

# デザインレシピに基づいた初学者のための学習環境

能勢純弥 叢悠悠 増原英彦 (東京工業大学)



デザインレシピ[Felleisen+ '18] : プログラムを作ることによって問題解決する一連の流れ

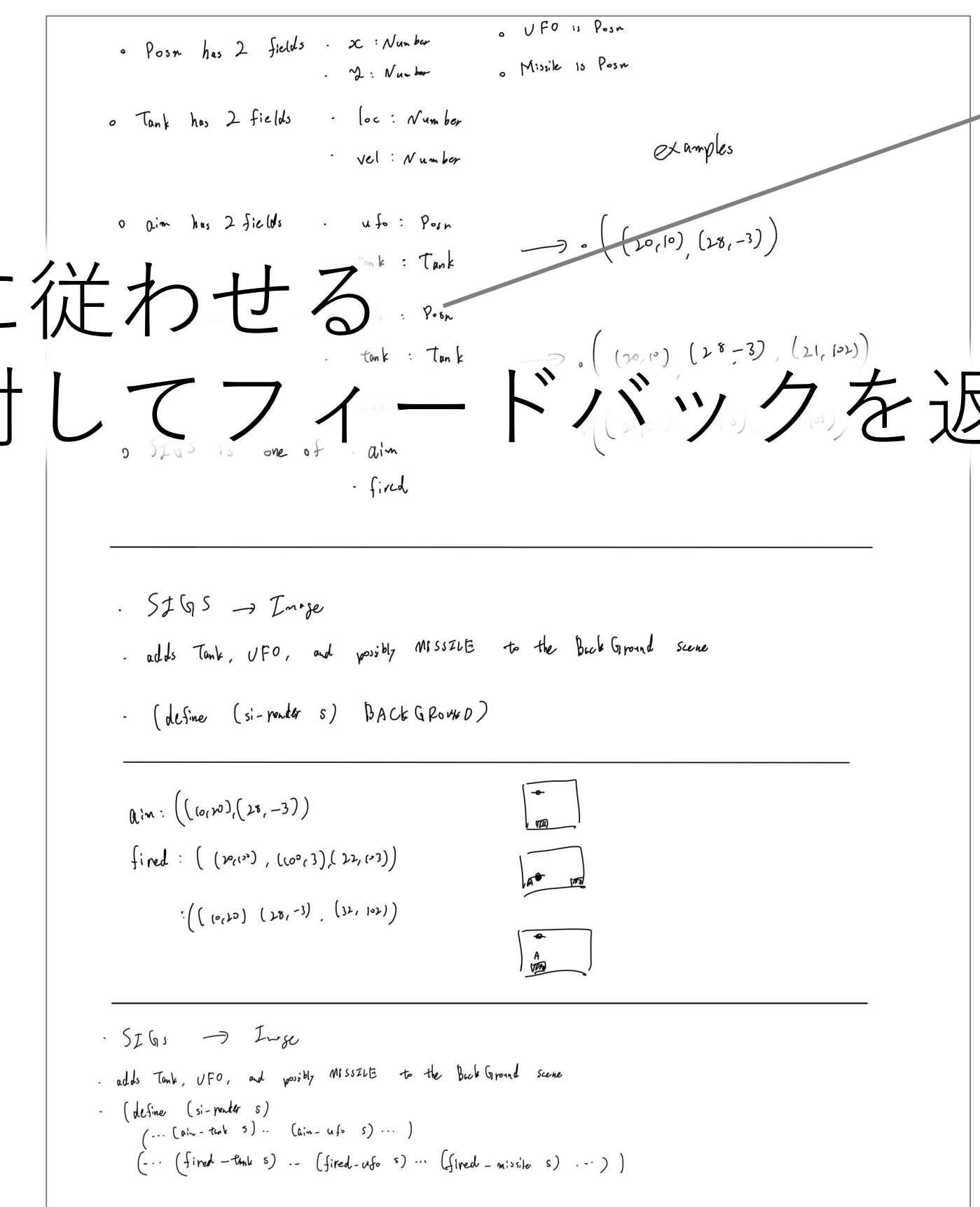
提案: ブロック言語による支援環境

1. データ定義・データ例の作成
2. 目的文・契約分・ヘッダ
3. 入出力の例
4. 関数のテンプレート
5. 関数定義
6. テスト

従来: 教育者の監督下で、手書きなどが中心

## 教育者の役割

- ①ステップを順番に従わせる
- ②中間ステップに対してフィードバックを返す



### Mio

①前のステップが完了しないと次に進ませない制約

②機械上で表現でき、書きやすい形を与えることでフィードバックが可能

```

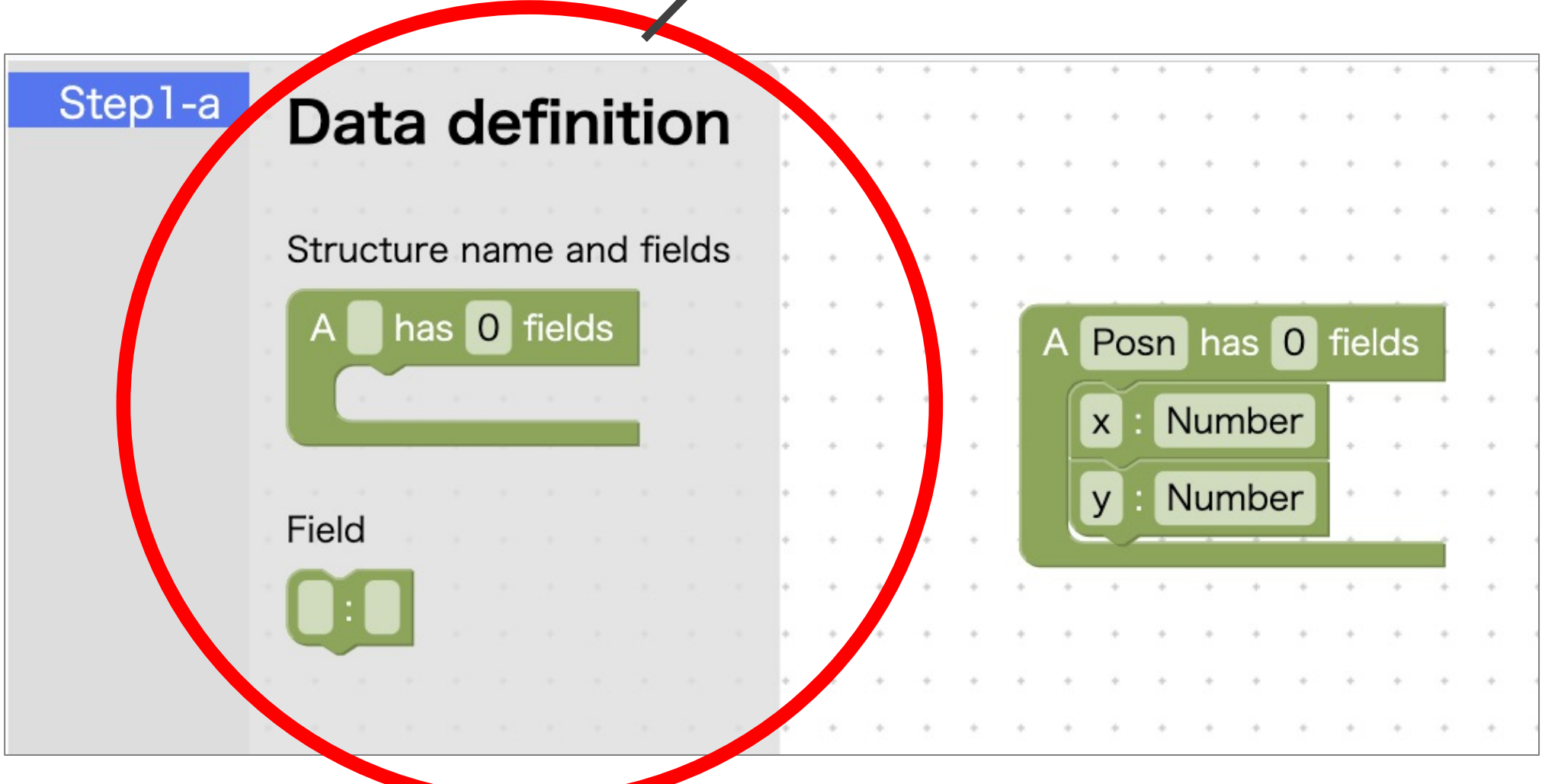
1 (define-struct posn [ x y ])
2 (make-posn 3 4)
3
4 (make-posn 5 12)
5
6 (check-expect (distance-to-0 (make-posn 3 4)) 5)
7
8 (check-expect (distance-to-0 (make-posn 5 12)) 13)
9
10 ;; compute the distance from p to 0
11 ;; Posn -> Number
12 (define (distance-to-0 p)
13   ... (posn-x p) ...
14   ... (posn-y p) ...
15 )
16
17 0
  
```

関数定義以降のステップはテキスト形式で実践

## Mioの機能

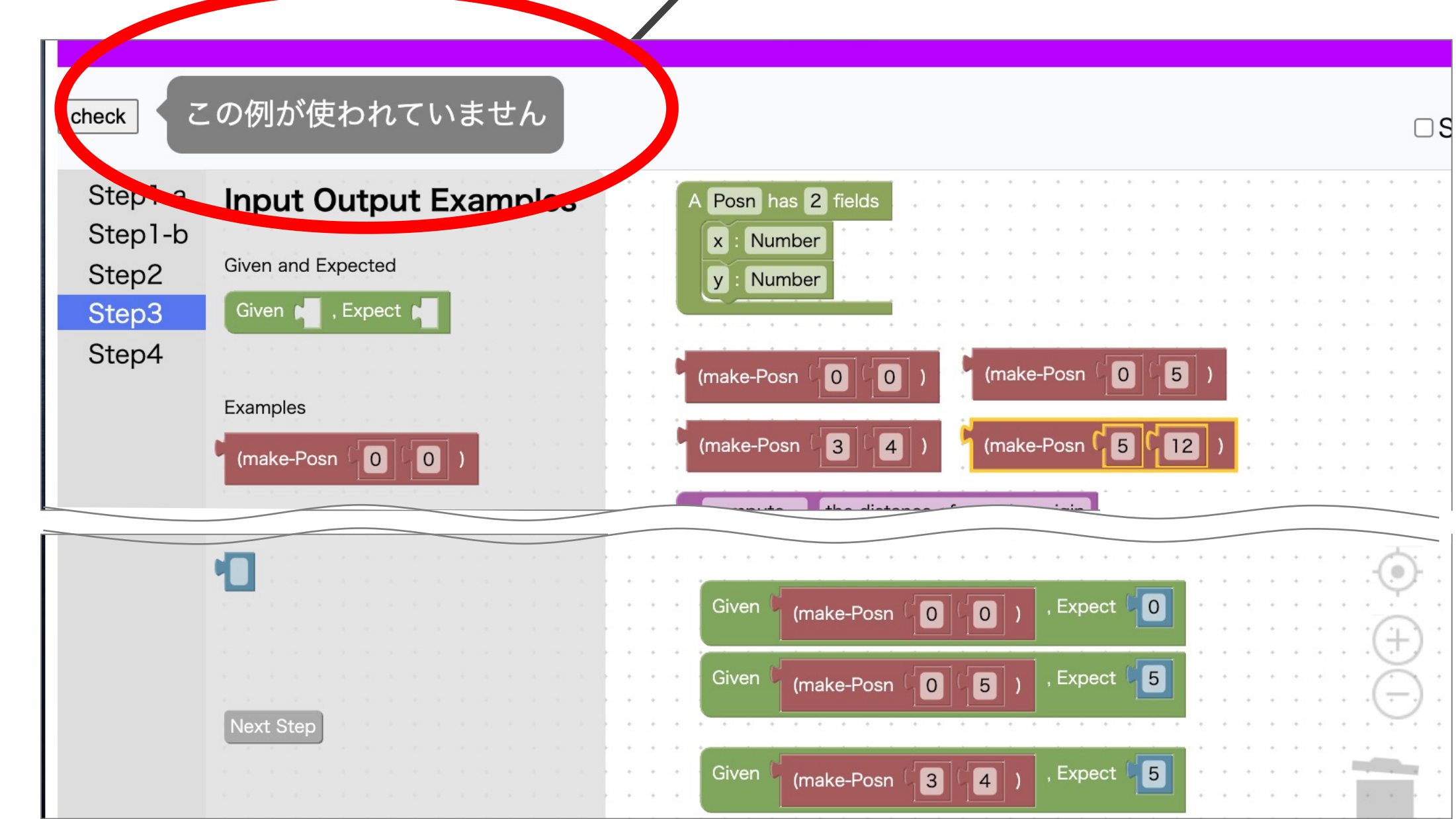
デザインを記述するブロック言語

データ定義を行うことと、それに必要な要素を明示



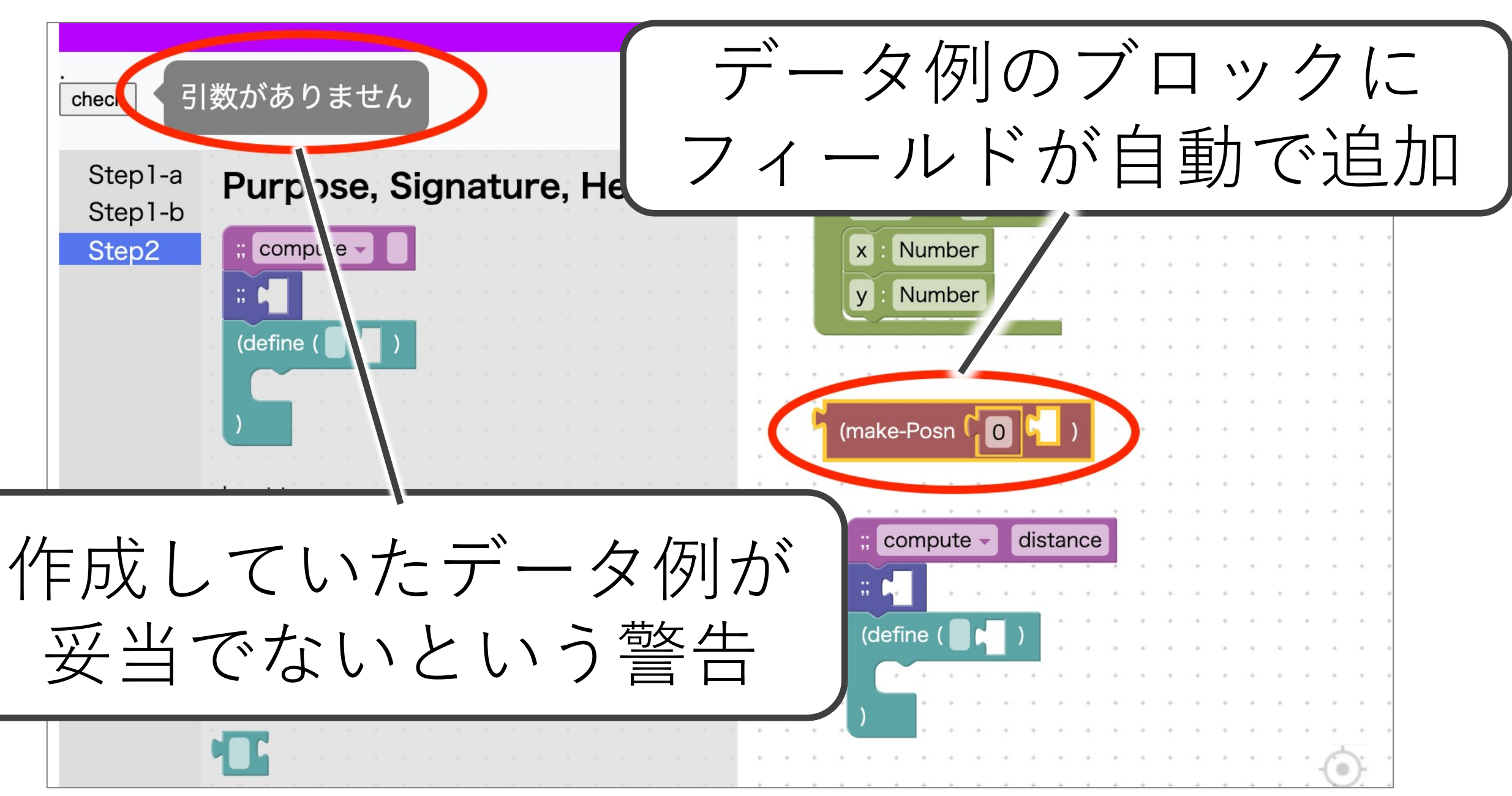
入出力例の網羅性などを検査

データ例が全て入力例として使われているか確認



自由な手戻りと矛盾検出

Step2の最中にStep1のデータ定義を修正  
データ例を修正せずにStep2に戻ると



作成していたデータ例が妥当でないという警告

## 今後の課題

### 1: 様々なパターンへの拡張

- ・構造的再帰のある関数
  - ・ネストした構造
  - ・アキュムレータを使用した関数 etc...
- =>いずれもブロックを追加することで対応可能

### 2: ユーザ実験

対象: 複数の学習レベルの学生  
 手法: いくつかの問題を、従来の方法とMioを用いてデザインレシピに従って解いてもらいアンケート調査

- ・従来の自由形式との比較
- ・フィードバックの内容やタイミング
- ・前後のステップの際の矛盾検出 etc.