

The background image shows a large, white, multi-story building with a church tower and a courtyard with a fountain and a lion statue. The building has a dark roof and many windows. The church tower is on the right side, with a cross on top. The courtyard in the foreground has a central fountain and a lion statue. The overall scene is a well-maintained campus or estate.

Dagstuhl Seminar Report

Educational Programming Languages and Systems

Youyou Cong (Tokyo Institute of Technology)

Seminar Overview

Participants background:

- CS/PL
- Cognitive science

Structure:

- 10-min talks (Mon-Wed)
- Breakout (Thu & Fri)



Program Design by Blocks (Youyou Cong)

- Design in blocks, code in text
- Received positive feedback from students

Step1-a テンプレート
Step1-b マッチ文
Step2 パターンマッチの対象
Step3
Step4

shape

ケース
の場合は

Square()
Triangle(,)

使うもの
...
...

Shape は 2 つのケースを持つ

Square (length : Double)
Triangle (base : Double height : Double)

s = Square(3) を受け取ると、9 を返す
t = Triangle(4 , 5) を受け取ると、10 を返す

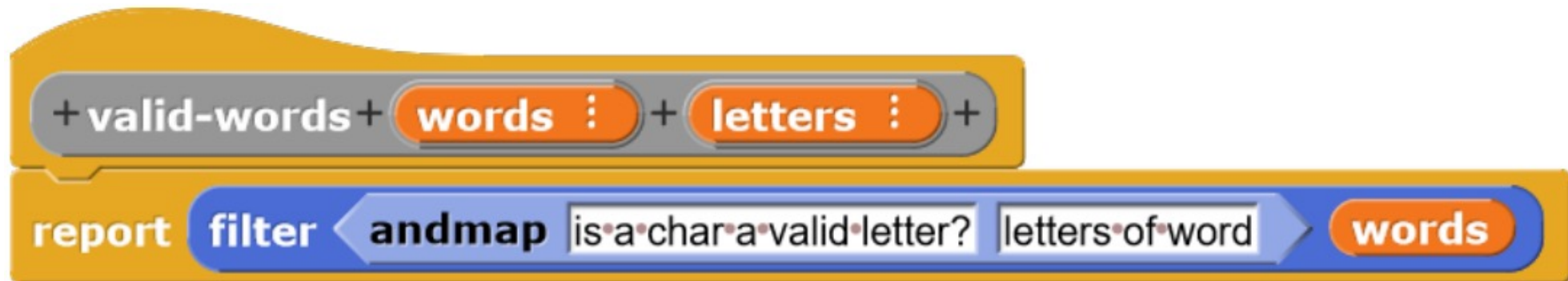
目的文 図形の面積を求める
関数 area は shape : Shape を受け取り Double を返す

パターンマッチの対象 shape
Square(length) の場合は ... length ...
Triangle(base , height) の場合は ... base ... height ...

```
1 abstract class Shape
2 case class Square(length: Double) extends Shape
3 case class Triangle(base: Double, height:
4 Double) extends Shape
5
6 s = Square(3)
7 t = Triangle(4, 5)
8
9 area(Square(3)) == 9
10 area(Triangle(4, 5)) == 10
11
12 //図形の面積を求める
13 def area(shape: Shape): Double = {
14 shape match {
15 case Square(length) => ...length...
16 case Triangle(base,height) => ...base...
17 ...height...
18 };
19 }
20 }
21
22
```

Program Planning via Higher-Order Functions (Shriram Krishnamurthi)

- Higher-order functions as primitives for planning
- Used to observe how students understand/use HOFs



PLTutor (Amy Ko)

- Semantic rules as causal relations
- Effective for learning tracing skills

Code	x == 0
Instruction	Push 0 onto the stack
Stack	0

Hedy (Feliene Hermans)

- Gradual learning via language levels
- Keywords in non-English languages

```
// level 1  
print hello world
```

```
// level 4  
print 'hello world'
```

```
// Japanese mode  
かけ hello world
```

Adaptive Parsons Problems (Barbara Ericson)

- Coding by dragging code fragments
- Support intra/inter-problem adaption

The interface is divided into two main sections: "Drag from here" (left, light blue background) and "Drop blocks here" (right, light green background). At the bottom, there are two buttons: "Check Me" (light blue) and "Reset" (dark grey).

Drag from here: A light blue box contains two code blocks. The top block is "counter = counter + 1" and is labeled "correct code block" with an arrow pointing to it. Below it is a "distractor" label with an arrow pointing to the same code block.

Drop blocks here: A light green box contains a sequence of code blocks. The first is "counter = 0". The second is a "while" loop: "while counter <= 10:". The third is "print(counter)". The fourth is "counter = counter + 2". An arrow from the "correct code block" in the "Drag from here" section points to this block in the "Drop blocks here" section.

Evening Panels

1. Teaching at scale
2. Evaluation
3. AI in education



Brainstorming Session

- What studies should we do together?
- What have we learned from building, deploying, and maintaining tools?



Non-academic Activities



What I liked about (this) Dagstuhl

- Small but diverse
- Not too packed, nor sparse
- Friendly and encouraging
- Good COVID policy



Links

- [Program design by blocks](#)
- [Plan composition via HOF](#)
- [PLTutor](#)
- [Hedy](#)
- [Adaptive Parsons problems](#)
- [Amy Ko's blog post](#)