多言語プログラミング

↪ プログラミングにおける骨格

電気技術研究会

NIT, Nara Collage

February 21, 2023

概要

講義目的と進め方

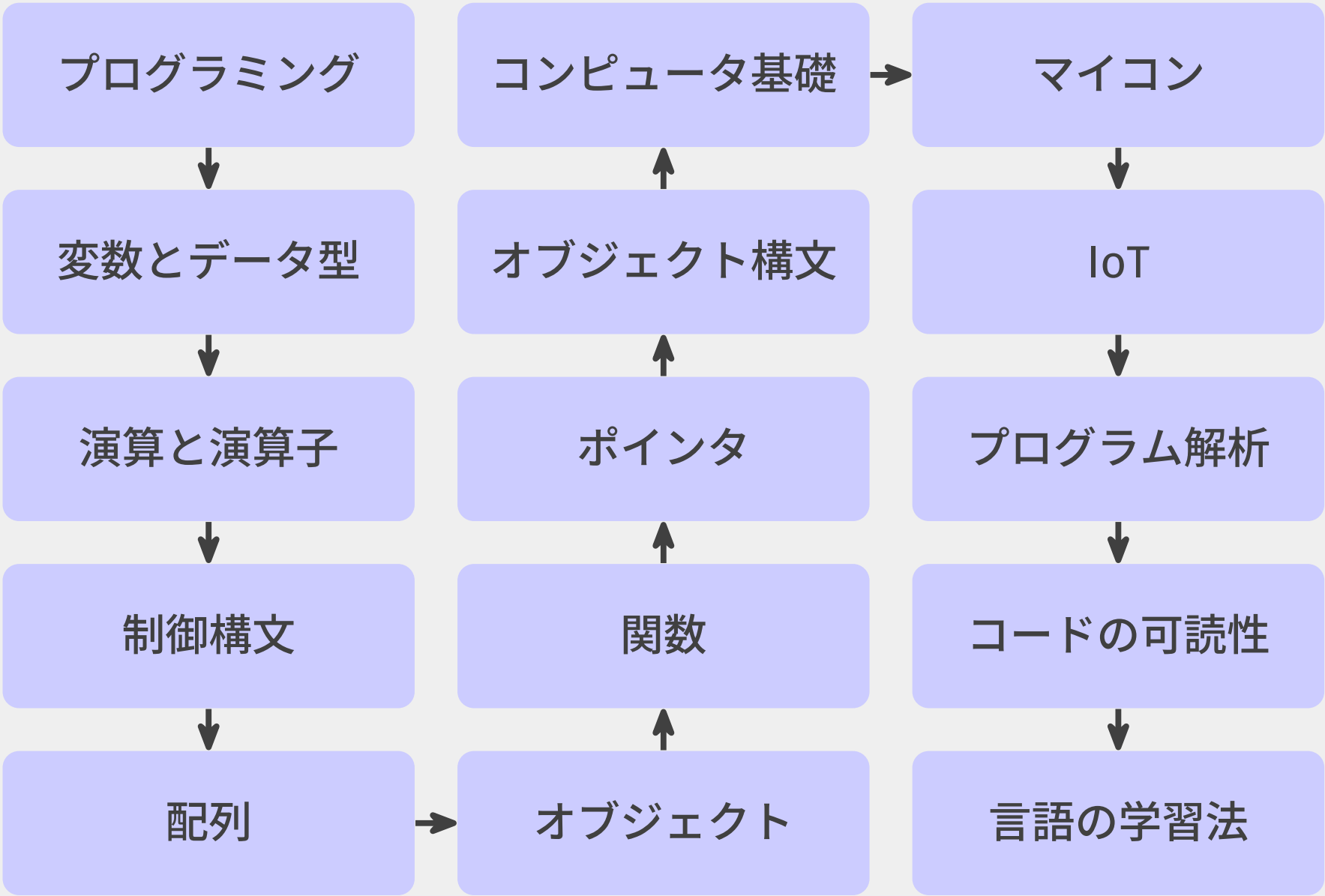
講義目的

- 多言語にわたるプログラミングを独学する術を身に付ける
- 基本的な用語や技術を学び 活かせる力を身に付ける
- マイコンやウェブなど 広い範囲に適応できる人材育成

進め方

- 重要な章ごとにまとめて講義
- １日（２～３時間）１～２章分を進めたい
- 各章ごとに演習がある

カリキュラム


プログラミング
変数とデータ型
演算と演算子
制御構文
配列
オブジェクト
関数
ポインタ
オブジェクト構文
コンピュータ基礎
マイコン
IoT
プログラム解析
コードの可読性
言語の学習法

プログラミングとは

進度


プログラミング
変数とデータ型
演算と演算子
制御構文
配列
オブジェクト
関数
ポインタ
オブジェクト構文
コンピュータ基礎
マイコン
IoT
プログラム解析
コードの可読性
言語の学習法

プログラミングについて

プログラミングどんなもの？

- プログラミングには様々な種類が存在
  ウェブ マイコン ゲーム...etc
- 大まかには２種類
  オブジェクト指向と構造化プログラミング（後述）
- 骨格は基本的に共通
  関数や変数など 基本的な骨格がある

プログラミング言語

- 言語は多数ある
  C, C++, C#, JavaScript, VisualBasic, Python, LaTeX
- 本教材では主に JavsScript と Arduino による C++を使用

基本的な骨格とは

- 冒頭でライブラリの宣言
- メインループで処理
  UI 等の場合ではイベントを処理
- サブルーチン（関数）にて
  繰り返し処理

C 言語の例

```c
#include <stdio.h>

int main(void){
  printf('Hello World');
  return 0;
}
```

Arduino の例

```c
void setup(){
  pinMode(2, OUTPUT);
}
void loop(){
  digitalWrite(2, HIGH);
  delay(200);
  digitalWrite(2,LOW);
  delay(200);
}
```

演習１：問題
↪ プログラミングとは

JavaScript 動作確認

1. VSCode の作業フォルダを作成
2. 'Hello World' を表示させる

HTML ファイルへの埋め込み

- html ファイルの<script>タグ内
- ディスプレイさせる関数
  `document.writeln('文字列')`
- 文末には ;をつける
- 文字列は 'か "で囲う

html テンプレート

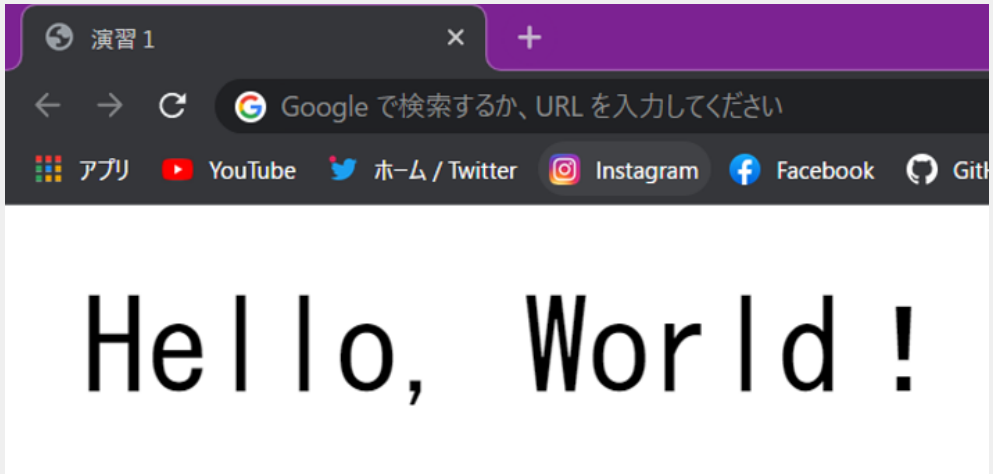
```
<html>
<head>
<meta
  http-equiv="Content-Type"
  content="text/html; charset=UTF-8">
<title>Hello, World!</title>
</head>
<body>
<pre>
<script type="text/javascript">
  // ここにスクリプトを記述
</script>
<noscript>
  JavaScriptが利用できません。
</noscript>
</pre>
</body>
</html>
```

演習１：解答

↪ プログラミングとは

演習１

```
document.writeln('Hello, World!');
```



変数とデータ型

進度

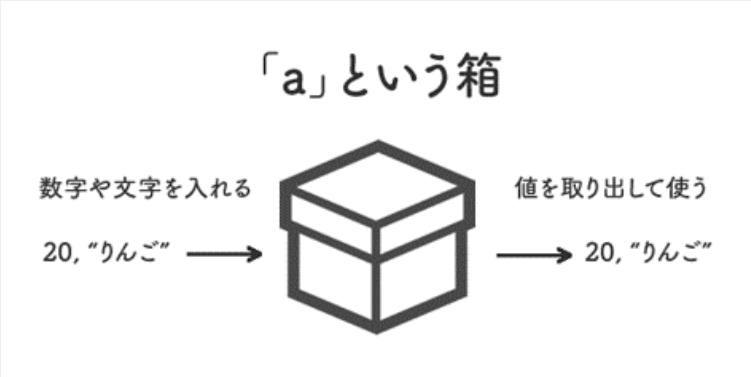

プログラミング
変数とデータ型
演算と演算子
制御構文
配列
オブジェクト
関数
ポインタ
オブジェクト構文
コンピュータ基礎
マイコン
IoT
プログラム解析
コードの可読性
言語の学習法

変数

↪ 変数について

変数とは

- データを入れる箱
- 入れるデータによって種類がある
- 箱をひとまとめにしたもの　▷　配列



変数の基本要素

- 宣言
  変数を作成すること
- 初期化
  変数の作成とともに数値を割り当てること
- 代入
  変数の値を上書きすること

変数
↪ 宣言

- どのような言語でも**基本おなじ** (python では宣言が暗黙)
- **(修飾子) + 型 + 変数名**で宣言

```
let x;
int x;
```

- 変数名は**予約語**以外なら原則何でも可

変数
↪ 初期化

- 宣言とともに値を割り当てる
  (python では宣言と初期化がセット)
- 型 + 変数名 = 初期値で宣言 + 初期化をセットで

```
let x = 100;
int x = 100;
```

- 静的型付けでは初期値の型が一致している必要がある。

~~`int x = 'a';`~~

初期化をしないと...

ランダムに適当な値（不定値）が割り当てられる
javascript では"undefined"と出力される
▶ 予測してない結果になる可能性がある
▼
基本的に宣言と初期化はセットでする

変数

↪ 代入

- 変数を上書きして内部の値を更新する
- 宣言後 任意の場所で 変数名 = 値

```
let x = 100;
x = 200;
```

- =は実は演算子

演習２－１：問題

↪ 変数とデータ型

`let` 型の変数を用いて Hello World を表示させる

1. **宣言**は **型 (今回は `let`) + 変数名 (適当)**
2. **初期化**は 宣言 **= 初期値**
3. 文字列は ’か ”で囲う
4. `document.writeln(変数名)` を使用して表示させよう

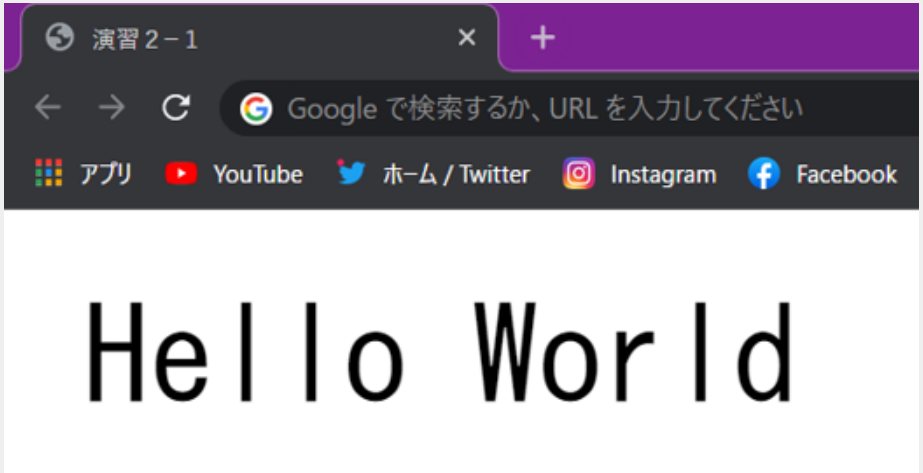
演習２－１：解答

↪ 変数とデータ型

演習２－１

```
let str='Hello World';

document.writeln(str);
```



データ型

↪ データ型について

データ型とは

- 変数に入るデータを予め決めておかなければいけない
  $\Longrightarrow$ 予め決める言語を静的型付け言語という
- 整数 (`int`) 型や 文字 (`char`) 型など多数ある

※**`var`** 型や **`let`** 型は直接なデータ型ではない

Table: 様々なデータ型

分類	名前	値の例
論理型	**`Boolean`**	True, false
文字型	`char`	a, b, c
文字列型	**`String`**	abc
整数型	`int`	1, 333
浮動小数点型	`float`, `double`	0.5, 0.0093
配列型	**`Array`**	[1,2,3]
オブジェクト型	**`Object`**	x:1, y:2, z:3

言語によってバラツキがあるので 使用したい言語で確認する

データ型
↪ 動的型付けについて

- JavaScript では**自動**でデータ型が割り当てられる
  $\Longrightarrow$ **動的型付け**言語という
- よく使われるのは `var` と `let` の 2 種類
- `var` は**宣言の重複**ができ `let` は不可
- `var` は後述の**スコープ**が特殊なので 基本的には `let`を使用

データ型

↪ データ型の使い分け

サイズによる使い分け

変数の**データ型**には**サイズ**がある

オーバーフロー

サイズを超えると**エラー**が起きたり
予想しない結果になることも

※ 使用する言語で変数のサイズを確認して使用しよう

演習２－２：問題

↪ 変数とデータ型

var 型と **let** 型の違いを確認する（**宣言の重複**）

1. **var** 型の変数１を**宣言+初期化**する
2. **var** 型の変数１をもう一度**宣言+初期化**し 表示
3. **let** 型の変数２を**宣言+初期化**する
4. **let** 型の変数２をもう一度**宣言+初期化**し 表示

※ 変数については演習２－１を参照

`document.writeln('')` を使用し 表示

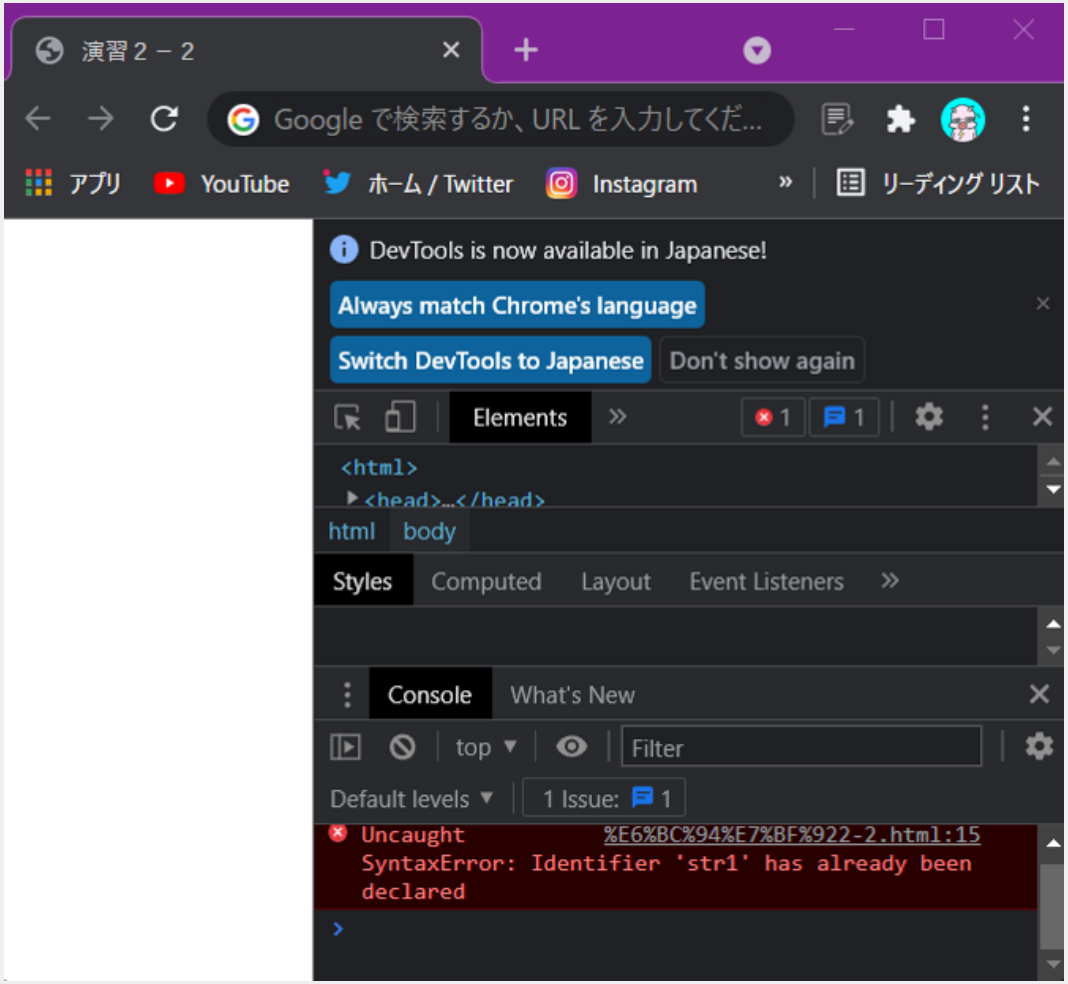
演習２－２：解答

↪ 変数とデータ型

演習２－２

```
var str = 'Hello World';
var str = 'Hello JS';
document.writeln(str);

let str1 = 'Hello World';
let str1 = 'Hello JS'
document.writeln(str1);
```



演算と演算子

進度


プログラミング
変数とデータ型
演算と演算子
制御構文
配列
オブジェクト
関数
ポインタ
オブジェクト構文
コンピュータ基礎
マイコン
IoT
プログラム解析
コードの可読性
言語の学習法

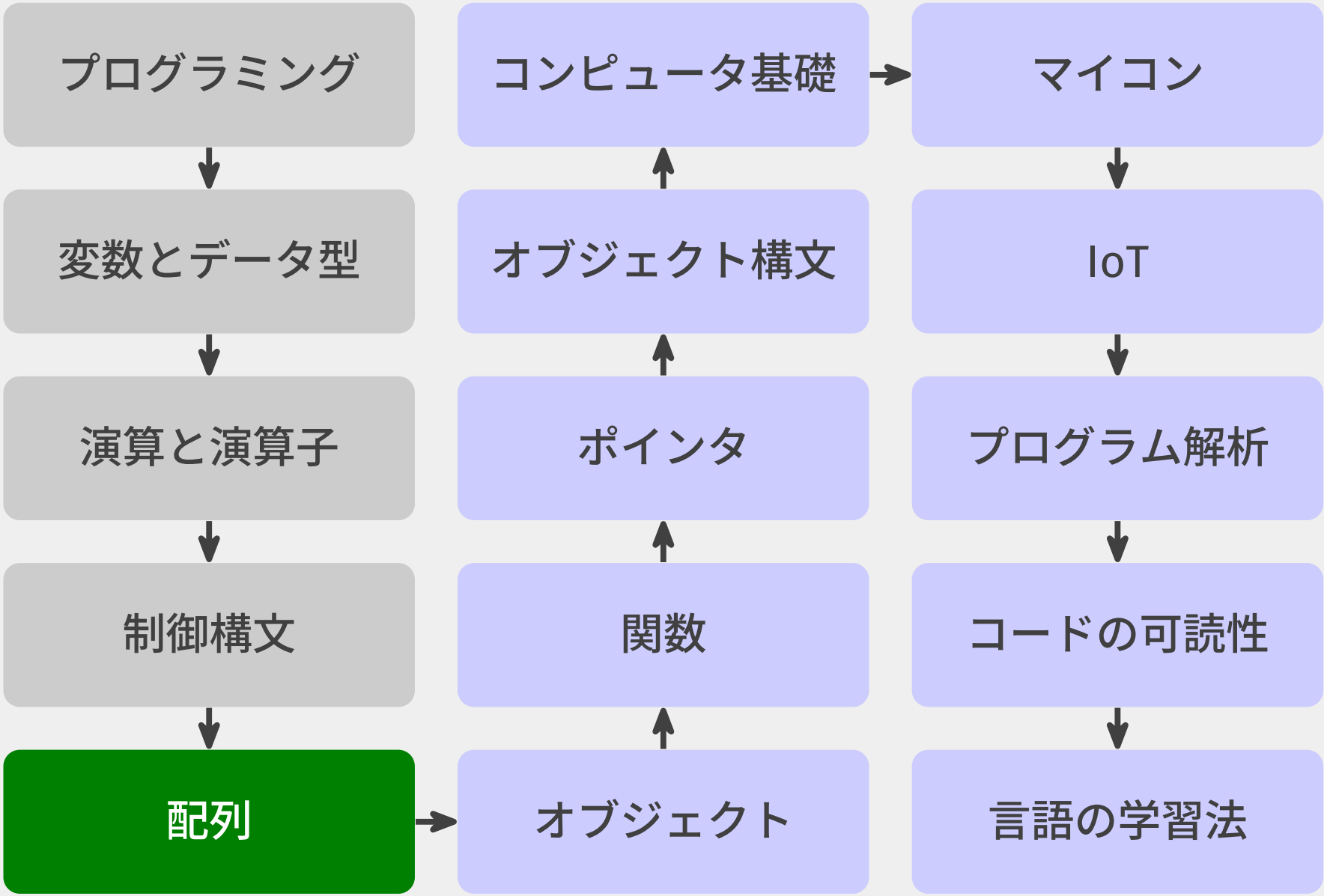
制御構文

進度


プログラミング
変数とデータ型
演算と演算子
制御構文
配列
オブジェクト
関数
ポインタ
オブジェクト構文
コンピュータ基礎
マイコン
IoT
プログラム解析
コードの可読性
言語の学習法

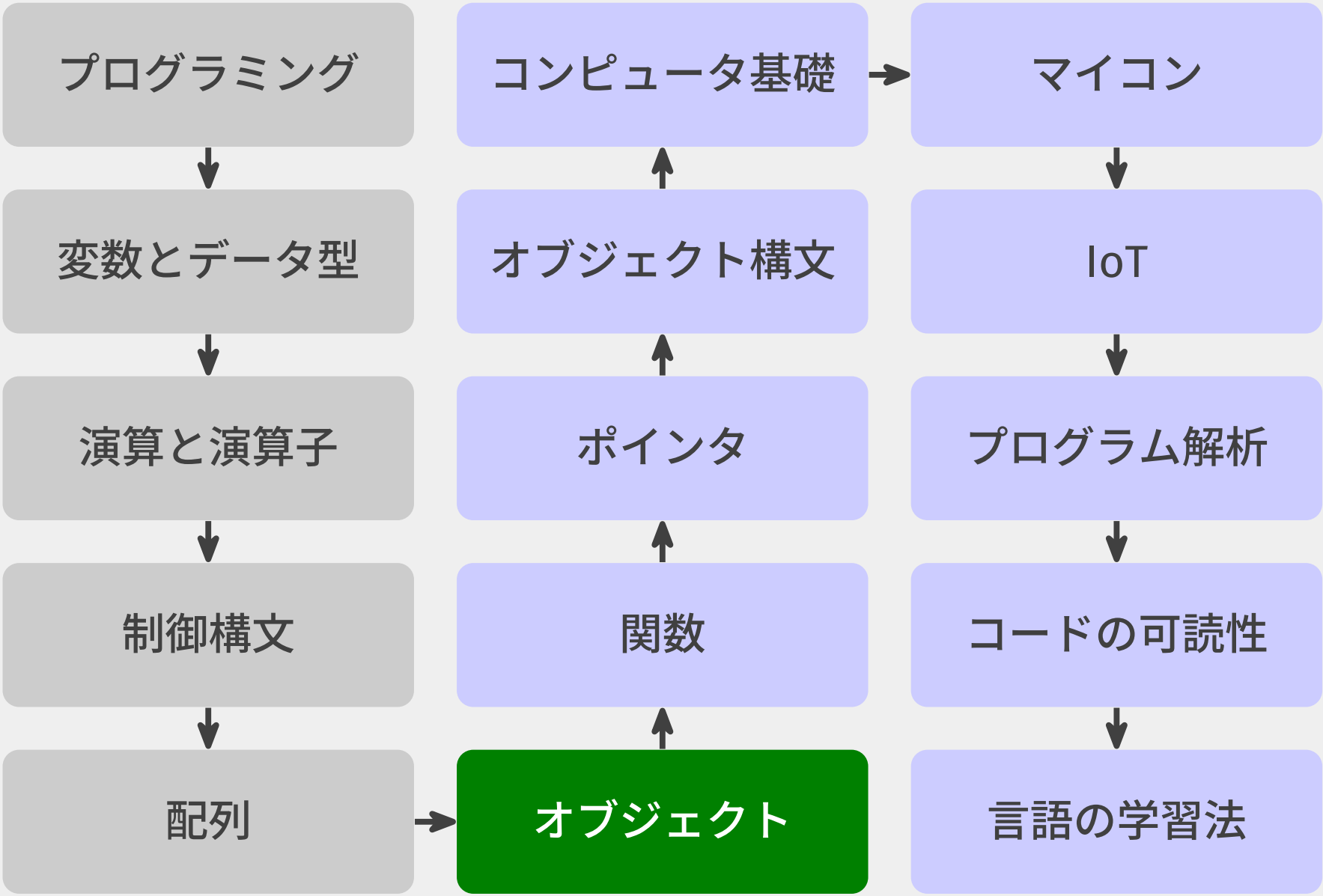
配列

進度


プログラミング
変数とデータ型
演算と演算子
制御構文
配列
オブジェクト
関数
ポインタ
オブジェクト構文
コンピュータ基礎
マイコン
IoT
プログラム解析
コードの可読性
言語の学習法

オブジェクト

進度


プログラミング
変数とデータ型
演算と演算子
制御構文
配列
オブジェクト
関数
ポインタ
オブジェクト構文
コンピュータ基礎
マイコン
IoT
プログラム解析
コードの可読性
言語の学習法

関数

進度


プログラミング
変数とデータ型
演算と演算子
制御構文
配列
オブジェクト
関数
ポインタ
オブジェクト構文
コンピュータ基礎
マイコン
IoT
プログラム解析
コードの可読性
言語の学習法

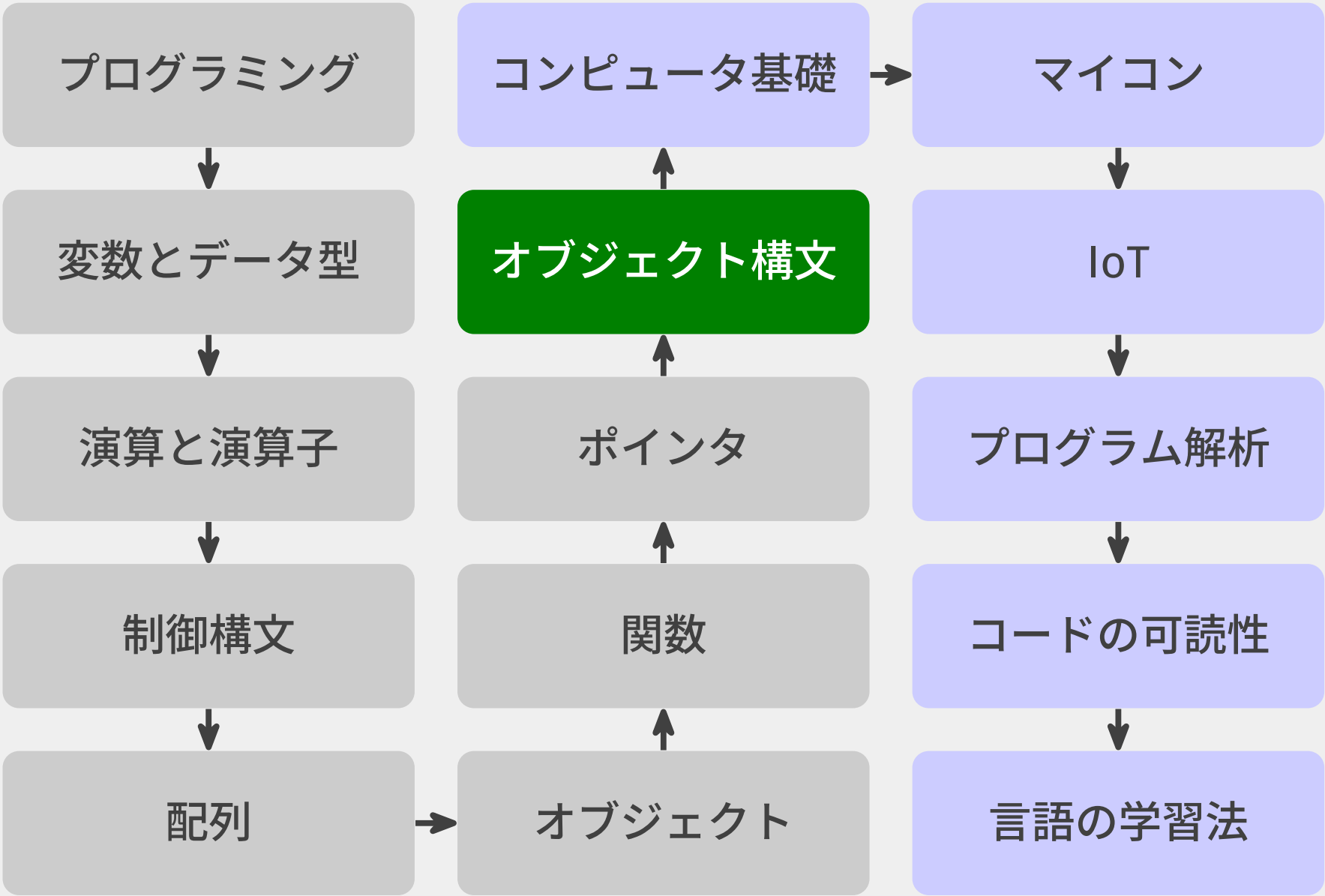
ポインタ

進度


プログラミング
変数とデータ型
演算と演算子
制御構文
配列
オブジェクト
関数
ポインタ
オブジェクト構文
コンピュータ基礎
マイコン
IoT
プログラム解析
コードの可読性
言語の学習法

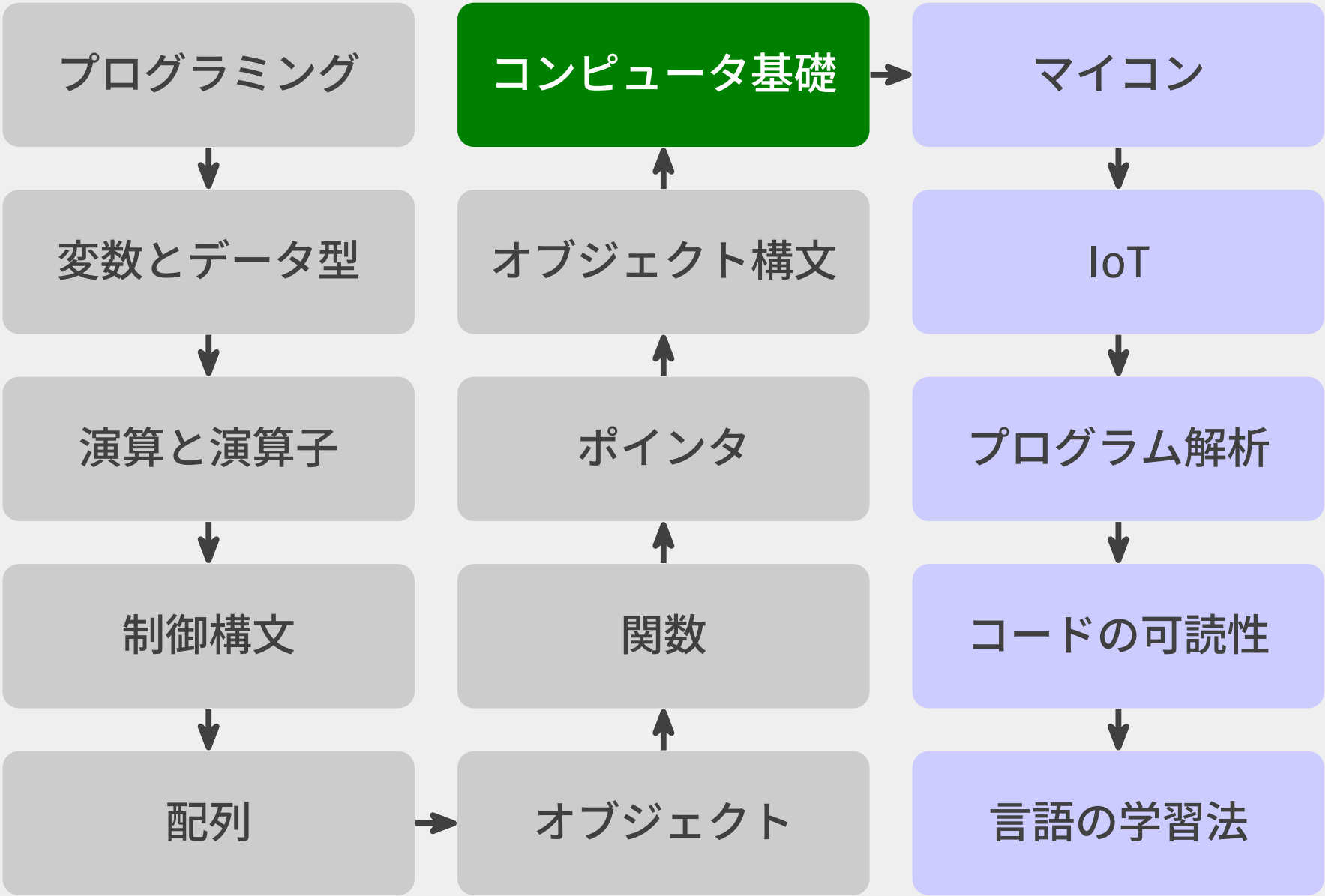
オブジェクト指向構文

進度


プログラミング
変数とデータ型
演算と演算子
制御構文
配列
コンピュータ基礎
オブジェクト構文
ポインタ
関数
オブジェクト
マイコン
IoT
プログラム解析
コードの可読性
言語の学習法

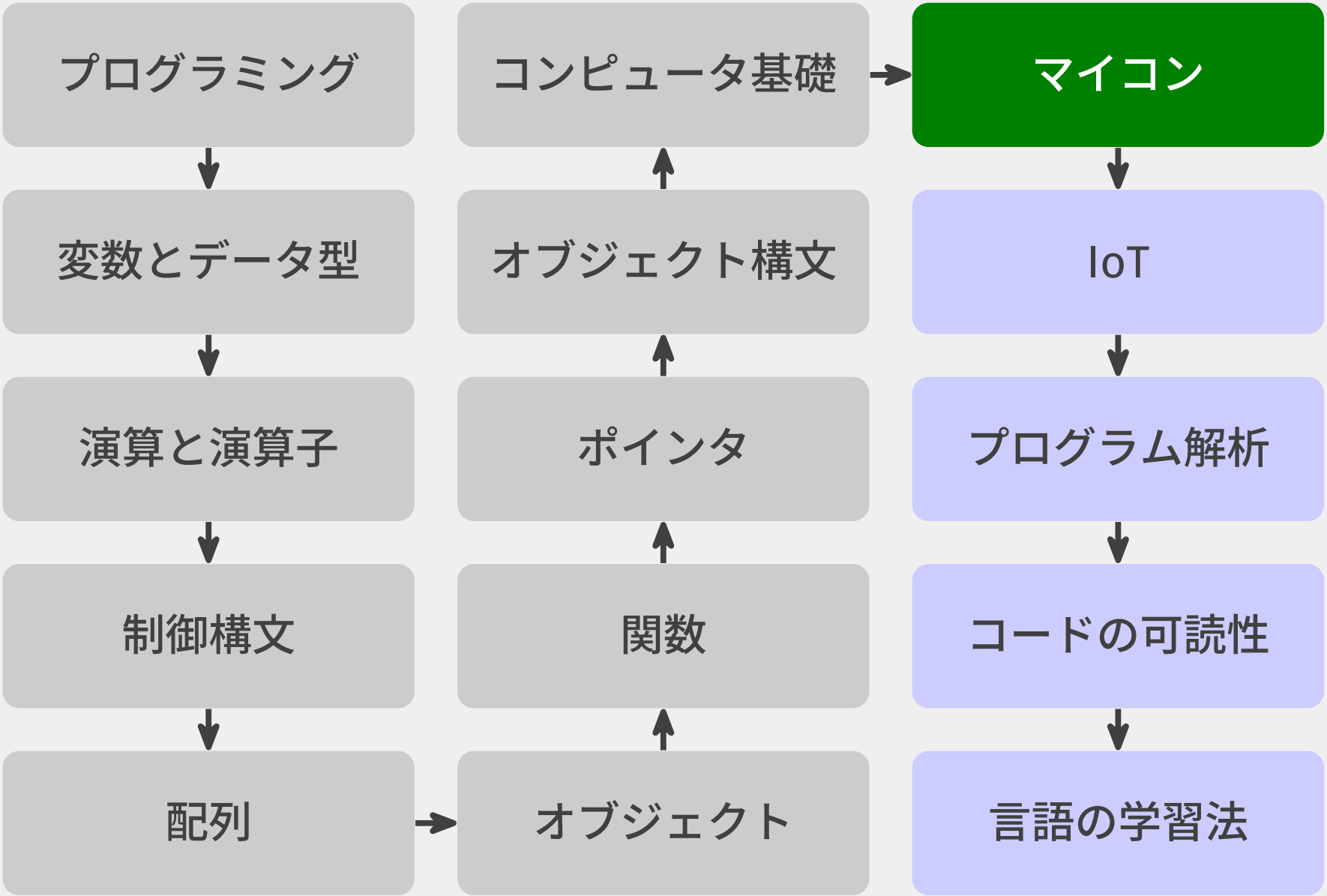
コンピュータ基礎

進度


プログラミング
変数とデータ型
演算と演算子
制御構文
配列
コンピュータ基礎
オブジェクト構文
ポインタ
関数
オブジェクト
マイコン
IoT
プログラム解析
コードの可読性
言語の学習法

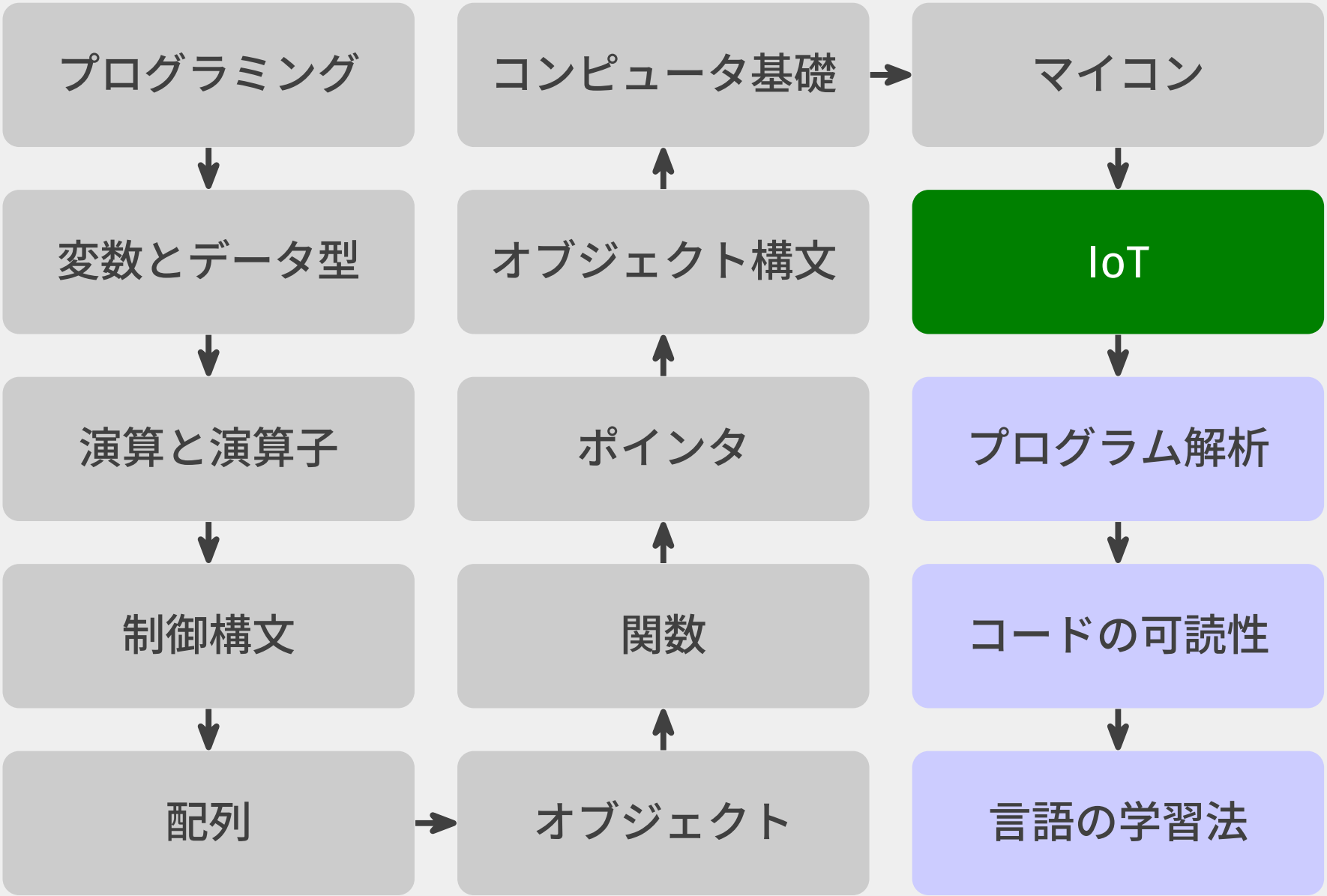
マイコン

進度


プログラミング
変数とデータ型
演算と演算子
制御構文
配列
オブジェクト
関数
ポインタ
オブジェクト構文
コンピュータ基礎
マイコン
IoT
プログラム解析
コードの可読性
言語の学習法

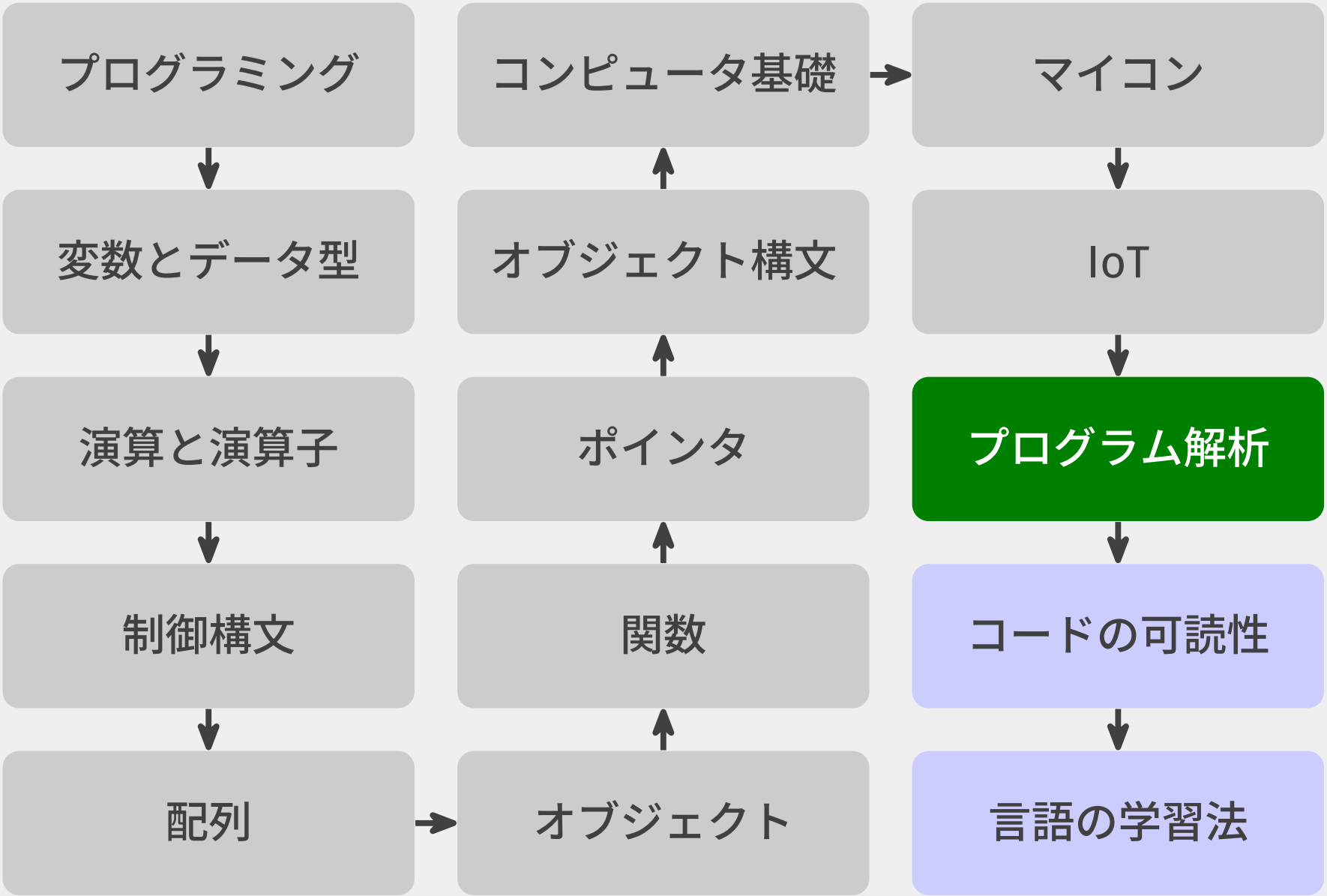
IoT

進度


プログラミング
変数とデータ型
演算と演算子
制御構文
配列
オブジェクト
関数
ポインタ
オブジェクト構文
コンピュータ基礎
マイコン
IoT
プログラム解析
コードの可読性
言語の学習法

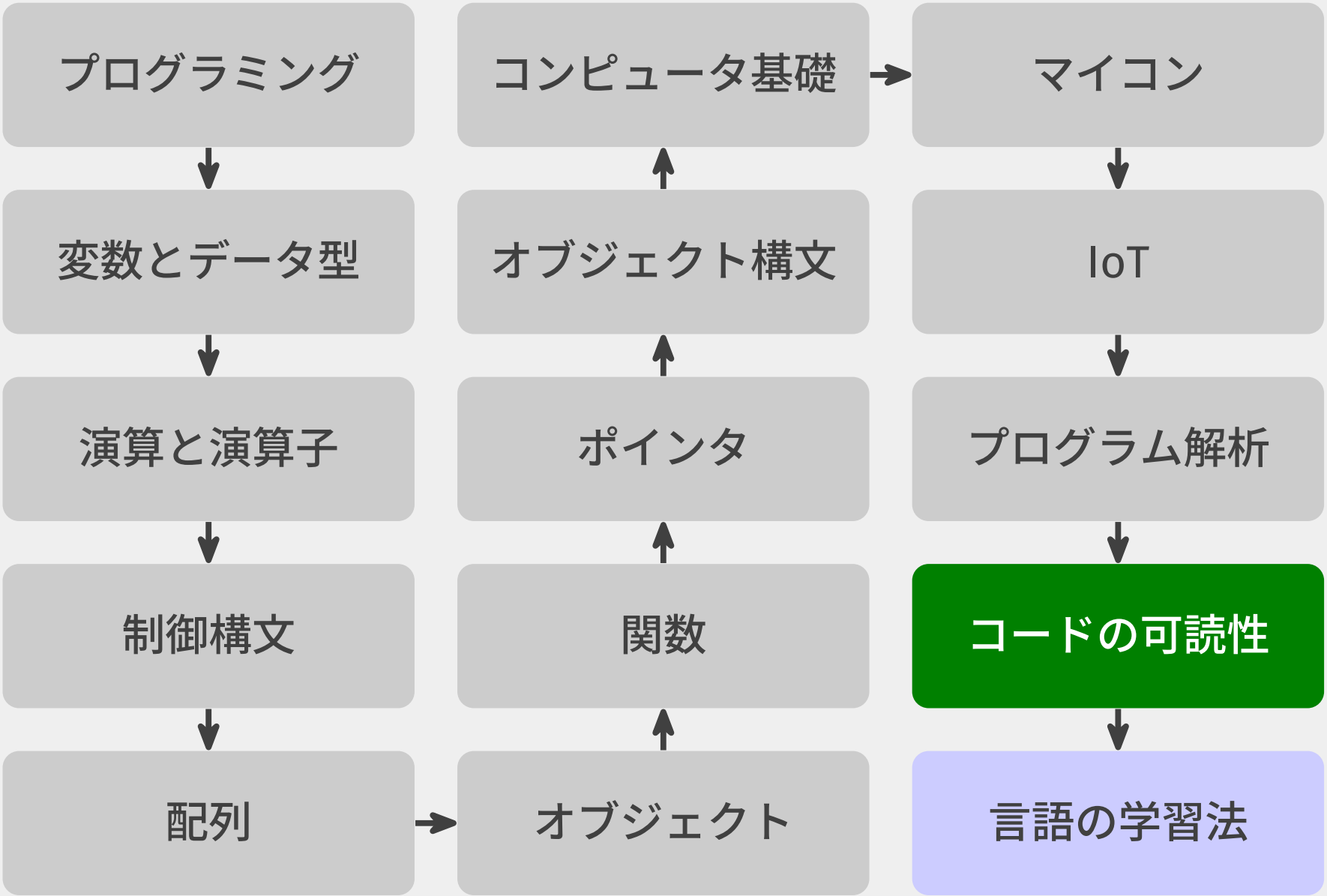
プログラム解析

進度


プログラミング
変数とデータ型
演算と演算子
制御構文
配列
オブジェクト
関数
ポインタ
オブジェクト構文
コンピュータ基礎
マイコン
IoT
プログラム解析
コードの可読性
言語の学習法

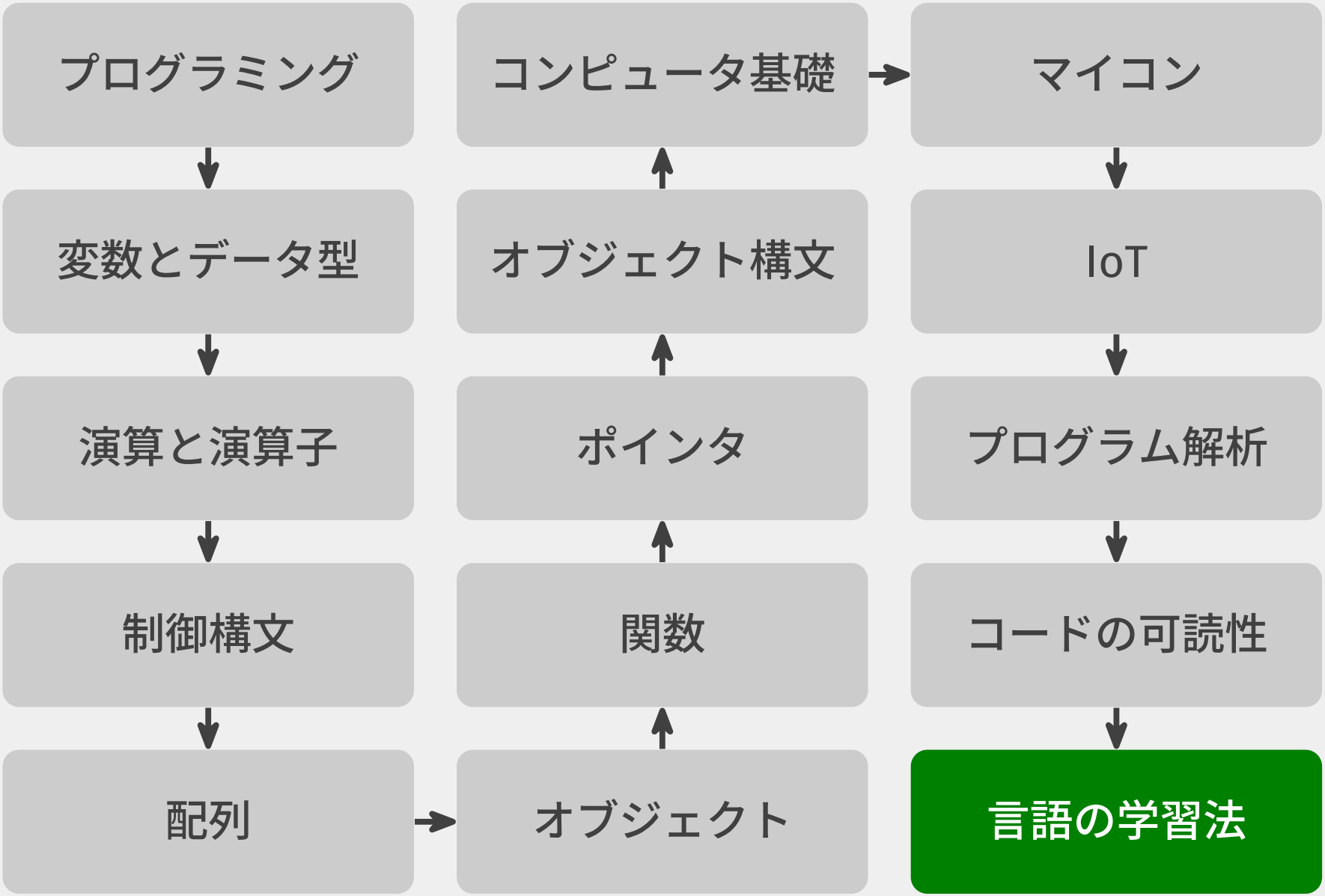
コードの可読性

進度


プログラミング
変数とデータ型
演算と演算子
制御構文
配列
オブジェクト
関数
ポインタ
オブジェクト構文
コンピュータ基礎
マイコン
IoT
プログラム解析
コードの可読性
言語の学習法

言語の学習法

進度


プログラミング
変数とデータ型
演算と演算子
制御構文
配列
オブジェクト
関数
ポインタ
オブジェクト構文
コンピュータ基礎
マイコン
IoT
プログラム解析
コードの可読性
言語の学習法

問題解決能力