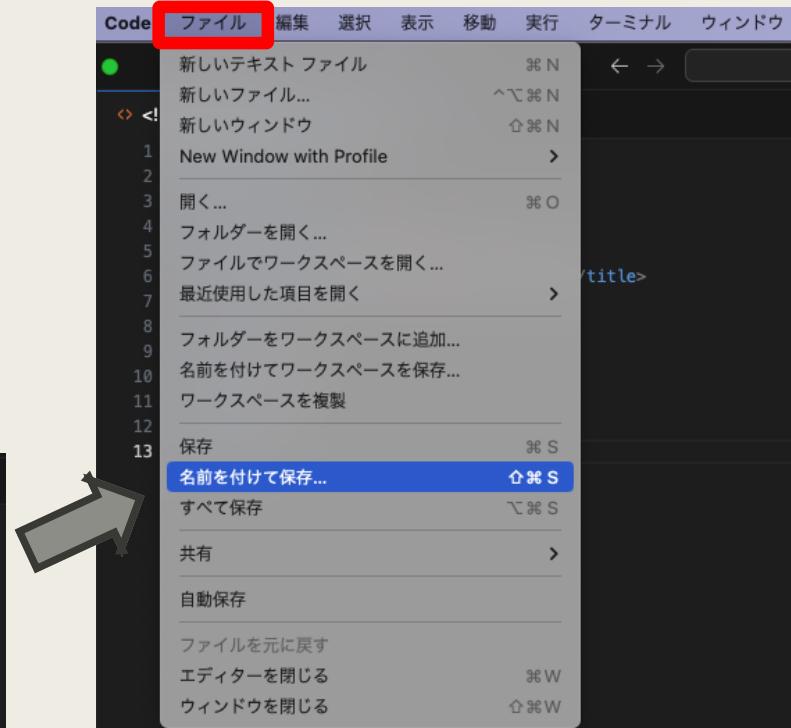
# [準備] コードの新規作成②

いつもの作業

- VSCode を起動し「ファイル」から「新しいテキストファイル」を選択。
- その後、さきほどコピーした文書をペースト（Ctrl + V）して「名前をつけて保存」。



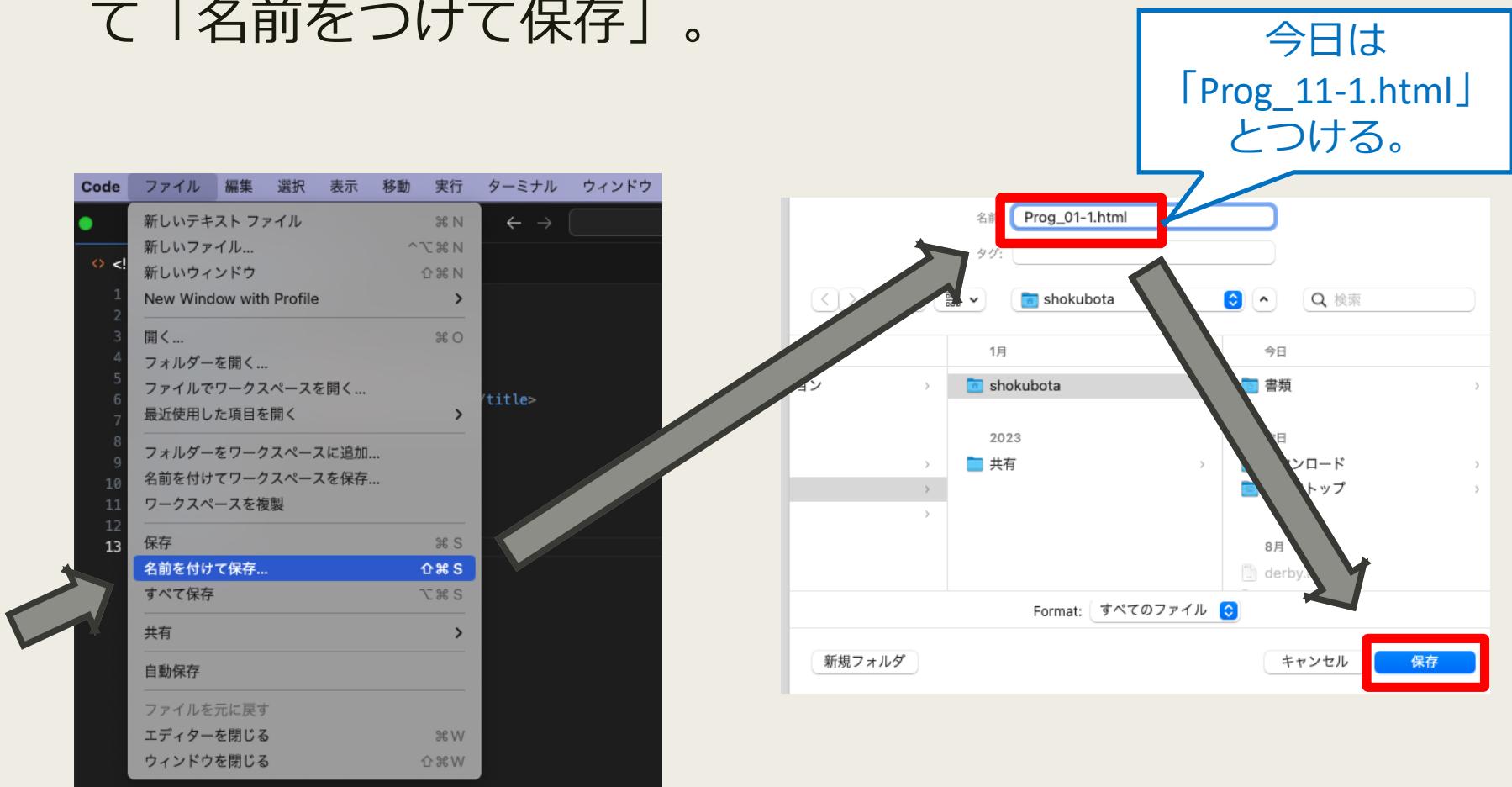
Ctrl + V



# [準備] コードの新規作成②

いつもの作業

- VSCode を起動し「ファイル」から「新しいテキストファイル」を選択。
- その後、さきほどコピーした文書をペースト (Ctrl + V) して「名前をつけて保存」。



# [準備]作業環境を整える

いつもの作業

- 保存したhtmlファイルをダブルクリックして開いておく。
- PCの画面をふたつに分け、片方はブラウザ、もう片方はvsCodeを開いておくと便利。

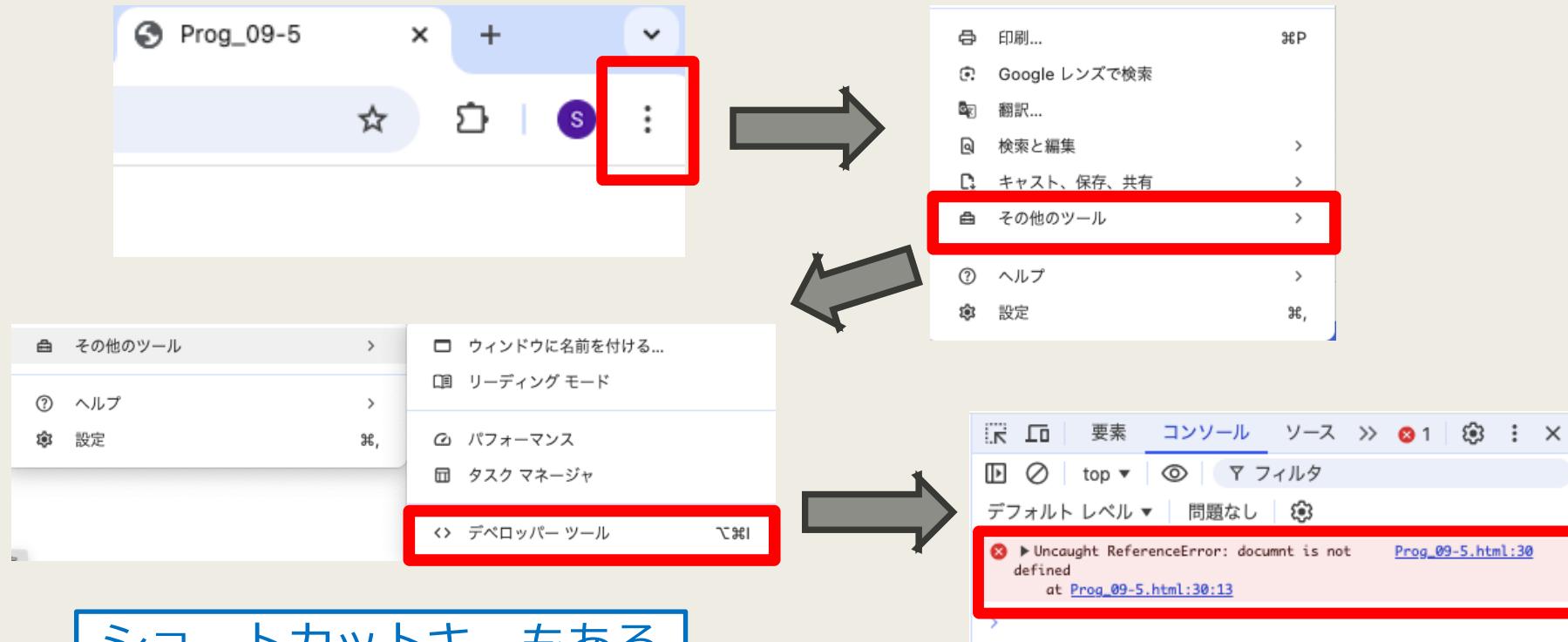


```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>ここにページのタイトルを書きます</title>
</head>
<body>
    こんにちは！<br>私の名前は久保田匠です。
</body>
</html>
```



# [再掲]デベロッパーツール

- 画面に何も表示されないときや、途中までしか表示されないときはプログラムに間違いがある可能性が高い。
- そのときは「デベロッパーツール」を開き、何行目でエラーが発生しているか見てみよう。



30行目でエラーが発生。  
document が未定義と言われている  
(スペルミスが発生していた)

# [復習]オブジェクト

- 深入りせず「使えれば良し」、  
ということで進めた。
- 「データ」と「データに関する操作」をセットにしたものを  
**オブジェクト**という。
- 「データ」のことを**プロパティ**といい、「データに関する操作」を**メソッド**という。
- 例として2次元ベクトルをオブジェクトとして考える。

**プロパティ**  
(2次元ベクトルがもつデータ)

x座標

y座標

**メソッド**  
(2次元ベクトルに関する操作)

大きさを計算する

他のベクトルとの  
和やスカラー倍を計算する

他のベクトルとの内積を計算

あまり気にしなくてよい

- これまで扱ってきた `document.write(***)` は、`document` オブジェクトの `write` メソッド。
- `console.log(***)` は `console` オブジェクトの `log` メソッド。

# [復習]インスタンスの生成（実体化）

具体的なデータを作成する作業

- 前回は Math, String, Date オブジェクトを紹介した。
- オブジェクトを使用するときは通常 インスタンスの生成（実体化）を行う必要がある。
  - 2次元ベクトルオブジェクトを例に考えると、インスタンスの生成は「具体的に考える2次元ベクトルを作る」ことに相当する。  
*あまり気にしなくてよい*
- ただし、あらかじめ実体化されているオブジェクトや、インスタンスを持たない例外的なオブジェクトもある。
  - document オブジェクトや console オブジェクトはあらかじめ実体化されているオブジェクトの例。
  - Math オブジェクトはインスタンスを持たないオブジェクトである。
  - String オブジェクトや Array オブジェクトは、インスタンスの生成を明示的に行わなくても内部的にインスタンスが生成され、そのまま利用できるオブジェクト。

# [復習]Math オブジェクト

- Math オブジェクトは本講義全体を通して重要である。
- Math オブジェクトのプロパティは  
Math.プロパティ名  
の形で使用する。
- Math オブジェクトのメソッドは  
Math.メソッド名(引数1, 引数2, ...) の形で使用する。
- Math.random() は0以上1未満の数を生成する。
- Math.floor(a) は引数 a の小数点以下を切り捨てる。
- 前回は、このふたつを組み合わせることで指定した範囲の乱数を発生させた。

**Mathオブジェクトのプロパティ  
(数学に関するあらゆる定数)**

円周率 $\pi$

自然対数の底 e

:

**Mathオブジェクトのメソッド  
(数学に関するあらゆる操作)**

絶対値を返す

sin や cos の値を返す

最大値 (最小値) を返す

0以上1未満の乱数を返す

小数点以下を切り捨てる

:

# [再掲]第10回以降に学ぶこと

逆行列  
行列  $\begin{bmatrix} -5 & 0 & 1 \\ -4 & -1 & 1 \\ -2 & -2 & 1 \end{bmatrix}$  の逆行列は

ボタンをクリックすると  
答えが表示される

である。

答え

逆行列

行列  $\begin{bmatrix} -5 & 0 & 1 \\ -4 & -1 & 1 \\ -2 & -2 & 1 \end{bmatrix}$  の逆行列は

$\begin{bmatrix} 1 & -2 & 1 \\ 2 & -3 & 1 \\ 6 & -10 & 5 \end{bmatrix}$  である。

答え

[第14回]

きれいな数式を  
表示する

[第10回]  
乱数を発生させる

[済]  
生成した問題に  
対して答えを計算  
(透明色で表示)

[第13回]  
ボタンを押したときに  
特定の処理を行う

[第12回]  
処理のかたまりを  
定義する

■ 第11回では配列を扱う。

- 配列はひとつの変数名で複数のデータをまとめて管理できるようにしたもの。
- 例えば、上の例で配列を使わずにプログラムすると、問題の行列と答えの行列の各成分で合計18個の変数を用意しなければならない。

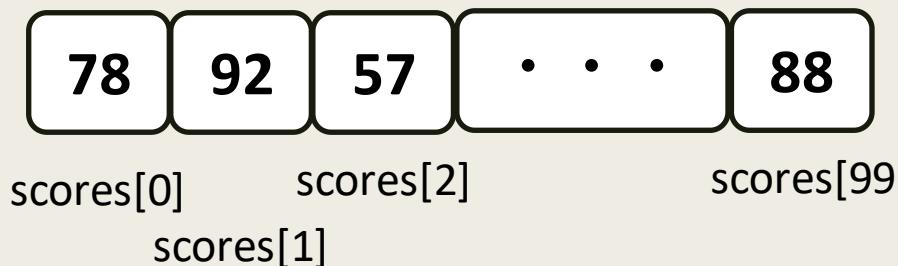
# 配列

- 配列とは、添字（そえじ）と呼ばれる番号を使ってひとつの変数名で複数のデータをまとめて管理できるようにしたもの。
- 配列を使わない場合、100名の学生の点数を管理する状況では、個別に100個の変数を宣言する必要がある。
- これに対して、配列を使えば scores のようなひとつの変数名だけで100個のデータをまとめて管理できる。

```
let score1 = 78;  
let score2 = 92;  
let score3 = 57;  
:  
let score100 = 88;
```

これまでの方法では  
100個の変数を  
宣言する必要がある

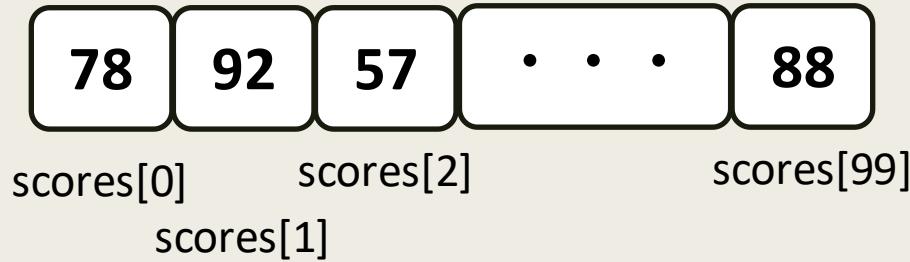
配列「scores」



配列はひとつ宣言すれば  
あとはまとめて管理できる

# 配列名と添字

配列「scores」



- 配列に格納された個々の値を **要素** という。
- それぞれの要素には次の形式でアクセスする。  
    配列名[添字]   (上の例では `score[0]` で 78 を取得)
- 上記のように配列名のあとに添字を大括弧 (`[ ]`) で囲って記述する。
- 添字の番号は「0」から始まることに注意。
- 100個の要素からなる配列の最後の要素の添字は 99 である。

配列の最後の番号 = 要素数 - 1

# 点数の合計を求める

配列の最後の番号 = 要素数 - 1

- 配列は次の形式で宣言する。

```
let 配列名 = [要素, 要素, ...]
```

- 配列に含まれる要素の個数は 配列名.length で取得できる。
- 次のコードを入力してみよう。

```
1  <!DOCTYPE html>
2  <html>
3
4  <head>
5      <meta charset="UTF-8">
6      <title>Prog_10-1</title>
7  </head>
8
9  <body>
10     <p>
11         <script>
12             let scores = [50, 70, 37, 90, 67];
13             let total = 0;
14             for(let i=0; i < scores.length; i++){
15                 total += scores[i];
16             }
17             document.write("合計は " + total + " 点です。");
18         </script>
19     </p>
20  </body>
21
22  </html>
```

12行目はサンプルコード  
からコピペできる

合計は 314 点です。

※配列を使わない場合は  
for文が使えない。

# [演習]点数の平均を求める

- 先のプログラムの修正し、配列の要素の平均を求めるプログラムを完成させよ。

```
1  <!DOCTYPE html>
2  <html>
3
4  <head>
5      <meta charset="UTF-8">
6      <title>Prog_10-2</title>
7  </head>
8
9  <body>
10     <p>
11         <script>
12             let scores = [50, 70, 37, 90, 67];
13
14
15
16
17
18
19         </script>
20     </p>
21 </body>
22
23 </html>
```

考えてみよう

平均は 62.8 点です。

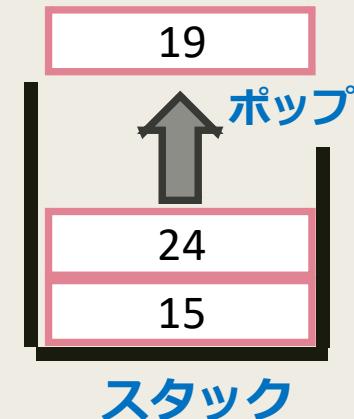
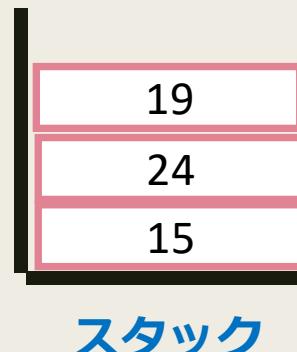
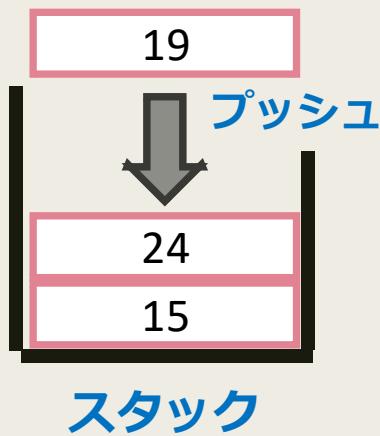
# Array オブジェクト

- JavaScript で配列を作った場合、それは Array オブジェクトとして振る舞う。
  - Array とは配列のこと。
- よって、Array オブジェクトのプロパティやメソッドが利用できる。
- さっき使った 配列名.length は Array オブジェクトのプロパティのひとつ。
- Array オブジェクトの主なメソッドは教科書の p193 を参照。
- Array オブジェクトのメソッドのうち、この授業では pop と push を紹介するが、その前にスタックと呼ばれるデータ構造について説明する。

インスタンスの生成を  
明示的に行わなくてよい。

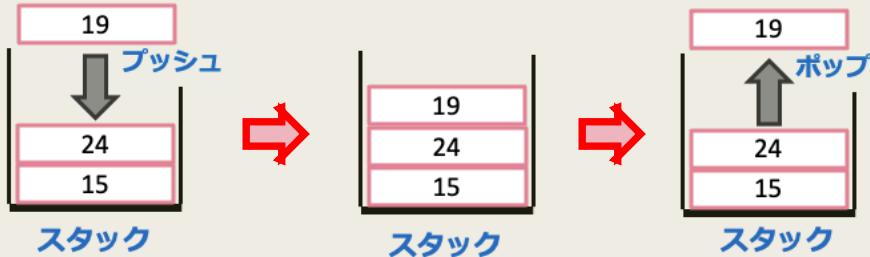
# スタック、ポップ、push

- 後に入れたデータが先に出る構造になっているものを **スタック** という。
  - カゴに物が入っている様子をイメージするとよい。
  - スタックに対して、先に入れたデータが先に出る構造になっているものを **キュー** という。
- スタックにデータを入れる操作を **push** という。
- スタックからデータを取り出す操作を **pop** という。



# push メソッドと pop メソッド

- push メソッドは配列の最後に新しい要素を追加する。
- pop メソッドは配列の最後の要素を削除し、その削除した要素を返す。



```
let array = [15,24];
array.push(19);
console.log(array); // [15,24,19]
let x = array.pop();
console.log(array); // [15,24]
console.log(x); // 19
```

- 配列を宣言するとき、要素や要素の個数がその時点では決まっておらず、とりあえず配列を用意だけしておきたい状況がある。
- そのようなときは要素をひとつももない配列を宣言しておくのが便利である。

```
let array = [];
```

 のように宣言する
- 要素をひとつももない配列を 空配列（からはいれつ） という。

# [演習] 5より大きい数を選ぶ

- 次のプログラムの空欄をうめ、与えられた配列の要素のうち、5より大きいものだけを残した配列を求めよう。

```
1  <!DOCTYPE html>
2  <html>
3
4  <head>
5      <meta charset="UTF-8">
6      <title>Prog_10-3</title>
7  </head>
8
9  <body>
10     <p>
11         <script>
12             let array = [6, 1, 10, -3, 5, 23, 0, -12, 8, 4];
13             let greaterThanFive = [];
14             for(l
15                 i
16             )
17         }
18     }
19     document.write("5より大きい要素を残した配列は [" + greaterThanFive + "] です。");
20
21     </script>
22     </p>
23
24  </body>
```

12行目はサンプルコード  
からコピペできる

考えてみよう

5より大きい要素を残した配列は [6,10,23,8] です。

# 配列のコピー

- 次のコードを入力してみよう。
- どんな出力になるか予想しながら入力しよう。

```
11 |     <script>
12 |         let a = 3;
13 |         let b = a;
14 |         a = 5;
15 |         document.write("a の値は"+ a + "で、b の値は" + b + "です。<br>");
16 |         let A = [1,2,3];
17 |         let B = A;
18 |         A[0] = 10;
19 |         document.write("配列Aは [" + A + "] で、配列Bは [" + B + "] です。");
20 |     </script>
```

a の値は5で、b の値は3です。  
配列Aは [10,2,3] で、配列Bは [10,2,3] です。

- 配列 B の出力結果は予想通りだろうか？
- なぜこのような出力になるのだろうか？

# [復習]変数（第5回「変数と演算」より）

- プログラミング言語において最も重要な要素のひとつに、値を一時的に格納する**変数**がある。
- **変数**は、値を入れる箱のようなもの。
- それぞれの箱には**変数名**と呼ばれる名前がある。
- **アドレス**と呼ばれる、箱を置いてある場所を表す数字の列もある（後の学習のために頭の片隅に入れておくとよい）。

例：年齢を管理する変数「age」を用意して 19 を代入



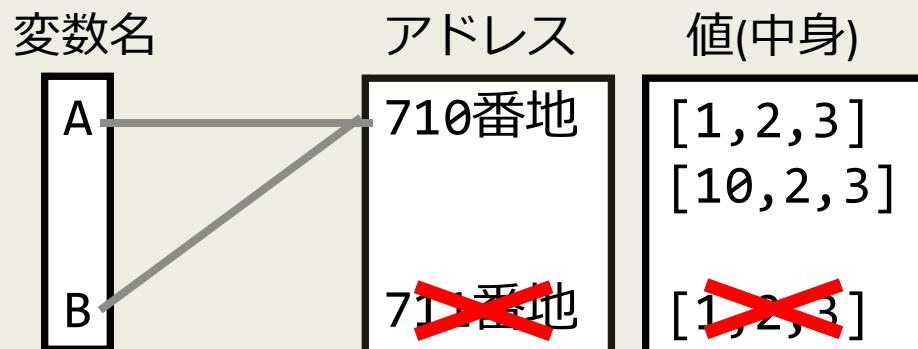
実際のアドレスは 0x7ffee10b2a4c のようにもっと複雑

```
b = a;
```

# Pass-by-Sharing

- JavaScript に限らず多くの言語において、変数に変数を代入するときは、次のように処理される。
- その変数が数値・文字列・真偽値の場合は変数の中にあるデータそのものがコピーされる（人間の直感に近い処理）。
- これを **値渡し** という。
- 一方、その変数がオブジェクト（配列など）の場合は、その変数のアドレスをコピーする。
- これを **共有渡し** もしくは **参照の値渡し** という。
  - 英語では *Pass-by-Sharing* と呼ばれ、和訳はまだ定まっていないらしい。

```
let A = [1,2,3];
let B = A;
let A[0] = 10;
```



# よく分からなかつた人向けの説明



大場つぐみ・小畠健『DEATH NOTE』より

**配列をコピーするときは注意しましょう**

# [演習]配列をコピーする

- それでは配列 A の中身を配列 B にコピーしたいときはどうすればよいだろうか？
- for文を使って配列Aのひとつひとつの値をコピー（push）すればよい。
- 出力が次になるように、以下のプログラムの空欄をうめよう。

配列Aは [10,2,3] で、配列Bは [1,2,3] です。

```
1  <!DOCTYPE html>
2  <html>
3
4  <head>
5      <meta charset="UTF-8">
6      <title>Prog_10-5</title>
7  </head>
8
9  <body>
10 <p>
11     <script>
12         let A = [1,2,3];
13         let B = [];
14
15         考えてみよう
16
17         A[0] = 10;
18         document.write("配列Aは [" + A + "] で、配列Bは [" + B + "] です。");
19     </script>
20 </p>
21 </body>
22
23 </html>
```

# [演習]教科書を熟読しよう

- 今日の内容は教科書の p176～p193 がベースになっている
- 残った時間で自分でも該当箇所を熟読してみよう。
- 配列の宣言の仕方が授業のやり方と少し違う（教科書ではオブジェクトであることを強調した方法をとっている）ので、その違いも踏まえながら読んでみよう。
- 授業で解説していないコードは自分でも入力してみてどのような出力結果になるか確かめてみよう。